Le menu des 27 prochaines semaines …

-0-

Informations diverses avant de prendre … la route

Avertissement:

Le document que vous avez entre les mains a été rédigé dans l'objectif d'un cours java « projeté» sur écran, et dispensé à des étudiants présents en amphi. En aucun cas, ce simple « powerpoint » ne se substitue à un ouvrage exhaustif dont vous pouvez trouver de bons exemples ci-dessous ou sur INTERNET. Le but est de faciliter le travail des étudiants en fournissant un support visuel, et écrit sous forme de slides que peut suivre séquentiellement l'enseignant. L'étudiant doit personnaliser son cours en prenant des NOTES (j'insiste!) Pour illustrer certains points, quelques passages ont été empruntés chez SUN ou parfois chez certains auteurs dont les productions ont été déposées sur le WEB, ou en librairie, et dont je donne les références ci-après. Pour des raisons de copyright, il est interdit sans autorisation, de les utiliser sous quelques manières que ce soient (copie, re-engineering, …) dans une intention lucrative, par exemple la rédaction d'un ouvrage en vue d'une vente, ou une formation JAVA dans le cadre d'un stage payant. En cas de manquement à ce conseil, le lecteur étant averti, l'auteur du document « Programmation objet » -votre serviteur- se dégage de toutes responsabilités quant aux conséquences. En cas d'usage universitaire par un collègue, s'il a un moment, qu'il m'en avertisse seulement: (jean-jacques.montois@univ-rennes1.fr).

Références bibliographiques:

SUN Microsystème, **java.sun.com**

DIP Genève, Alexandre Maret & Jacques Guyot

G. Falquet - Université de Genève 1 –Cours JAVA

P. Ducrot - ENSI de Caen – Cours JAVA

"Introduction à la programmation objet" par J. Brondeau, Ed: Dunod

"Programmation JAVA" par JF Macary, Ed: Eyrolles

Cours en ligne de Bruno Kostrzewa Prof. de Maths au Lycée Faidherbe

"JAVA" par M. Morisson & al. Ed: S&SM

"Le livre de JAVA premier langage" par A. Tasso, Ed: Eyrolles

"du C/C++ à JAVA" par E. Puybaret, cours web payant sur www.eteks.com, et un ouvrage

Le must!!

Excellent …

Excellent …

Excellent …

Bonnes résolutions … mutuelles!

Quelques sentences pour la route …

L'erreur est humaine, mais pardonner est hors des capacités du système d'exploitation
Proverbe informatique

«Si vous ne pouvez le faire bien, rendez le beau.»
Bill Gates, PDG et fondateur de Microsoft.
[A quel produit Microsoft pensait-il Bill?]

«J'ai toujours rêvé d'un ordinateur qui soit aussi facile à utiliser qu'un téléphone. Mon rêve s'est réalisé. Je ne sais plus comment utiliser mon téléphone.»
Bjarne Stroustrup, auteur du langage C++

« La machine a gagné l'homme, l'homme s'est fait machine, fonctionne et ne vit plus »
Alahauna Gandhi

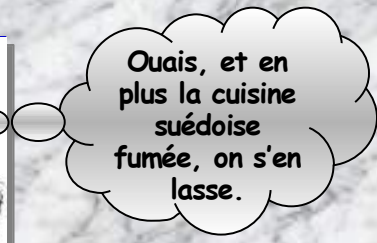
Digressions philosophiques ...

Il m'est difficile, de résister au plaisir de citer dans cet humble cours de java, les célèbres préceptes du non moins célèbre "discours de la méthode", formulés par notre grand philosophe logicien français, René Descartes. De mon avis, Descartes exprime dans les quatre règles qu'il énonce, l'attitude intellectuelle d'un informaticien face à l'élaboration d'un projet informatique (...et bien d'autres sciences !). J'ose même imaginer que le fameux langage PASCAL à la base de la programmation structurée, eusse pu se nommer le "langage DESCARTES" eut égard aux concepts de la logique du raisonnement, avancés par Descartes; - "cogito ergo sum" - affirmation logique s'il en est! Il eut été justice d'associer son nom à la technologie informatique dans la mesure où Pascal possède déjà un "triangle", un "pari", et une machine arithmétique. Voyons ces fameuses règles …

… je crus que j'aurais assez des quatre [préceptes] suivants, pourvu que je prisse une ferme et constante résolution de ne manquer pas une seule fois à les observer:
Le premier était de ne recevoir jamais aucune chose pour vraie que je ne la connusse évidemment être telle: c'est à dire, d'éviter soigneusement la précipitation, et la prévention, et de ne comprendre rien de plus en mes jugements, que ce qui se présenterait si clairement et si distinctement à mon esprit que je n'eusse aucune occasion de le mettre en doute.
Le second, de diviser chacune des difficultés que j'examinerais en autant de parcelles qu'il se pourrait, et qu'il serait requis pour mieux les résoudre.
Le troisième, de conduire par ordre mes pensées, en commençant par les objets les plus simples et les plus aisés à connaître, pour monter peu à peu comme par degrés jusqu'à la connaissance des plus composés: et supposant même de l'ordre entre ceux qui ne se précèdent point naturellement les uns les autres.
Et le dernier, de faire des dénombrements si entiers et des revues si générales que je fusse assuré de ne rien omettre.

Ainsi, les principes généraux des langages objets et structurés apparaissent complètement dans ces 4 préceptes !

P'tite histoire : *savez-vous de quoi est mort le grand René Descartes? non? Et bien à cause de la Reine Christine de Suède! Celle-ci avait un goût prononcé pour l'étude des mathématiques la nuit dans son château (à chacun ses fantasmes). Le château n'étant évidemment pas chauffé au fuel ou à l'électricité, Brrr! Elle faisait lever notre brave René (alors à son service; faut bien boulotter !) pour lui fournir la quintessence de son brillant esprit. A ce régime, Descartes finit par attraper une bonne fluxion de poitrine, que l'on guérissait en Suède en appliquant ... de la neige sur la poitrine! Combattre le mal par le mal (véridique! Georges Boole est mort de la même façon. Sa femme usait du même moyen. A croire que la logique mène toujours au 0 ... degré!)*



Attitudes, et comportements du bon programmeur …

Curieux – Attachez vous à mieux connaître votre environnement de travail. N'hésitez pas à surfer dans les répertoires système afin de mieux connaître la structure hiérarchique du système de fichiers, ouvrir les fichiers qui s'y trouvent pour en apprécier leur contenu (sans les modifier!), lire les fichiers *.h, *.lib, *.doc, … Une règle cependant: ne jamais altérer les informations visionnées. Retenez, en informatique, la curiosité n'est pas un vilain défaut, c'est une excellente qualité! Ah, j'oubliais, repérez un cador en info, ces passionnés dont les yeux brillent dès qu'on prononce if-then-else. Tachez de devenir copain, c'est sûr, il mettra un point d'honneur à répondre à vos interrogations!

Précis – La programmation requiert une précision diabolique: là où il faut un point virgule, n'écrivez pas une virgule, et quand bien même vous recompileriez sans cesse votre source jusqu'à la fin des temps, l'ordinateur vous répondra invariablement : « Error: Invalid Descriptor » ou autre injure incompréhensible. Souvenez-vous, il est plus têtu que vous, il ignore le temps qui s'écoule, vous non!

Méthodique - C'est pas grave si vous êtes d'un naturel désordonné, vous aimez «répartir vos affaires », d'autres ont la manie de tout ranger, chacun sa façon de vivre. Soyons tolérant. Mais en programmation point de fantaisie, pratiquez l'ordre et la méthode , et puis relisez les 4 règles de Descartes dans le « discours de la Méthode » pour vous forger une ligne de conduite.

Persévérant, pugnace- « *100 fois sur le métier il faut remettre l'ouvrage* … ». Si le programme ne fonctionne pas, prenez du recul, «laisse béton», aller prendre un verre avec un pote à la « Caravelle » (après les cours, *of course*), parlez de tous et de rien, dégagez vous l'esprit, et reprenez le lendemain votre programme; vous verrez, c'est souverain. « fiat lux »!

Autonome- ne comptez que sur vous-même pour résoudre les mystères d'un programme qui ne veut point fonctionner malgré vos efforts durant 4h de TP; et surtout éviter les conseils des messieurs YAKA-FAUCON, « j'sais-tout , sur tous ». Souvenez-vous des vers de La Fontaine « *S'agit-il d'un conseil? La Cour en conseillers foisonne; faut-il exécuter, on ne trouve plus personne* ». Vous pouvez aussi solliciter l'aide d'un prof, on ne sait jamais, il peut trouver le bug!

Cool- face à un programme récalcitrant; surtout pas de « gros mots » (enfin pas de manière trop sonore, et seulement le mot de Cambronne; … c'est être patriote!!), et ne cassez pas l'ordinateur en tapant dessus, il n'y est pour rien!! Quoique … ?

Non-conformiste Habillez-vous comme vous le sentez, venez à mes cours en kilt écossais, en indien cheyenne, ou en bure de moine si çà vous chante (ma préférence va au chapeau à plumes, perruque, pourpoint et rhingrave), mais surtout, respectez les standards informatiques, et les règles de l'écriture de programme, sinon aucune chance que votre soft fonctionne avec d'autres.

C++ attitude …

Bjarne Stroustrup, créateur du C++, en plein travail …

A méditer …

Jadis, les hommes édifièrent une tour à Babel city. Ils firent monter du parpaing le plus haut possible afin de montrer leur force et leur grandeur (genre tour Montparnasse, ou Worla Trade Center - Enfin, ce qu'il en reste -, mais vraiment en plus grand !). Dieu voyant cela, décida de punir l'orgueil des hommes en leur imposant l'usage de très très nombreux langages et dialectes afin de semer la confusion et faire voir ainsi qui était le maître (non mais!). *« … tous les habitants de la terre parlaient encore la même langue. Ils voulurent construire tous ensemble une ville et une tour dont le sommet atteindrait les cieux. Alors Dieu, ne voulant pas les foudroyer, confondit leurs langages pour qu'ils ne puissent plus continuer de parler et travailler ensemble. Ils cessèrent de bâtir leur tour, qui s'écroula, se séparèrent, et se dispersèrent.* ***C'est ainsi que sont nés les différents langages*** *… » [Bible]*

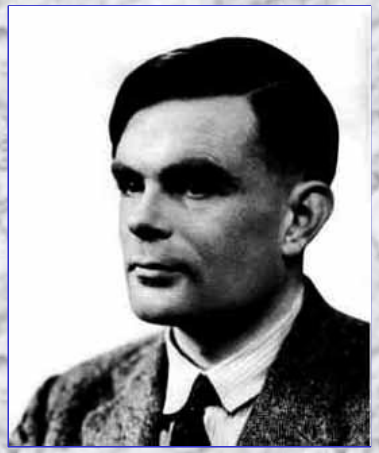
Ainsi, aujourd'hui pour écrire nos programmes, nous utilisons des tas de langages (C, C++, C#, Perl C, JAVA, PASCAL, Eiffel, ADA, fortran, lisp, basic, PHP, JSP, XML, HTML, …), des tas de protocoles de communication (TCP/IP, NETBEUI, IPX, ..) [environ 600 langages recensés; quand aux protocoles de communications, on ne les dénombre plus!].
Or les langages assembleur(s), basic, fortran, cobol, ada, pascal, c, c++, java, lisp, prolog, etc etc sont bien sûr tous incompatibles les uns des autres. Ainsi, Dieu, 1ier programmeur système de l'Univers, n'avait pas oublié l'informatique future! «Progranticipation » divine. Et depuis 70 ans, l'homme continue fébrilement, frénétiquement, pathétiquement à construire moult idiomes informatiques, et pérenniser ainsi la malédiction divine …

A savoir …

Trois géants! Ils ont contribué aux fondations de l'informatique. Retenez ces noms, ces visages, ils ont eu chacun un destin grandiose et une fin ... misérable, tragique. Harcelés, frappés par la bêtise humaine, surtout lorsqu'elle agit en meute bien-pensante, fraternelle disent-ils, et se réfugie, derrière la «normalité» grégaire,intolérante de leur Vérité! Pourtant ces personnalités appartiennent à ces magnifiques «locomotives» qui tirent l'humanité, à ces héros de la pensée constamment conspués, décriés, mais en définitive … immortels! Albert Einstein disait souvent: ***« Il y a 2 choses d'infini: l'Univers, et la bêtise humaine; quoique je n'ai pas encore acquis une certitude absolue pour …l'Univers!»***

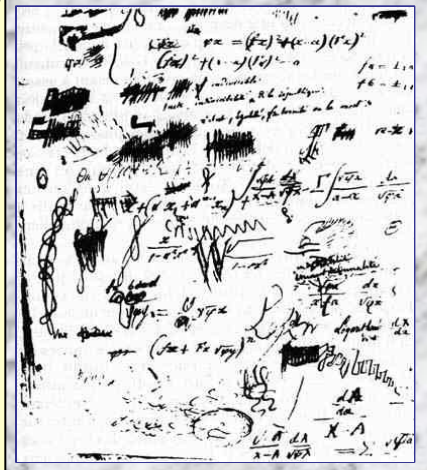
Adélaïde of Byron (Ada pour les intimes) [1816-1852], fille du poète romantique anglais Lord Byron (ses poésies exaltent le héros rebelle). Ada était membre de la « jet society » anglaise et épouse du Comte of Lovelace de 30 ans son aîné. Egérie du savant Lord Babagge (inventeur de la machine analytique), elle attacha son nom à des études sur les systèmes de traitement automatique des données. Ada formula, la première, les concepts théoriques de la programmation qu'elle appliqua à la « machine analytique » de Babagge. Le grand Von Neumann devait se souvenir de cela quand il mit au point sa propre architecture informatique en … 1949. Ada est considérée comme le 1ier programmeur. Du fait d'un comportement non conforme aux usages de la société puritaine de l'époque («dilapidation» de son patrimoine pour soutenir ses recherches, adultère, jeu, dettes, mépris des crétins importants, …), Ada fut internée dans un asile sur la demande de sa mère, la redoutable lady Noll Byron (femme d'une grande culture), jusqu'à la fin de son existence (1852) où elle mourut d'un cancer dans des conditions particulièrement atroces, d'abandon, de dénuement, sans aucune compassion! Destin et fin tragique à rapprocher de celui de Camille Claudel, elle même égérie du sculpteur Rodin, et morte exactement dans les mêmes conditions misérables, implorant son intransigeante mère. Décidément, à cette époque, il ne faisait pas bon pour une femme d'être libre comme un … homme! [Très belle exposition de Camille Claudel à Dinan automne 2005, musée des Jacobins]. Le monde scientifique lui a rendu cependant hommage en donnant son prénom (ADA) au meilleur langage de programmation qui soit. Noter l'étrange destinée du père et de la fille morts tous deux à 36 ans dans des conditions violentes.

L'anglais **Alan Turing**, professeur au King' collège responsable durant la seconde guerre mondiale du bureau du chiffre. Mathématicien surdoué, à 24 ans il apporte une contribution décisive aux fondements logiques de l'informatique. Il définit les principes théoriques des « machines logiques ». Il donne une réponse « mécanique » au problème de la décidabilité énoncé par le grand Hilbert. Grâce à lui, les messages codés par la machine de cryptage : ENIGMA en usage dans la marine allemande étaient régulièrement déchiffrés. Incontestablement Turing a permis d'écourter la durée du conflit économisant ainsi des centaines de milliers de vies. Le véritable vainqueur de la bataille de l'atlantique, c'est lui ! On lui doit aussi les premières recherches sur l'intelligence artificielle. Personnage farfelu, imprévisible, trop brillant et … homosexuel. Pour cette dernière raison, il fut mis à l'index, mis à l'écart et condamné par la justice anglaise à suivre un dur traitement hormonal ou bien être incarcéré. Turing choisit le traitement médical qui transforma quelque peu son apparence masculine. Un beau jour Alan finit par croquer une pomme qu'il avait enduit de cyanure (il aimait bien les pommes, et … blanche-neige qui représentait pour lui le symbole de la pureté). C'était sa manière de dire tchao à cette société qui l'avait si mal remercié ! Son destin est à rapprocher de celui d'Oscar Wilde.



Evariste Galois [1811-1832]; personnage balzacien à la Julien Sorel, ou Rastignac. Il fut l'un des plus grands mathématiciens de tous les temps, sinon le plus prodigieux. Ses travaux ont révolutionné les mathématiques; ils sont directement à l'origine de la cryptographie en informatique, de la théorie de l'organisation des structures cristallines (utile pour Pierre-Gilles Degenne, prix Nobel pour la découverte des cristaux liquides. Pensez à Evariste lorsque vous regardez l'écran LCD de votre portable); récemment, Andrew Wiles a pu bénéficier des travaux de Galois pour résoudre l'un des plus fameux problème mathématique: le grand théorème de Fermat. Il découvre les mathématiques à 15 ans, et meurt dans un duel stupide à 20 ans à cause « d'une infâme coquette … ». Jeunesse difficile, dont le père adoré, maire de Bourg-La-Reine s'est suicidé suite à une campagne venimeuse de diffamation. E. Galois, personnalité tourmentée, alternant entre des actions politiques [républicain passionné, il se jette dans la mêlée politique. Lors de la période des « 3 glorieuses » de juillet 1830, il monte sur les barricades parisiennes pour défendre la Liberté, et le droit de vote sérieusement écornée par Charles X], et des travaux mathématiques ponctués souvent par des séjours dans les prisons de St Pélagie, ou de la Force conséquemment à ses activités politiques, harcelé par la police secrète de Charles X. Mathématicien parfaitement incompris, souvent combattu avec hargne, l'audace des ses idées fait peur, il est trop en avance des perceptions de son temps, son approche est trop inductive. Il est, entre autre, l'inventeur de la théorie des groupes, qu'il applique à la résolution d'un des plus grands problèmes mathématiques irrésolus: La résolubilité des équations algébriques de degré N par l'étude des groupes de permutations issus des fonctions symétriques des racines. Les nombreux mémoires (rédigés à la plume d'oie) qu'il soumet à l'académie ou qu'il adresse aux grands mathématiciens de l'époque (Poisson, Cauchy, Fourier, …) sont soit égarés, soit négligés car incompréhensibles. Parfois méprisé, voire conspué par les sociétés savantes; on l'accuse même de plagiat des travaux du jeune mathématicien Abel (mort de tuberculose à 25 ans, misérablement, et dans le dénuement) qui travaillait aussi sur les équations algébriques. Refusé par 2 fois à l'entrée de polytechnique, en effet l'examinateur Mr Binet comprend mal les réponses d'Evariste, ce dernier excédé finit par lui lancer l'éponge à la figure « Voici, monsieur, ma réponse à votre question! », et il sort en claquant la porte . L'année suivante, il est exclu de la toute nouvelle école normale supérieure, toujours pour insubordination, et activité politique subversives. En effet, tandis que le Directeur pro-Charles X consigne ses condisciples dans l'enceinte de l'école afin qu'ils ne participent pas aux 3 jours d'émeutes qui allaient devenir les … « 3 glorieuses », Evariste fait le mur de l'école pour prendre part à ses journées historiques en tant que membre de la garde nationale. A 20 ans, Evariste découvre l'amour, non partagé, avec Stéphanie Poterin du Motel –une coquette- qui le conduit presque aussitôt à la mort lors d'un duel imbécile un matin de mai 1832, aux étangs de la Glacière à Paris. Abandonné par ses témoins sur le lieu de l'affrontement, il agonisera 24h presque sans soutien à l'hôpital Cochin, et le 31 Mai 1832, le monde perd l'un des esprits les plus prodigieux de tous les temps. Il sera enterré à la fosse commune du cimetière du MontParnasse (comme Mozart, étrange similarité de destin pour 2 esprits jumeaux: géniaux, rebelles, jeunes, et tourmentés). Fidèle à son esprit rebelle, la nuit précédant son duel, au lieu de s'entraîner au pistolet, ou tout au moins de dormir, Evariste préfère reconstituer l'essentiel de ses travaux égarés. Il se savait condamné, mais avait conscience de l'importance de ses Recherches pour l'Humanité. Il a donc résumé en quelques heures, dans le silence de sa petite chambre-logement, à la lumière vacillante de la bougie, l'essentiel de ses idées. Il faudra 50 ans pour que l'on comprenne enfin l'immense intérêt de ce testament scientifique légué à l'Homme. A mesure que l'aube pointe, le crissement de la plume se fait plus rapide, Evariste sait qu'il n'a plus le temps de préciser certains résultats «je n'ai pas le temps, je n'ai pas le temps, … », «le lecteur démontrera lui-même … » écrit-il. Parfois, Evariste s'insurge, et griffonne rageusement dans la marge du mémoire-testament « Liberté, égalité, fraternité, ou la mort … ». Le jour pointe à travers la petite fenêtre de sa chambre, Evariste fatigué range ses papiers, rédige à son frères ses dernières recommandations, enfile sa redingote, et tel un héros romantique, part vers son destin … Citoyen étudiant, lorsque tu votes, aie une pensée pour le malheureux Galois et ses compagnons républicains qui t'ont permis de faire ce geste! Quidam, lorsque tu tapotes le clavier de ton téléphone, de ton ordinateur, que tu surfes sur le WEB, pense à Evariste associé à l'avènement de ces technologies!

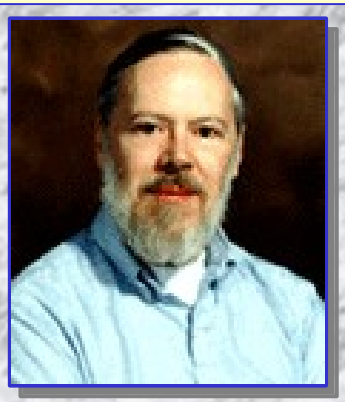
Evariste vers 18 ans.



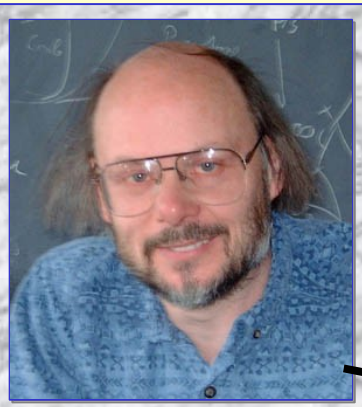
Dernière nuit, derniers écrits …

Les responsables de ce qui vous arrive aujourd'hui …

C'est par moi que tout commence comme concepteur du langage C : Dennis RITCHIE



Moi, j'ai continué avec le C++ :
Bjarne Stroustrup



Moi, j'ai innové avec java, proche du C++ :
James Gosling

Moi, c'est surtout le concept GNU (et plein d'autres choses: emacs, gcc …) qui vous permet d'utiliser plein de freeware, notamment linux, … : Richard Stalmann

Là, je réfléchis à mon programme C++

Bjarne Stroustrup

Là, sur la photo, c'est ma période mystique, Woodstock,Katmandou. (Non, non, je n'ai pas de lien de parenté avec Lui !)

Pourquoi les langages objet?

- **Représenter directement les entités du monde réel dans les environnements informatiques, sans nécessité de les déformer ou de les décomposer.**
- **Réutiliser et étendre les logiciels existants, à partir de bibliothèques spécialisables, et modifiables**
- **Facilités de prototypage rapide des applications sans créer le corps des procédures.**
- **Facilités d'exploitation du parallélisme pour la mise en oeuvre de structures multiprocesseur;**
- **Faciliter la conception d'interfaces homme/machine**
- **Accroître la productivité des développeurs de logiciel.**

Historique des langages objet

Approche objet:

- **Associe données et traitements dans une même entité.**
- **Née avec le langage SIMULA, : Dahl O. et Nygaard K.** *" SIMULA, An Algol based simulation language", ACM, 1966*
- **SIMULA a introduit les concepts de classe**
- **SIMULA est un langage structuré permettant de simuler des processus parallèles**
- **La notion d'abstraction de l'approche objet "types abstraits de données" a été formalisée dans les années 70:**

  GuttagJ.V, Horowitz E. Musser D., *"The design of data type spécifications", in current trends in programming methodology, Prentice Hall, Englewood Cliff, 1977*

  Liskow B, Zilles S., *"Specification techniques for data abstractions", IEEE Transaction on software Engineering, 1975*

Un type abstrait se compose d'une interface qui définit la vision offerte aux utilisateurs en termes d'opérations et d'une implémentation en terme de structures de données et d'algorithmes définissant les fonctions.

✓ **Émergence de la programmation objet avec l'apparition de SMALLTALK. Langage développé durant les années 70 au centre de recherche de XEROX (parc de Palo Alto).**

✓ **SMALLTALK intègre les technologies d'interaction avec l'utilisateur par icônes, fenêtres et souris. Ces technologies ont servi de base au Macintosh**

Les langages objets principaux

- **SIMULA - construit au départ pour résoudre des problèmes de simulation de processus parallèles**
  - ✓ **syntaxe proche du langage ALGOL**
  - ✓ **permet de définir des classes**
  - ✓ **héritage simple**
  - ✓ **mécanisme des fonctions virtuelles**
  - ✓ **classe Echéancier pour gérer des processus déclenchables à des instants précisés.**
- **SMALLTALK vise plutôt les applications interactives**
  - ✓ **conçu durant les années 70 par Golberg et Kay à Xérox**
  - ✓ **gère les concepts d'objet, de classe, d'héritage, d'envoi de messages.**
  - ✓ **orienté vers la communication homme/machine.**
  - ✓ **supporte les notions de polymorphisme**
- **ADA plutôt destiné aux applications multitâche temps réel**
  - ✓ **Conçu par J. Ichbiah pour le compte du DOD (70)**
  - ✓ **Langage d'usage général de type PASCAL.**
  - ✓ **ADA n'est pas exactement un langage objet.**
  - ✓ **Intègre la notion de type abstrait à travers le "paquetage" (package).**
  - ✓ **Les données, fonctions et procédures peuvent être exportées ou privées.**
  - ✓ **Intègre des mécanismes multitâches.**

- **C++ , développé en 80 par Bjarne stroustrup des laboratoires Bell.**
  - ✓ **Langage orienté P.O.O**
  - ✓ **Extension du langage C; C++ corrige les défauts du C vis à vis du typage.**
  - ✓ **C++ à mi-chemin entre un langage structuré (C) et un langage totalement P.O.O (SMALLTALK)**
  - ✓ **Concepts de classe, d'héritage, de polymorphisme emprunté à SIMULA.**
  - ✓ **Le concept de classe généralise le constructeur struct.**
  - ✓ **Encapsulation totale ou partielle des données.**
  - ✓ **droits d'accès *private*, *public*, *protected* des objets d'une classe.(sur-ensemble orienté objet du langage C).**
  - ✓ **Dérivation des classes. héritage multiple**
  - ✓ **Concept de fonction amie (friend)**
  - ✓ **Surdéfinition des fonctions et des opérateurs**
  - ✓ **En attendant la normalisation définitive du C++ ANSI, ce sont les publications d' AT&T qui servent de références:**
  - ✓ **versions 1.1(86), 1.2(87), 2.0(89), 3.0(91) [version de travail de l'ANSI];**
  - ✓ **Nouvelles possibilités d' E/S, basées sur la notion de "flux".**
  - ✓ **C++ permet de transiter doucement d'une programmation algorithmique (WIRTH) à une programmation P.O.O (préserver le "savoir faire" et l'existant).**

Histoire courte du langage Java et anecdotes

- **1990 : Naissance du langage Java, appelé initialement Oak (chêne ??).**
- **Langage créé par une équipe de Sun Microsystems, dirigée par James Gosling, effectuant le développement de concepts sur les logiciels incorporés dans les appareils électroniques de grande consommation. Contrainte majeure : différentes plateformes et microprocesseurs.**
- **1ère application: un contrôleur de poche interactif pour l'équipement de loisir domestique, destiné aux chaînes de télévision câblées numériques (en 1991!!).**
- **1995: Netscape Communications annonce que leur navigateur va supporter Java, ... Microsoft achète aussi une licence de Java (on ne sait jamais!?).**
- **Anecdote: Le nom Java proviendrait d'une large consommation de café durant le temps de développement du langage …**

Les navigateurs Java:

- **Il est possible de visualiser des applications Java dans un navigateur dans la mesure où celui-ci est compatible Java. Exemple: le navigateur HotJava (développé par Sun).**

Les navigateurs ci-dessous permettent d'exécuter du code JAVA mobile (applets).

- **Internet Explorer**
- **Netscape, …**

Un peu de terminologies …

1. **Applet**: Codes JAVA téléchargeables sur des machines clientes à partir d'un serveur WEB (APACHE, …). Ne contient pas de fonction main(), et s'exécute obligatoirement dans l'environnement d'un navigateur compatible JAVA (IE, NETSCAPE, …) intégrant une JVM.
2. **Midlet**: Codes JAVA téléchargeables à partir d'un serveur de Servlets (TOMCAT, …), sur des dispositifs mobiles (portables, PDA, …) à ressources matérielles limitées. Ne contient pas de fonction main(), et s'exécute obligatoirement dans l'environnement d'un équipement compatible JAVA intégrant une KVM ou CVM.
3. **Servlet**: Codes stockés sur un serveur et s'exécutant sur la requête d'une machine cliente afin de fournir un service de traitement ou de calcul.
4. **JVM**: Processeur virtuel, produisant un code « machine » intermédiaire (byte-code) à partir d'un fichier source java. Un interpréteur est ensuite nécessaire afin de convertir et d'exécuter les bytes-code en code assembleur de la machine cible.
5. **KVM**: Processeur virtuel « très allégé » afin de pouvoir être porté sur des machines embarquées (téléphones cellulaires, PDA, …) aux ressources matérielles limitées.
6. **CVM**: Processeur virtuel « moyennement allégé » afin de pouvoir être porté sur des machines embarquées aux ressources matérielles relativement limitées.
7. **RMI**: (Remote Methode Invocation) – API JAVA permettant le développement d'applications distribuées objet
8. **JSP, ASP**: (JavaServerpage) - Langages de script, type C (resp. sun, microsoft) permettant de créer des sites web dynamiques. Les scripts sont insérés dans des pages html à l'intérieur des balises <% et %>
9. **PHP**: Langage de script structuré, inséré dans une page html, exécuté au niveau du serveur, et permet de « dynamiser » la page html en fournissant un ensemble complet de fonctions utiles (mail, …); Remplace avantageusement le code CGI (écrit en C, PERL, …) au niveau des temps d'exécution. Dès que le code PHP est exécuté dans la page html, il ne figure plus dans le source récupéré par la machine client.
10. **Socket**: Point de connexion réseau (prise réseau) où doit se brancher d'autres machines informatiques afin de pouvoir établir des échanges entre elles.
11. **APACHE**: Application informatique configurée comme un serveur WEB, et permettant de « uploader » des pages html, wml, xml, … sur des machines clientes. Des services peuvent être exécutés pour le compte de client sur requêtes PHP, JSP, ASP, CGI, … des applets peuvent être téléchargées sur des machines clientes
12. **TOMCAT**: Application informatique configurée comme un serveur de servlet, et permettant sur requêtes de machines clientes, d'exécuter des services de traitement et/ou de calcul à partir de code java spécifiques: servlet
13. **mySQL**:

-1-

Généralités sur la technologie JAVA

Java, c'est quoi ?

☞ Java est un langage de programmation orienté objet (POO) créé par Sun. Il permet de développer des applications client/serveur pour le WEB (RMI, CORBA) et du code mobile (applet, midlet) sur workstations, téléphones cellulaires, PDA, cartes de paiement, …

☞ Un source Java est compilé pour fournir un binaire de type pcode (pseudo code machine). Les fichiers ainsi compilés peuvent être soit interprétés et exécutés localement; soit chargés via une page HTML dans un navigateur WEB compatible Java (intègre un « plugin » capable d'interpréter le pcode fourni par la page HTML et exécuter le programme java); soit téléchargés dans un dispositif mobile léger (téléphone cellulaire, PDA, …).

Java permet:

☞ De concevoir une application de type objet
☞ De mettre un peu plus de vie dans les pages WEB (html)
☞ De créer une vraie application graphique et interactive au sein d'une page html
☞ De valider le contenu d'un formulaire sur votre machine avant de l'envoyer au serveur

Où trouver outils et informations sur java ?

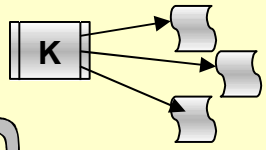
Les principales informations concernant JAVA peuvent être trouvées sur le site de Sun. Ce site permet de charger le JDK (Java Development Kit, version J2SE 1.5.0), et la documentation Java.

http://java.sun.com ou ftp://java.sun.com/pub/
http://www.javasoft.com/tutorial/index.html

Caractéristiques du langage Java (suite)

JAVA permet le multi-thread

- Une application Java peut lancer plusieurs tâches ou processus indépendants qui s'exécutent simultanément. Java intègre un exécutif multitâche permettant d'ordonnancer les "threads".



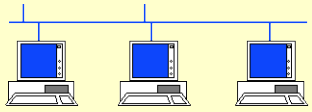
JAVA est sécurisé

- Java a intégré, dès sa conception, plusieurs mécanismes de sécurité visant à rendre les programmes fiables et à éliminer les risques de virus (vérification du "bytes codes", pas de manipulation de pointeurs, …). Les écritures sur disque ne sont pas autorisées dans les applets. Le code source n'est pas visible, seul le pcode l'est (*.class). Chaque groupe de classes (local, machine distante) est stocké dans un espace mémoire distinct par le chargeur de classes.

JAVA est simple

- Plus simple que le C++, Java n'a pas de pointeurs, source de nombreuses erreurs (mais des références). Il n'intègre ni l'héritage multiple, ni la surcharge d'opérateurs.

JAVA implémente le modèle client-serveur.



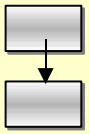
- Conçu pour les architectures distribuées; intègre la gestion des sockets TCP/IP et URL

Caractéristiques du langage Java (suite)

JAVA est orienté-objet

Java permet:

- **L'encapsulation (masquage d'information)**
- **L'héritage (déclarer une nouvelle classe comme extension d'une classe existante),**



- **La liaison dynamique (les appels à une opération ou à n'importe laquelle de ses redéfinitions dans les classes dérivées sont "résolus" au moment de l'exécution, en fonction du type de l'objet concerné).**
- **Une gestion automatique de la mémoire (les objets inutilisés sont désalloués par un système sousjacent, le garbage collector, sans intervention du programmeur)**

- **Une bibliothèque de classes standard très complète.**

JAVA est indépendant de la machine

- **Point fort de Java: un source Java compilé donne des bytes-code (opcodes 8 bits et opérandes de tailles variables); codes d'un pseudo-assembleur (pcode) d'une *machine virtuelle Java* (JVM)**

```
lload address;    22xxxx
ladd;             97
lstore address;   55xxxx
jsr address;      168 xx
```

Quelques instructions «assembleur» JVM

- Espace d'adressage 32 bits; méthode 32KB max; 256 variables par pile.
- L'exécution des bytes-codes nécessite un interpréteur Java simulant une machine virtuelle.
- Les logiciels de navigation compatibles Java intègrent un tel interpréteur.
- Applications Java, applets, midlets conçues en bytes-codes sont indépendantes de la machine physique.

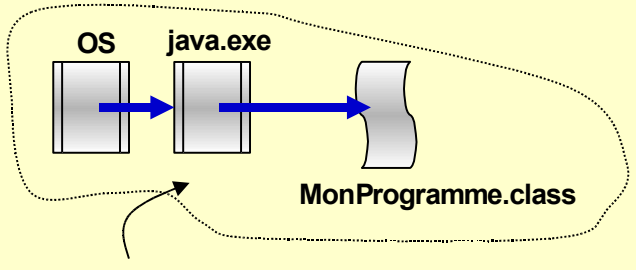
Caractéristiques du langage Java (suite)

JAVA est un langage de classe

- Un programme s'écrit en faisant référence directement à la classe principale:

```
class MonProgramme
{
    public void init() {
      resize (500,100);
      date = getParameter ("LAST_UPDATED");
      email = getParameter ("EMAIL"); }
      public void paint (Graphics g) {
          g.drawString(date + " by ",100, 15);
      }
}
```

le code source



- La compilation ne donne pas un fichier exécutable: MonProgramme.exe, mais un fichier intermédiaire MonProgramme.class constitué de pcodes
- Le pcode doit être transcris et exécuté en codes de la machine informatique utilisée, par la commande : Java MonProgramme.class ↲

JAVA gère les erreurs et exceptions

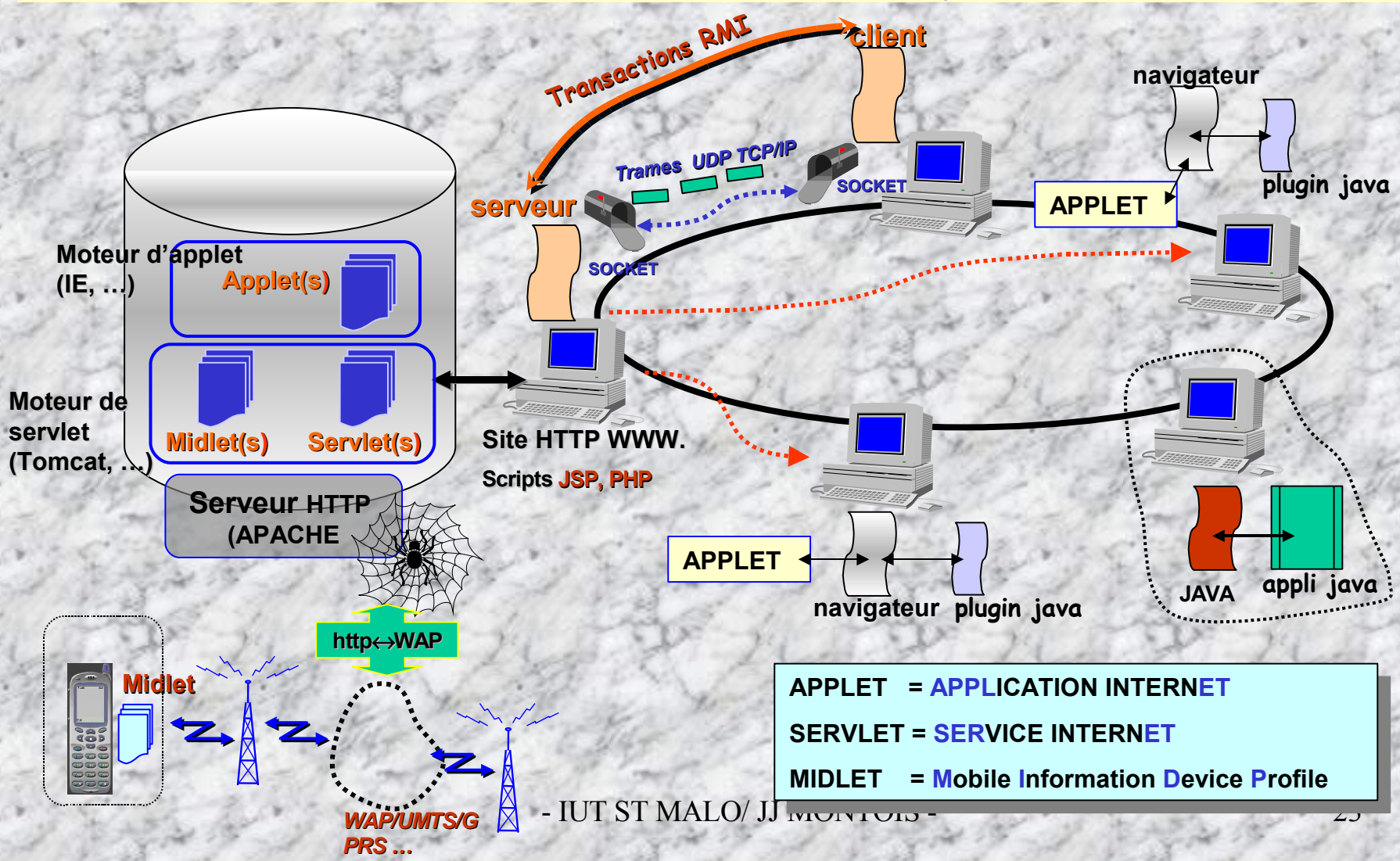
- Des segments de codes peuvent être surveillés pour lever les erreurs d'E/S ou de traitement.

```
...
try
{
    public void init() {
      resize (500,100);
      date = getParameter ("LAST_UPDATED");
      email = getParameter ("EMAIL"); }
      public void paint (Graphics g) {
          g.drawString(date + " by ",100, 15);
      }
}
catch (exception  <objet erreur>) {   //code de traitement de l'erreur }
```

Transmission d'un objet erreur de type exception

JAVA & LES APPLICATIONS DISTRIBUEES …

1. Programmation réseau « bas niveau » - SOCKET,STREAM TCP/IP UNIX/WINDOWS
1. Informatique communicante: Architecture Client-Serveur, RMI, …
1. Interface IHM évènementielle & Applet JAVA
1. Programmation de dispositifs légers mobiles - Développement d'applications CLDC MIDP: Servlet -Midlet sous JAVA ME et machine KVM,



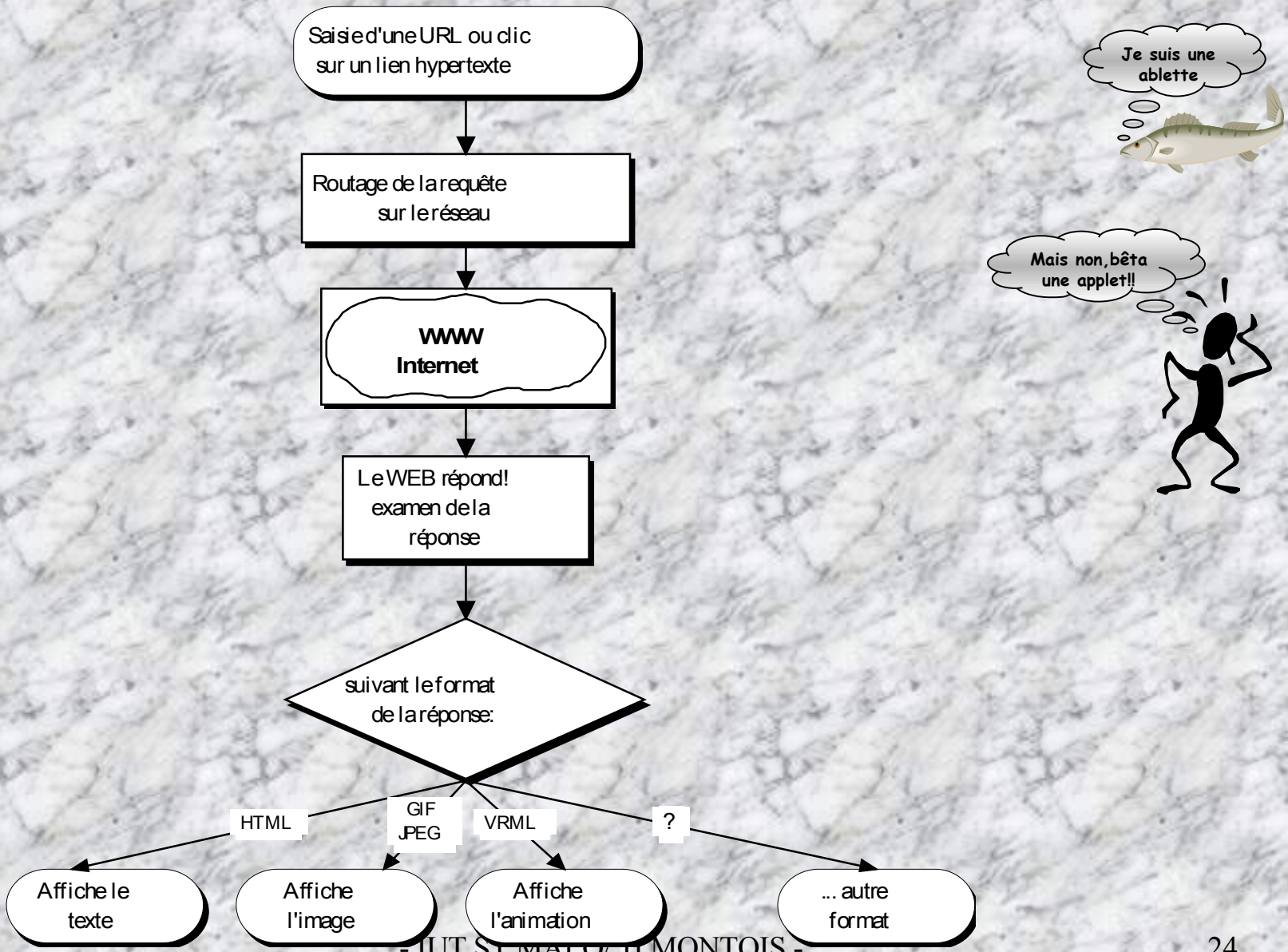
APPLET = APPLICATION INTERNET

SERVLET = SERVICE INTERNET

MIDLET = Mobile Information Device Profile

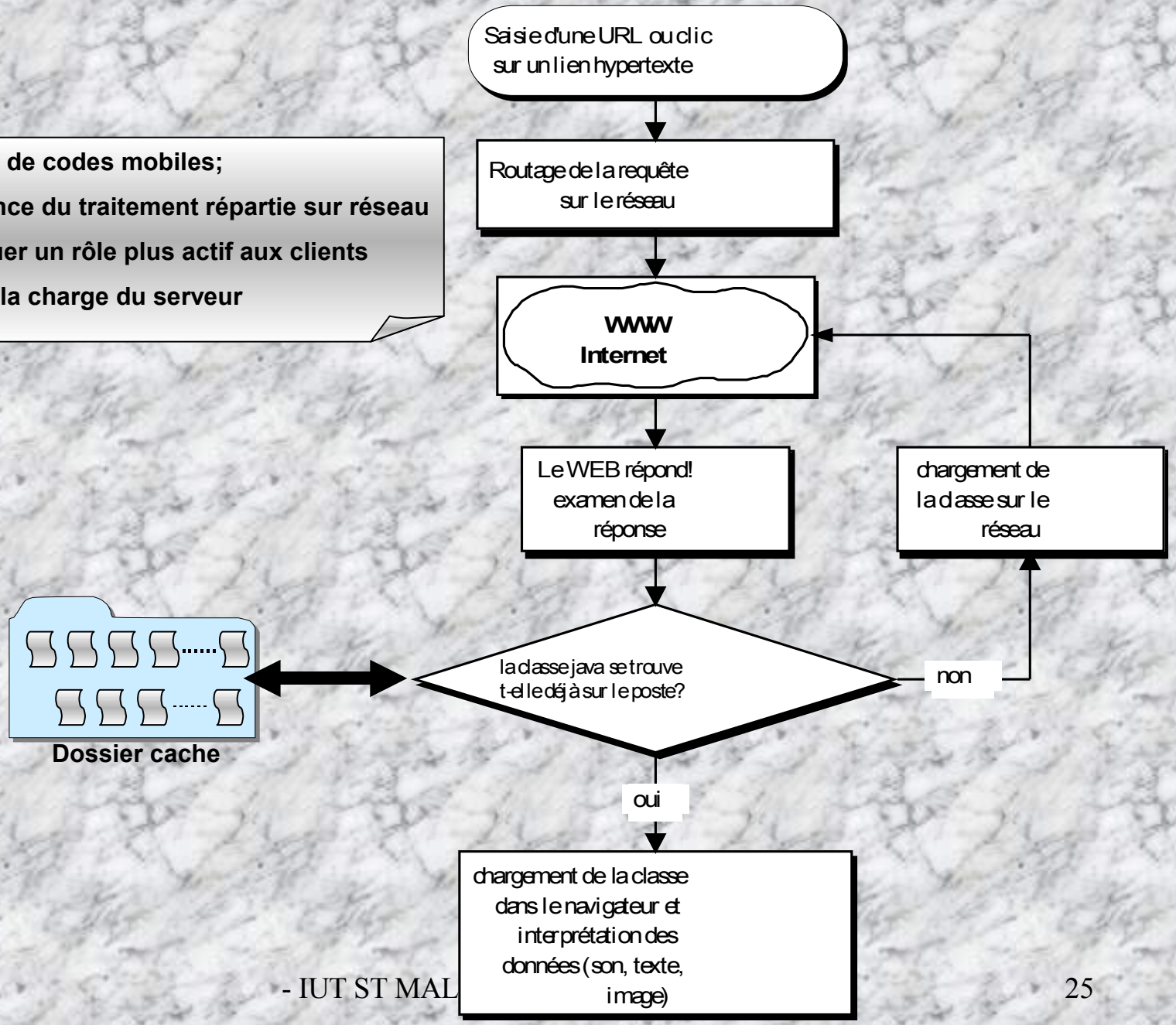
MAIS QU'EST DONC UNE APPLET? Schéma de chargement d'une applet …

1 - Requête classique à partir d'un lien hypertexte: aucune demande d'applet java:

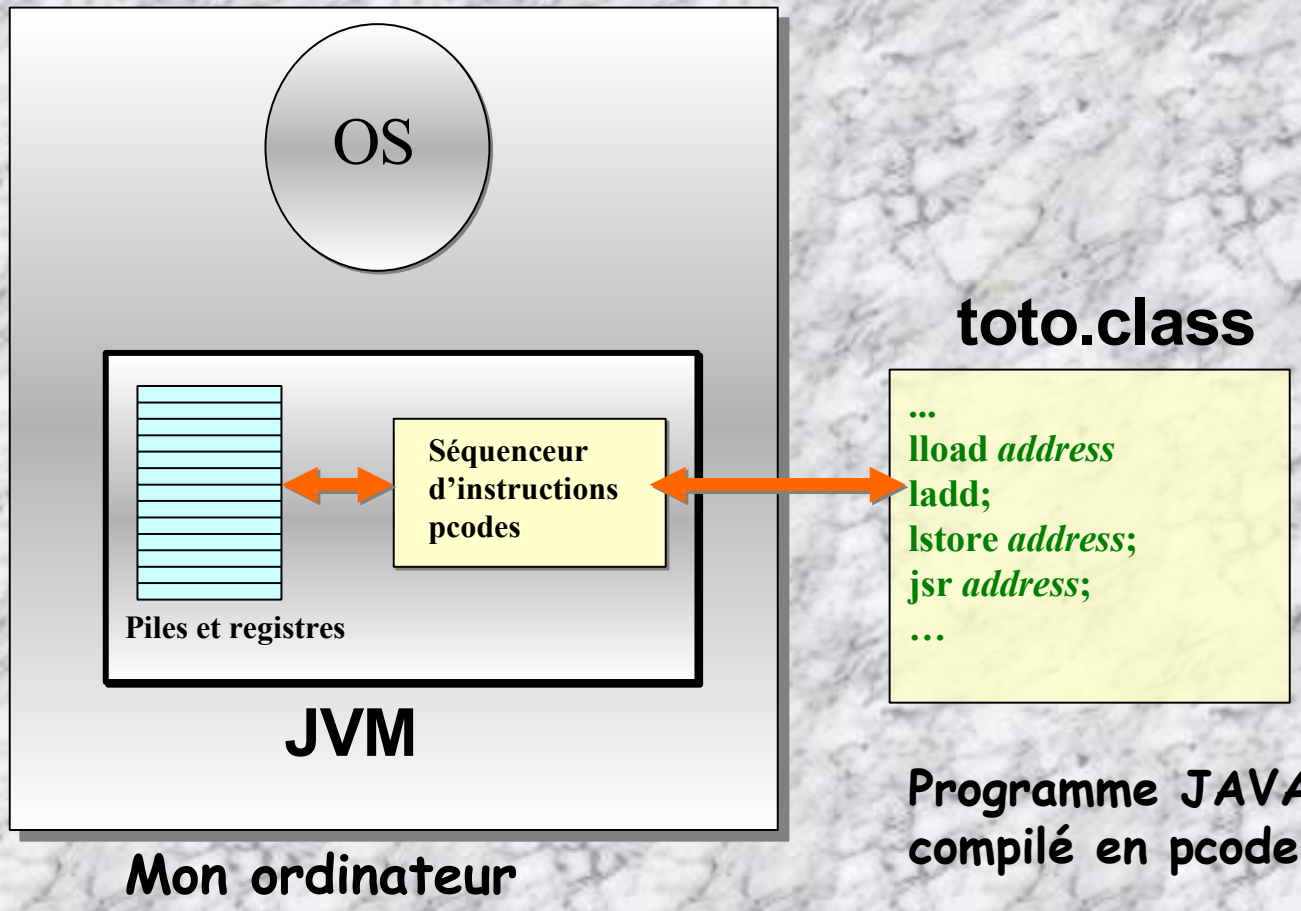


2 - Requête à partir d'un lien hypertexte: l'appel nécessite une applet java:

- ✓ **Concept de codes mobiles;**
- ✓ **Emergence du traitement répartie sur réseau**
- ✓ **Faire jouer un rôle plus actif aux clients**
- ✓ **Réduire la charge du serveur**



Le processeur virtuel de la machine JAVA – La JVM



CARACTERISTIQUES D'UNE JVM

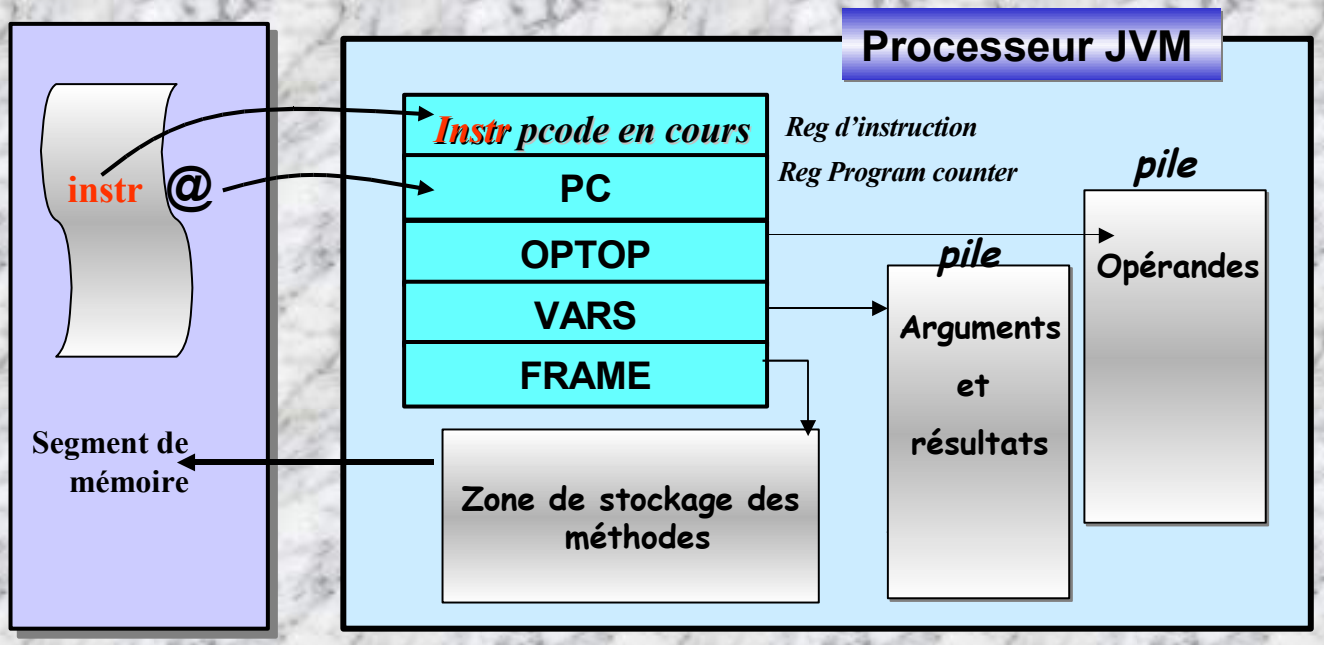
1. Jeu de registres internes de 32 bits pour stocker les données traitées, et à traiter selon l'instruction pcode présente dans le registre d'instruction.

   - **S** : registre pointeur (Stack) de pile vers la 1ière variable locale de la méthode en cours d'exécution.

   - **PC**: « Program Counter », registre indiquant l'@ de l'instruction pcode en cours d'exécution

   - **Optop, Frame** : registres pointeur: sommet de la pile des opérandes, et zone de stockage des méthodes

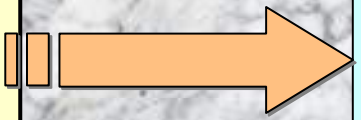
5. Jeu d'instructions « machine »: le **PCODE**
6. Une pile LIFO pour transmettre paramètres, espace des variables locales, et adresses de retour aux méthodes
7. Une pile FIFO d'opérandes pour stocker les arguments/résultats des instructions pcodes
8. Un segment de mémoire dans lequel s'effectue l'allocation/désalocation d'objets
9. Une zone de stockage des segments de pcodes des méthodes ainsi que la table des symboles.



* On trouve sur le site de SUN, des précisions sur la JVM, et son jeu d'instructions « machine »

Le programme DocFooter.java

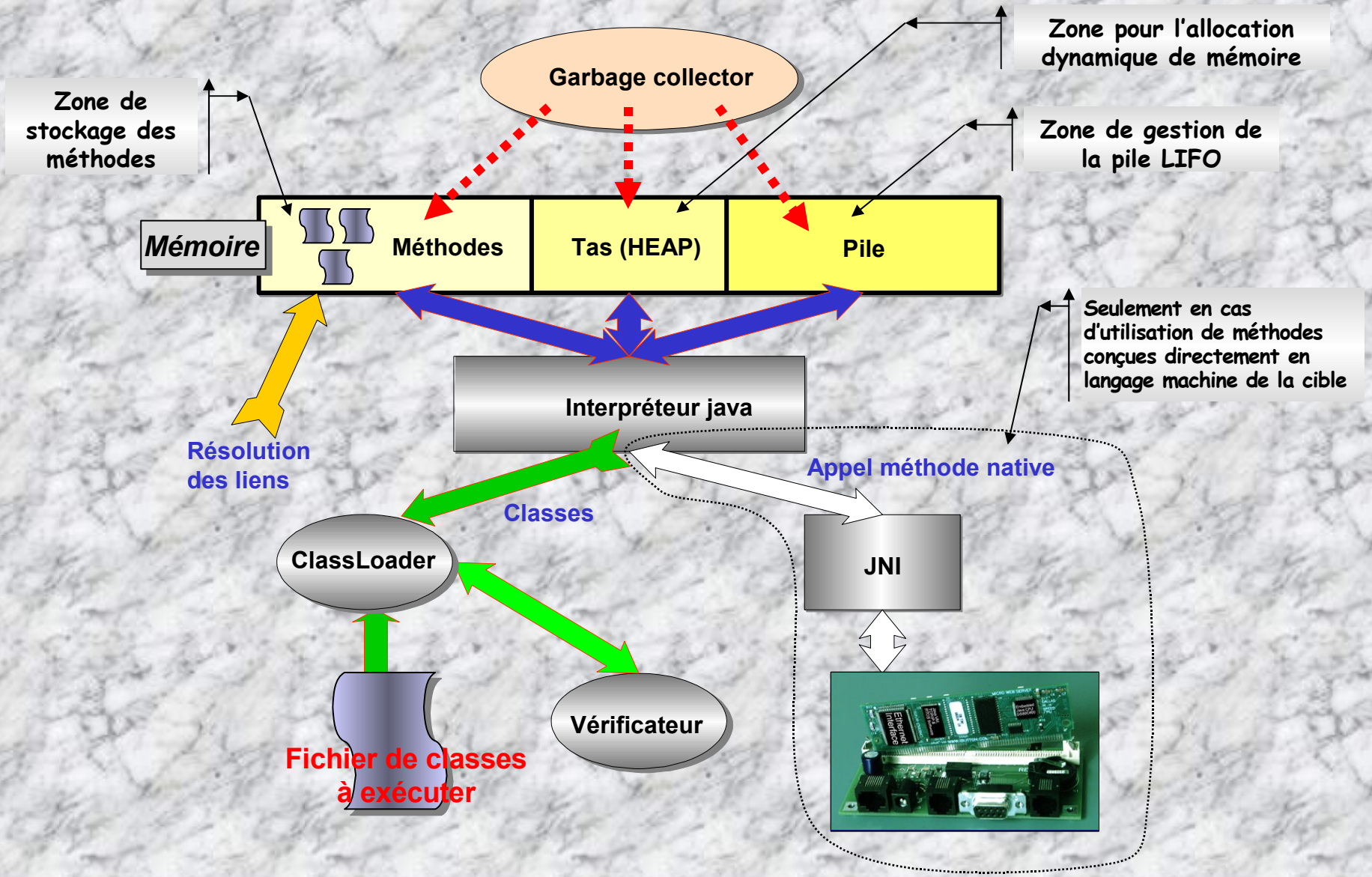
```
public class DocFooter extends Applet {
String date;
String email;
public void init() {
  resize (500,100);
  date = getParameter ("LAST_UPDATED");
  email = getParameter ("EMAIL"); }
  public void paint (Graphics g) {
      g.drawString(date + " by ",100, 15);
      g.drawString(email, 290,15);
    }
}
```



```
Method DocFooter()
  0 aload_0
  1 invokespecial #1
  4 return
Method void init()
  0 aload_0
  1 sipush 500
  4 bipush 100
  6 invokevirtual #2
  9 aload_0
 10 aload_0
 11 ldc #3
 13 invokevirtual #4
 16 putfield #5
 19 aload_0
 20 aload_0
 21 ldc #6
 23 invokevirtual #4
 26 putfield #7
 29 return
Method void paint (java.awt.Graphics)
  0 aload_1
  1 new #8
  4 dup
  5 invokespecial #9
  8 aload_0
  9 getfield #5
 12 invokevirtual #10
 15 ldc #11
 17 invokevirtual #10
 20 invokevirtual #12
 23 bipush 100
 25 bipush 15
 27 invokevirtual #13
 30 aload_1
 31 aload_0
 32 getfield #7
 35 sipush 290
 38 bipush 15
 40 invokevirtual #13
 43 return
}
```

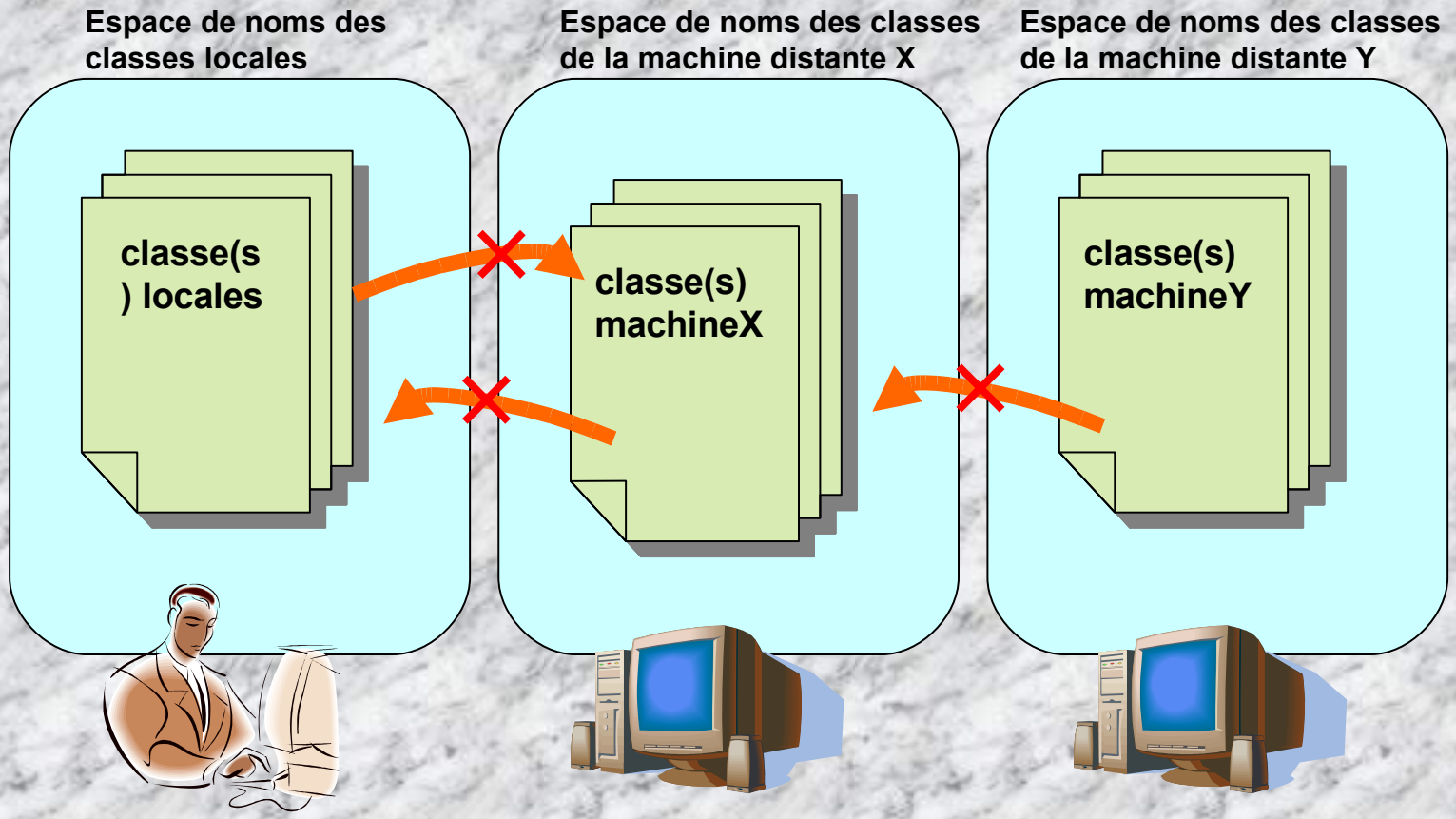
La compilation du programme DocFooter.class avec la commande javap -c DocFooter, donne le source assembleur JVM suivant:

Architecture de la machine virtuelle JAVA



Deux dispositifs permettant la sécurité

1. Le chargeur de classe: le ClassLoader

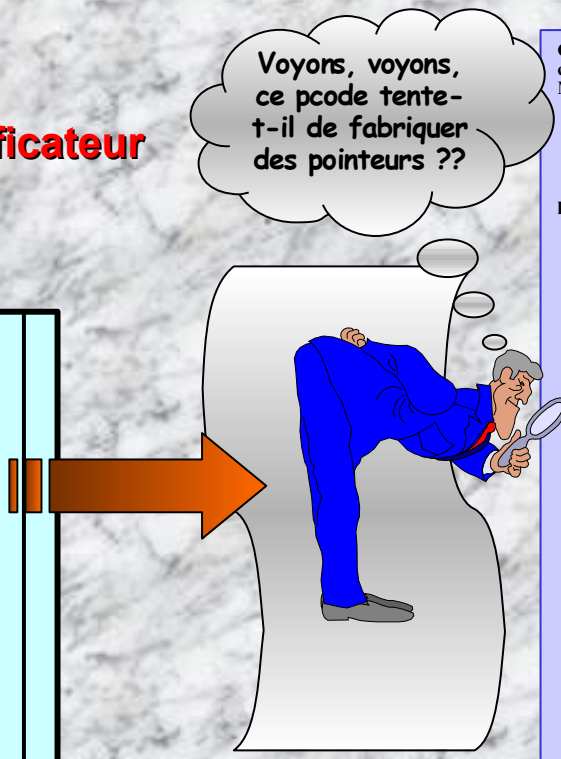


Lors de l'appel d'une classe donnée, la recherche s'effectue en priorité dans l'espace des classes locales. Ainsi, aucune classe téléchargée ne peut se faire passer pour une classe système, en écrasant l'original, même si elle modifie sont nom,

2. Le vérificateur de code: le vérificateur

MaClasse.java

```
class MaClasse {
public static void main (String args[]) {
 int A = 10; int B = 12; int X;
 System.out.println("L'entier A vaut "+ A);
 System.out.println("L'entier B vaut "+ B);
 if (A < B) System.out.println("A est plus petit que B");
  else if (A == B) System.out.println ("A est égal a B");
  else System.out.println ("A est plus grand que B");
 System.out.println("comptons de 1 à "+ A);
 int somme = 0;
 int fact = 1;
 for (int i = 1;i <= A;i++) {
    System.out.print(" "+i);
    somme += i;
    fact *= i;
    }
 System.out.println();
 System.out.println("la somme de tous les nombres de 1 à
"+ A + " vaut "+ somme);
 System.out.println("la factorielle de "+ A+" vaut "+ fact);
  }
}
```

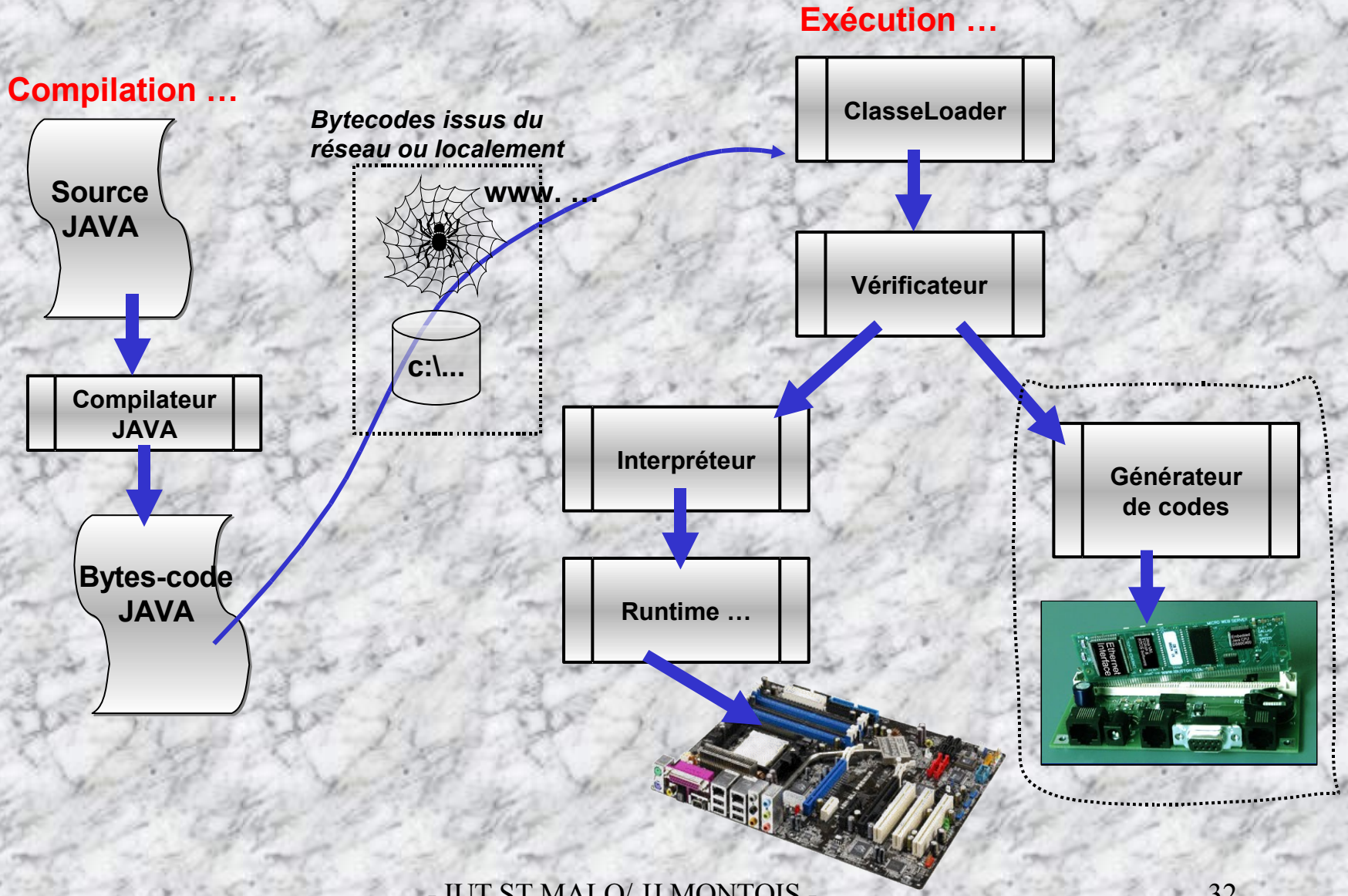


pcode de MaClasse.java

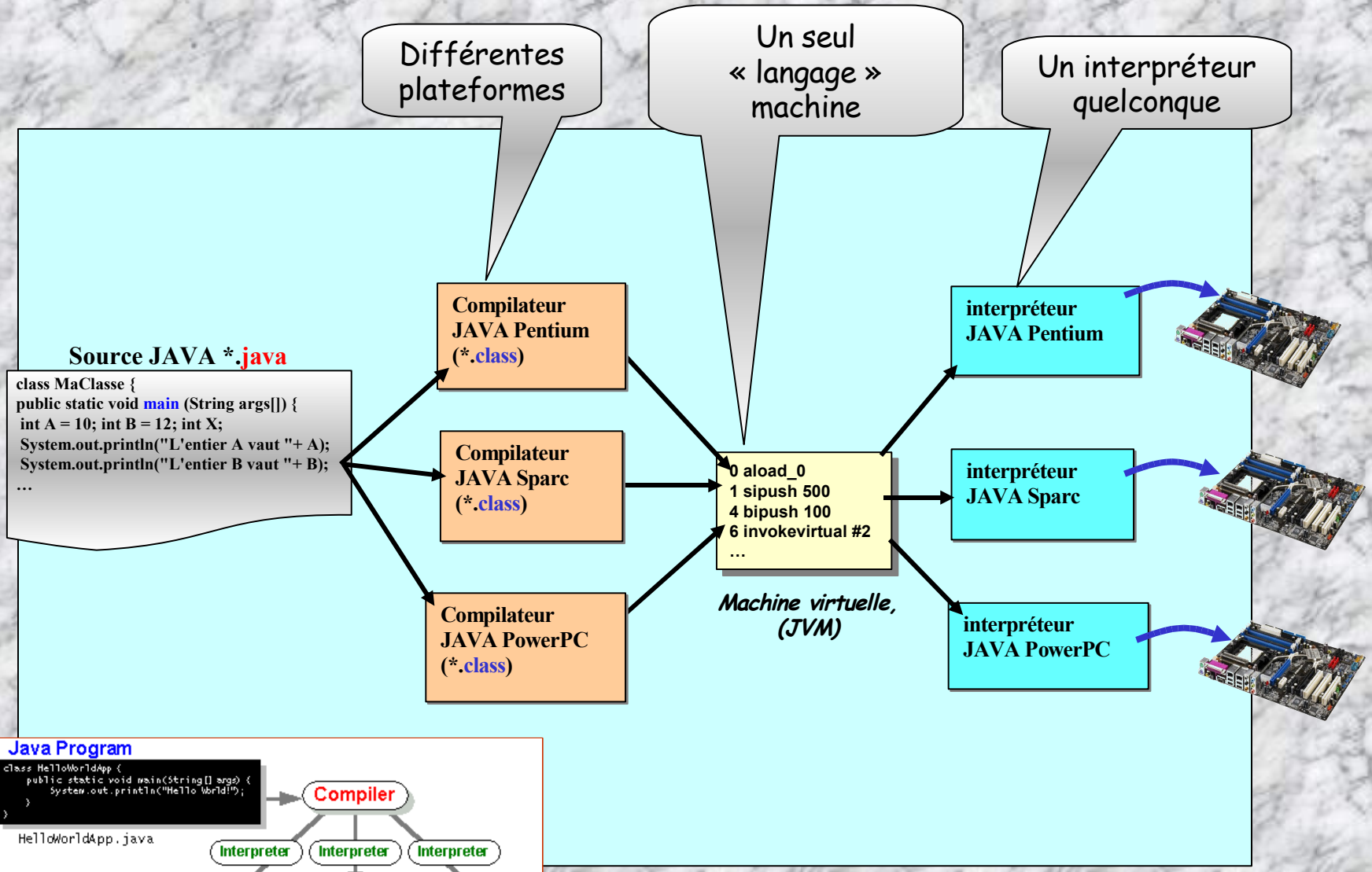
```
Compiled from "MaClasse.java"
class MaClasse extends java.lang.Object{
MaClasse();
 Code:
  0:  aload_0
  1:  invokespecial        #1;
  4:  return

public static void main(java.lang.String[]);
 Code:
  0:  bipush               10
  2:  istore_1
  3:  bipush               12
  5:  istore_2
  6:  getstatic            #2;
  9:  new                  #3;
  12: dup
  13: invokespecial        #4;
  16: ldc                  #5;
  18: invokevirtual        #6;
  21: iload_1
  22: invokevirtual        #7;
  25: invokevirtual        #8;
  28: invokevirtual        #9;
  31: getstatic            #2;
  34: new                  #3;
  37: dup
  38: invokespecial        #4;
  41: ldc                  #10;
  43: invokevirtual        #6;
  46: iload_2
  47: invokevirtual        #7;
  50: invokevirtual        #8;
  53: invokevirtual        #9;
  56: iload_1
  57: iload_2
  58: if_icmpge            72
  61: getstatic            #2;
  64: ldc                  #11;
  66: invokevirtual        #9;
  69: goto                 96
  72: iload_1
  73:                      iload_2
  74: if_icmpne            88
  77: getstatic            #2;
  80: ldc                  #12;
      ...
      ...
```

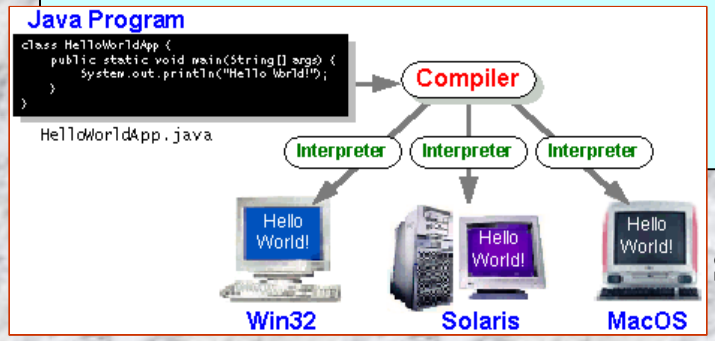
- ✓ **Le pcode ne provoque pas de dépassement de la pile d'opérande**
- ✓ **Les types d'arguments passés lors des appels de méthodes sont corrects**
- ✓ **Le pcode ne tente pas de conversion illégale de données (int, float, …) en pointeurs**
- ✓ **Le pcode ne tente pas de passer outre les restrictions d'accès (public, private, …)**

Séquencement des opérations de compilation, chargement, exécution d'une application JAVA



Conception multiplateforme d'un programme compilé en pcodes



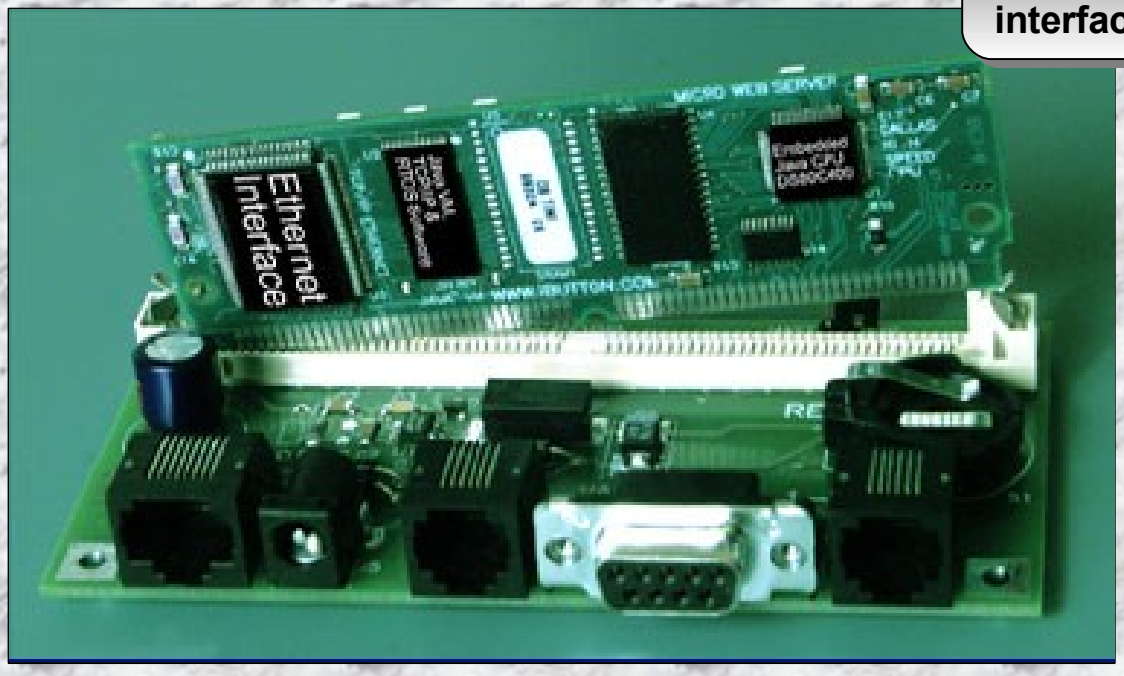


Processeur JAVA JVM « siliciumisé »

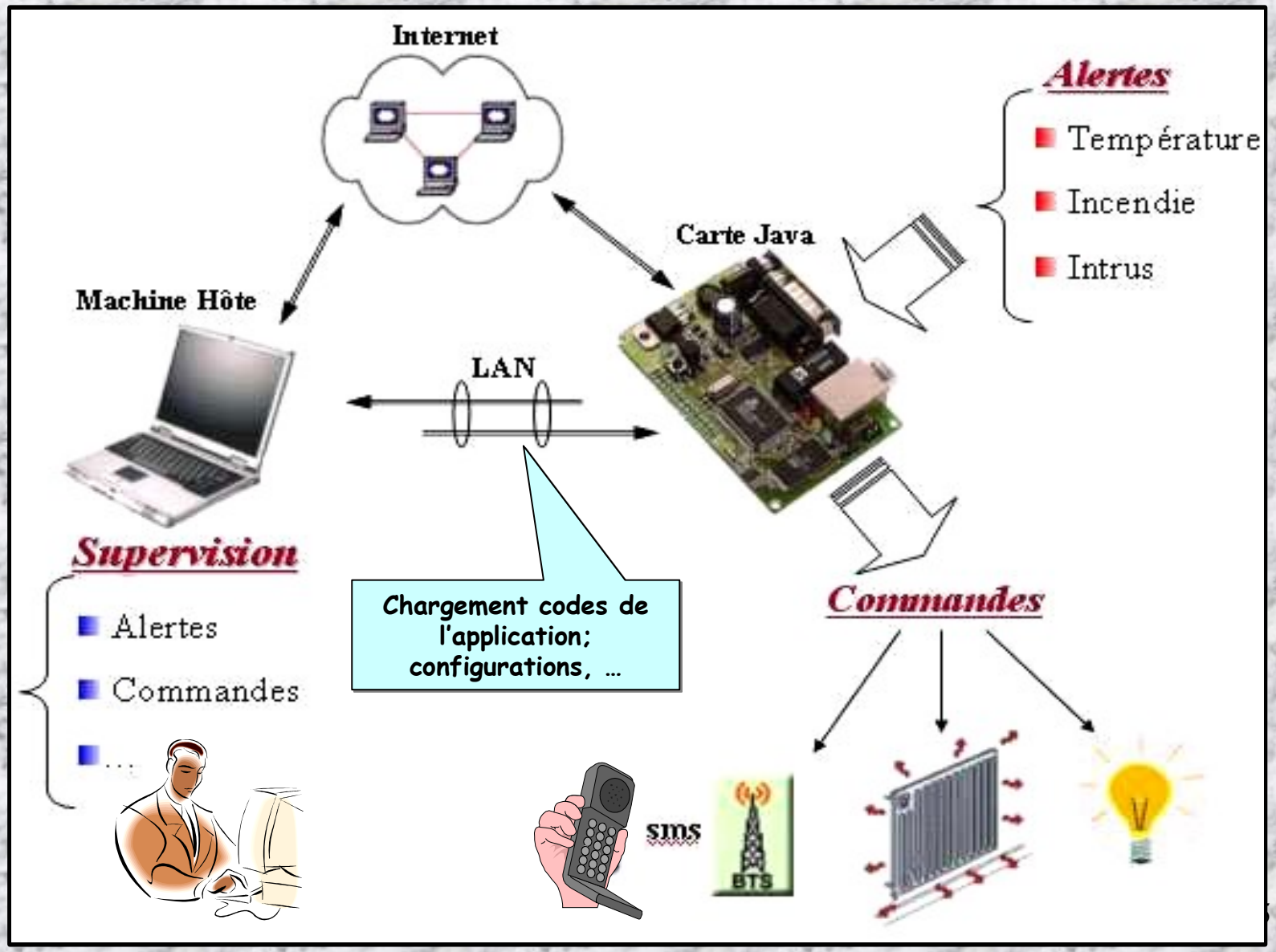
On peut trouver aussi un processeur JVM « en dur », ou « siliciumisé »

- **Architecture de base : Picojava (SUN 1996)**
- **But : faire exécuter du bytecode JAVA sans interpréteur**
- **Usage: appications embarquées**
- **OS TRe intégré**

Tini de Dallas semiconductor 's Support 68 pins avec processeur interface ethernet, RS232, I2C, CAN



EXEMPLE: Contrôle-Commande domotique d'un ou plusieurs équipements d'une maison. Pouvoir piloter à distance l'éclairage ou le chauffage de la maison, et être averti en toute heure d'un éventuel problème comme le déclenchement d'une alarme, risque d'incendie ou autres…

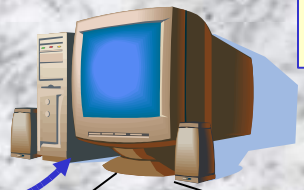


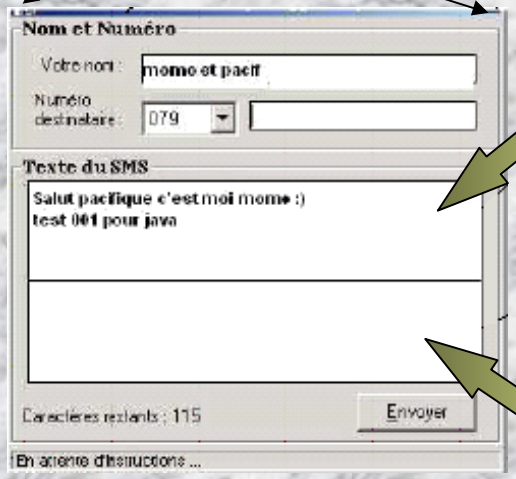
Système de commande à base de la carte JAVA IPJV-ES permettant d'établir la communication entre les différents capteur de température et de luminosité et un téléphone mobile via un modem WAVECOM.

Envoi/réception de sms

MODEM GSM WAVECOM

Carte JAVA IPJV-ES





Envoi de message sms

Message sms reçu

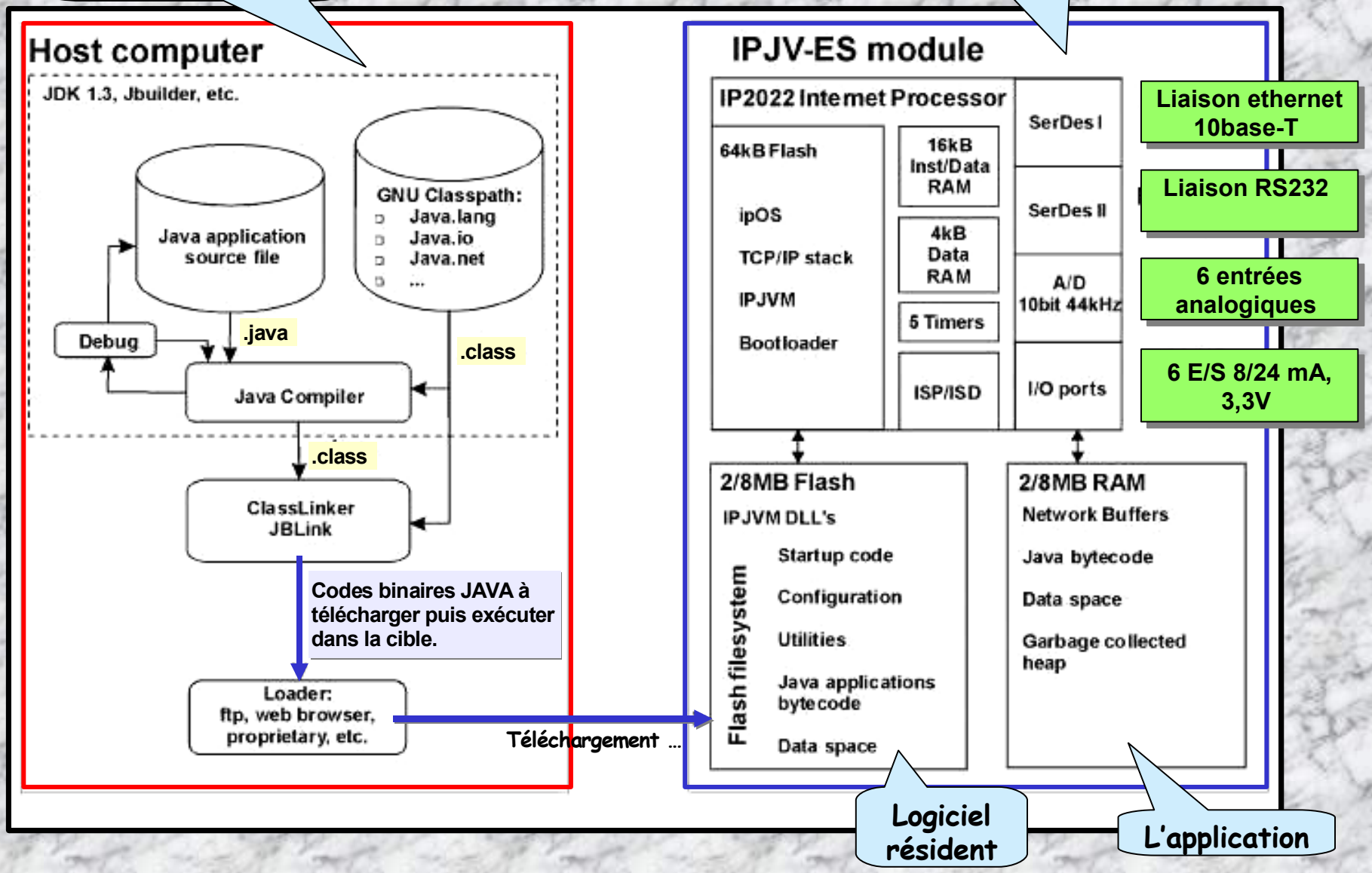
Processus commandés

Les systèmes de contrôle et de supervision utilisent de plus en plus le GSM et plus spécialement la fonctionnalité des messages courts (SMS).

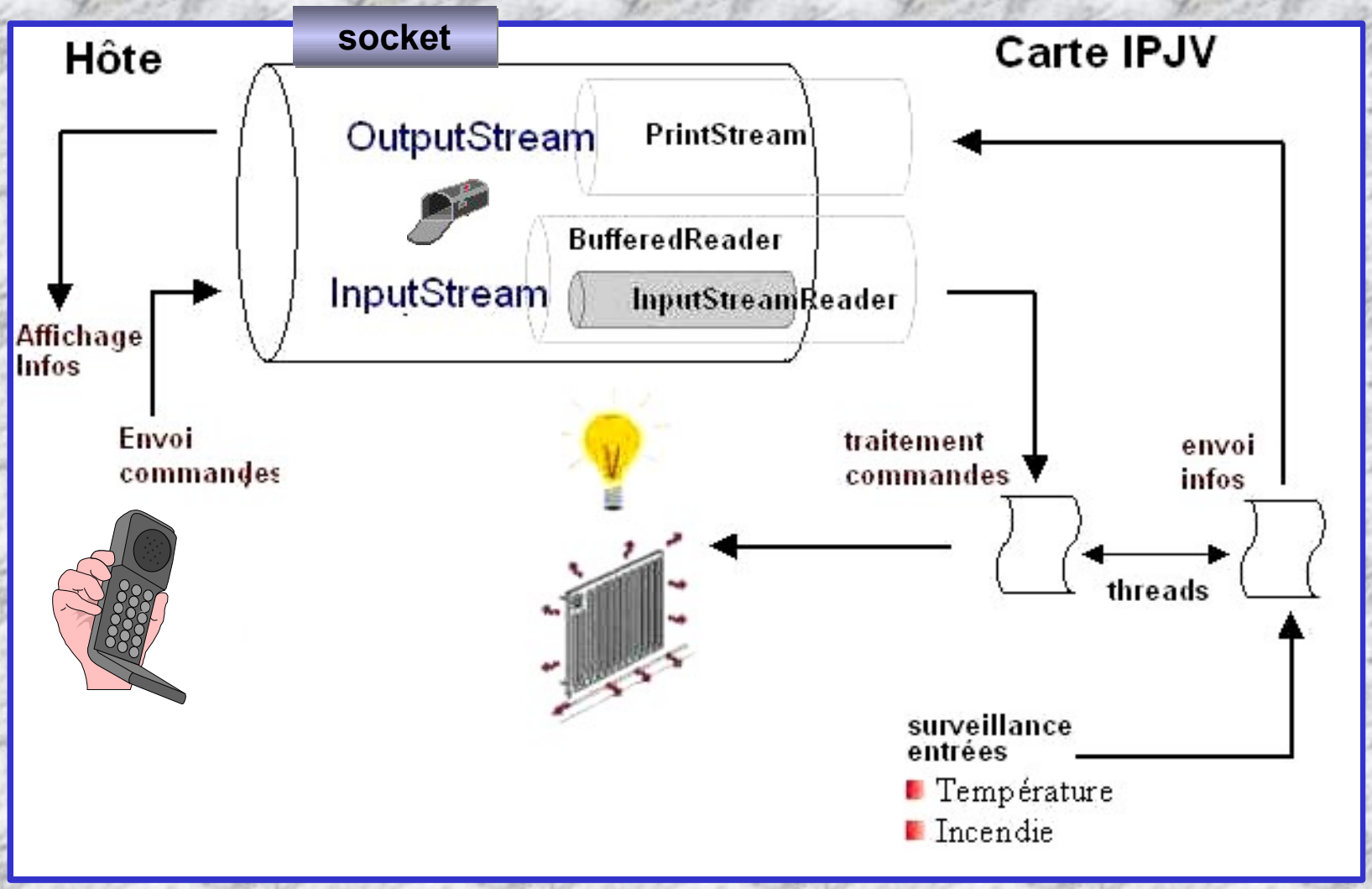
La carte JAVA exécute un programme qui prélève les informations via la liaison RS232.

Plateforme de développement logiciel

La carte JAVA



Architecture logicielle de l'application domotique …



Diverses plateformes spécialisées de développement JAVA

Plateforme JAVA ⇒ 3 sous-ensembles spécialisés:

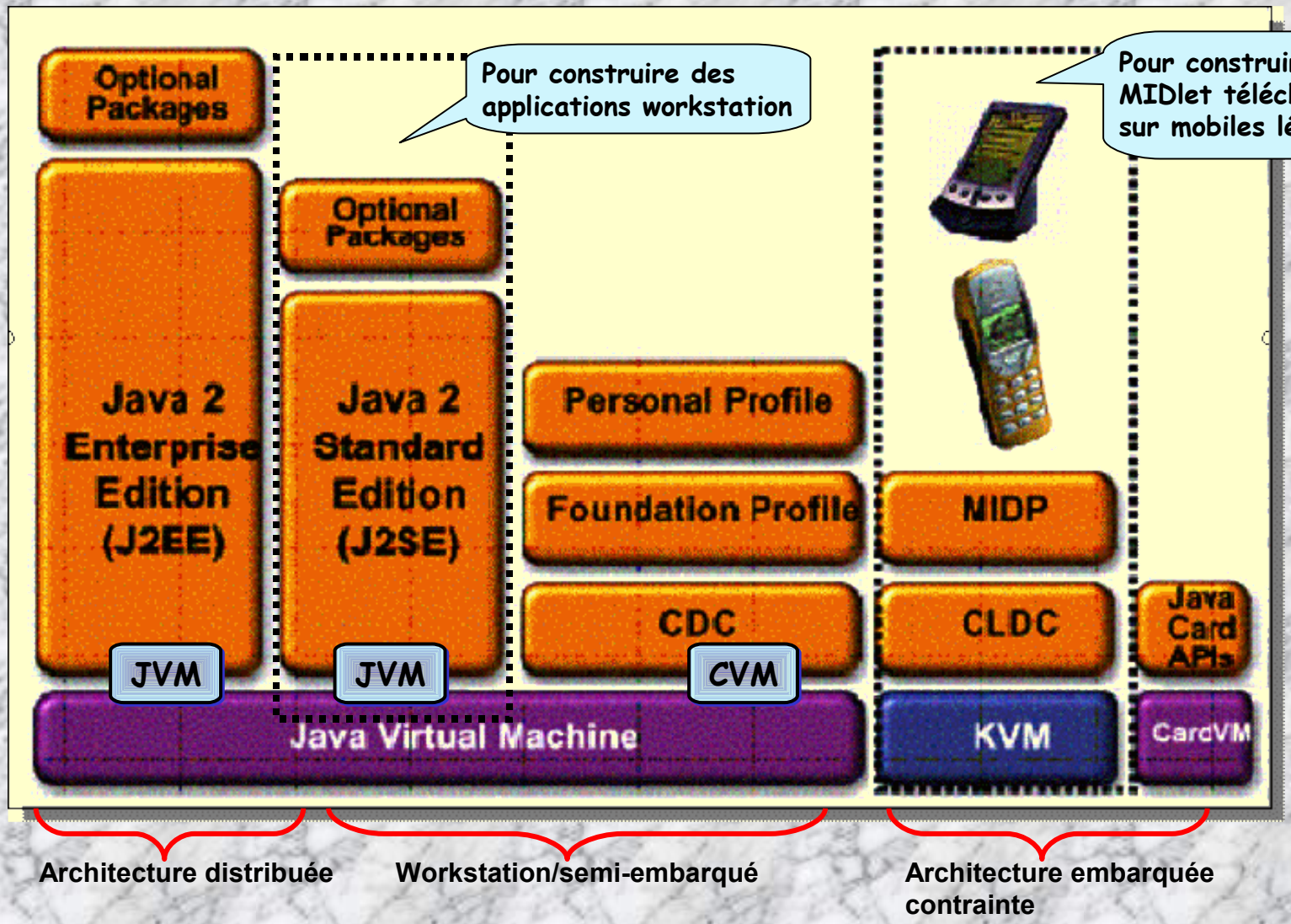
3. **J2EE** (JAVA 2 Enterprise Edition) – Développement et déploiement d'applications à destination des entreprises (applet, SERVlet) (Architectures B2C et B2B);

4. **J2SE** (JAVA 2 Standard Edition) – Développement et exécution de logiciels au niveau de l'utilisateur (APPlet).

5. **J2ME** (JAVA 2 Micro Edition) – Production d'applications Java pour les systèmes embarqués (MIDlet). Architecture définie avec des configurations (ressources minimales nécessaires pour accueillir la plateforme JAVA et des profils permettant de compléter la configuration avec des fonctionnalités spécifiques au matériel)

Java 1.2 c'est …
60 Packages
1 781 Classes et Interfaces
15 060 Méthodes!!

PLATEFORMES JAVA™ ME & MACHINES VIRTUELLES: JVM, KVM, …



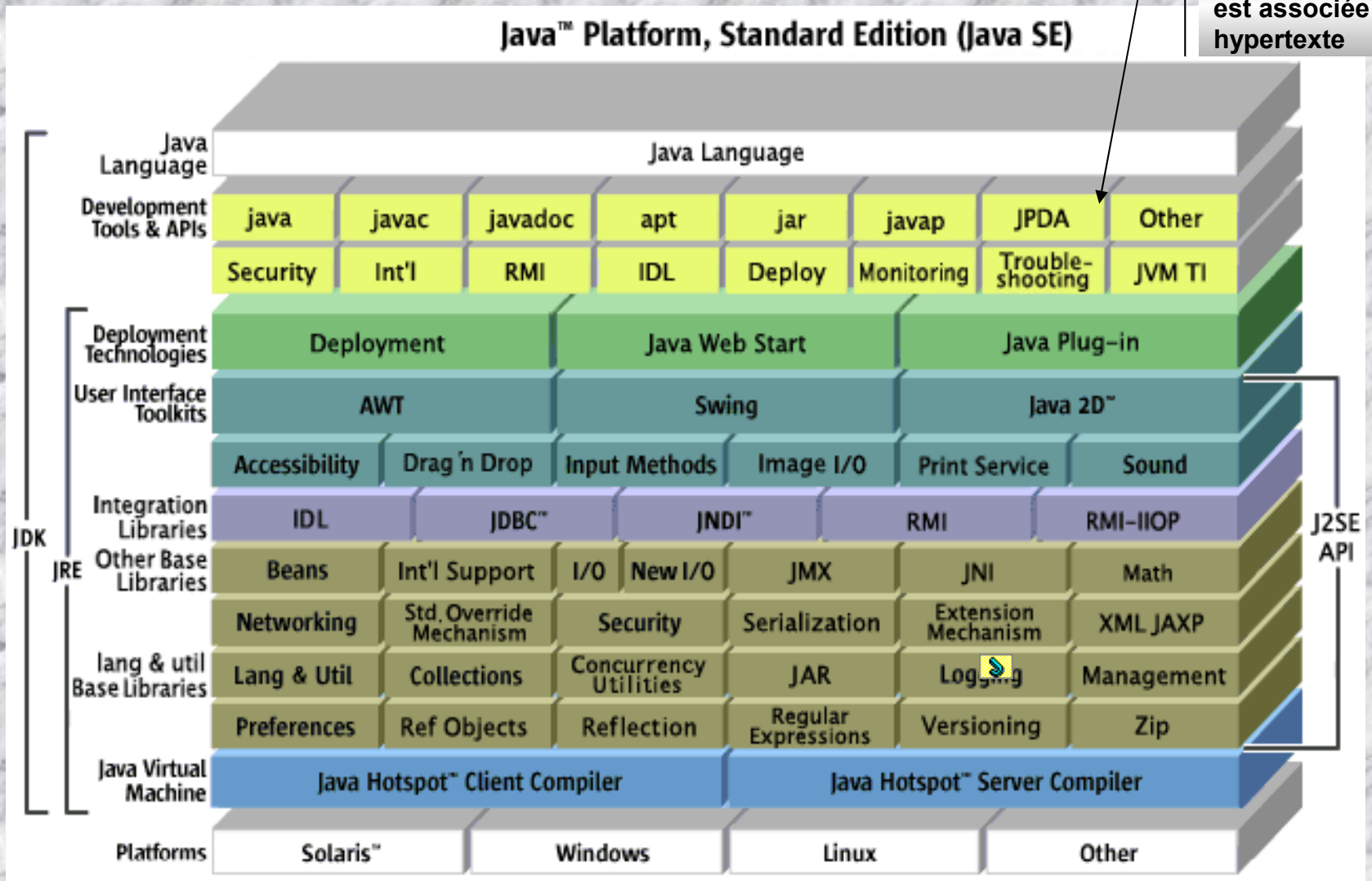
Plusieurs processeurs virtuels selon le type d'application: JVM, KVM, CVM, CardVM, …

Rien que la plateforme JAVA SE …

Pour avoir accès à toutes les infos du JAVA SE, ne pas hésiter à consulter le site:

http://java.sun.com/javase/index.jsp

Chacune de ces zones est associée à un lien hypertexte



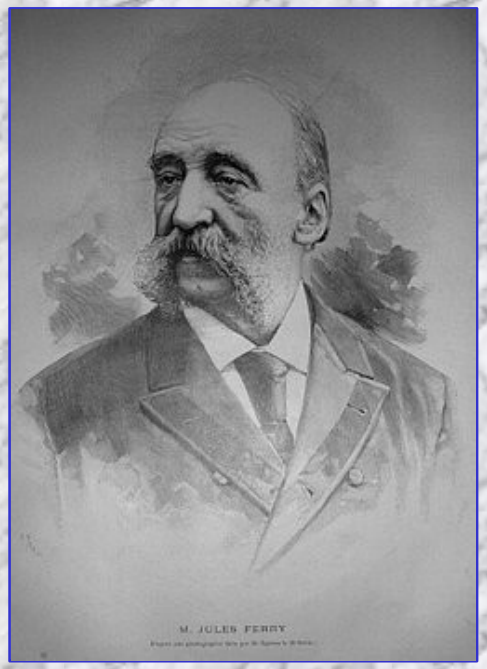
Différentes versions de JAVA Standard Edition …

- Depuis le J2SE1.4, l'évolution de Java est dirigée par le JCP (Java Community Process) qui utilise les JSR (Java spécifications Requests) pour proposer des ajouts et des changements sur la plateforme Java. http://www.jcp.org
- Le langage est spécifié par le JLS (Java Spécification Language).
- Les modifications du JSL sont gérées sous le code JSR 901

Versions JAVA	Nom de code	Année de sortie
Versions 1.0, 1.1, 1.2, 1.3		1991 à 2000
JCP → 1.4	Merlin	2001
1.4.1	Hopper	2002
1.4.2	Mantis	2003
1.5	Tiger	2004
1.5.1	Dragonfly	2006?
6	Mustang	fin 2006?
7	Dolphin	2008?

Versions JAVA	Taille compressée	Nombre de packages	Nombre de classes
Java 1.0		8	211
Java 1.1	8,6 Mo	23	477
Java 1.2	20 Mo	59	1524
J2SE 1.3	30 Mo	76	1840
J2SE 1.4	47 Mo	135	2990
J2SE 1.5	44 Mo	166	3270
Java SE 6	53 Mo		
Java SE 7	En construction		

Pause culturelle …

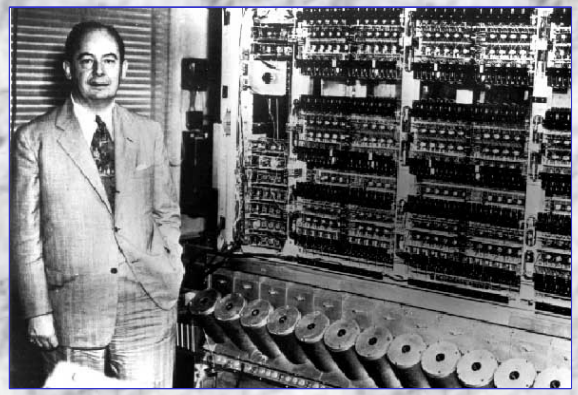


Jules Ferry

Ministre de l'Instruction publique du 4 février 1879 au 23 septembre 1880 , il attache son nom aux lois scolaires :

- Collation des grades universitaires retirée à l'enseignement privé (12/03/1880)
- Dispersion des congrégations religieuses non autorisées (29/03/1880)
- Gratuité de l'enseignement primaire (16/06/1881)
- Extension aux jeunes filles du bénéfice de l'enseignement secondaire d'État (21/12/1881)
- Loi relative à l'obligation et à la laïcité de l'enseignement (28/03/1882)
- Création d'une École Normale féminine à Sèvres et d'une agrégation féminine (13/07/1882)

[d'après WIKIPEDIA]



John Von Neumann

John à côté du 1er dispositif automatisé de traitement des données que l'on nommait pas encore ordinateur. John Von Neumann est bien trop peu connu, pourtant il est à l'origine de l'architecture encore utilisée aujourd'hui par tous les processeurs !

Chapitre consacré aux outils de développement de classes JAVA. A passer dans un 1er temps, puis à étudier lors des exercices et TP à venir

-2-

Outillages d'édition, de compilation, d'exécution du code JAVA

L'environnement « basique » de compilation selon la plateforme J2SE

- Cet environnement de développement s'appelle aussi JDK (Java Development Kit).
- Seuls les environnements Windows 32 bits (Windows NT, Windows 95) et SUN sont supportés.
- De façon identique aux environnements SUN et Windows 32 bits, le compilateur JAVA se nomme javac et son appel se fait par la syntaxe suivante :

javac -<options> *sourcefichier*.java

- ?	Permet d'obtenir la liste des options du compilateur
-classpath *chemin*	Spécifie le chemin utilisé par javac pour localiser les classes.
-d *repertoire*	Specifie le répertoire racine où stocker les classes générées.
-encoding *nom*	Specifie le nom du fichier source.
-g	Autorise la génération d'information pour le debugging.
-nowarn	Inhibe les warnings.
-O	Optimiser le code compilé par "inlining" les méthodes static, final, et private.
-verbose	Générer des détails lors de la compilation

Hiérarchie des répertoires du JDK

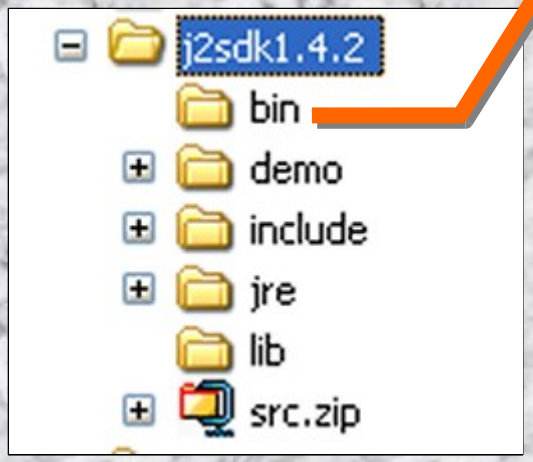
c:\j2sdk1.4.2

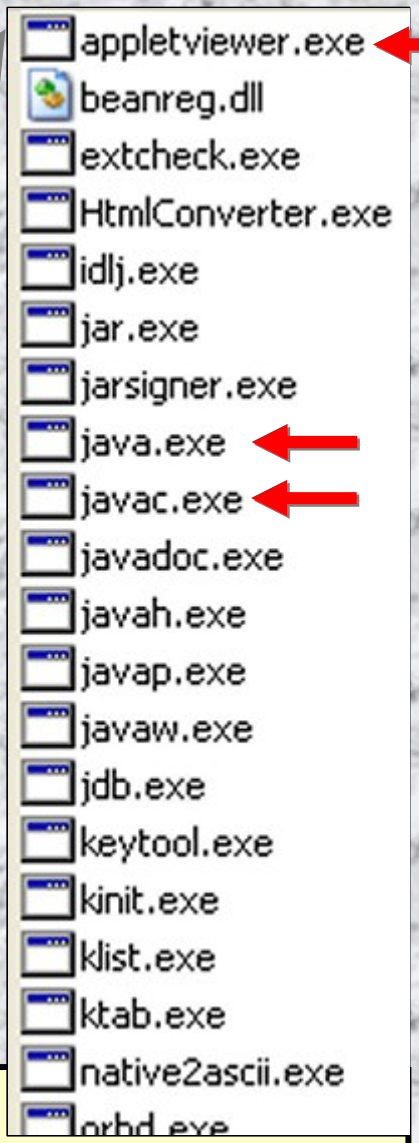
c:\ j2sdk1.4.2 \bin

c:\ j2sdk1.4.2 \demo

c:\ j2sdk1.4.2 \include

c:\ j2sdk1.4.2 \lib

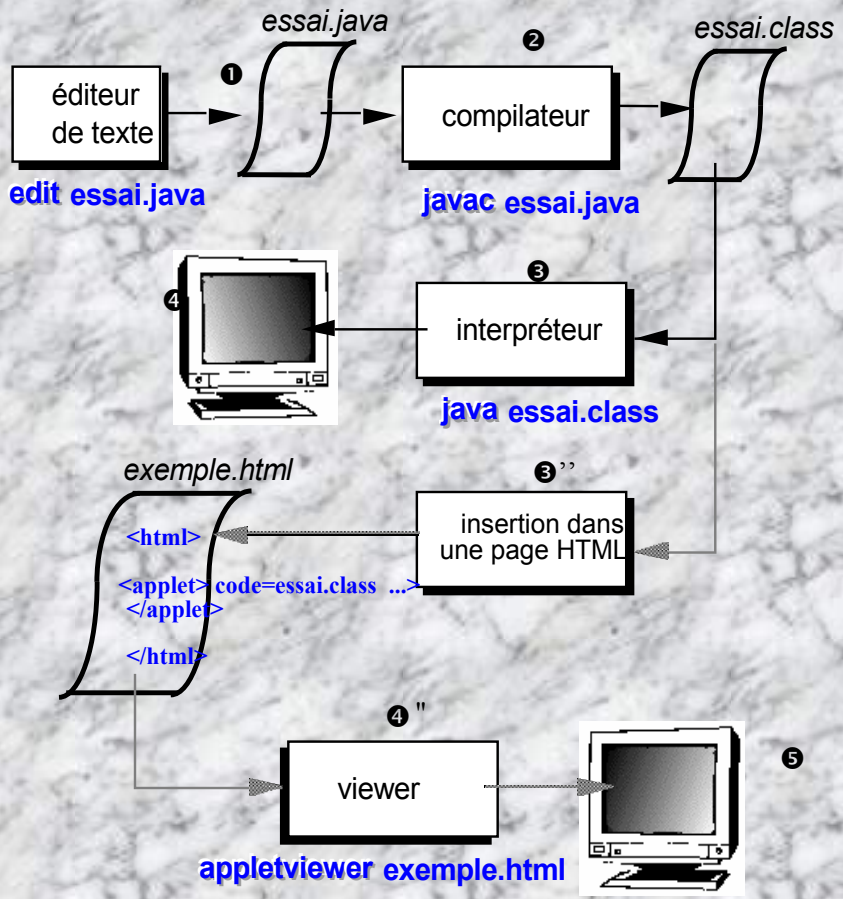




Trois commandes essentielles dans c:\jsdk1.4.2\bin:

java.exe	**Interpréteur java pour exécuter les programmes *.java**
javac.exe	**Compilateur traduisant le texte source *.java en pcode *.class**
appletviewer.exe	**Visionneur d'applet qui permet d'exécuter un fichier *.html contenant l'applet.**

Schéma de développement de programme JAVA à l'aide du JDK de Sun:



❶ On écrit le texte source du programme JAVA à l'aide d'un éditeur de texte (edit, worpad). Le fichier doit être suffixé par l'extension *.java*

❷ On compile le texte *essai.java* avec l'utilitaire **javac.exe**, celui-ci produit le fichier *essai.class*

❸ On lance l'interpréteur **java.exe**, qui charge la classe essai (trouvée dans le fichier *essai.class*). Puis exécute la méthode *main* de cette classe.

❸'' Eventuellement, on peut insérer la classe (une applet) dans une feuille exemple.html en utilisant la balise HTML: <applet> .... </applet>

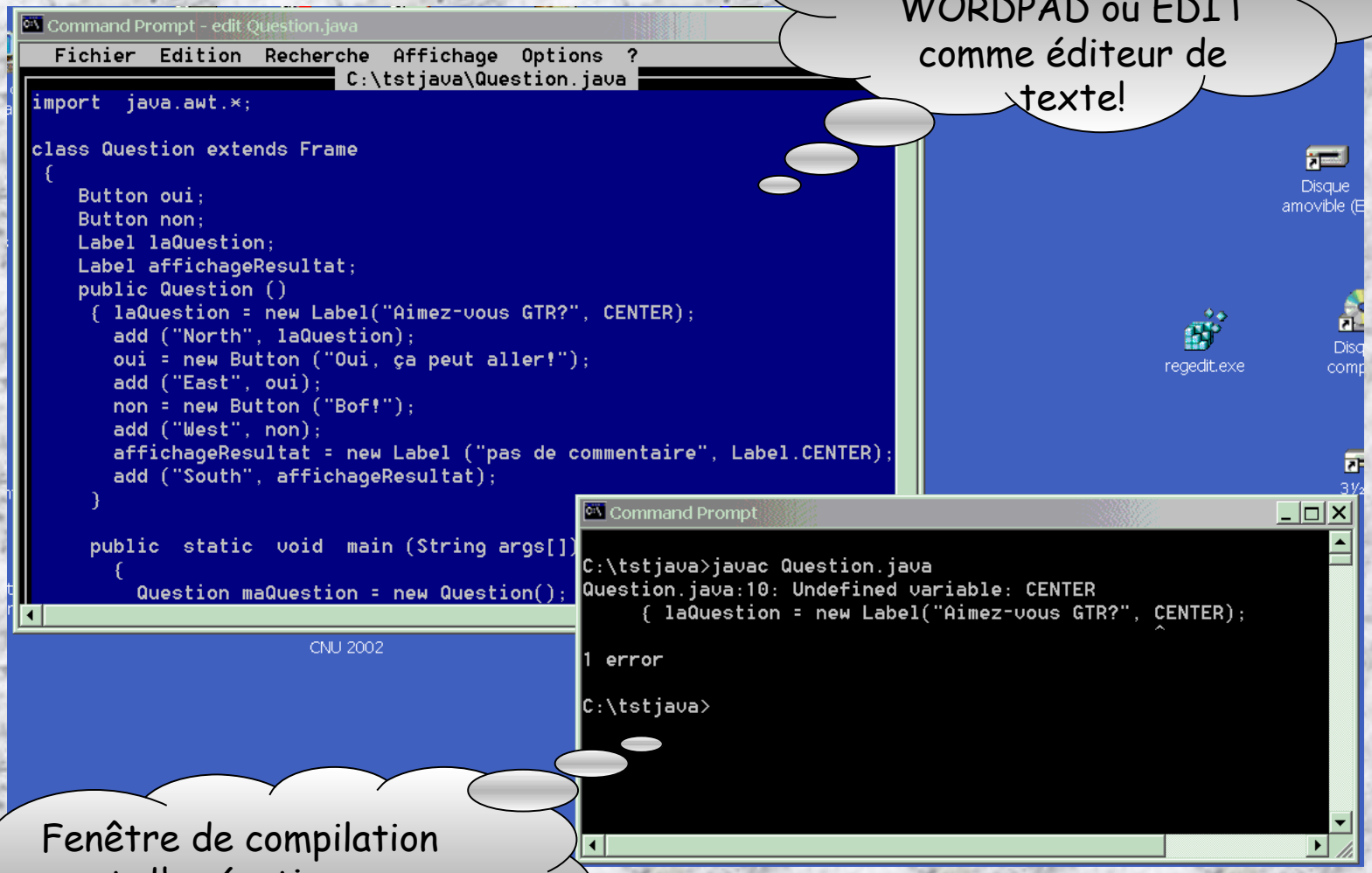
❹ L'exécution du programme affiche ses résultats

❹" On appelle ensuite l'utilitaire viewer d'applet: **appletviewer** pour exécuter la page exemple.html (ou hotjava)

❺ L'affichage de la page html s'exécute ...

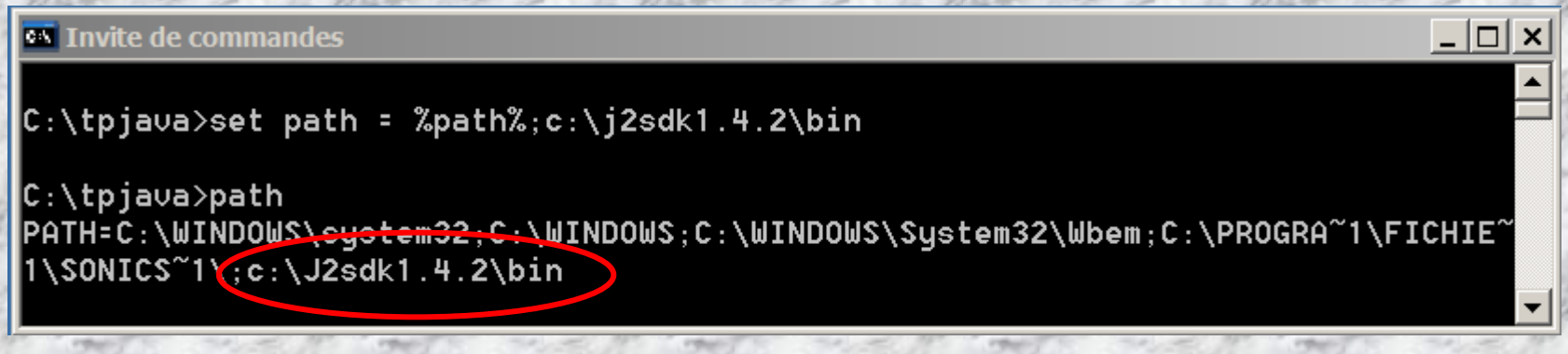
Le nom du fichier source *.java doit être absolument identique (casse comprise) au nom de la classe à compiler: essai.java ↔ public class essai { ... }

Un OS mutitâches (win9x, winNT, linux, ...) fournit un EDI de "campagne" ...

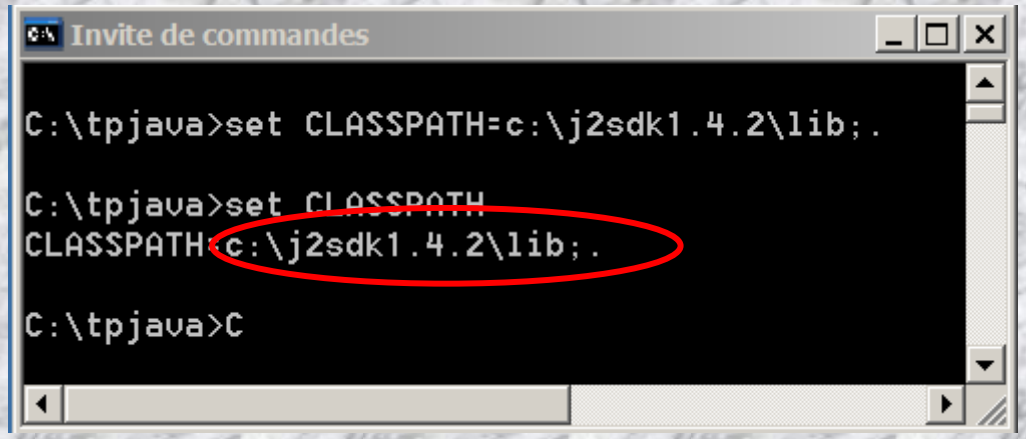


mais … 2 variables système « vitales » à régler préalablement …

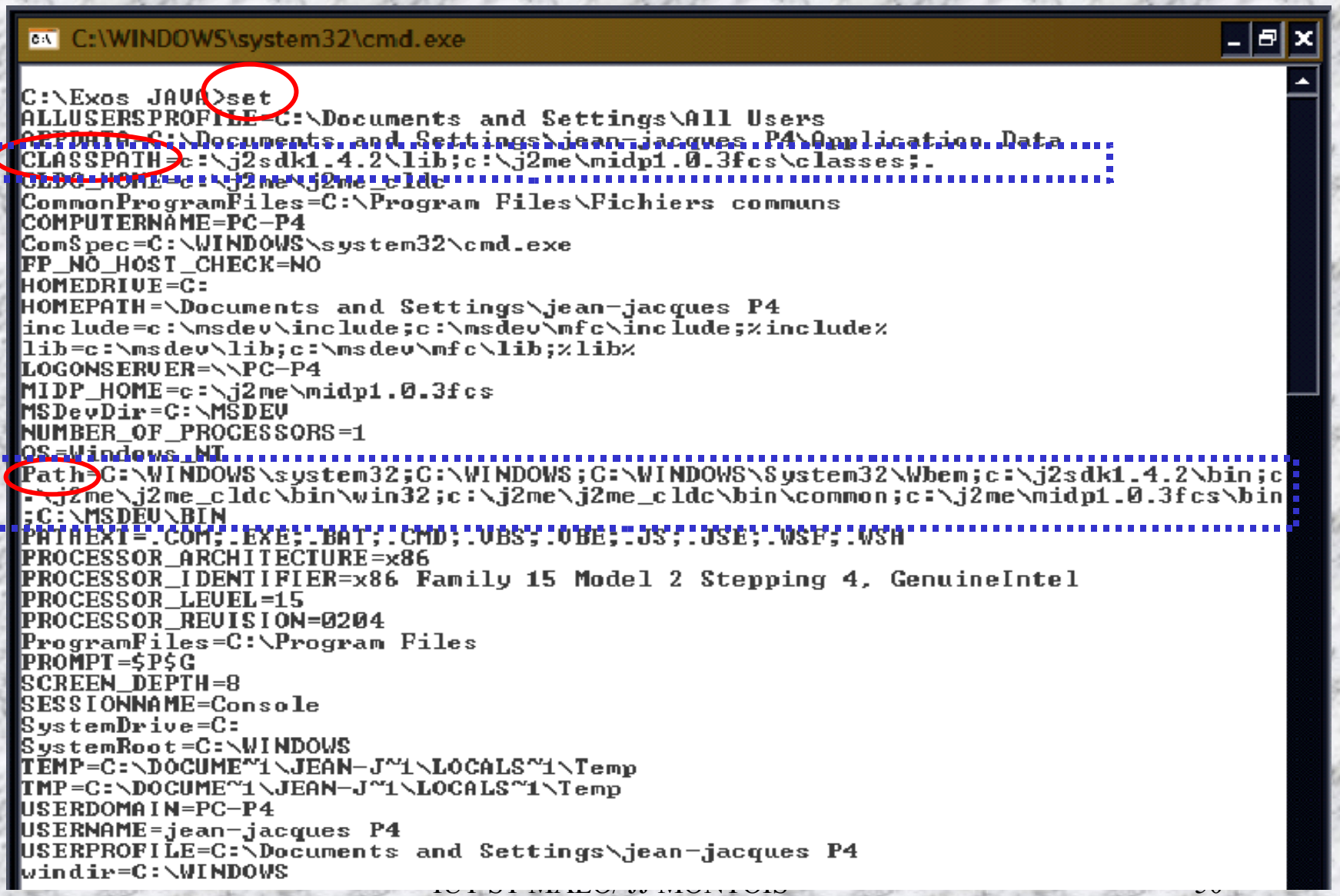
PATH = %PATH%;c:\j2sdk1.4.2\bin



CLASSPATH = c:\j2sdk1.4.2\lib;.

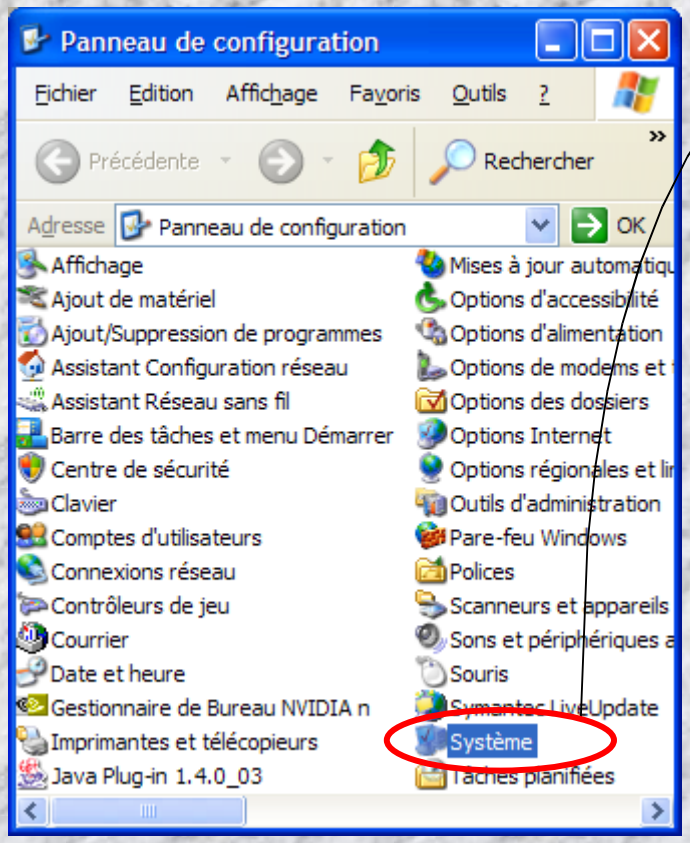


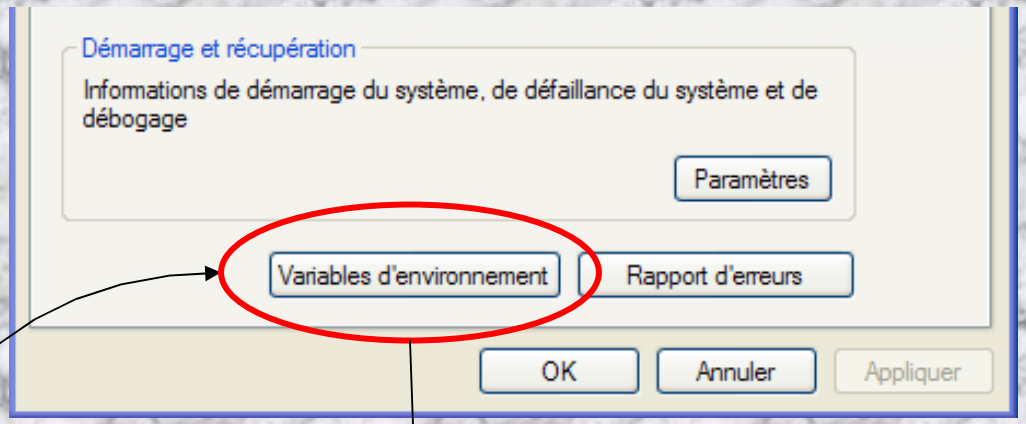
Pour connaître l'ensemble des variables systèmes et utilisateurs de sa machine informatique (Windows ou Unix): exécuter la commande en ligne: set



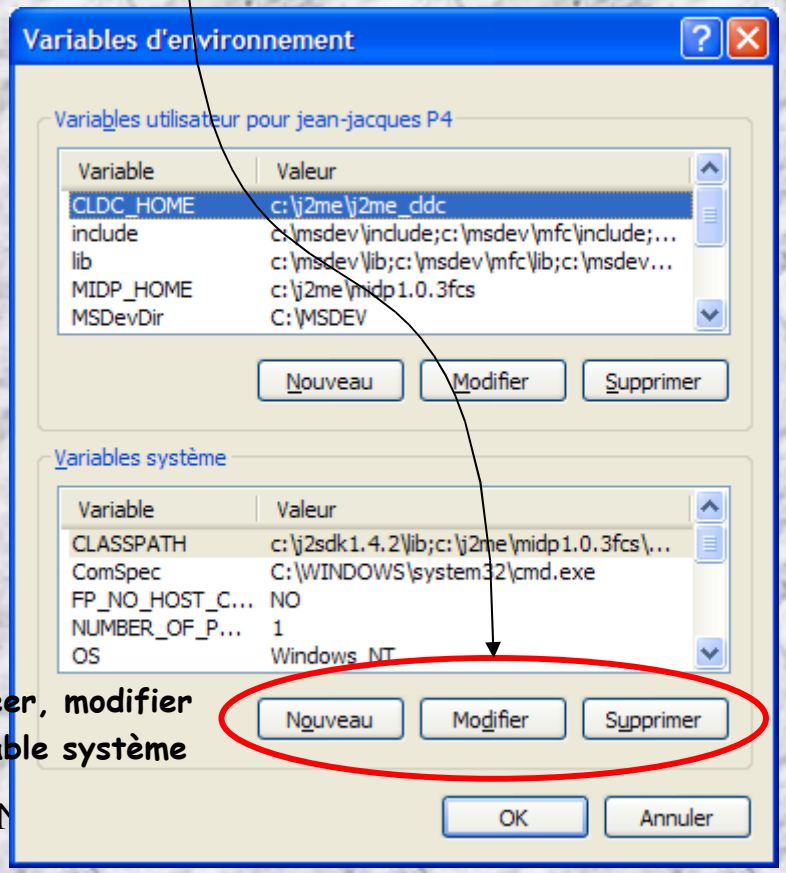
Sous windows XP …

❶ Ouvrir le panneau de configuration et cliquer sur System





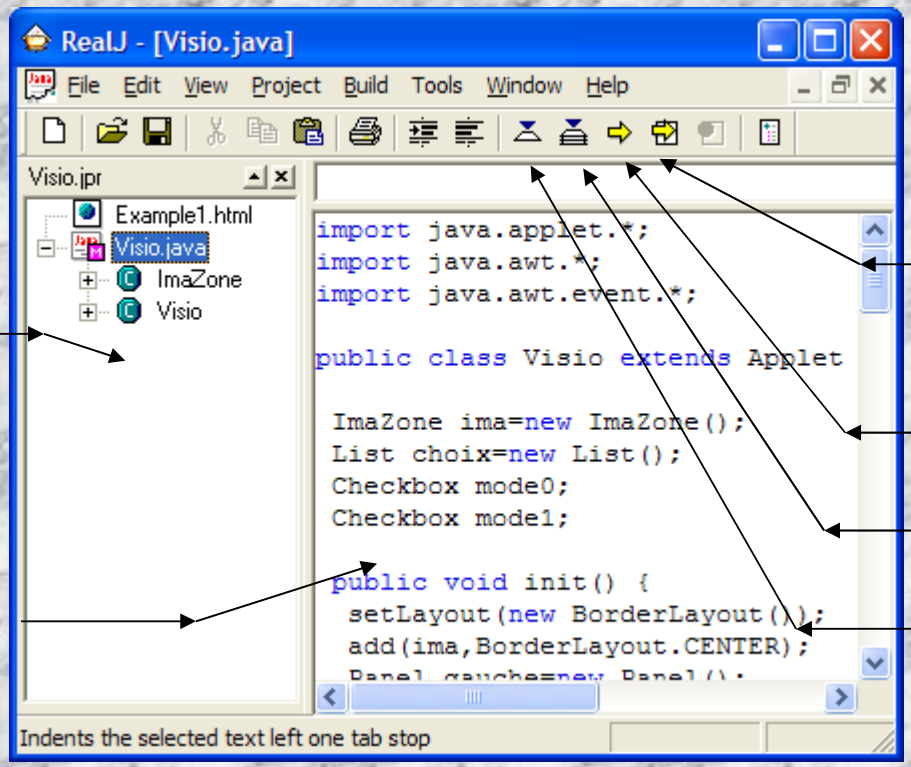
❷ cliquer



❸ Créer, modifier une variable système

Un EDI plus sophistiqué et … gratuit: RealJ

- Une interface simple pour construire: Une application, une applet
- La plateforme J2sdk doit être au préalable installée



Fenêtre donnant la structure de la classe principale

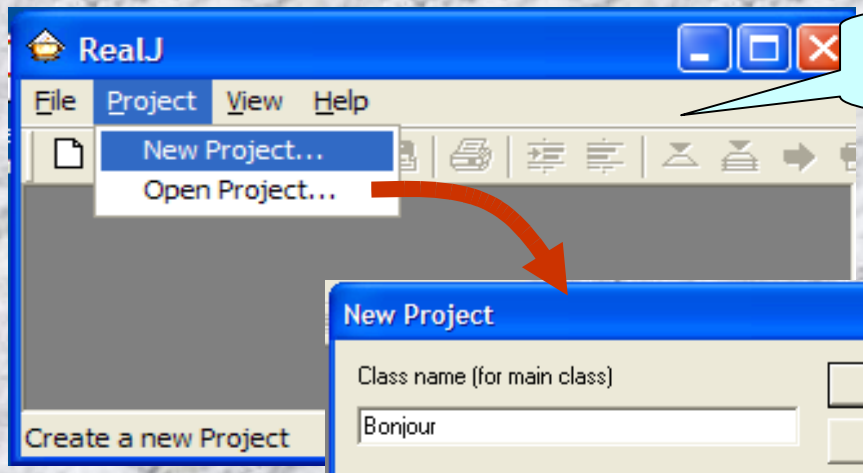
Fenêtre d'édition du fichier *.java

Exécuter une applet

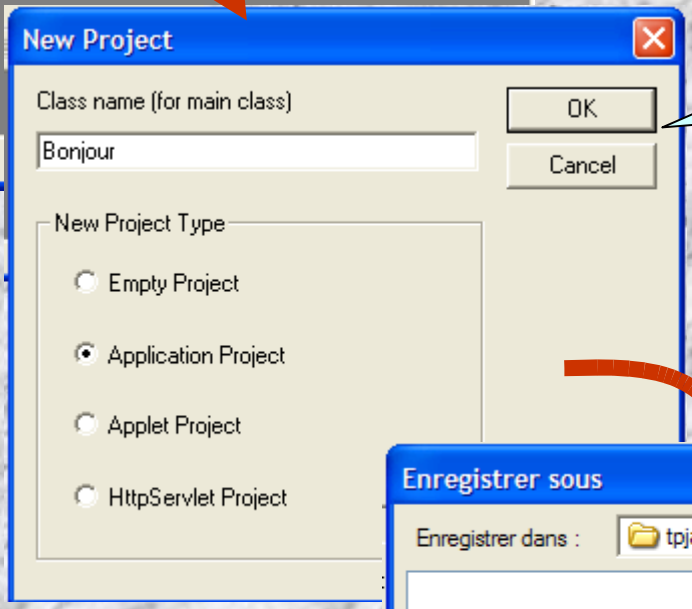
Exécuter une application

Construire le fichier *.class

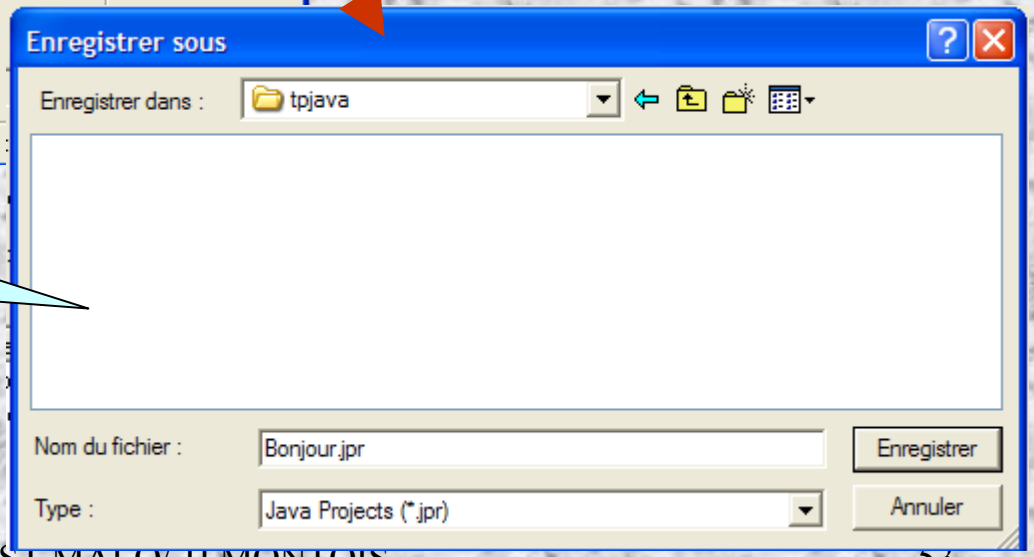
Compiler le fichier *.java



1°) Avant toute chose, créer un projet



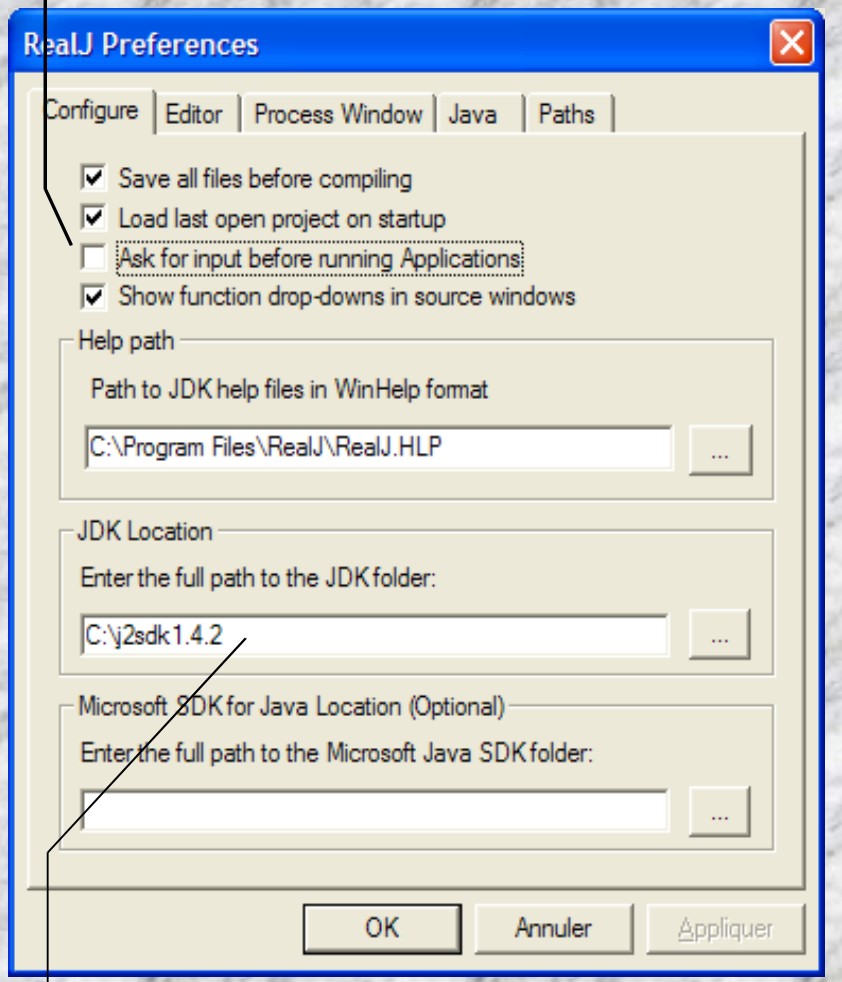
2°) Donner le nom de la classe principale: Bonjour. Si c'est une application (méthode main), cocher la case Application Project

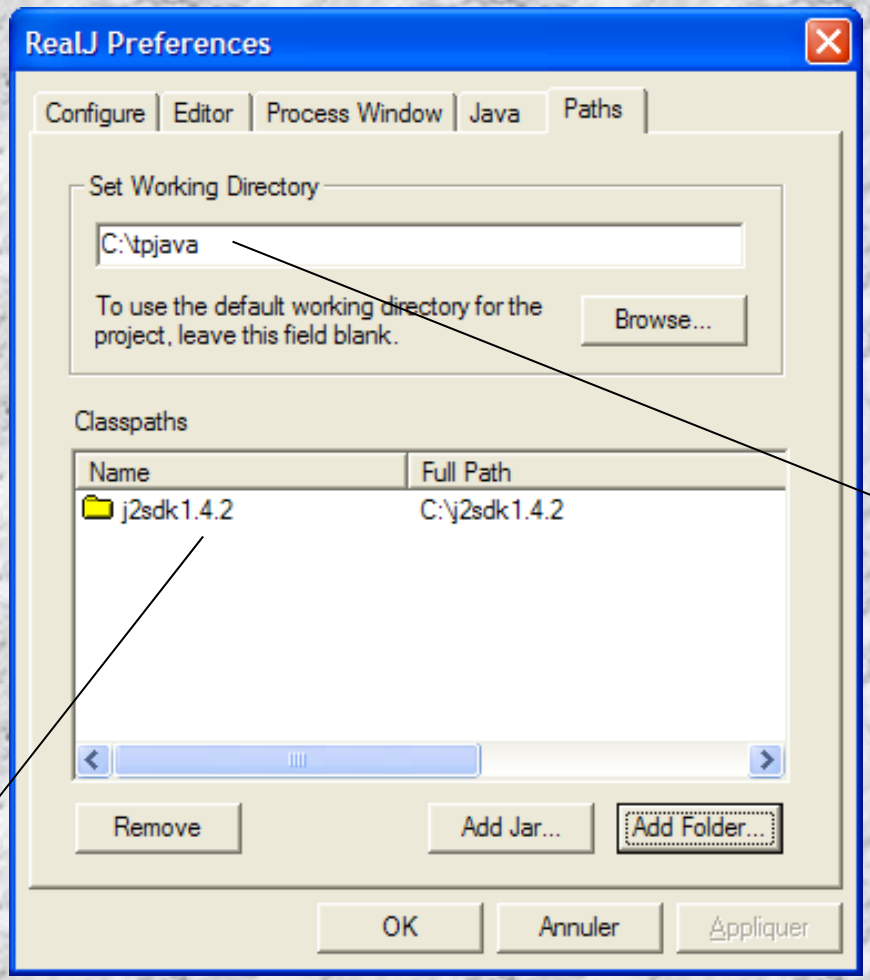


3°) Enregistrer le fichier projet Bonjour.jpr dans un répertoire de travail

Tous les réglages et configurations sont stockés dans le fichier *.jpr

4°) Cocher ici, seulement dans le cas de saisie d'arguments en ligne

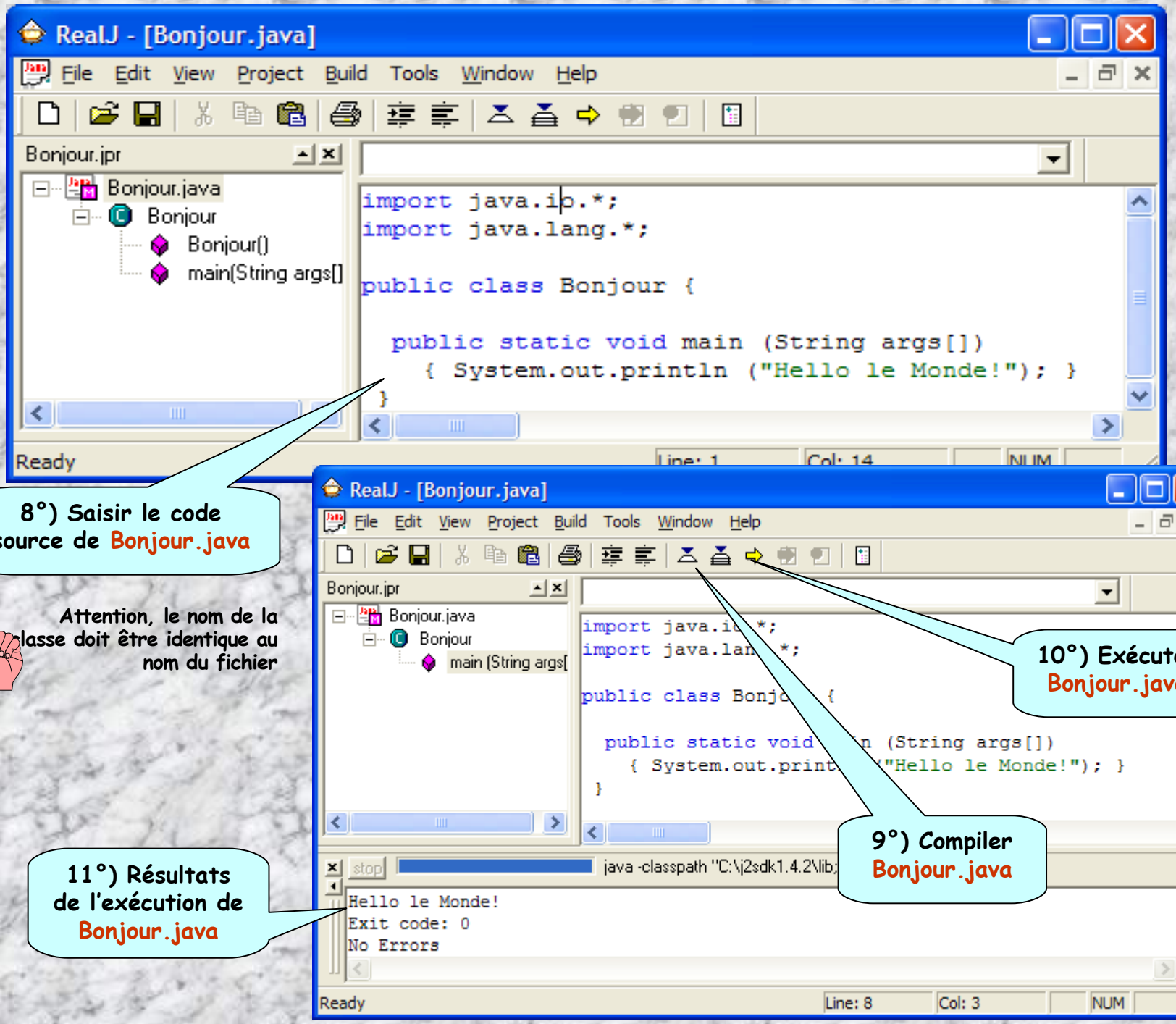


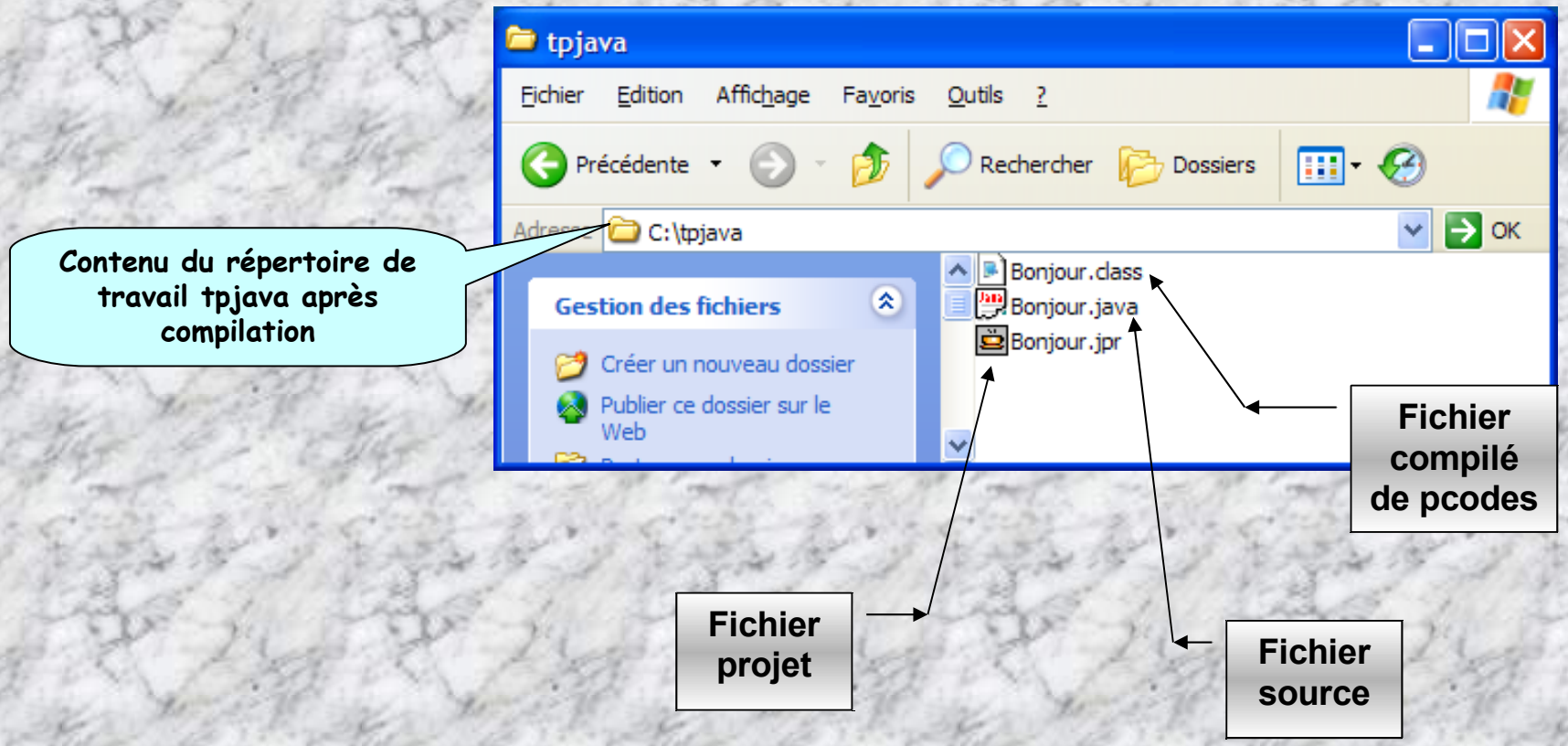


5°) Préciser le répertoire où se trouve la plateforme JAVA Sun

6°) Préciser le répertoire où se trouve les librairies de classes JAVA

7°) Préciser le répertoire où sera déposé tous vos fichiers


RealJ - [Bonjour.java]
File Edit View Project Build Tools Window Help
Bonjour.jpr
Bonjour.java
Bonjour
Bonjour()
main(String args[]
import java.io.*;
import java.lang.*;
public class Bonjour {
public static void main (String args[])
{ System.out.println ("Hello le Monde!"); }
}
Ready
Line: 1
Col: 14
NUM
8°) Saisir le code source de Bonjour.java
Attention, le nom de la classe doit être identique au nom du fichier
RealJ - [Bonjour.java]
File Edit View Project Build Tools Window Help
Bonjour.jpr
Bonjour.java
Bonjour
main (String args[
10°) Exécuter Bonjour.java
9°) Compiler Bonjour.java
stop
java -classpath "C:\j2sdk1.4.2\lib;
Hello le Monde!
Exit code: 0
No Errors
11°) Résultats de l'exécution de Bonjour.java
Ready
Line: 8
Col: 3
NUM



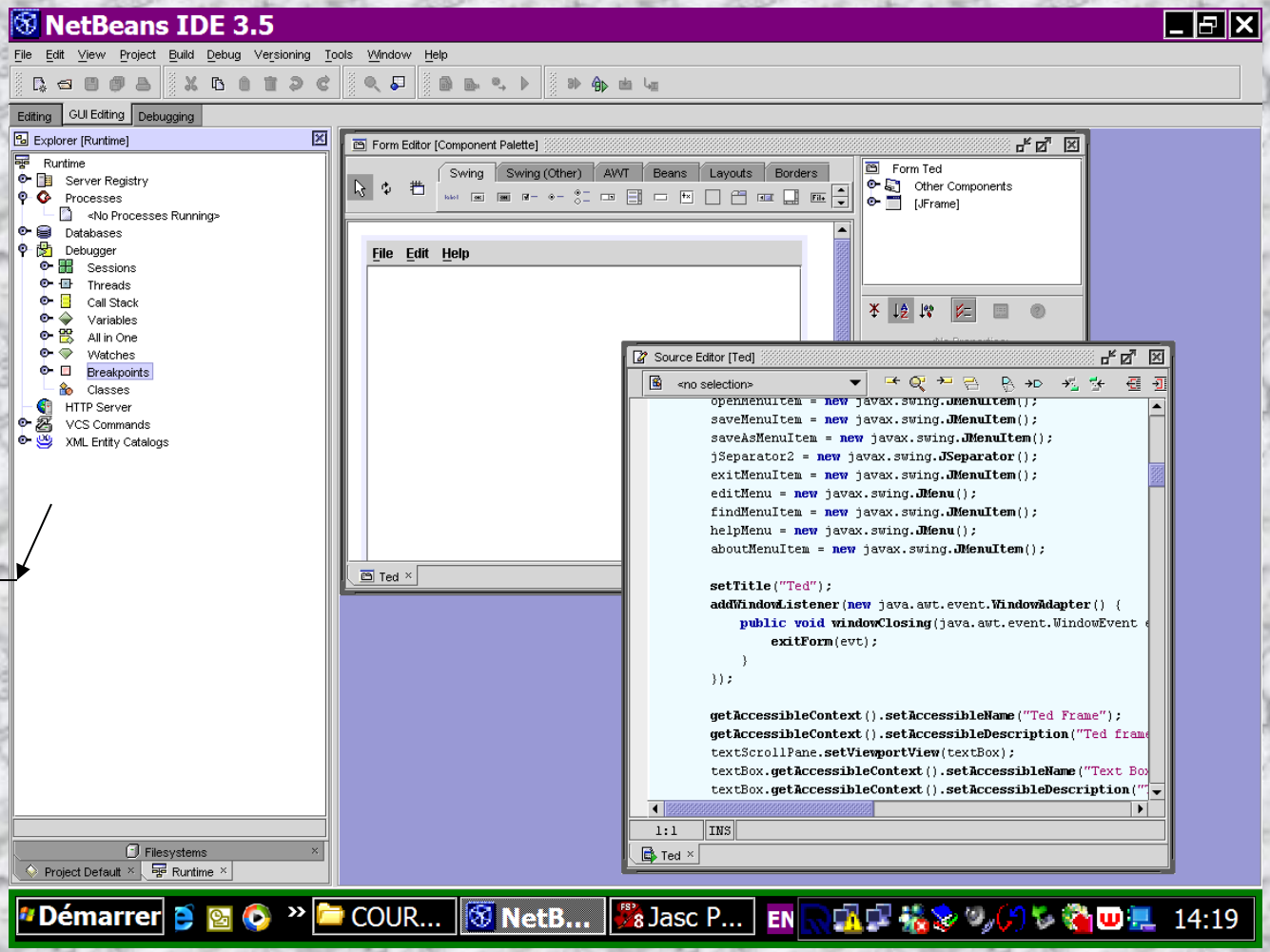
ClassNotFoundException

C'est un message d'erreur qui apparaît souvent lorsque la variable CLASSPATH est mal configuré, entre autre, lorsqu'on a pas définit le répertoire de travail où se trouve notre classe recherchée par le compilateur.

L'EDI sophistiqué, complet, et … gratuit: Netbeans de Sun

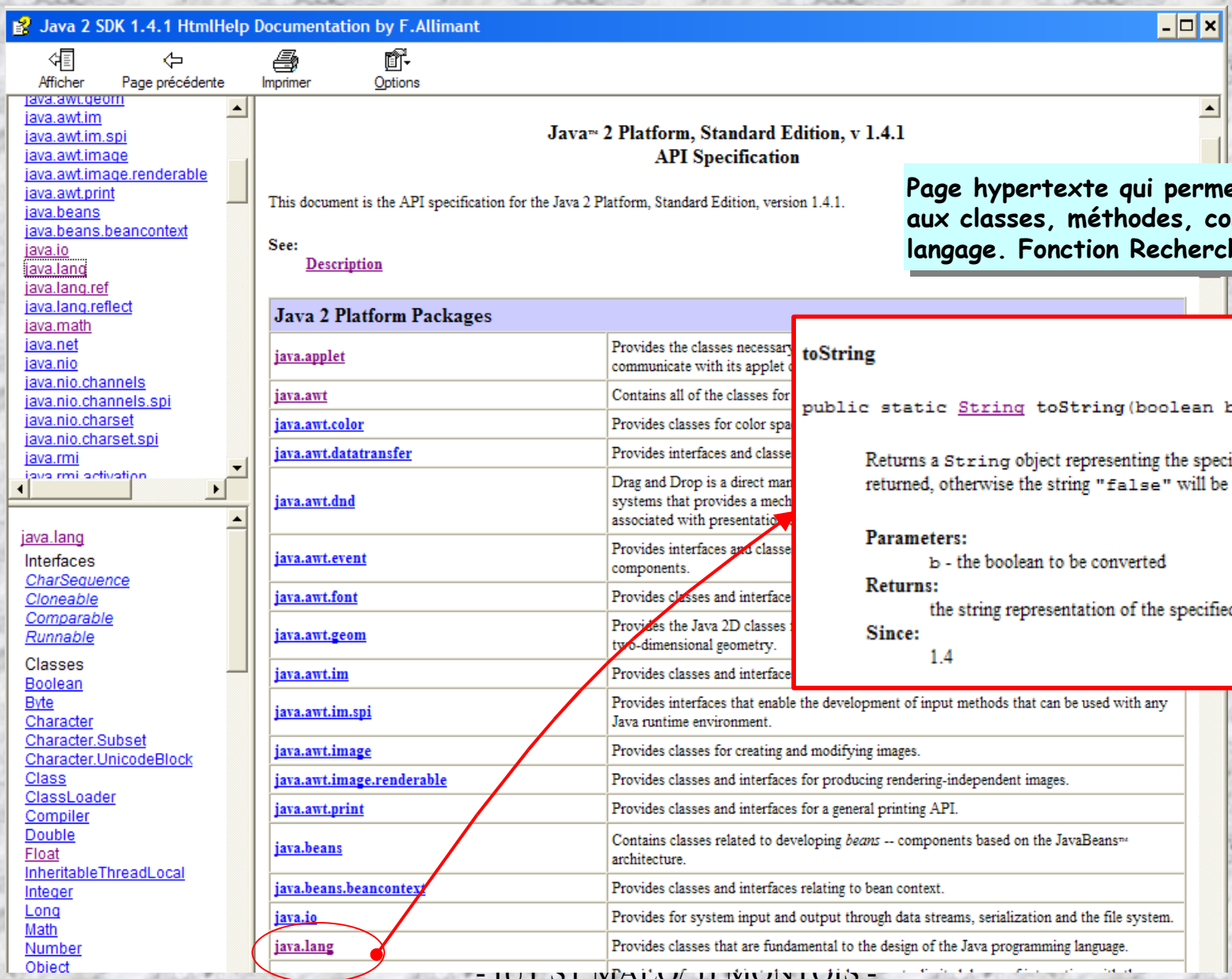
Permet dans le même EDI:

- ✓ **La construction, et l'exécution, la mise au point d'applications,**
- ✓ **La construction, l'exécution, la mise au point d'applets de midlets, de servlet, …**
- ✓ **Intègre les plateformes J2ME, J2SE, J2EE**



Plateforme professionnelle, nécessitant une station de travail « musclée »: PENTIUM 2 à 3 Ghz, au moins 512Mo de RAM, Windows XP ou LINUX

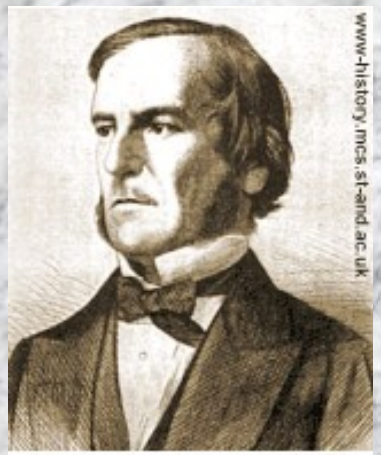
Documentation en ligne des API(s) JAVA (télécharger site Sun)



Page hypertexte qui permet d'accéder aux classes, méthodes, constantes du langage. Fonction Recherche disponible.

Pause culturelle ...

011010010111 ...



George Boole (1815-1864), père fondateur de la logique moderne.

Il y a cent quatre-vingt cinq ans naissait **George Boole**, le père fondateur de la logique moderne qui a contribué à l'avènement de l'informatique. C'est travaux posent les bases de ce qu'on nomme l'algèbre booléenne. George Boole y développe une nouvelle forme de logique, à la fois symbolique et mathématique. Le but : traduire des idées, des concepts en équations, leur appliquer certaines lois (et, ou, non) et retraduire le résultat en assertions logiques.

L'histoire commence comme un conte de Charles Dickens, dans le décor d'une ville industrielle d'Angleterre. George Boole naît le 2 novembre 1815 à Lincoln, dans le Lincolnshire. Issu d'une famille pauvre, il n'aura pas les moyens financiers d'aller à l'université. Ses capacités intellectuelles sont cependant remarquables ; seul (ou presque), il apprend le latin, l'allemand, le français et l'italien qu'il maîtrise déjà à l'adolescence. Obligé de travailler pour soutenir sa famille, il devient enseignant à 16 ans. Quatre ans plus tard, il fonde et dirige sa propre école. C'est à ce moment que le jeune Boole, décidément autodidacte modèle, se plonge dans l'étude des mathématiques auxquelles son père l'avait initié dès l'enfance. Bénéficiant des moyens de l'Institut de Mécanique de sa ville, il se confronte aux œuvres d'Isaac Newton, Pierre-Simon Laplace et Joseph-Louis Lagrange. Mais très vite, il commence ses propres recherches. En 1839, il publie sa première étude dans le *Cambridge Mathematical Journal*. Cette publication lui permet de s'imposer petit à petit comme une personnalité importante du monde des mathématiques. D'après [http://www.histoire-informatique.org/portraits/]

-3-

les classes , les objets

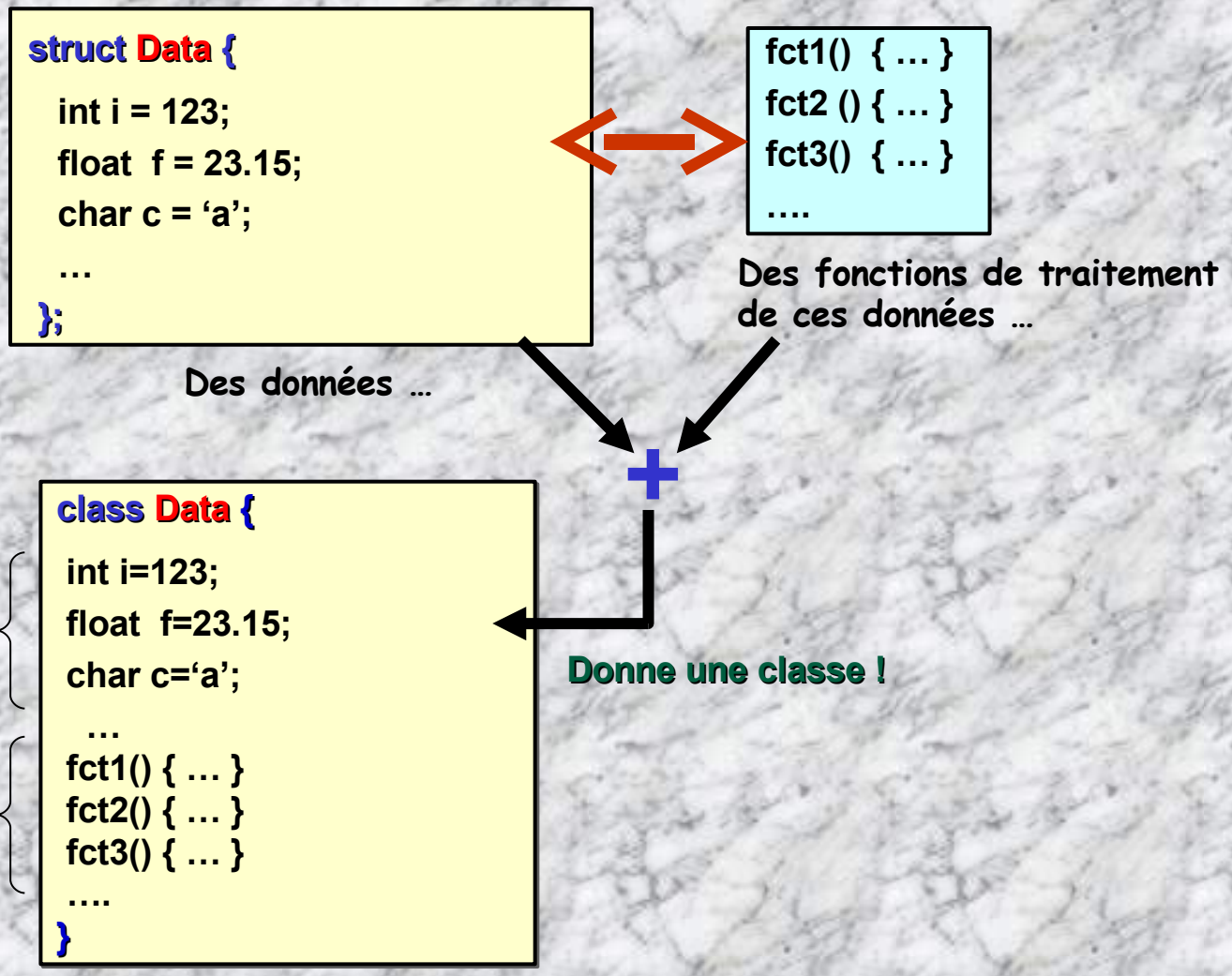
- 1ères notions -

Objets inanimés, avez-vous donc une âme
Qui s'attache à notre âme et la force d'aimer ?...
LAMARTINE

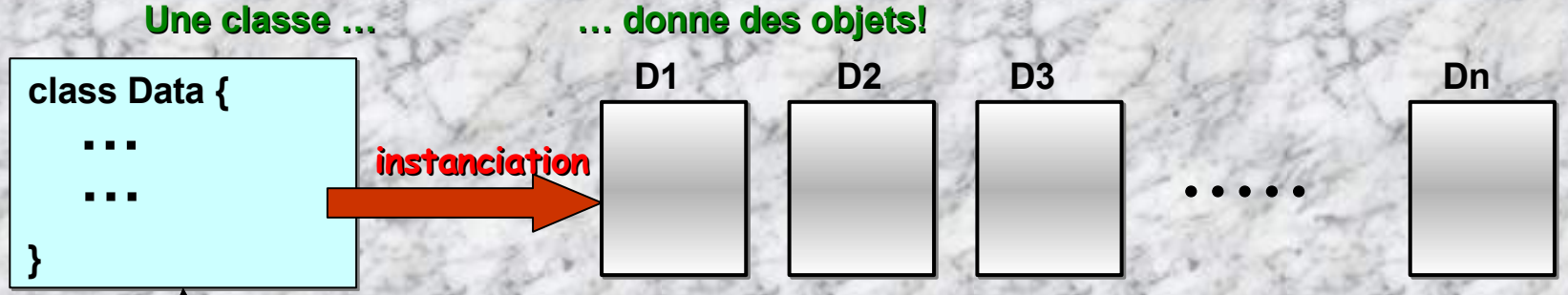
Dans ce chapitre, nous étudions la notion de classe, agrégat de données+méthodes, laquelle permet, comme un moule, de créer des objets qui possèdent différentes propriétés: réservoir de méthodes spécifiques, de variables, surcharge des méthodes, …

Associer les données et les fonctions qui les traitent …

Schéma classique:

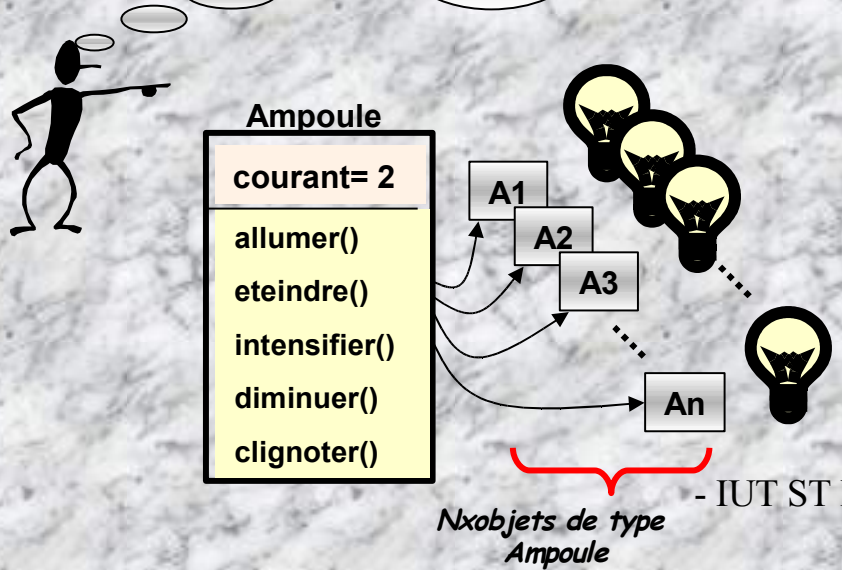


Avec une classe ont fabrique des objets (on dit instancier) …



Une classe représente un **moule**, avec lequel, on fabrique autant d'objets distincts que l'on souhaite

On peut assimiler la définition d'une classe comme la création d'un nouveau type de donnée




```
public class MonProgramme {
  public static void main ()
   {
     Data D1, D2, D3;   //on déclare les objets
     D1=new Data();
     D2=new Data();
     D3=new Data();   //on crée les objets
     …
   }
class Data {
     …
     …
 }
}//fin MonProgramme
```

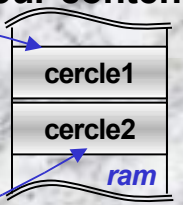
Le programme est lui-même une classe

☞ Une classe est un modèle ou encore un moule avec lequel on crée des objets par une opération d'instanciation.

classe + instanciation = objet

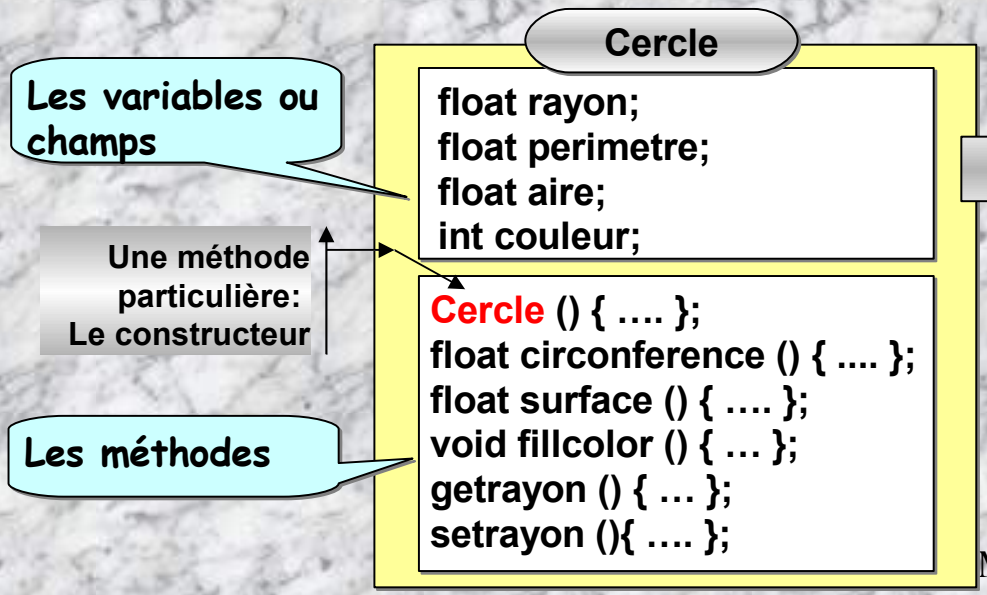
☞ La déclaration d'un objet crée une référence, mais il n'est pas créé physiquement. Il faut utiliser l'opérateur new créant dynamiquement un espace mémoire pour contenir l'objet

```
Cercle cercle1= new Cercle(5);
Cercle cercle2=new Cercle(10);
```



Création de 2 espaces mémoire

☞ Une classe contient des variables et des méthodes

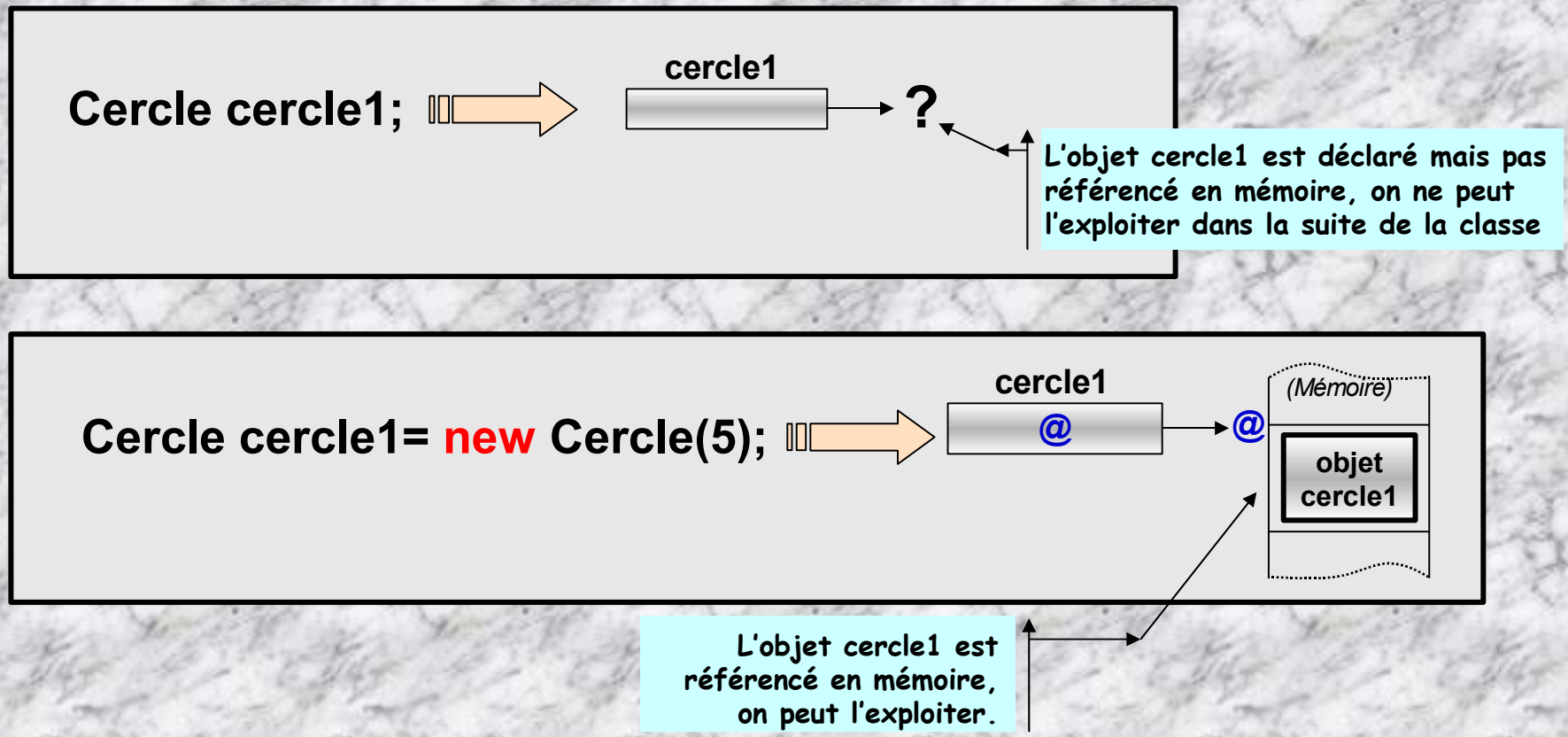


usage

```
classe Cercle {
 public static void main ()
  {
   Cercle cercle1 = new Cercle(15);
   Cercle  cercle2 = new Cercle(10);
   cercle1.fillcolor();
   float s=cercle2.surface();
 ….
   }
}
```

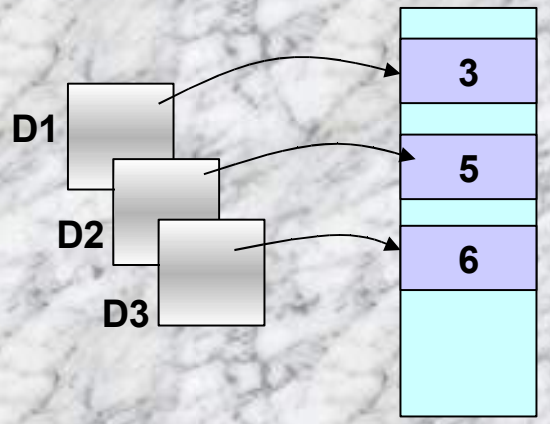
Allocation dynamique de mémoire. Opérateur new …

- En langage JAVA, il n'y a pas d'instruction de gestion dynamique de la mémoire comme en lange C et C++ (malloc, calloc, free, delete, …). En effet, l'allocation de mémoire est automatique, et le garbage collector gère les restitutions mémoire au système (pas de free, ni de delete).
- Une seule directive : new permet de créer un espace mémoire pour y placer l'instance d'une classe (un objet)



☞ Accès aux données d'un objet:

```
public classe MonProgramme {
  public static void main ()
    {
      Data D1 = new Data();  //déclaration et réservation mémoire
      Data D2 = new Data();
      Data D3 = new Data();
      D1.i = 3;
      D2.i = 5;
      D3.i = 6;
      …
    }
…
}//fin
```



Chaque objet gère un espace de mémoire distinct

Méthode principale

NomObjet•<nomVariable> = <*valeur d'initialisation*>

☞ Accès aux méthodes d'un objet:

```
public classe MonProgramme {

  public static void main ()
   {
      Data D1 = new Data();  //déclaration et réservation mémoire
      Data D2 = new Data();
      Data D3 = new Data();

      int  entier   =  D1.fct1();
      float  reel   =  D2.fct2();
      char  caract =  D3.fct3();
      …

   }

…

}//fin
```

<variable> = NomObjet•<nomfct>(arg1, arg2, … , argn)

Variable pour recevoir le résultat retourné par la méthode

Liste des arguments passés à la méthode

La surcharge de méthodes

Une méthode est dite surchargée si elle permet plusieurs passages de paramètres différents.

Exemple:

```
int  somme (int a, int b)          { return (a+b); }
int  somme (int a, int b, int c)   { return (a+b+c); }
float somme (float a, float b)     { return (a+b); }
...
int  argA = 1, argB = 2, argC = 9;
float f1 = 1.3, f2 = 2.6;
int entSomme = somme (argA, argB, argC);
float fSomme =  somme (f1, f2);
```

Le compilateur reconnaît la méthode grâce à sa signature

☞ **Créer une surcharge de méthodes, autorise le traitement d'une même action avec des arguments de types différents.**

Exemple:

- matrice **Multiplication (matrice m1, matrice m2);**
- complexe **Multiplication (complexe c1, complexe c2);**
- int **Multiplication (int i1, int i2);**

Une méthode particulière: le constructeur

Un constructeur est une méthode appelée automatiquement lors de l'instanciation de l'objet; ainsi une initialisation des attributs de l'objet sera systématiquement exécutée par le constructeur.

- **Le constructeur est identifié par son nom qui est celui de la classe et par le fait qu'il est sans type retourné.**
- **Par défaut de constructeur, le compilateur recherche un constructeur sans paramètre dans la classe supérieure.**
- **Il peut y avoir plusieurs constructeurs surchargés (avec des arguments différents à la création des objets)**

```
class MaClass {
  private String uneChaine;
  MaClass (String s) { uneChaine = s; }
  MaClass () { uneChaine = " " ; }
  …
}
```

Les constructeurs initialisent soit une chaîne s, soit une chaîne vide à la création de l'objet

```
class MonProgramme {
  …

public void static main (String argv[]) {
  MaClass toto = new MaClass ( "il fait beau" );
  MaClass titi = new MaClass ();
  …


  finalize () { … }  //si besoin!
}
```

Il n 'y a pas de destructeur comme en C++, cependant, on peut utiliser la méthode finalize() pour effectuer des opérations juste avant la sortie de la classe:

```
void finalize() { … }
```

Exemple: Surcharges des constructeurs de la classe String (extrait doc Sun):

String() Initializes a newly created String object so that it represents an empty character sequence.

String (byte[] bytes) Constructs a new String by decoding the specified array of bytes using the platform's default charset.

String (byte[] ascii, int hibyte) **Deprecated.** *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the String constructors that take a charset name or that use the platform's default charset.*

String (byte[] bytes, int offset, int length) Constructs a new String by decoding the specified subarray of bytes using the platform's default charset.

String (byte[] ascii, int hibyte, int offset, int count) **Deprecated**. *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the String constructors that take a charset name or that use the platform's default charset.*

String (byte[] bytes, int offset, int length, String charsetName) Constructs a new String by decoding the specified subarray of bytes using the specified charset.

String (byte[] bytes, String charsetName) Constructs a new String by decoding the specified array of bytes using the specified charset.

String (char[] value) Allocates a new String so that it represents the sequence of characters currently contained in the character array argument.

String (char[] value, int offset, int count) Allocates a new String that contains characters from a subarray of the character array argument.

String (String original) Initializes a newly created String object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.

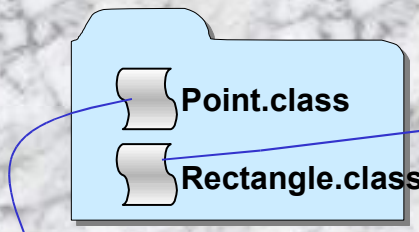
String (StringBuffer buffer) Allocates a new string that contains the sequence of characters currently contained in the string buffer argume

Un petit exemple pour la route …

```
Class TestString {  …
    char UnTableau [] = {'a', 'b', 'c', 'd'};
    String s1= new String ()  //string vide
    String s2= new String (" il fait beau ");   //s2 vaut " il fait beau "
    String s3 = new String (s2);   //que vaut s3?
    String s4 = new String (UnTableau);
    String s5 = " l'homme est un prédateur pour l'homme ";
…
```

Un exemple complet …

On considère 2 classes indépendantes situées dans un même répertoire (par ex.)




```
public class Point {
    public int x = 0;
    public int y = 0;
    public Point (int a, int b)
    { x = a; y = b; }
}
```

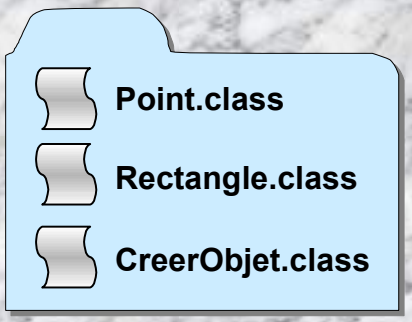
Remarquer que les 2 fichiers source: Point.java, et Rectangle.java vont être compilé séparément, et produire chacun un fichier de pcodes *.class

```
public class Rectangle {
        public int largeur;      //largeur du rectangle
        public int hauteur;      //hauteur du rectangle
        public Point origine;  //origine, coing gauche inférieur du rectangle

    //Créer un rectangle avec la taille spécifiée d'origine (0,0)
     public Rectangle (int l, int h) { this (new Point(0, 0), l, h); }

  //Créer un rectangle avec l'origine p, et la taille spécifiée (l, h).
    public Rectangle (Point p, int l, int h) {
     origine = p;
     largeur = l;
     hauteur = h;
    }

  // Bouger le rectangle sur l'origine spécifiée.
    public void move (int x, int y) {
     origine.x = x;
     origine.y = y;
    }

 //Retourne la surface calculée du rectangle.
    public int surface () { return largeur * hauteur; }
}
```

2 constructeurs

2 méthodes

Exploiter les classes Point, et Rectangle en construisant une classe CreerObjet incorporant ces 2 classes:




```
public class CreerObject {
public static void main (String[] args) {
    Point une_origine = new Point(23, 94);
    Rectangle R1= new Rectangle (une_origine, 100, 200);
    Rectangle R2 = new Rectangle(50, 100);

    System.out.println (“Largeur de R1: " + R1.largeur);
    System.out.println ("Hauteur de R1: " + R1.hauteur);
    System.out.println (“Surface de R1: " + R1.surface());

    R2.origine = une_origine;   // configure position de R2
    System.out.println ("Position x de R2: " + R2.origine.x);
    System.out.println ("Position y de R2: " + R2.origine.y);
    R2.move(40, 72);
    System.out.println ("Position x de R2: " + R2.origine.x);
    System.out.println ("Position y de R2: " + R2.origine.y);
  }
}
```

créer un objet Point initialisés, et 2 objets Rectangle initialisés

Affiche largeur, hauteur, surface de R1

bouge R2 et affiche sa nouvelle position

A l'exécution …

```
Largeur de R1: 100
Hauteur de R1: 200
Surface de R1: 20000
Position x de R2: 23
Position y de R2: 94
Position x de R2: 40
Position y de R2: 72
```

java CreerObjet.class ↵

Passage d'arguments dans une méthode

- Le mode de passage des arguments dans les méthodes dépend de la nature des paramètres.
- Le passage de données par pointeur ou adresse comme en C/C++, est remplacé avantageusement en JAVA par le passage par référence, similaire à un passage par adresse, mais en conservant la référence nominative de la donnée (var et plus de *var ou &var)

En JAVA, on passe les paramètres:

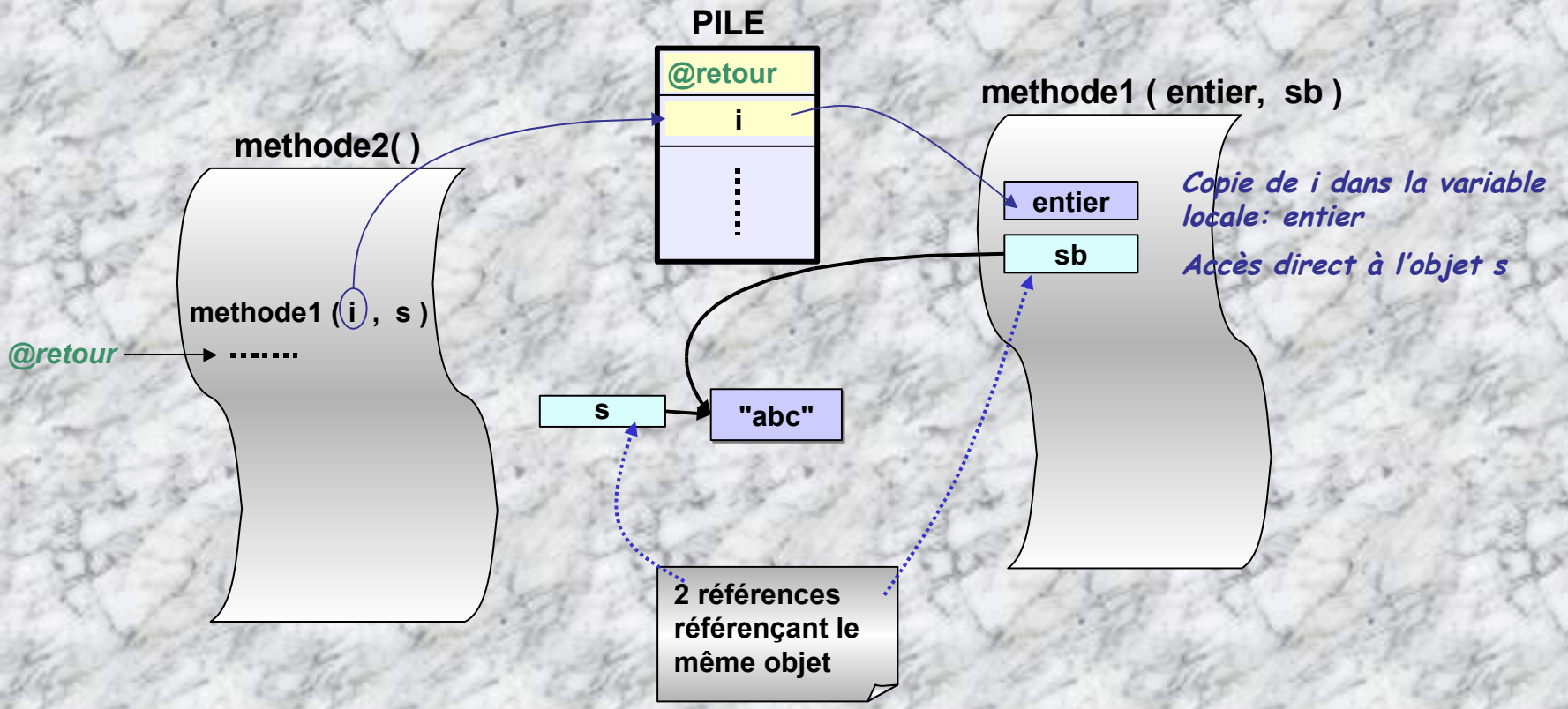
- **Par référence pour les objets**
- **Par copie ou valeur pour les types primitifs**

```
public class MaClasse
 {
   void methode1(int entier, StringBuffer sb)
    { entier++; sb.append("d");}

   void methode2()
    {
      int i = 0;
      StringBuffer s = new StringBuffer("abc");
      methode1(i, s);
      System.out.println("i=" + i + ", s=" + s);    // i=0, s=abcd
    }
}
```

Je passe un entier

Je passe une référence

- ✓ **Le passage par valeur utilise le biais de la pile CPU; la variable i est recopié dans celle-ci; methode1() utilise une copie de i. Impossibilité de modification de la valeur originale.**
- ✓ **Le passage par référence est similaire à un passage par adresse. La methode1() crée une adresse (ou référence) temporaire référençant directement la valeur originale de l'objet. Attention, possibilité de modifier la valeur originale, faire attention si ce n'est pas souhaitable.**



Cycle de vie d'un objet (ex. tiré de eTeks): Exécution de Exemple.methode1();

Exemple.java

```
public class Exemple {
  public static void methode1()
    {  Classe1 objet = new Classe1();
       methode2( objet );
       objet.nombre = 12;
    }
  static void methode2 (Classe1 objet1)
   { objet1.nombre = 11;
  }
}//fin
class  Classe1 {
  int nombre;
  Classe1 ()
   { nombre = 10; }
  finalize ()
   { nombre = 0;}
}//fin
```

1.new Classe1() crée une instance de Classe1 sur laquelle est invoqué Le constructeur de Classe1

instance Classe1 | nombre=10

1.La référence objet nouvellement créée désigne ce nouvel objet

objet → instance Classe1 | nombre=10

1.L'appel de methode2() crée une référence objet1, copie de objet qui désigne donc le même objet

objet, objet1 → instance Classe1 | nombre=10

1.La variable nombre de l'objet est Modifiée via la référence objet1 Désignant cet objet

objet, objet1 → instance Classe1 | nombre=11

1.A la sortie de methode2(), la référence objet1 (pas l'objet qu'elle désigne) est supprimée: l'objet n'est plus désigné par objet.

objet → instance Classe1 | nombre=11

1.La variable nombre de l'objet est Modifiée via la référence objet Désignant cet objet.

objet → instance Classe1 | nombre=12

1.A la sortie de methode1(), la Référence objet est supprimée: L'objet n'est plus désigné par Aucune référence et peu être supprimé.

instance Classe1 | nombre=12

Un peu plus tard …

Le Garbage Collector intervient pour Supprimer tous les objets qui ne sont Plus désignés par aucune références. Il appelle pour chacun d'eux leur méthode finalize().

instance Classe1 | nombre=12

Le garbage Collector libère la mémoire prise par l'objet.

instance Classe1 | nombre=12

Un exemple introductif pour « se faire la main »: Observer la structure d'un programme java. [Les dispositifs d'E/S seront étudiés plus précisément, ultérieurement dans le cours …]

Exercice: L'objectif de cet exercice est de faire simplement connaissance avec l'environnement de compilation.
1°) Construire un répertoire c:\tpjava
2°) Dans ce répertoire créer, à l'aide de l'éditeur de texte EDIT, le fichier source Exemple1.java dont le texte est donné ci-dessous. On utilisera la plateforme jdk (voir chpt « **Outillages d'édition, de compilation, d'exécution d'un code JAVA »**)
3°) Compiler le source Exemple1.java, et produire le fichier de pcodes Exemple1.class. Visionner ce pcode: javap –c Exemple1
4°) Exécuter le fichier Exemple1.class à l'aide de l'interpréteur java.

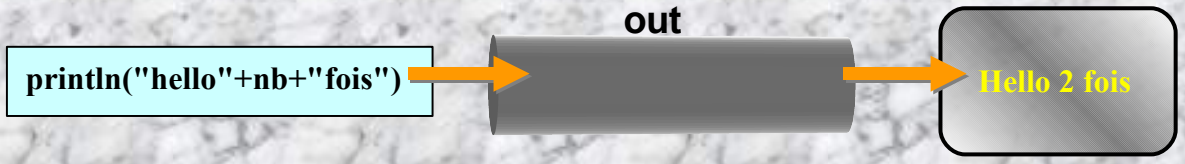
Importation de la bibliothèque de classes d'E/S

Méthode principale

Méthode println permettant l'affichage sur le flux out

```
import java.io.*;
public class Exemple1
{ public static void main (String args[])
    { int nb = 2;
      System.out.println("hello" + nb + "fois");
    }
}
```

Opérateur + de concaténation

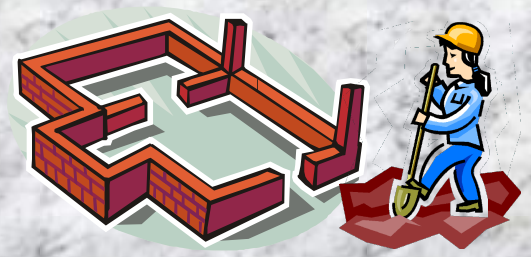


La méthode println n'accepte que des string en arguments et convertit automatiquement tout argument d'un autre type en string.

-4-

Langage JAVA

Les bases …



- Les types de données,
- les conventions d'écriture,
- Les différents types de commentaires;
- Les principales bibliothèques
- Les opérations d'E/S

…

Définitions du langage, les mots et les types du langage

- Les commentaires sont indiqués par: // commentaire de ligne, ou /* ... */ commentaire de bloc de plusieurs lignes, ou /** ... */ commentaire précédent une variable, fonction, classe, afin d'être traité par l'utilitaire javadoc qui construit un fichier documentation HTML du source java .
- Le langage est composé de mots de l'alphabet de a à z, de A à Z et de 0 à 9 pour les premiers caractères. Les suivants peuvent contenir des caractères accentués ou des symboles autres que les 26 lettres de l'alphabet.
- Les identificateurs commencent par une lettre ou les caractères: _ ou $.
- Les entiers sont représentés en base 10, 16 ou 8 avec ou sans point décimal. Les caractères en base 16 sont précédés des symboles 0x ou 0X; les caractères en base 8 sont précédés du symbole 0.
- Les entiers de plus de 32 bits finissent par le symbole L ou l.
- Les entiers peuvent être en simple précision terminés par d ou D (2.0d) ou double précision terminés par f ou F (2.0F).
- Les entiers représentés en notation exponentielle utilisent les symboles e ou E (2.02e2 = 202)
- Les booléens sont représentés par des valeurs true ou false.
- Les caractères sont représentés entre simples quotes ('A') , et les chaînes de caractères entre doubles quotes " …"

Types de base:

Etendue de définition

Les opérateurs utilisables avec les types correspondants

Entiers:

boolean	true/false	==, !=, !, &, ^, |, &&, ||
byte	-128.→.127	==, !=, <, >, <=, >=,+, -, *, /, %, ++, --,<<, >>, >>>, &, |, ^ , (transtypage)
short	-32768.→.32767	voir byte
int	-2147483648 → 2147483647	voir byte
long	-9223372036854775808 → 9223372036854775807	voir byte
char	'\u0000' → '\uffff'	==, !=

Flottants:

float	NaN, -1.4023984 **e-45** → 3.40282347 **e38**	==, !=, <, >, <=, >=, +, -, *, /, %, ++, --, <<, >>, >>>, &, |, ^ , (transtypage)
double	NaN, -4.94065645841243544 **e-324** → 1.79769313486231570 **e308**	voir float

- ✓ **boolean (true/false), byte (1 byte), char (2 bytes), short (2 bytes), int (4 bytes), long (8 bytes), float (4 bytes), double (8 bytes).**
- ✓ **Les variables peuvent être déclarées n'importe où dans un bloc.**
- ✓ **Les affectations non implicites doivent être « *castées* » (sinon erreur à la compilation).**

- Représentation des réels dans le standard IEEE 754. Un suffixe **f** ou **d** après une valeur numérique permet de spécifier le type. Exemples : double x = 145.56d ; float y = 23.4f ;
- Sans suffixe, le réel est assimilé à un double. float f = 23.65 ; //Erreur
- Pour des raisons de sécurité, les opérateurs ne peuvent être surchargés en JAVA (par exemple: pas de + sur une opération vecteur) comme en C++.
- Une constante est déclarée comme une variable avec le mot final en en-tête: `final int N=18;`
  Les constantes sont généralement écrites en majuscule. De façon générale, tout élément déclaré *final* ne pourra être redéfini ou modifié par la suite (par surcharge en cas d'héritage).
  `final int MACONSTANTE = 0;`
- Une variable doit être obligatoirement initialisée lors de sa déclaration, avant son exploitation!

```
int i;
i = i+1
```

```
int i = 0;
i = i+1
```

Conventions …

- Nom de classe → Chaque mot commence par une majuscule — CercleRond
- Nom de fonction → Chaque mot, sauf le 1ier, commence par une majuscule — maFonction
- Constante → Chaque mot est en majuscule, séparé par un souligné — CONSTANTE_PI

Testons les types numériques

```
class TestsTypes {
   final int CONSTANT=1 ;
   public static void main (String args[])
    {
      boolean test=true ;
      byte octet ;
      short petit ;
      int entier ;
      long entier_l ;
      char car ;
      float flottant ;
     double flottant_l ;
     //Ecrit la valeur des variables en sortie
     System.out.println ("La valeur du booléen test est <" + test + "> et sa négation <" + !test +">");
     //Traitement des entiers
      petit = 10 ;
      entier = 1000 * (int) petit - 10 + 3/4 ;
      entier_l = 10000 * (long) entier ;
      System.out.println ("Mes entiers ont pour valeurs : " + petit  + " " + entier + " " + entier_l) ;
     //Traitement des doubles
      flottant = (float) 1e308 ;    /*1e308 est un double par défaut */
      flottant_l = 10.0 * (double) flottant ;
     System.out.println ("Voici ce qui se passe pour un dépassement de capacité " + flottant*10) ;
     System.out.println ("Voici la valeur de pi : " + Math.PI ) ;
     System.out.println ("Voici le resultat d'une division par 0 : " + flottant/Math.sin(0.0)) ;
     //Traitement d'un caractère
      car = 'a' ;
      System.out.println("Voici un caractère : <" + car + ">") ;
    } //fin main
}
```

Résultat de l'exécution :

```
La valeur du booleen test est <true> et sa negation <false>
Mes entiers ont pour valeurs : 10 9990 99900000
Voici ce qui se passe pour un depassement de capacite inf
Voici la valeur de pi : 3.14159
Voici le resultat d'une division par 0 : inf
Voici un caractere : <a>
```

Le programme ci-dessous déclare 8 variables de différents types.

```
public class MaxVariablesDemo {
 public static void main (String args[])
 {
   // les entiers
   byte   largestByte    = Byte.MAX_VALUE;
   short largestShort    = Short.MAX_VALUE;
   int    largestInteger = Integer.MAX_VALUE;
   long   largestLong    = Long.MAX_VALUE;

  // les nombres réels
  float    largestFloat  = Float.MAX_VALUE;
  double largestDouble = Double.MAX_VALUE;

  char unChar = 'S';
  boolean unBooleen = true;

  System.out.println( "Le byte le plus grand: " + largestByte);
  System.out.println(  "L'entier court le plus grand: " + largestShort);
  System.out.println( "L'entier le plus grand: " + largestInteger);
  System.out.println( "L'entier long le plus grand:  " + largestLong);
  System.out.println( " Le plus grand float:  " + largestFloat);
  System.out.println( "Le plus grand double:  " + largestDouble);
  System.out.println( "valeur du bouléen:  " + unBooleen);
  if (Character.isUpperCase(unChar)) { System.out.println(" Le caractère  " + unChar + « est
majuscule"); }
   }
 }
```

Résultats à l'exécution …

```
stop   java -classpath "C:\j2sdk1.4.2\lib;." MyApp
Le byte le plus grand:          127
L'entier court le plus grand: 32767
L'entier le plus grand:         2147483647
Le long le plus grand:          9223372036854775807
Le plus grand float:            3.4028235E38
Le plus grand double:           1.79769313486231570E308
Valeur du booléen:              true
Le caractère S est majuscule
```

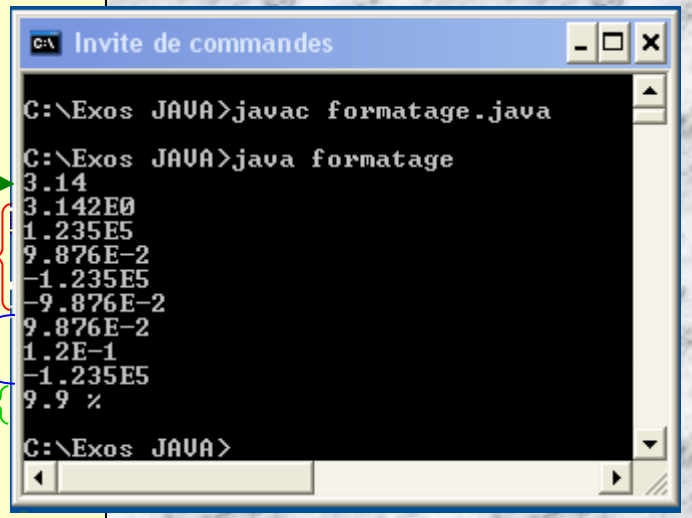
Affichage des résultats.

✧ *Les E/S println-read, seront vues plus loin dans le cours*

FORMATAGE DECIMAL - Exemple d'affichage formaté de nombres réels -

Vincent MAGNIN (EUDIL)

```
import java.text.NumberFormat;
import java.text.DecimalFormat;

public class Formatage
{
  public static void main (String[] args)
  {
    //Première méthode :
    NumberFormat  monFormat = NumberFormat.getNumberInstance();
    monFormat.setMinimumFractionDigits(2);
    monFormat.setMaximumFractionDigits(2);
    System.out.println( monFormat.format (Math.PI));

    //Seconde méthode :
    DecimalFormat monFormatDecimal = new DecimalFormat("0.000E0");
    System.out.println ( monFormatDecimal.format(Math.PI) );
    System.out.println ( monFormatDecimal.format(123456) );
    System.out.println ( monFormatDecimal.format(0.098765) );
    System.out.println ( monFormatDecimal.format(-123456) );
    System.out.println ( monFormatDecimal.format(-0.098765) );

    DecimalFormat monFormatDecimal2 = new DecimalFormat("0.###E0");
    System.out.println (monFormatDecimal2.format(0.098765));
    System.out.println (monFormatDecimal2.format(0.12));
    System.out.println (monFormatDecimal2.format(-123456));

    //Afficher sous forme de pourcentage :
    DecimalFormat monFormatDecimal3 = new DecimalFormat("###.# %");
    System.out.println ( monFormatDecimal3.format(0.098765) );
  }
}
```



Différents types de commentaires

Commentaire de bloc de plusieurs lignes :

```
/*
  …
*/
```

Peut intégrer des commentaires lignes //

Commentaire sur une seule ligne:

```
// ….
```

Commentaires destinés au générateur de documentation javadoc :

```
/**
  @param …
  @return …
  @see …
  ….
  ….
*/
```

@ est le préfixe de la variable de commentaire

- @deprecated
- @exception
- @param
- @return
- @see
- @author
- @version

* Réalisation pratique de la documentation:

- ☞ On génère la documentation : javadoc *angle.java* ↲
- ☞ On obtient un fichier HTML angle.html que l'on peut ouvrir avec n'importe quel navigateur

```
public class angle {

 public angle() { … }
 public angle(double ini) { … }

 /**
  méthode qui ramène la valeur entre 0 et 360
 */
 public void recadre() { … }
 …
 /**
  méthode de conversion en radians
  @return La valeur de l'angle en radians
 */
 public double toRadian() { return valeur*Math.PI/180.0; }

 /**
  addition de 2 angles
  @param a angle ajoute a VALEUR
  @see #multiplier
 */
 public void ajouter(angle a) { … }
 …
 …

 /**
  Fonction sinus
  @return la valeur radian
  @see #cosinus
  @see #tangente
 */
 public double sinus() { return Math.sin(toRadian());}
 …
}
```

SUMMARY: NESTED | FIELD | CONSTR | METHOD

Class angle

```
java.lang.Object
  └ angle
```

toRadian

```
public double toRadian()
```

methode de conversion en radians

Returns:
La valeur de l'angle en radians

ajouter

```
public void ajouter(angle a)
```

addition de 2 angles

Parameters:
a - angle ajoute a VALEUR

See Also:
multiplier(int)

sinus

```
public double sinus()
```

Fonction sinus

Returns:
la valeur radian

See Also:
cosinus(), tangente()

Extrait du fichier HTML …

public class **angle**
extends java.lang.Object

SUMMARY: NESTED | FIELD | CONSTR | METHOD

Class angle

java.lang.Object
  └ **angle**

Constructor Summary

angle()	
angle(double ini)	

Method Summary

void	**ajouter**(angle a) addition de 2 angles
double	**cosinus**() Fonction cosinus
void	**multiplier**(int n) multiplication par un entier
void	**recadre**() methode qui ramene la valeur entre 0 et 360
double	**sinus**() Fonction sinus
double	**tangente**() Fonction tangente
double	**toRadian**() methode de conversion en radians
java.lang.String	**toString**() mise en forme pour affichage

ajouter

```
public void ajouter(angle a)
```

addition de 2 angles

Parameters:
a - angle ajoute a VALEUR
See Also:
multiplier(int)

multiplier

```
public void multiplier(int n)
```

multiplication par un entier

Parameters:
n - un entier
See Also:
ajouter(angle)

sinus

```
public double sinus()
```

Fonction sinus

Returns:
la valeur radian
See Also:
cosinus(), tangente()

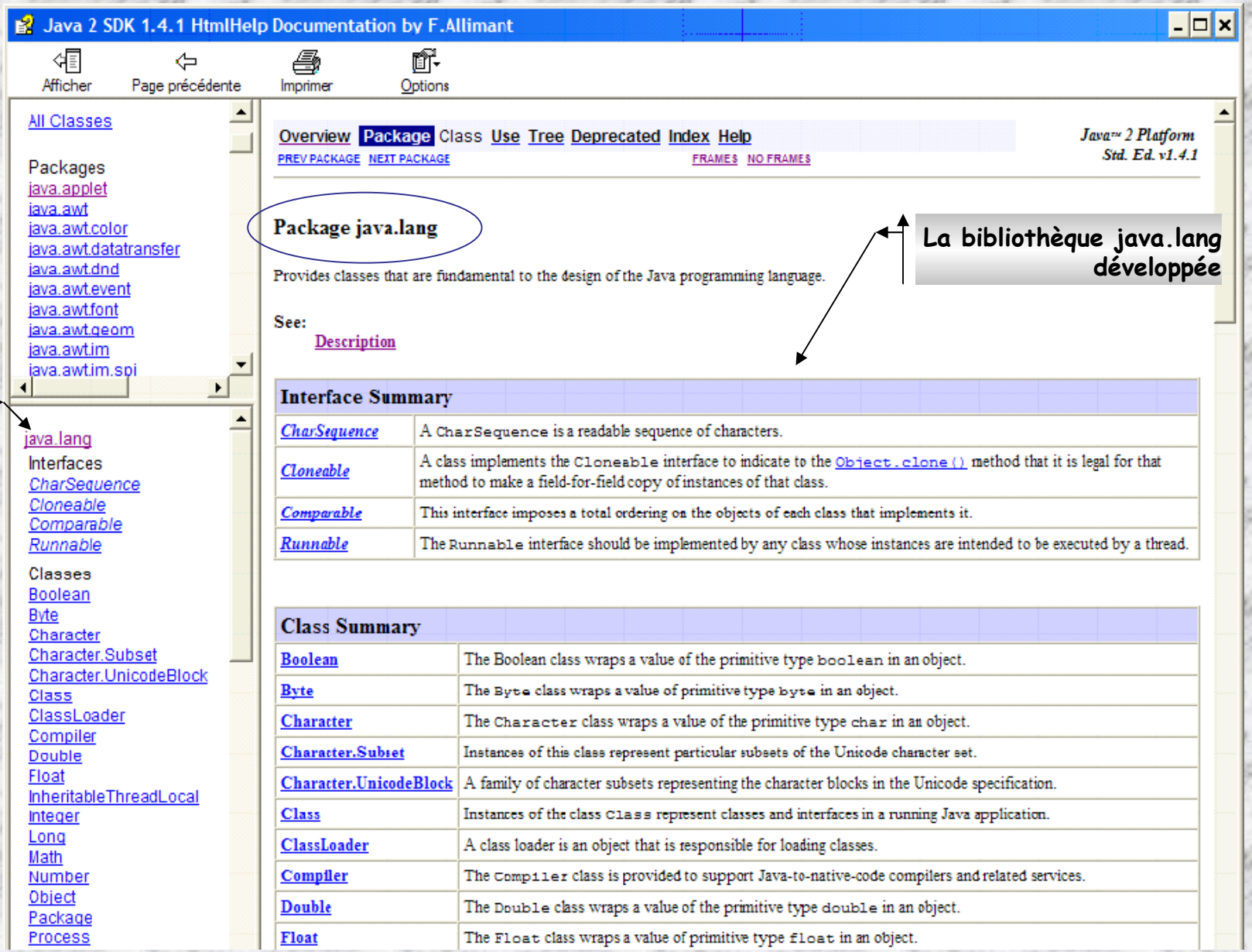
Les grands packages, ou bibliothèques de classes ….

java.applet	importé par défaut dans tout programme.
java.awt	graphismes standards et gui.
java.awt.image	manipuler les images.
java.awt.peer.	
java.io	entrees/sorties.
java.lang	classes liées au langage (fonctions mathématiques ...)
java.net	manipuler les mécanismes réseaux (socket, URL, …)
java.util	structures de données (token, vecteur...).

Il y a plein de méthodes intéressantes dans ces bibliothèques de classes. Soyez curieux (qualité 1ère du programmeur), aller voir les APIs java (usage d'un fichier html d'aide) !

API jdk141.lnk

Le fichier HTML d'aide sur les APIs java



Deux bibliothèques importantes!

1 - Le package **java.lang.** …

Le package java.lang est chargé automatiquement, ses classes sont donc toujours utilisables.

On y trouve, entre autres :

- la classe Object dont dérivent toutes les autres classes
- les classes représentant les types numériques de bases : Boolean, Byte, Double, Float, Integer, Long
- la classe Math qui fournit des méthodes de calcul des fonctions usuelles en mathématiques
- les classes Character, String et StringBuffer pour la gestion des caractères et chaînes de caractères
- la classe System utilisée pour afficher du texte sur la console DOS.

2- Le package **java.util.** …

Ce package contient un ensemble de classes très utiles dans l'élaboration de certaines fonctionnalités, qu'il serait inutile de réécrire:

- Calendar: gérer les questions d'heures, de dates, d'années, …
- Bitset: gérer un vecteur de bits sur lesquels on exécute des opérations logiques, ET, OU, NON, …
- LinkedList: gérer des listes chaînée - ajout, extraction, insertion, …
- Vector: gérer des listes d'objets - ajout, extraction, insertion, …
- Timer: gérer le scheduling de tâches
- Date: gérer le temps
- Random: gérer des variables aléatoires

Et bien d'autres classes …

Conversions de types - type wrapper (ou enveloppe)

☝ **Chaque type de base est associée à une classe de type wrapper ou type enveloppe.**

int → Int, float → Float, byte →Byte, long →Long , double →Double, …

les classes wrapper ...

Voir API SUN …

Boolean,
Byte
Integer,
Long,
Float,
Double,
Character,
String

float floatValue()	Retourne l'entier associé à l'Integer comme un float.
String toString (int i)	Retourne un objet String représentant l'int i spécifié.
String toBinaryString (int i)	Retourne une représentation i d'un int non signé en base 2 sous forme d'une String.
valueOf (String s)	Retourne l'objet Integer intégrant la chaîne s.
int parseInt (String s)	Analyse l'argument s et le retourne en int signé.
…	

intValue ()	Retourne la valeur de ce Float comme un int
longValue ()	Retourne la valeur de ce Float comme un long.
parseFloat (String s)	Retourne un float initialisé avec l'argument String spécifiée.
toString (float f)	Retourne la représentation String de l'argument float .
valueOf (String s)	Retourne l'objet Float intégrant le float représenté par l'argument string s.
…	

☞ **Les classes wrapper stockent une valeur du type correspondant, elles offrent des méthodes permettant les tâches de conversion (int ↔ String, float ↔ int, …)**

☞ Grâce à ces classes, on peut par exemple effectuer la conversion d'une chaîne de caractères en nombre entier, un nombre float en chaîne de caractères ou en type entier,

```
public static void main (String args[])
  {
   int i= 100, j;
   char unCaractere = '8';
   String uneChaine = String.valueOf (i);
   String uneAutreChaine = "14503";
   int unEntier = Integer.parseInt (uneAutreChaine);
   int unAutreEntier = Character.digit(unCaractere,10);
   float x = 1.123F;
   String chaineFloat = Float.toString (x);
   float xReverse = Float.parseFloat(chaineFloat);
   Float X = new Float (x);
   int  entier = X.intValue();
   String chaineBin = Integer.toBinaryString(i);

   System.out.println("unCaractere:     "+unCaractere );
   System.out.println("uneAutreChaine:  "+uneAutreChaine);
   System.out.println("unEntier:        "+unEntier);
   System.out.println("unAutreEntier:   "+unAutreEntier);
   System.out.println("chaineFloat :    "+chaineFloat );
   System.out.println("xReverse :       "+xReverse );
   System.out.println("entier:          "+entier);
   System.out.println("chaineBin :      "+chaineBin );
  }
```

Insertion du float x dans un objet Float X en vue de conversion en entier

```
unCaractere: 8
uneAutreChaine: 14503
unEntier: 14503
unAutreEntier: 8
chaineFloat : 1.123
xReverse : 1.123
entier: 1
chaineBin : 1100100
Exit code: 0
No Errors
```

☞ C'est aussi un moyen de promouvoir un type simple en objet Ex: **Float X = Float (1.123);**

Les chaînes de caractères

☞ Les chaînes de caractères sont des **objets** , ils peuvent-être créés selon deux types :

- Le type **String** crée une chaîne constante de caractères [java.lang.String]

- Le type **StringBuffer** crée une chaîne de caractères modifiable [java.lang.StringBuffer]

1 – String

```
java.lang.Object
  |
  +--java.lang.String
```

• Déclarer une chaîne:

```
String machaine = "ceci est une chaine CONSTANTE";
ou bien ..
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

• Passer tous les caractères en minuscules:

```
String c = machaine.toLowerCase();
```

• Passer tous les caractères en majuscule:

```
String c = machaine.toUpperCase();
```

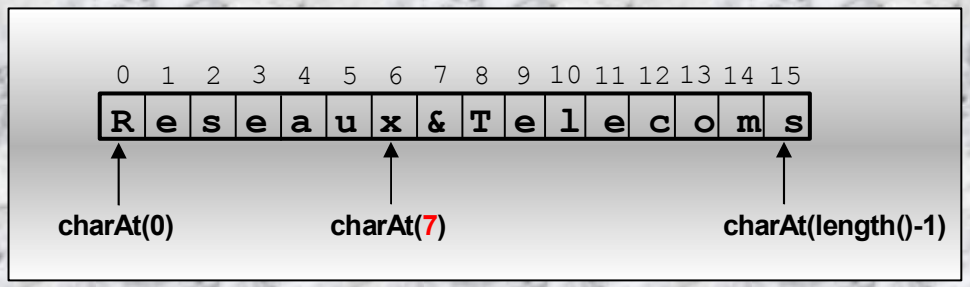
• Connaître la longueur de la chaîne:

```
int taille = machaine.length();
```

• Comparer deux chaînes:

```
boolean c = machaine.equals(autrechaine);
```

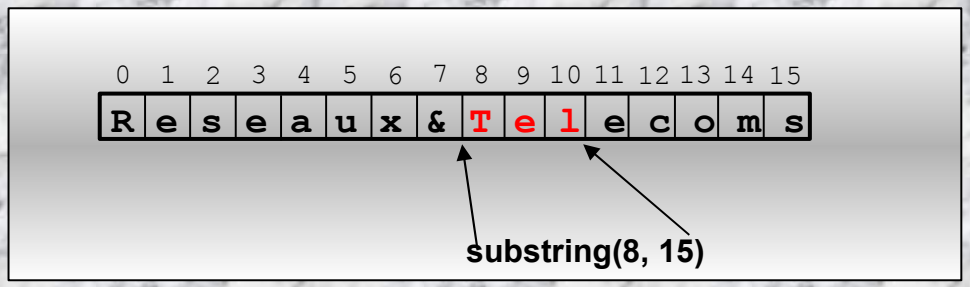
·Récupérer un caractère à un index donné:




```
char c = machaîne. charAt(6);
```

```
String maChaine =  "Reseaux&Telecoms";
char unCar = maChaine.charAt (7);
```

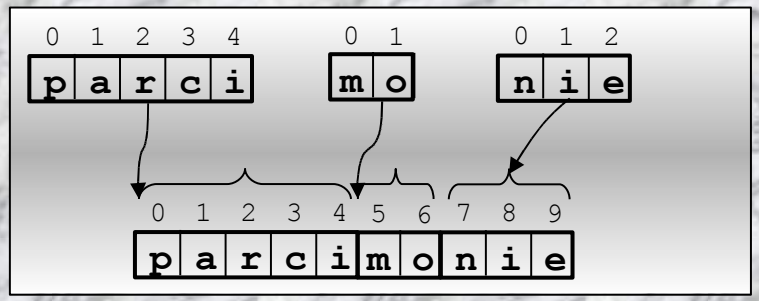
· Retourner une sous-chaîne:




```
String souschaine = machaine.substring(3, 8)
```

```
String maChaine = "Reseaux&Telecoms";
String sousChaine = maChaine.substring (8, 10);
```

• Concaténer avec une chaine:



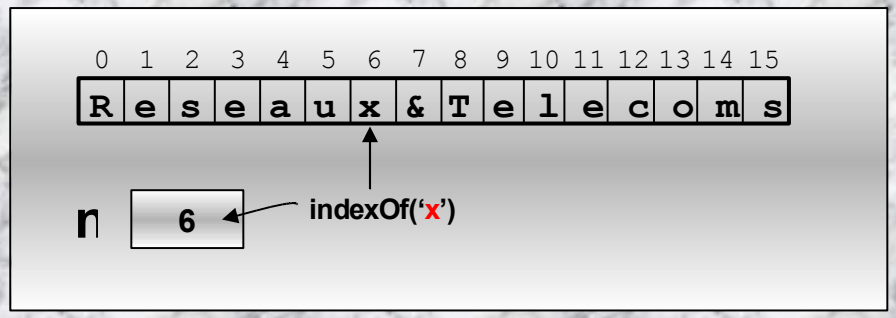

```
String nouvellechaine = machaine.concat(String str)
```

Ou bien…

```
"parci".concat("mo").concat("nie")   //retourne "parcimonie"
```

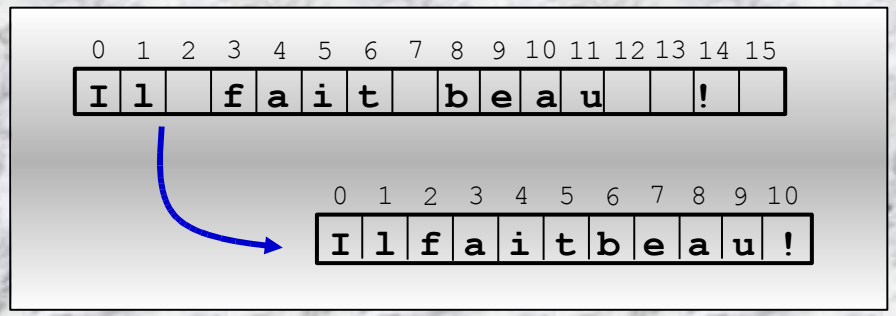
• Trouver dans une string l'index de la 1ière occurrence d'un caractère :

```
int n = machaine.indexOf('x');
```



• Oter tous les caractères d'espace \u0020, TAB ...

```
String c = machaine.trim();
```



• Convertir en chaîne un booléen, un caractère, un entier, un flottant, ...

```
static String valueOf (boolean ou char ou int ou long ou float ou double);

Boolean  b=true;
String unbooleen = valueOf(b)   //this.ValueOf(b) car  méthode static
```

Constructeurs de la classe String

Extrait doc Sun

String() Initializes a newly created String object so that it represents an empty character sequence.

String (byte[] bytes) Constructs a new String by decoding the specified array of bytes using the platform's default charset.

String (byte[] ascii, int hibyte) **Deprecated.** *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the String constructors that take a charset name or that use the platform's default charset.*

String (byte[] bytes, int offset, int length) Constructs a new String by decoding the specified subarray of bytes using the platform's default charset.

String (byte[] ascii, int hibyte, int offset, int count) **Deprecated**. *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the String constructors that take a charset name or that use the platform's default charset.*

String (byte[] bytes, int offset, int length, String charsetName) Constructs a new String by decoding the specified subarray of bytes using the specified charset.

String (byte[] bytes, String charsetName) Constructs a new String by decoding the specified array of bytes using the specified charset.

String (char[] value) Allocates a new String so that it represents the sequence of characters currently contained in the character array argument.

String (char[] value, int offset, int count) Allocates a new String that contains characters from a subarray of the character array argument.

String (String original) Initializes a newly created String object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.

String (StringBuffer buffer) Allocates a new string that contains the sequence of characters currently contained in the string buffer argume

Plusieurs façons d'initialiser une variable String

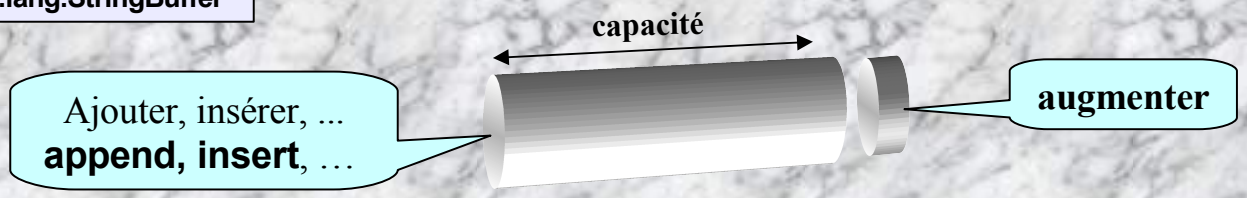
```
Class TestString {  …
    char UnTableau [] = {'a', 'b', 'c', 'd'};
    String s1= new String ()  //string vide
    String s2= new String (" il fait beau ");   //s2 vaut " il fait beau "
    String s3 = new String (s2);   //que vaut s3?
    String s4 = new String (UnTableau);
    String s5 = " l'homme est un prédateur pour l'homme ";
…
```

Les méthodes de gestion de chaînes String …

charAt(int)	renvoie le nieme caractère de la chaîne
compareTo(String)	compare la chaîne avec l'argument
concat(String)	ajoute l'argument à la chaîne et renvoie la nouvelle chaîne
endsWith(String)	vérifie si la chaîne se termine par l'argument
equalsIgnoreCase(String)	compare la chaîne sans tenir compte de la casse
indexOf(String)	renvoie la position de début à laquelle l'argument est contenu dans la chaîne
lastIndexOf(String)	renvoie la dernière position à laquelle l'argument est contenu dans la chaîne
lenght()	renvoie la longueur de la chaîne
replace(char,char)	renvoie la chaîne dont les occurrences d'un caractère sont remplacées
startsWidt(String int)	Vérifie si la chaîne commence par la sous chaîne
substring(int,int)	renvoie une partie de la chaîne
toLowCase()	renvoie la chaîne en minuscule
toUpperCase()	renvoie la chaîne en majuscule
trim()	Ote les caractères non significatifs de la chaîne

2 – StringBuffer

```
java.lang.Object
  |
  +--java.lang.StringBuffer
```



- **Constructeurs d'une chaîne modifiable**

- **StringBuffer()**	**Contruit un buffer pouvant contenir initialement 16 caractères.**
- **StringBuffer**(int *length*).	**Contruit un buffer pouvant contenir initialement *length* caractères.**
- **StringBuffer**(String *machaine*)	**Contruit un buffer contenant initialement *machaine***

- **Ajouter à l'objet StringBuffer un char, boolean, int, long, float, double, String, …**

append(boolean), **append**(char), **append**(char[]), **append**(char[], int, int), **append**(double), **append**(float), **append**(int), **append**(long), **append**(Object), **append**(String)

- **Insérer dans l'objet StringBuffer à l'index *int*: char, boolean, int, long, float, double, String, …**

insert(*int*, boolean), **insert**(*int*, char), **insert**(*int*, char[]), **insert**(*int*, double), **insert**(*int*, float), **insert**(*int*, int), **insert**(*int*, long), **insert**(*int*, Object), **insert**(*int*, String)

Les arguments sont traduit automatiquement en type String avant d'être ajoutés ou insérés.

Exemple:

```
StringBuffer s1 = new StringBuffer ("1 + 1 = ");
StringBuffer s2 = new StringBuffer ("les pneus sont en contact  ");
s1.append (1+1);
s2.append ("avec la route.");
s2.insert (10, "sont les éléments d'un vélo qui ");
System.out.println(s2);
```

Deviner l'affichage de s1, s2?

- **Longueur et capacité de la StringBuffer:**

length()	**Retourne le nombre de char dans StringBuffer.**
capacity()	**Retourne la capacité en nombre de char de la StringBuffer.**

- **Inverser la chaîne de char:**

reverse()	**Inverse l'ordre des caractères dans le buffer**

- **Obtenir un caractère précis:**

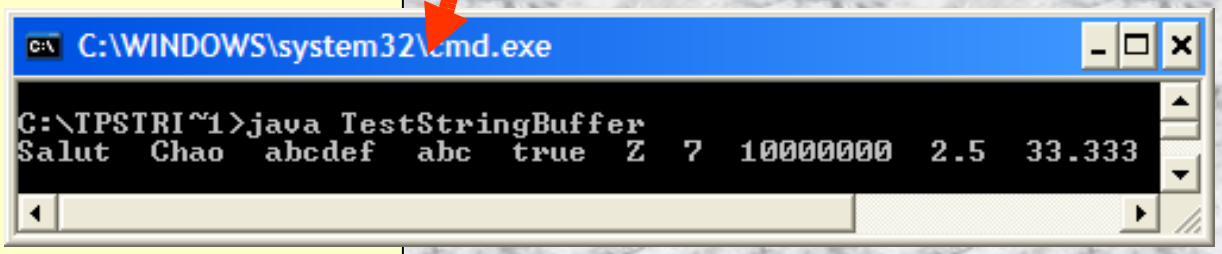
charAt(int)	**Retourne le caractère à l'index spécifié dans la chaîne**
getChars(int, int, char[], int)	**Caractères extraits de la StringBuffer et copiés dans char[]**

- **Insérer un caractère à un index donné:** machaîne. **setcharAt**(1,'a');

Un exemple pour illustrer le côté « fourre-tout » des StringBuffer …

```
public class TestStringBuffer {

public static void main( String args[] ) {
      Object o =  "Salut";
      String s =  "Chao";
      char tableauCar[] = { 'a', 'b', 'c', 'd', 'e', 'f' };
      boolean b = true;
      char c = 'Z';
      int i = 7;
      long l = 10000000;
      float f = 2.5f;
      double d = 33.333;
      StringBuffer buf = new StringBuffer();
      buf.append( o ); buf.append( "  " );
      buf.append( s ); buf.append( "  " );
      buf.append( tableauCar ); buf.append( "  " );
      buf.append( tableauCar, 0, 3 ); buf.append( "  " );
      buf.append( b ); buf.append( "  " );
      buf.append( c ); buf.append( "  " );
      buf.append( i ); buf.append( "  " );
      buf.append( l ); buf.append( "  " );
      buf.append( f ); buf.append( "  " );
      buf.append( d );
   }
}
```

Expérimentons …



Les tableaux

- Les tableaux sont des objets, ce sont des références auxquelles il faut allouer de la place mémoire. Ils sont définis par la syntaxe suivante:

```
char chaine[ ] = new char[24];
```

Le mot new permet d'allouer la mémoire du tableau pour 24 éléments *char*.

- Les tableaux de tableaux se déclarent de la façon suivante :

```
char chaine[ ][ ] = new char[24][3];
```

Taille d'un élément = 3 caractères

24 éléments dans le tableau

- La longueur d'un tableau peut être déterminée par .length :

```
char chaine[][] = new char[24][3];
System.out.println (chaine.length) ; //imprime 24
System.out. println (chaine[0].length) ;   //imprime 3
```

☞ On peut initialiser un tableau à sa création:

```
char montableau[ ] = {'a', 'b', 'd', '1', '2'};
int montableau[ ][ ] = {{0,1},{2,3}};
UnObjet  TableauObjet[ ] = { UnObjet(…) , UnObjet(…) , UnObjet(…) }
```

L'exécution du constructeur initialise l'objet

☞ Les éléments du tableau sont indexés de 0 à (length-1)

```
class Debordement
 {
  public static void main (String argv [])
   {
     int Tableaux [ ] = new int [4];
     System.out.println("Longueur : " + Tableaux.length);
     Tableaux [3] = 3;
     Tableaux [4] = 4;
   }
 }
```

Ce programme va-t-il fonctionner correctement?

Restitution des ressources

le programmeur n'a pas à restituer explicitement les espaces mémoire alloués dynamiquement par l'invocation de la clause new, en effet, le - garbbage collector- (thread de faible priorité) récupère les zones de mémoire qui ne sont plus utilisées lors de l'exécution. En outre, les fichiers ouverts sont automatiquement refermés lors de la fin d'exécution du programme. La méthode destructeur n'est plus vraiment justifiée.

Exemple: On veut initialiser 5 noms dans un tableau Noms[], puis les afficher

Initialisation interne:

```
String Noms[] = { "Durand" , "Dupont" , "Martin" , "Bartin" , "Fallot" , "Maudut" , "Zan"}

  for (int i=0 ; i<=4 ; i++) { System.out.println (Noms[i]); }
```

création du tableau contenant des chaînes de caractères (String)

affichage des noms

Initialisation externe:

```
import java.lang.StringBuffer;


 String Noms[] = new String[5];          //Déclare le tableau de chaînes de caractères
 StringBuffer s = new StringBuffer();  //chaîne modifiable de caractères
 char c;

 for (int i=0 ; i<=4 ; i++) {
     try { while ((c= (char)System.in.read() ) !=  '\n') { s.append(c);} }
     catch (Exception e)  { System.out.println("Erreur: "+ e.toString());}
     Noms[i] = s.toString();  //conversion Stringbuffer->String
  }

  for (int i=0 ; i<=4 ; i++) { System.out.println (Noms[i]); }
```

Ajoute le caractère c à la chaîne s

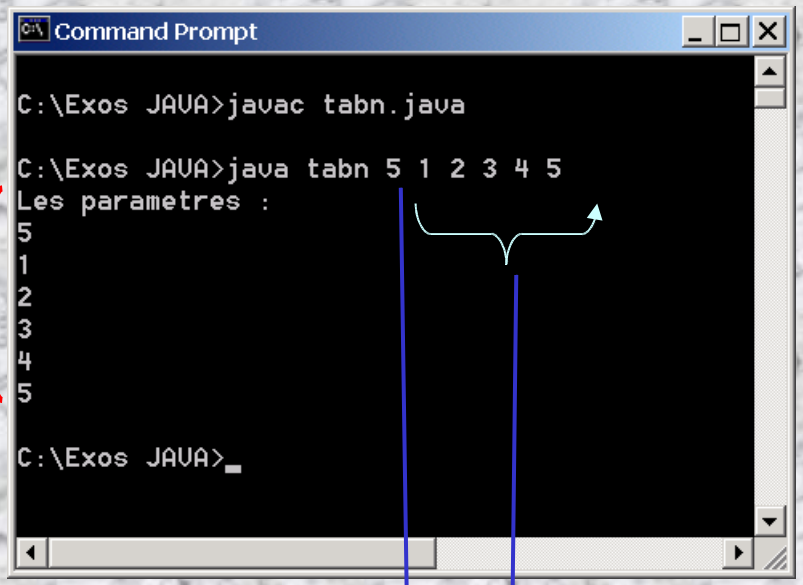
Saisie des 5 chaînes de caractères (String) – codes d'E/S surveillés

affichage des noms

Conseil: Etudier préalablement les méthodes d'E/S: System.in.**read**(…), et System.out.**println** (…), la transmission d'arguments en ligne, les **String**, la classe wrapper **Integer** pour les opérations de conversion.

Exercice:
Construire un programme JAVA **tabn** qui permet de remplir une table de N entier saisis au clavier. Le nombre N, puis les entiers seront saisis en arguments en ligne du programme.

Exécution …



Nombre d'entiers à saisir

Saisie des entiers

Corrigé tabn:

```
public class tabn {
 public static void main (String args[]) {
  int n=args.length;
  if (n==0) {
   System.out.println ("Pas de données!.");
   System.exit(1);
  }
  int tab[]=new int[n];
  for (int i=0; i<n; i++) { tab[i]=Integer.parseInt
(args[i]);}
  System.out.println("Les parametres :");
  for (int i=0; i<n; i++) System.out.println(tab[i]);
 }

}//finclass
```

Le tableau args stockera les arguments en ligne

Nombre d'arguments en ligne saisis

Si le nbr d'arguments saisis est insuffisant, quitter le programme

Créer l'objet tableau tab pour stocker les N entiers saisis au clavier sous forme d'objet String

Convertir les String en entier à l'aide de la classe wrapper Integer

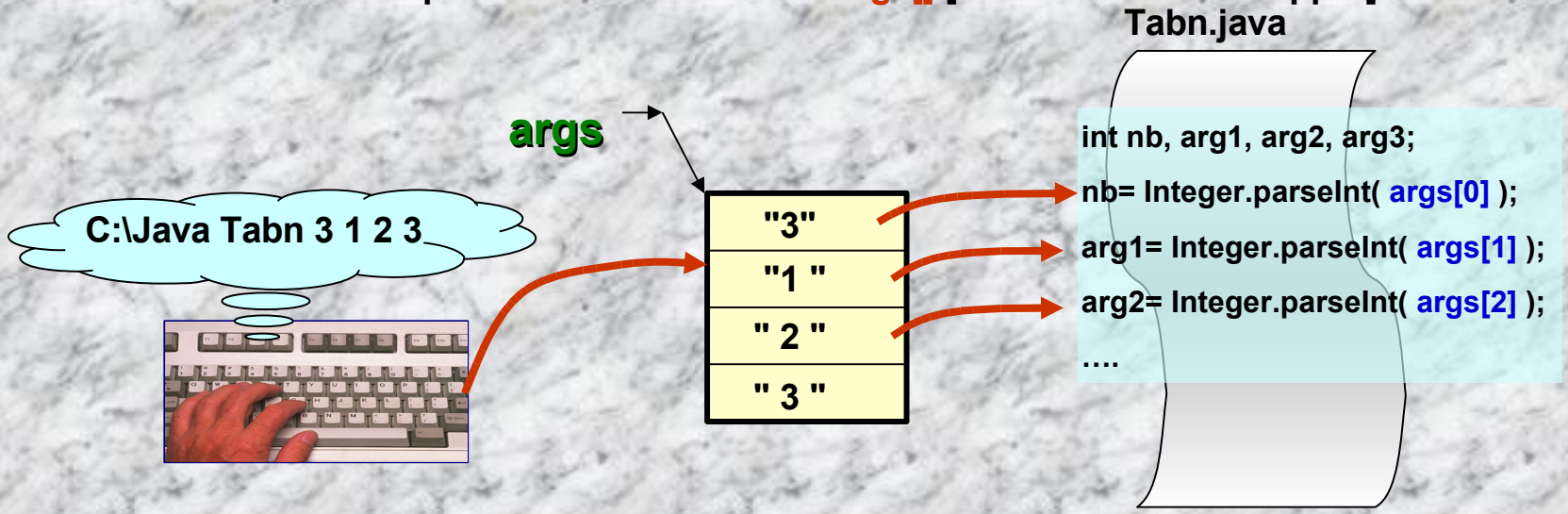
Affichage des entiers saisis en ligne de commande

NOTA: Pour faciliter la programmation JAVA, 2 choses utiles ….

1ière chose: Transmettre des arguments en ligne:

Revenir sur ces aspects au moment des travaux pratiques

- Les arguments peuvent être définitivement inclus dans un programme ou fournis au moment de l'exécution, dans ce cas, on utilise un tableau « passerelle » args[ ].
- Les arguments sont systématiquement traduits sous forme de chaîne de caractères (String). Il est de la responsabilité du programmeur d'effectuer la conversion dans le type adéquat au moment de la récupération dans le tableau args[] [voir les classes wrapper]

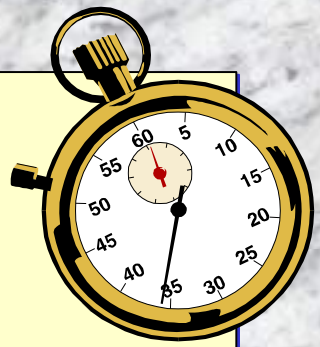


Exécuter ce petit programme pour voir l'effet de la cde: java Echo 1 un 1.1 …

```
public class Echo {
  public static void main (String args[]) {
        for (int i = 0; i < args.length; i++) System.out.println(args[i]); }
}
```

2ième chose: Calculer le temps d'exécution d'un bloc d'instructions ….

Utiliser la méthode système **currentTimeMillis()** présente dans la classe System:



```
class TestTemps {
  public static void main (String args[])
  {
     long cptdeb, cptfin;  // compteurs de ms

        …
    cptdeb = System.currentTimeMillis ();
        for (int i = 1; i <= nbrcpt; i++)
        System.out.println(i);
    cptfin = System.currentTimeMillis ();

    System.out.println («Temps d'exécution:  » + (cptfin - cptdeb) + « ms »);
        …
  }
}
```

Encadrer le bloc d'instructions dont on veut déterminer le nbr de ms écoulé.

Afficher la différence des cpt de ms

Il y a plein de méthodes intéressantes dans la classe System.
Soyez curieux, aller voir!

CLASSES d'OPERATEURS MATHEMATIQUES

Java.lang.Objet
|
+-- java.lang.Math

☝ La class **Math** contient des méthodes pour exécuter de nombreuses opérations de base telles que les opérateurs mathématiques exponentiel, logarithme, racine carré, & fonctions trigonométriques.

1. Math.abs(…), Math.min(…), Math.max(…), Math.floor(…), Math.round(…), Math.ceil(…), …
2. Math.sin(…), Math.cos(…), Math.tan(…), Math.asin(…), Math.acos(…), Math.atan(…), Math.atan2(…), Math.toDegrees(…), Math.toRadians(…), …
3. Math.log(…), Math.exp(…), Math.pow(…), Math.sqrt(…), …

Remplir une table de 12 entiers aléatoires

```
class  TabAleatoire {
   public static void main (String args[])
     {   int matable[ ] = new int[12];
         for (int i = 0; i<matable.length; i++)
           {
             matable[i] = (int)(Math.random ()*100);
           }
     }
}
```

Remplir une table de 256 échantillons de sinusoïde

```
class  TabSinus {
   public static void main (String args[])
     {   int tabsin[ ] = new int[256];
         tabsin[0] = 0;
         for (int i = 1; i< tabsin.length; i++)
           { tabsin[i] = (int)255*Math.sin (2*PI/i);}
     }
}
```

Exemples de méthodes mathématiques:

```
public class TestExponentiel {
   public static void main(String[] args) {
      double x = 11.635;
      double y = 2.76;
      System.out.println("Valeur de e : " +  Math.E);
      System.out.println("exp(" + x + ")  : " +    Math.exp(x));
      System.out.println("log(" + x + ") : " +     Math.log(x));
      System.out.println("pow(" + x + ", " + y + ") : " + Math.pow(x, y));
      System.out.println("sqrt(" + x + ") : " +    Math.sqrt(x));
   }
}
```

```
Valeur de e : 2.71828
exp(11.635): 112984
log(11.635) : 2.45402
pow(11.635, 2.76) : 874.008
sqrt(11.635) : 3.41101
```

```
public class TestTrigo {
   public static void main(String[] args) {
      double degre = 45.0;
     double radians = Math.toRadians (degre);

     System.out.println("Valeur de pi : " + Math.PI);
     System.out.println("sinus de " + degre +  " : " + Math.sin(radians));
     System.out.println("cosinus de " + degre +  " : " + Math.cos(radians));
     System.out.println("tangente de " + degre +  " : " + Math.tan(radians));
    System.out.println("arcsinus de " + Math.sin(radians) + " : " +  Math.toDegrees(Math.asin(Math.sin(radians))) );
     System.out.println("arccosinus de " + Math.cos(radians) + " : " +   Math.toDegrees(Math.acos(Math.cos(radians))) );
    System.out.println("arctangente de " + Math.tan(radians) + " : " +  Math.toDegrees(Math.atan(Math.tan(radians))) );
   }
}
```

```
Valeur de pi: 3.141592653589793
sinus de 45.0 : 0.8060754911159176
cosinus de 45.0 :-0.5918127259718502
tangente de 45.0: -1.3620448762608377
arcsinus de 45.0: NaN
arccosinus de 45.0:NaN
arctangente de 45.0: 1.570408475869457
```

Toutes les méthodes du package Math …

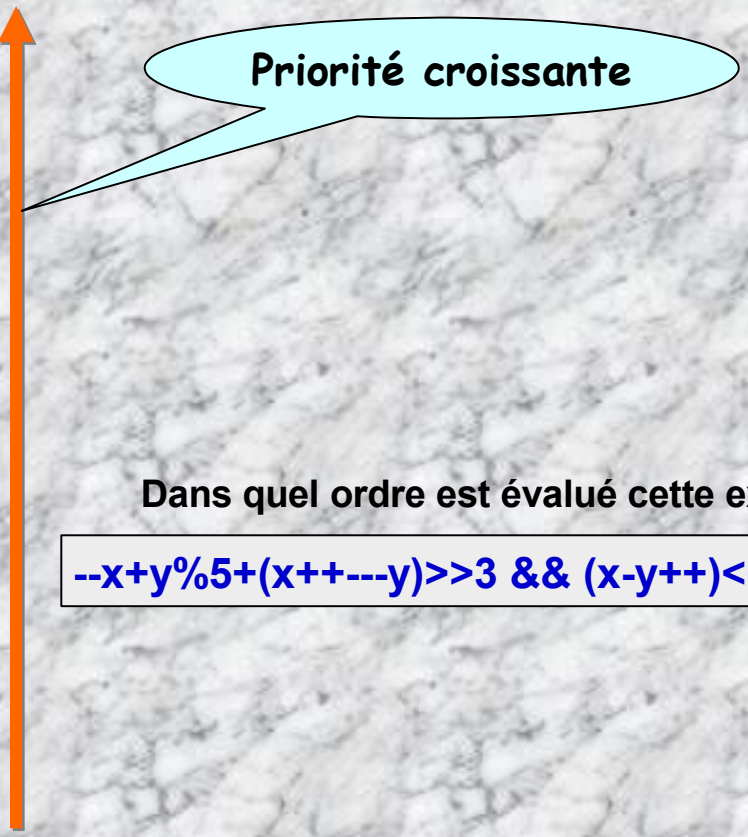
Consulter les APIs Sun pour préciser leur fonctions, leurs arguments, etc

```
static double  abs (double a)
static float   abs (float a)
static int     abs (int a)
static long    abs (long a)
static double acos (double a)
static double asin (double a)
static double atan (double a)
static double atan2 (double y, double x)
static double ceil (double a)
static double cos (double a)
static double  exp (double a)
static double  floor (double a)
static double IEEEremainder (double f1, double f2)
static double log (double a)
static double max (double a, double b)
static float   max (float a, float b)
static int     max (int a, int b)
static long    max (long a, long b)
static double  min (double a, double b)
static float   min (float a, float b)
static int     min (int a, int b)
static long    min (long a, long b)
static double  pow (double a, double b)
static double  random ()
static double  rint (double a)
static long    round (double a)
static int     round (float a)
static double  sin (double a)
static double  sqrt (double a)
static double  tan (double a)
static double  toDegrees (double angrad)
static double  toRadians (double angdeg)
```

Les opérateurs

- Ces opérateurs ont la même signification que leur homonyme en C.
- Les opérateurs par ordre de priorité croissante sont les suivants :

```
.  []  ()
++ -- ! ~  instanceof
*  /  %
+  -
<<   >>   >>>
< >  <=  >=
== !=
&
^
|
&&
||
? :
= op=
,
```



Dans quel ordre est évalué cette expression ??

```
--x+y%5+(x++---y)>>3 && (x-y++)<<3 ? x=100; : x=200;
```

☞ Les chaînes de caractères « comprennent » l'opérateur + qui permet de concaténer deux chaînes et par voie de conséquence l'opérateur += est valide pour les chaînes.

```
String s = " parci " + "monie ";
ou bien …
 s += "usement";
```

Deviner la valeur de s?

☞ L'opérateur instanceof permet de tester si un objet est une instance de la classe passée en paramètre (ou de l'une de ses sous-classes).

```
class voiture { …. }
…
public static void main ()
{
 voiture 2CV;
...
  if ( 2CV instanceof voitures ) ….
…
}
```

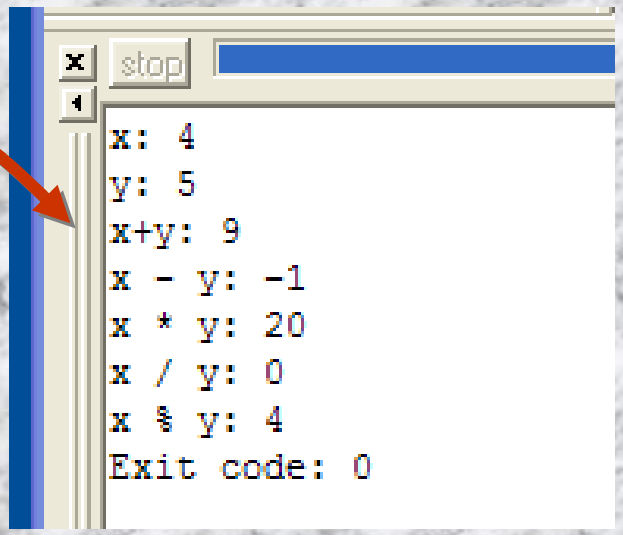
Opérateurs aritmétiques:

c:\java OpArithmétique <arg1 <arg2> [CR]

Avec des entiers ...

```
public class OpArithmétique {


public static void main(String args[])
 {
   int x = Integer.parseInt(args[0]);
   int y = Integer.parseInt(args[1]);
    System.out.println ("x: " + x);
    System.out.println ("y: " + y);
    System.out.println ("x+y: "   + (x + y));
    System.out.println ("x - y: " + (x-y));
    System.out.println ("x * y: "+ x*y);
    System.out.println ("x / y: "+ x/y);
    System.out.println ("x % y: "+ x%y);
 }
}
```

Collecte des arguments de ligne, avec conversion String → int



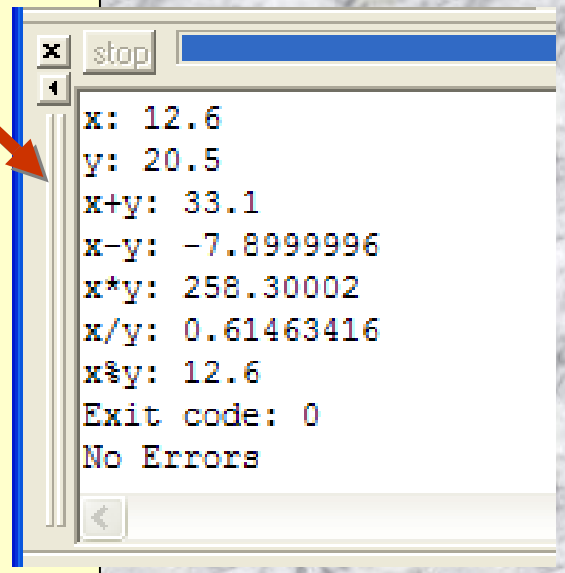
Usage de la plateforme realJ

...

Avec des flottants ...

Collecte des arguments de ligne, avec conversion String → float

```
class OpFlottant
 {
  public static void main (String args[])
   {
    float x = Float.parseFloat(args[0]);
    float y = Float.parseFloat(args[1]);

    System.out.println("x: " + x);
    System.out.println("y: " + y);
    System.out.println("x+y: " + (x + y));
    System.out.println("x-y: " + (x - y));
    System.out.println("x*y: " + x * y);
    System.out.println("x/y: " + x / y);
    System.out.println("x%y: " + x % y);
      }
  }
```



Opérateurs logiques (bitwise):

Collecte des arguments de ligne,

le ET, le OU, le OU exclusif

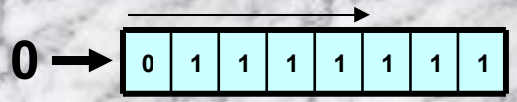
```
class BitaBit
 {
  public static void main (String  args[ ] )
   {
    int x = Integer.parseInt (args[0]);
    int y = Integer.parseInt (args[1]);
    System.out.println (x);
    System.out.println (y);
    System.out.println (x & y);
    System.out.println (x | y);
    System.out.println (x ^ y);
   }
 }
```

le complément

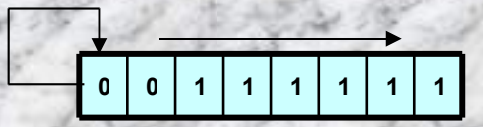
```
class BitaBitComp
 {
  public static void main (String  args[ ])
   {
     int x = Integer.parseInt (args[0]);
     System.out.println(x);
     int y = ~x;
     System.out.println(y);
   }
 }
```

Opérateurs de décalage:

```
class Shift
 {
  public static void main (String  args[ ])
   {
    int x = 7;
    System.out.println("x= "  + x);
    System.out.println("2xdécalage logiques à droite   :" + x >> 2);
    System.out.println("1xdécalage à gauche            :" + x << 1);
    System.out.println( "1 décalage arithmétique droite:" +  x >>> 1);
   }
 }
```



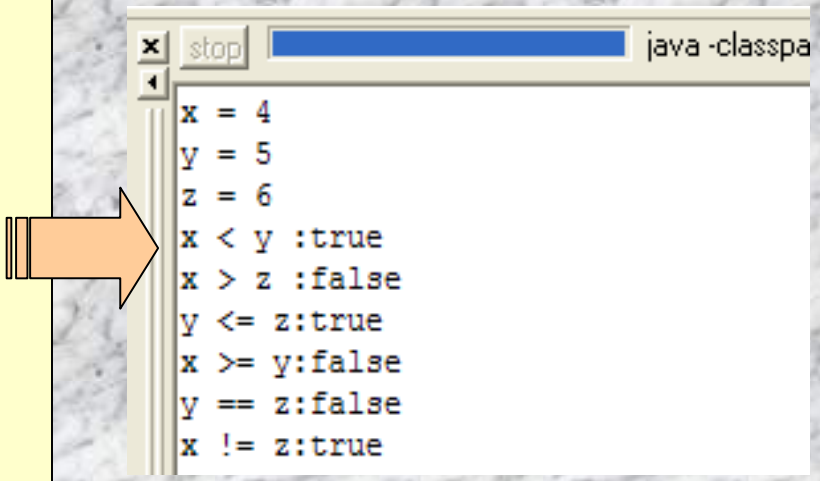
Décalage logique
d'un mot binaire



Décalage arithmétique d'un nombre
signé. Conservation du signe

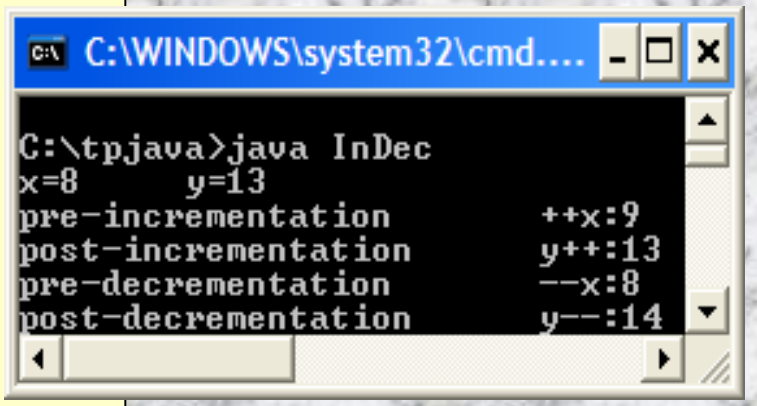
Opérateurs relationnels:

```
class Relation
 {
  public static void main (String  args[ ])
   {
    int x = Integer.parseInt (args[0]);
    int y = Integer.parseInt (args[1]);
    int z = Integer.parseInt (args[2]);

    System.out.print ("x = "        + x);
    System.out.println ( "y = "     + y);
    System.out.println ( « z = "     + z);

    System.out.println ("x < y :"  + (x < y));
    System.out.println ("x > z :"  + (x > y));
    System.out.println ("y <= z:"  + (y<=z));
    System.out.println ("x >= y:"  + (x>=y));
    System.out.println ("y == z:"   + (y==z));
    System.out.println ("x != z:"    + (x!=z));
   }
 }
```



Opérateurs de post/pré incrémentation (décrémentation):

```
class IncDec {
  public static void main (String  args[ ])
   {
    int x = 8, y = 13;
    System.out.println(x); System.out.println(y);
    System.out.println( "pré-incrémentation: " +  ++x);
    System.out.println("post-incrémentation: " +  y++);
    System.out.println("pré-décrémentation: " +  --x);
    System.out.println("post-décrémentation: " +  y-- );
   }
 }
```



Opérateur de concaténation:

```
class Concatenation {
  public static void main (String  args[ ])
   {
     String chaine1 = "Java " + "est ";
     String chaine2 = "presque " + "du C!";
     System.out.println(chaine1 + chaine2);
   }
 }
```

Les instructions de

Les instructions de contrôle sont quasiment les mêmes que celles utilisées en C/C++

IF-THEN-ELSE

if (*expr condit*) {instruction(s);} else {instruction(s);}

```
class IfPrenom {
 public static void main (String args[])
  {
    char Initiale;
    System.out.println("Entrer la première initiale du prénom:");
   try{ Initiale = (char)System.in.read(); } catch(Exception e) {System.out.println("Erreur");}
    if (Initiale == -1) System.out.println ("Prénom curieux!?");
    else if (Initiale == 'j') System.out.println("Votre prénom est Jules?");
    else if (Initiale == 'v') System.out.println("Votre prénom est Vincent?");
    else if (Initiale == 'z') System.out.println("Votre prénom est Zorro?");
    else System.out.println("Impossible de dire votre prénom!");
  }
}
```

Expression conditionnelle ternaire …

On peut utiliser aussi l'expression ternaire suivante:

expression condit ? expr1 : expr2 ≡ if (expression condit) expr1 else expr2

L'expression ternaire est évaluée de la façon suivante :

1°) l'expression <expression condit> est évaluée. Le résultat est *true* ou *false*

2°) Si elle est *true*, la valeur conditionnelle est celle de *expr1*. *expr2* n'est pas évaluée.

3°) Si elle est *false*, la valeur conditionnelle est celle de *expr2*. *expr1* n'est pas évaluée.

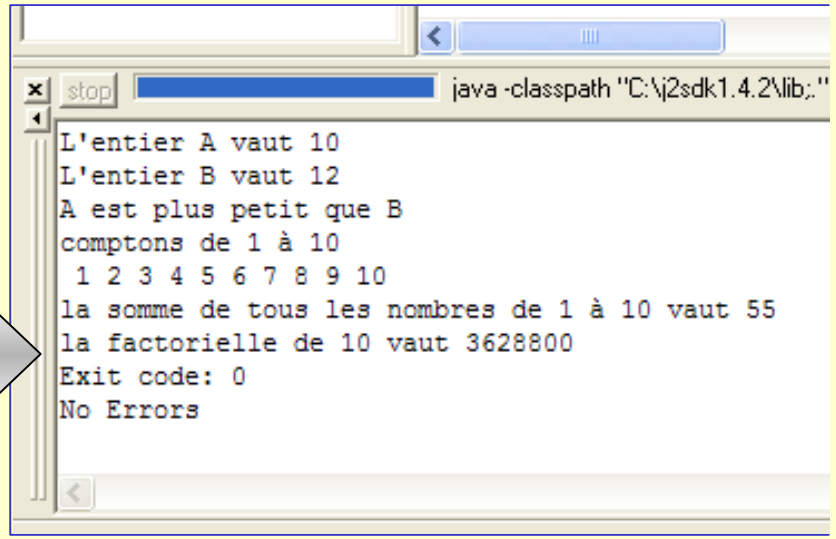
Exemple:

```
int x=-4, y=+2;
(x+3) >0 && (y-3) <0? x++: --y;
```

Quelle est l'expression évaluée?

```
class ExempleSimple {
public static void main (String args[]) {
 int A = 10; int B = 12; int X;
 System.out.println ("L'entier A vaut "+ A);
 System.out.println ("L'entier B vaut "+ B);
 if (A < B) System.out.println ("A est plus petit que B");
  else if (A == B) System.out.println ("A est égal a B");
  else System.out.println ("A est plus grand que B");
 System.out.println ("comptons de 1 à "+ A);
 int somme = 0;
 int fact = 1;
 for (int i = 1;i <= A;i++) {
    System.out.print(" "+i);
    somme += i;
    fact *= i;
  }
 System.out.println();
 System.out.println("la somme de tous les nombres de 1 à "+ A + " vaut "+ somme);
 System.out.println("la factorielle de "+ A+" vaut "+ fact);
 }
}
```

résultats



SUIVANT LE CAS-FAIRE

Un caractère ou un entier

```
Switch ( expr1 )
{
    case cond1: { instruction(s); }; [break;]
    case cond2: { instruction(s); }; [break;]
    ...
    default: { instruction(s); };
}
```

```
class ChoixPrenom {
  public static void main (String  args[ ])
  {
     char Initiale;
     System.out.println("Entrer la première initiale du prénom:");
   try { Initiale = (char)System.in.read();  } catch (Execption e) {System.out.println("Erreur");}
     switch (Initiale) {
          case -1:  System.out.println Prénom curieux!?"); break;
          case 'j':   System.out.println("Votre prénom est Jules!"); break;
          case 'v':  System.out.println("Votre prénom est Vincent!"); break;
          case 'z':   System.out.println("Votre prénom est Zorro!"); break;
          default:   System.out.println("Impossible de dire votre prénom!");
       }
   }
}
```

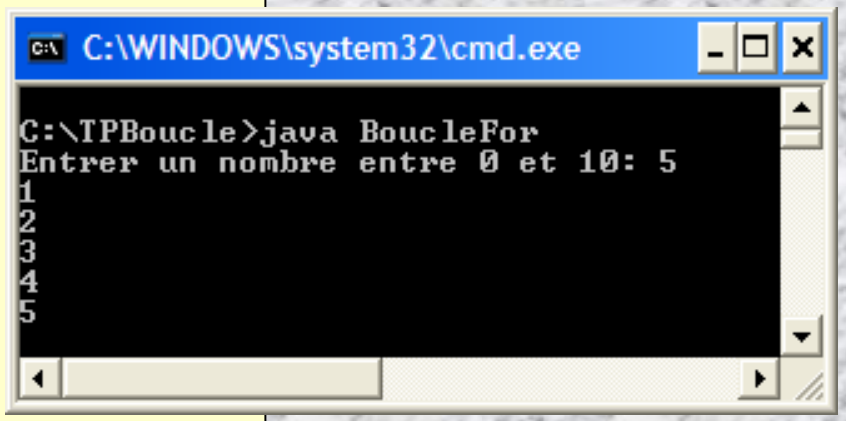
BOUCLE ITERATIVE

for ([*expr1*]; [*expr2*]; [*expr3*]) { *instruction* (s); }


```
class Salut
{
  public static void main (String  args[ ])
  {
    int i;
    for (i = 0; i < 5; i++)
     {
       int k;
       System.out.println ("Salut!");
     }
  }
}
```

Un exemple un peu plus compliqué ...

```
class BoucleFor
{
  public static void main (String args[])
  {
    char n;
    int  nbrcpt;
    System.out.println("Entrer un nombre entre 0 et 10:");
   try {
         n = (char)System.in.read();
        nbrcpt = Character.digit(n, 10);
        } catch (Exception) { //traitement erreur }

   if ((nbrcpt > 0) && (nbrcpt < 10))
    {
      for (int i = 1; i <= nbrcpt; i++)
        System.out.println(i);
    }
    else
      System.out.println("le nombre hors intervalle!");
  }
}
```



TANTQUE-FAIRE

while (*exprcondit*) { *instruction(s)*; }

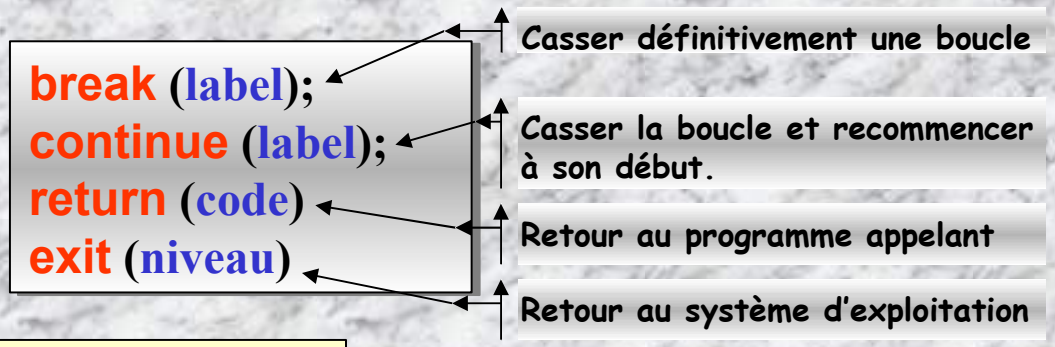
Insérer try-catch, sinon erreur à la compilation

```
class BoucleWhile
{
  public static void main (String args[])
 {
    char input;
    int nbrcpt;
    System.out.println Entrer un nombre entre 0 et 10:");
    try {n = (char)System.in.read(); } catch (Exception e) {System.out.println("Erreur");}
    nbrcpt = Character.digit(n, 10);
    if ((numToCount > 0) && (nbrcpt < 10))
    {
      int i = 1;
      while (i <= nbrcpt) {
        System.out.println(i);
        i++;
      }
    }
    else
      System.out.println( "Nombre hors intervalle!");
  }
}
```

FAIRE- TANTQUE

```
do { instruction(s);  } while ( exprcondit );
```

CASSER LA BOUCLE



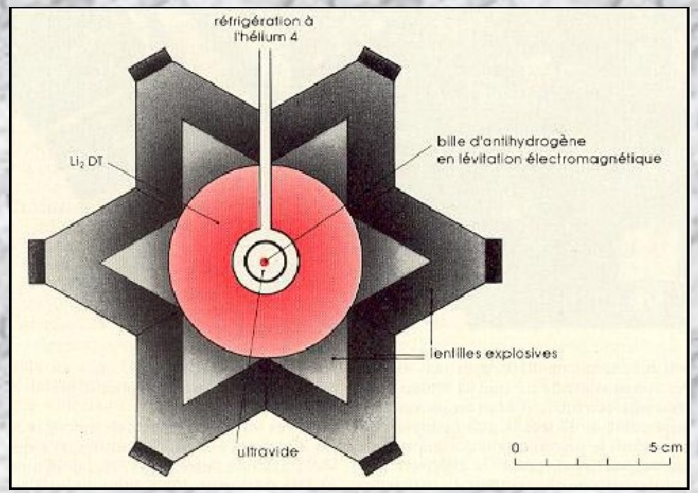

```
class BreakBoucle
 {
  public static void main (String args[])
  {
    int i = 0;
    do {
          System.out.println("Je suis dans la boucle!");
           i++;
           if (i > 100) break;
         }while (true);
  }
 }
```

Une petite pause culturelle …

1932 : Albert Einstein arrive comme réfugié aux États-Unis, après avoir fui le régime nazi en Allemagne (Dieu merci, les américains, toujours très pragmatique, lui ont donné de bon cœur la *« green card »*. Comment la planète eut-elle évoluée si Albert s'était réfugié en France, comme le Recteur Charléty le lui avait proposé trois ans auparavant, lors de son passage à Paris en décembre 1929, pour y recevoir à la Sorbonne le titre de Docteur *honoris causa* ?).

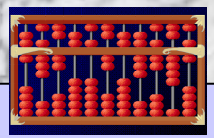
Quelques bonnes phrases d'Albert:

3. *« Il y a 2 choses d'infini, l'Univers, et la bêtise humaine; quoique je n'ai pas acquis une certitude absolue pour … l'Univers! »*
4. *« Asseyez-vous une heure près d'une jolie fille, cela passe comme un minute ; asseyez-vous une minute sur un poêle brûlant, cela passe comme une heure : c'est cela la relativité ».*
5. *« Celui qui est capable de marcher en rang au son d'une musique, alors cette personne je la méprise; c'est pas erreur que la nature l'a doté d'un cerveau, la moelle épinière eut suffit »*



1952 : Explosion de la première Bombe à hydrogène américaine (toujours plus fort, toujours plus haut, toujours plus détonnant, çà décoiffe ! À quand la bombe qui rase gratis ?)

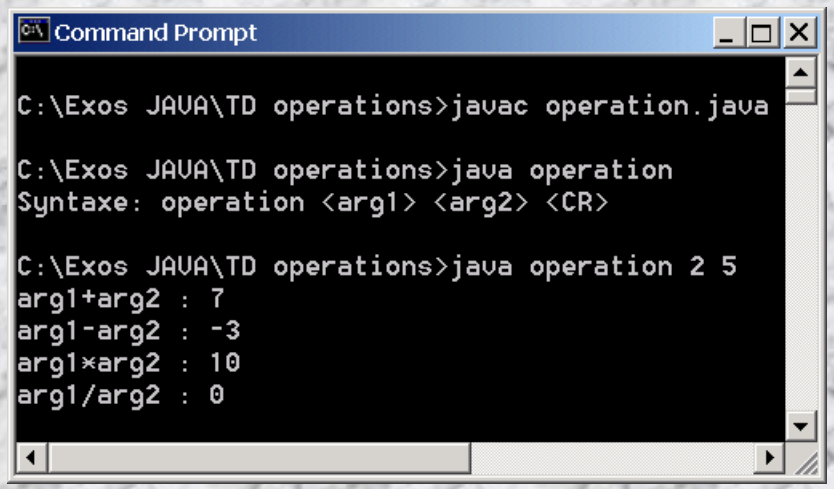
Six exercices:

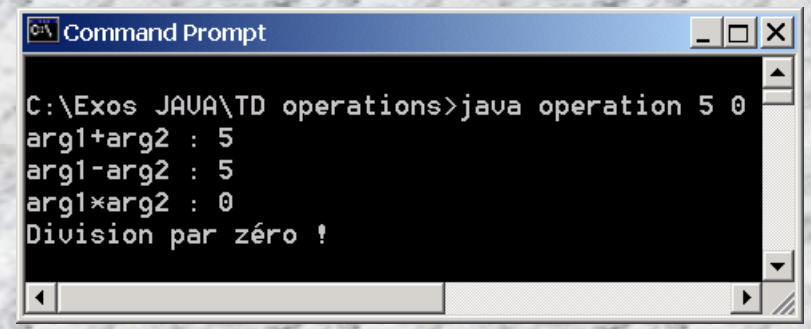


Conseil: Etudier préalablement les méthodes d'E/S: System.in.read(…) , et System.out.println (…), puis la transmission d'arguments en ligne, puis la classe wrapper Integer pour les opérations de conversion.

1. Quatre opérations

Ecrire un programme qui lit deux entiers passés en paramètres sur la ligne de commande et affiche leur somme, leur différence, leur produit et leur quotient. Attention, tester la division par 0.



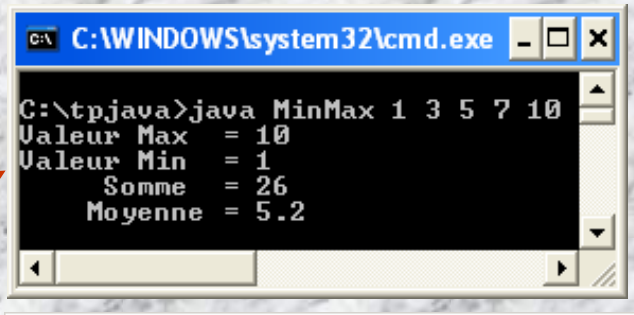


2. Compter de 5 en 5

Ecrire un programme qui compte de 5 en 5 de 0 jusqu'à 100 puis affiche la somme des nombres trouvés. Ecrire deux versions : l'une utilisant une boucle FOR et l'autre utilisant une boucle WHILE. Donner le temps d'exécution en ms pour une valeur final=500 puis 1000. Utiliser la méthode **System.currentTimeMillis()** qui retourne un type long (nbr de ms)

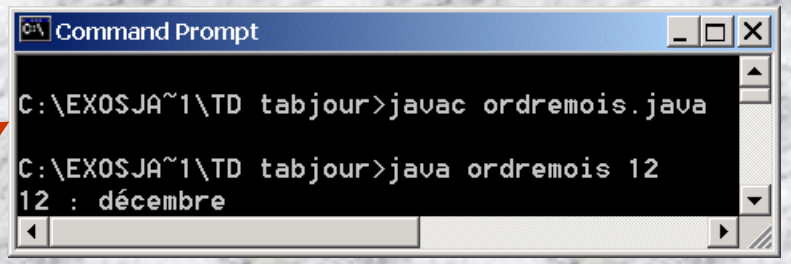
3. Minimum, maximum et somme

Ecrire un programme qui saisie N entiers en arguments de ligne de commande, les insère dans un tableau, puis calcule et affiche le minimum, le maximum, la moyenne, et la somme de ces nombres. Ecrire une variante en donnant seulement N en argument de ligne, et en insérant les N entier à partir du programme.



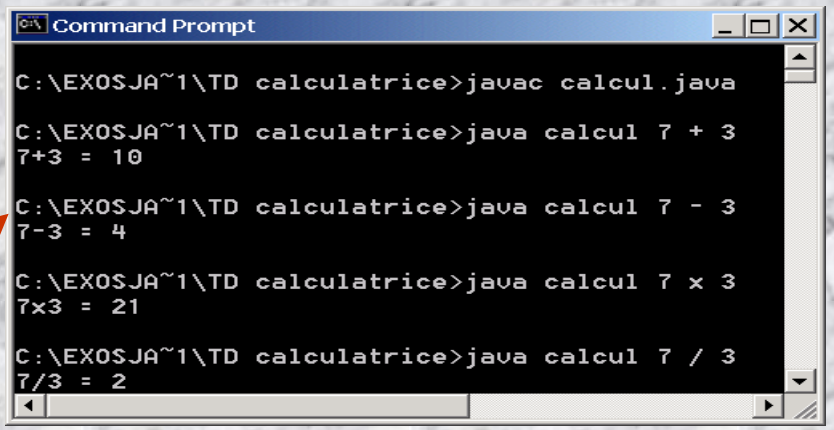
4. Mois

Ecrire un programme qui lit le nombre entier entre 1 et 12 passé en paramètre et qui affiche le nom du mois correspondant. On pourra utiliser un tableau ou l'instruction switch.



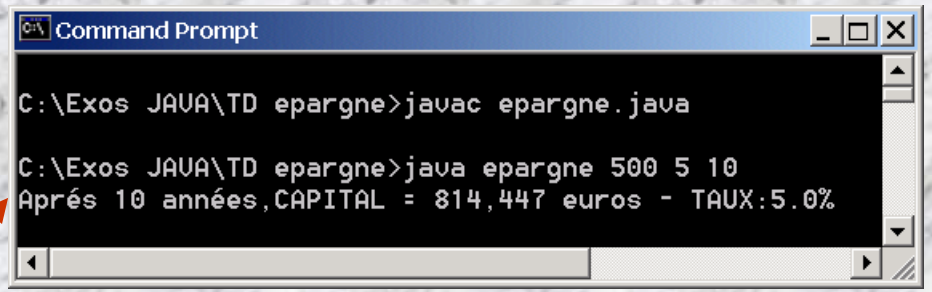
5. Calculette

Ecrire le programme qui lit 3 paramètres : un nombre entier, un caractère représentant l'opération (+,-,x ou /) et un second nombre entier, puis qui effectue le calcul ainsi indiqué et affiche le résultat. (Eviter les divisions par 0). Nota : ne pas utiliser le signe * comme paramètre, il a une signification particulière: référence aux fichiers d'un répertoire.



6. Epargne et intérêts

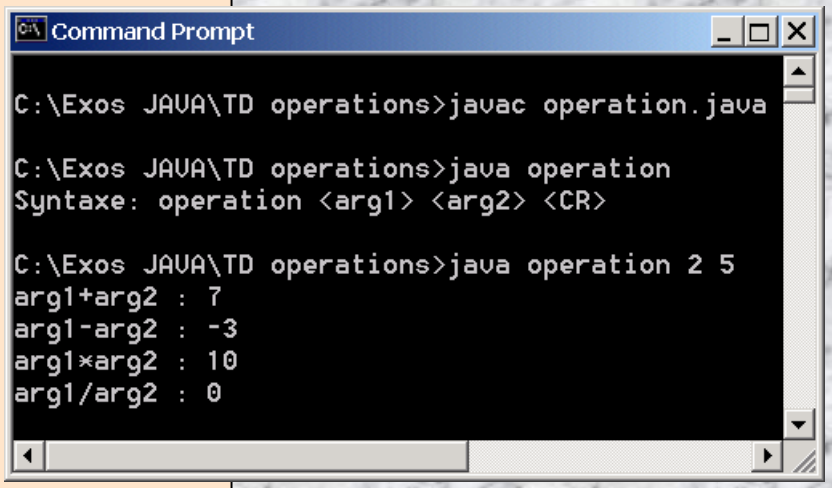
Soit le dépôt d'un **CAPITAL** à la caisse d'épargne; vous désirez savoir son évolution avec un taux d'intérêt composé de **TAUX**/an au bout de 1an, 2 ans, 3 ans, …, 10 ans, **N** années. Ecrire le programme JAVA qui indique cette évolution en indiquant en paramètres sur la ligne de commande: CAPITAL, TAUX, N



Corrigé 1. – 4 opérations

```
public class operation {
 public static void main(String args[]) {
  if (args.length<2) {
   System.out.println("Syntaxe: operation <arg1> <arg2> <CR>");
   System.exit(1);
  }
  else {
   int n0=Integer.parseInt (args[0]);
   int n1=Integer.parseInt (args[1]);
   int som=n0+n1;
   int diff=n0-n1;
   int prod=n0*n1;
   System.out.println("arg1+arg2 : "+som);
   System.out.println("arg1-arg2 : "+diff);
   System.out.println("arg1*arg2 : "+prod);
   if (n1==0) System.out.println("Division par zéro !");
   else {
          int quot=n0/n1;
        System.out.println("arg1/arg2 : "+quot);
          }
  }
 }
}
```

Convertir en integer les données saisies en ligne de cde



Corrigé 2. – Compter de 5 en 5-

```
public class cinq {
 public static void main(String args[]) {
   int somme=0;
   double date0 = System.currentTimeMillis ();
   for (int i=0; 5*i<=1000; i++) {
     somme=somme+ 5*i;
     System.out.println(5*i);
   }
  double date1 = System.currentTimeMillis ();
  System.out.println("Somme : "  + somme);
  System.out.println("Temps d'exécution : "+(date1 - date0)+" ms");
 }
}
```

Section de code dont on veut connaître le temps d'exécution

Retourne le nbr de ms écoulés

On fait la différence entre les deux temps écoulés qui encadre la section de code considérée

Corrigé 3. – Minimum, maximum, somme et moyenne -

```
public class MinMax {

public static void main(String args[])
 {
   if(args.length<1) {
       System.out.println("Syntaxe: java MinMax arg1 arg2 ... argN <CR>");
       System.exit(1);
     }
   int TAB[] = new int[args.length];  //tableau selon nombre d'arguments saisis
   int MAX=0,MIN=0,SOMME=0;
   float MOYENNE=0.0F;
   for (int i=0;i<args.length;i++)  TAB[i]=Integer.parseInt(args[i]);

   for (int i=0;i<args.length;i++)
      {  MAX=MIN=TAB[0];
         if (MAX<TAB[i]) MAX=TAB[i]; //Recherche du maximum
         if (MIN>TAB[i]) MIN=TAB[i]; //Recherche du minimum
         SOMME +=TAB[i];
         MOYENNE = 1.0F*SOMME/args.length; //1.0F force une division réelle
       }
   System.out.println("Valeur Max  = " + MAX);
   System.out.println("Valeur Min  = " + MIN);
   System.out.println("    Somme  = " + SOMME);
   System.out.println("   Moyenne = " + MOYENNE);
 }
}
```

Test de saisie

Insertion des données dans TAB[]

Traitement des données de TAB[]

Forcer une division réelle

Affichage des résultats

Corrigé 3. (Variante) – Minimum, maximum, somme et moyenne -

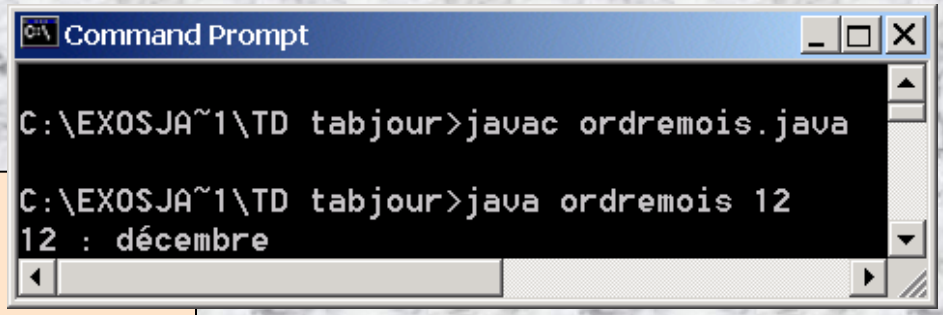
```
import java.io.*;

public class MaxMin2 {
 public static void main(String args[]) {
    int MAX=0,MIN=0,SOMME=0;
    int i=0, N;
    float MOYENNE=0.0F;
    char c;

StringBuffer tmp = new StringBuffer();
   if(args.length<1)  { System.out.println ("Syntaxe: java MaxMin2 arg1 <CR>"); System.exit(1); }
   int TAB[]= new int[N= Integer.parseInt (args[0])];
   System.out.println("Saisir " + N + " entiers:");

   while (i<N)  { //saisie des N entiers
        tmp = new StringBuffer(); //raz tmp
        try {
              while ( (c= (char)System.in.read()) != '\r')   tmp.append(c);
              System.in.read(); // oter caractere '\n'
              TAB[i]= Integer.parseInt(tmp.toString()); i++;
             }
        catch (IOException e) { System.out.println("Erreur1: "+ e.toString()); System.exit(1); }
        catch(NumberFormatException e) { System.out.println("Erreur2: Format numerique incorrect"); }
      }

    for (i=0;i<N;i++){System.out.println("Entier dans TAB["+i+"] ="+TAB[i]);}
    for (i=0;i<N;i++){
          MAX=MIN=TAB[0];
          if (MAX<TAB[i]) MAX=TAB[i];
          if (MIN>TAB[i]) MIN=TAB[i];
          SOMME +=TAB[i];
          MOYENNE = 1.0F*SOMME/N;  //1.0F force une division réelle
       }
    System.out.println("Valeur Max  = " + MAX);
    System.out.println("Valeur Min  = " + MIN);
    System.out.println("     Somme  = " + SOMME);
     System.out.println("    Moyenne = " + MOYENNE);
 }//finmain
}//finclass
```

Récupération du nombre N d'entiers à saisir

Saisie de N entiers

Traitement

Corrigé 4. – Mois -




```
public class mois {
 public static void main(String args[]) {
  int n=Integer.parseInt (args[0] );
  String rep;
  switch (n) {
   case 1 : rep="janvier"; break;
   case 2 : rep="février"; break;
   case 3 : rep="mars"; break;
   case 4 : rep="avril"; break;
   case 5 : rep="mai"; break;
   case 6 : rep="juin"; break;
   case 7 : rep="juillet"; break;
   case 8 : rep="août"; break;
   case 9 : rep="septembre"; break;
   case 10 : rep="octobre"; break;
   case 11 : rep="novembre"; break;
   case 12 : rep="décembre"; break;
   default : rep="Ce n'est pas un numéro de mois";
  }
  System.out.println(n+" : "+rep);
 }
}
```

Corrigé 5. – Calculette -

```
public class calcul {

 public static void main(String args[]) {
  if(arg.lenght<3) {
    System.out.println ("Syntaxe: calcul <arg1> <op> <arg2> CR");
    System.exit(1);
    }
  int n1=Integer.parseInt(args[0]);
  String op=args[1];
  int n2=Integer.parseInt(args[2]);
  int rep=0;
  for (int i=0; i<3; i++) System.out.print(args[i]);
  System.out.print(" = ");
  switch (op.charAt(0) ) {
   case '+' : rep=n1+n2; break;
   case '-' : rep=n1-n2; break;
   case 'x' : rep=n1*n2; break;
   case '/' :
    if (n2==0) {
     System.out.println("Division par zéro !");
     System.exit(1);
    }
    else rep=n1/n2;
    break;
   default :
    System.out.println("Opérateur inconnu.");
    System.exit(2);
  }
  System.out.println(rep);
 }

}
```

Contrôle du nbr d'argument saisis. Retour à l'OS si erreur, avec rappel syntaxe

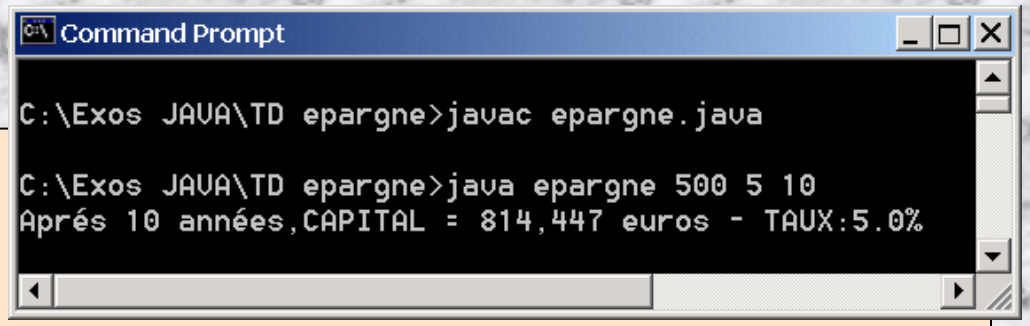
Conversion des arguments saisis sous forme d'objet String en entier à l'aide de la classe wrapper Integer

Calcul des 4 opérations sur les opérandes saisis

Contrôle et gestion des erreurs par retour à l'OS: exit(1) et exit(2)

Affichage du résultat du calcul des 4 opérations sur les opérandes saisis

Corrigé 6 – Epargne et intérêts -




```
import java.text.DecimalFormat;
public class epargne {

 public static void main(String args[]) {
  if (args.length <3) {
    System.out.println("Syntaxe: epargne  <capital> <taux> <nbr ans> CR");
    System.exit(1);
  }
  double capital= Double.parseDouble (args[0]);
  double taux=Double.parseDouble (args[1]);
  int    n=Integer.parseInt (args[2]);
  for (int i=1; i<=n; i++) capital=capital*(1+taux/100.0);
  DecimalFormat f = new DecimalFormat();
  String moncapital = f.format(capital);
  System.out.println("Après "+n+" années,CAPITAL = "+moncapital+" euros"+" - TAUX:"+taux+"%");
```

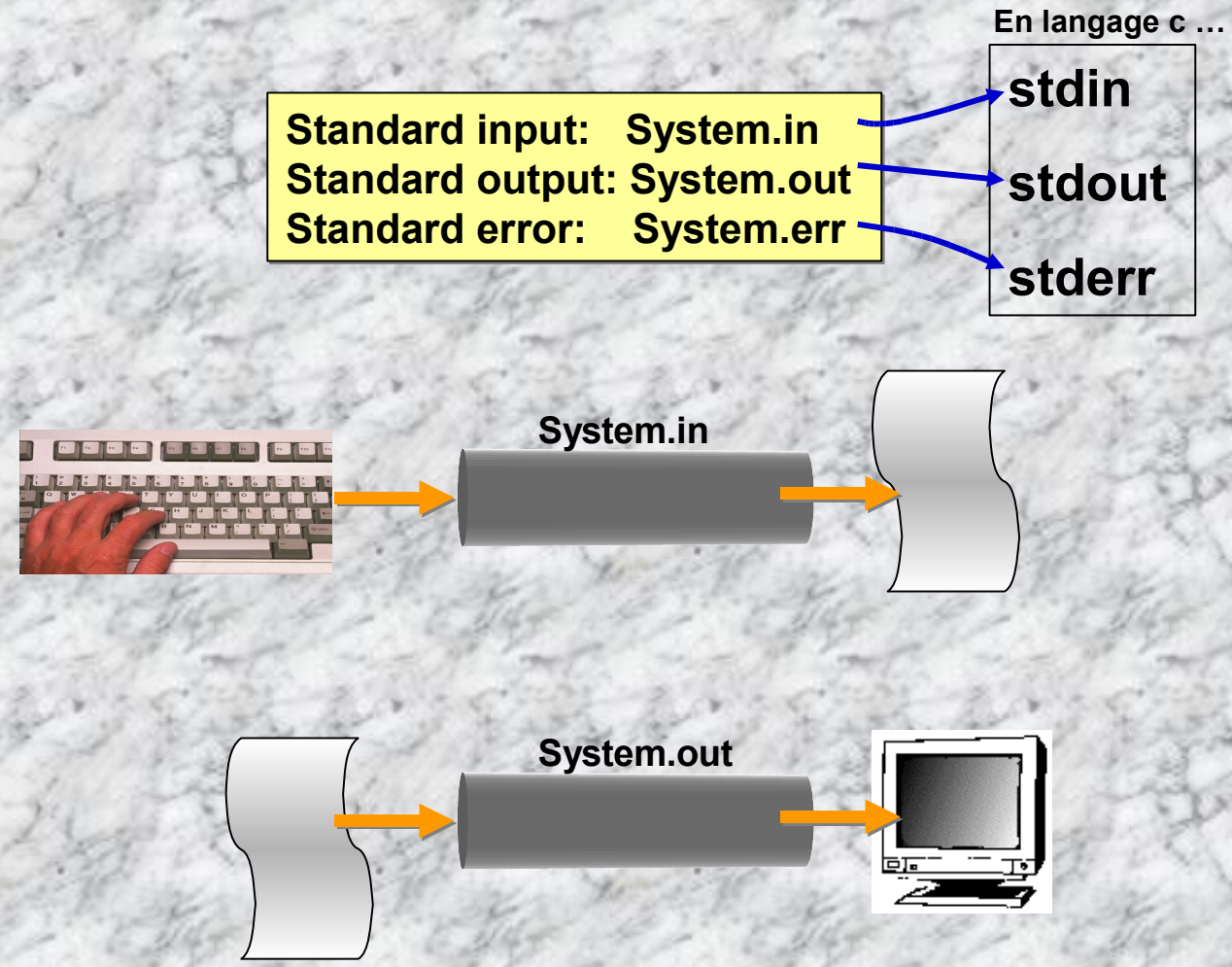
-5-

Les entrées et sortie standards de base – clavier, écran

Dans ce chapitre, nous étudions les entrées/sorties de base: println() et read() de la classe System.

Les entrées et sortie standards de base – clavier, écran

- La classe System fournie trois flux: flux de sortie: out, flux d'entrée: in , flux d'erreur: err. Les entrée-sortie sont définies dans le paquetage de classes java.io.*

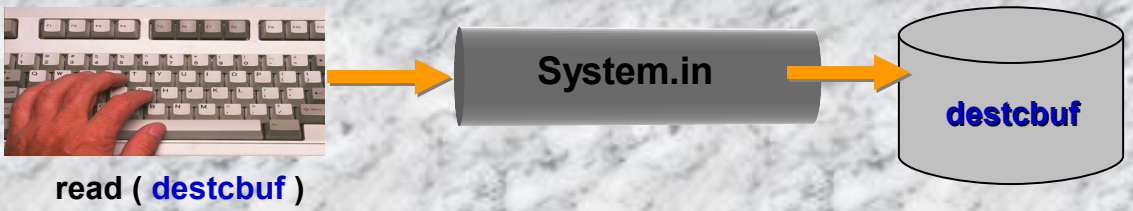


- Les flux in, out, sont en fait deux objets possédant respectivement les méthodes **println** (…) pour afficher les données sur l'écran et **read** (…) pour saisir des données au clavier.

System.out.println(…) et System.in.read (…)

Lire le clavier …

• int **read** ()	Lire un seul caractère.
• int **read** (char[] destcbuf)	Lire des caractères dans le tableau destcbuff.
• int **read** (char[] destcbuf, int off, int len)	Lire *len* caractères du tableau à partir de *off* caractères.



read (destcbuf)

Afficher à l'écran …

Sans retour de ligne:

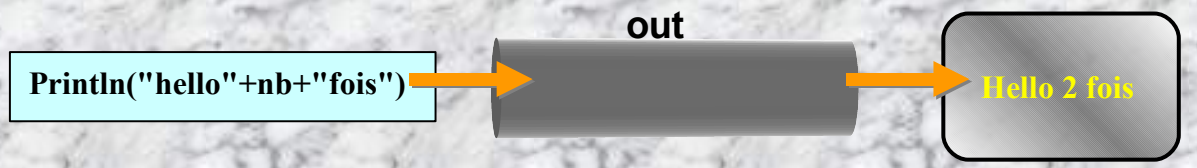
print(boolean b), **print**(char c), **print**(double d), **print**(float f), **print**(int i), **print**(long l), **print**(Object obj), **print**(String s), **print**(char[] s).

Avec retour de ligne:

println(boolean x), **println**(char x), **println**(double x), **println**(float x), **println**(int x), **println**(long x), **println**(Object x), **println**(String x), **println**(char[] x).

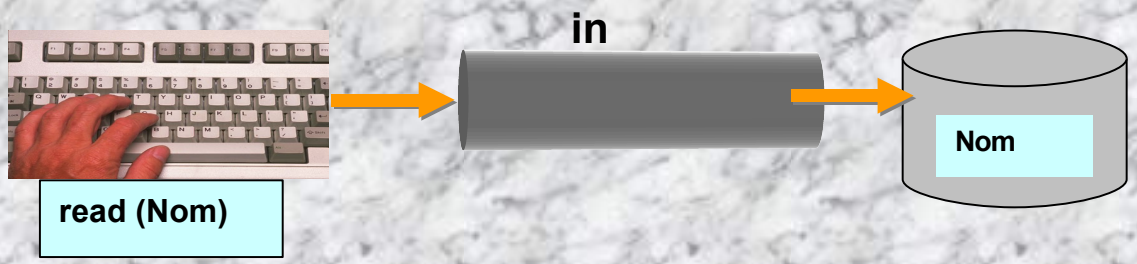
1ier exemple: Introduisons l'incontournable exemple introductif d'introduction

```
import java.io.*;
public class Exemple1
 { public static void main (String args[])
    { int nb = 2;
      System.out.println("hello "+ nb + "fois");}
 }
```



La méthode println n'accepte que des string en arguments et convertit automatiquement tout argument d'un autre type en string.

2ième exemple: effectuons une lecture du clavier




```
import java.io.*;
public class Exemple2
 { public static void main (String args[])
   { byte Nom[] = new byte [100]; //création d'un buffer
      System.out.println("Salut tout le monde!!");
      System.out.println("Alors, votre nom ...?");
      System.in.read (Nom); //entrée des caractères
      System.out.println("Bienvenue " + Nom);
   }
 }
```

Le compilateur nous informe clairement qu'il faut surveiller l'entrée de donnée en captant toutes exceptions issues des entrées-sorties.

Résultat de la compilation:

```
c:\exojava>javac Exemple2.java
exo2.java:8: Exception java.io.IOException must be caught, or it must be declared in the
throws clause of this method.
      System.in.read (Nom); //entrée des caractères
                    ^
1 error
```

3ième exemple: effectuons une lecture du clavier protégée

```
import java.io.*;
public class Exemple3
 { public static void main (String args[])
    { byte Nom[] = new byte [100]; //création d'un buffer
      System.out.println("Salut tout le monde!!");
      System.out.println("Alors, votre nom ...?");
      try
       { System.in.read (Nom); //entrée des caractères
       }
      catch (Exception e)
         { System.out.println("Erreur: "+ e.toString());}
      System.out.println("Bienvenue "+ Nom);
    }
 }
```

Résultat de la compilation:

```
C:\exojava>java Exemple3
Salut tout le monde!!
Alors, votre nom ...?
toto
Bienvenue [B@75dfb47a
```

Cela compile mais il semble y avoir encore un problème! Lequel?

Lecture clavier protégé

4ième exemple: effectuons une lecture du clavier protégée avec un bon usage des string car un byte n'est pas un char (qu'on se le dise!)

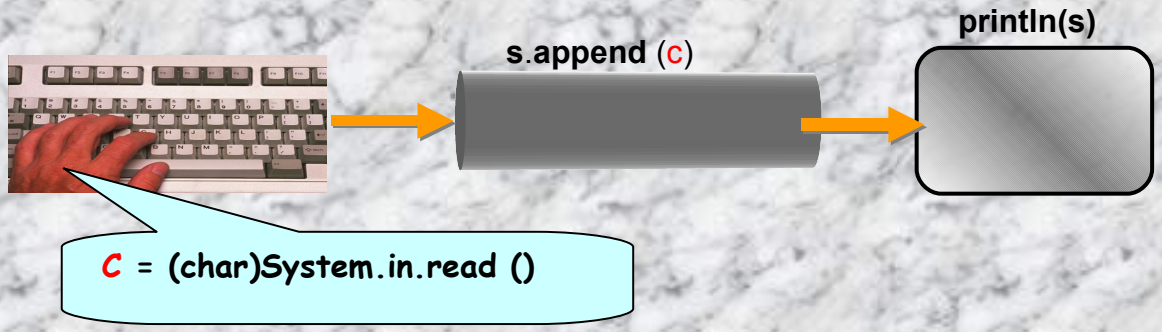
```
import java.io.*;
public class Exemple4
 { public static void main (String args[])
    { byte Nom[] = new byte [100]; //création d'un buffer
      System.out.println("Salut tout le monde!!");
      System.out.println("Alors, votre nom ...?");
      try
       { System.in.read (Nom); //entrée des caractères }
      catch (Exception e)
         { System.out.println("Erreur: "+ e.toString());}

        String s = new String (Nom, 0); //conversion en string
        System.out.println("Bienvenue"  + s);
```

Résultat de la compilation:

```
c:\exojava>javac Exemple4.java
Note: exemple4.java uses or overrides a deprecated API.  Recompile with
"-deprecation" for details.
1 warning
c:\exojava>java exemple4
Salut tout le monde!!
Alors, votre nom ...?
toto
Bienvenue toto
```

Voilà, ça marche! Au passage, on constate que le compilateur version 1.2.2 a constaté (**1 warning**) l'usage de certaines informations obsolètes (**deprecated API**) mais encore utilisables.

5ième exemple: Même exemple mais avec une stringbuffer




```
import java.io.*;
public class Exemple5
 { public static void main (String args[])
    {
      StringBuffer s = new StringBuffer();
      char c;
      System.out.println("Taper n'importe quoi ...");
      try
        {while((c=System.in.read())!= '\n') { s.append(c);}}
      catch (Exception e)
         { System.out.println("Erreur: "+ e.toString());}

        System.out.println(s); //Affiche la chaine saisie
    }
 }
```

Résultat de la compilation:

```
c:\exojava>javac Exemple5.java

c:\exojava>java Exemple5
Taper n'importe quoi ...
je tape, et je tape ... <CR>
je tape, et je tape ...
```

Lire une String au clavier …

```
...
 StringBuffer tmp;
 char C= '\0';
try {
      while ((C=(char) System.in.read()) !='\n')
       { if (C != '\r')   tmp.append(C); }
      String S = tmp.toString(); //conversion StringBuffer->String
   }
catch (IOException e)
     {
      System.out.println("Erreur de frappe");
      System.exit(1);
     }
...
```

Au départ, la chaîne de réception est vide

Lecture caractère par caractère et concaténation avec la chaîne de réception

Traitement de l'erreur, et retour à l'OS

Cette procédure permet de rentrer au clavier tous types de données sous le format d'une String, puis, à l'aide des méthodes des types wapper, procéder à une conversion vers le type initial.

C = (char)System.in.read ()

caractère
entier
réel
chaîne

→ Créer une String S → Convertir S en int, double, float, …

```
Integer.parseInt (S)
Double.parseDouble(S)
Float.parseFloat(S)
…
```

Nouveauté SUN: la classe Scanner …

Depuis l'arrivée de la nouvelle plateforme JAVA version 5.0, labellisée Tiger, SUN a implémenté une nouvelle classe d'analyse des flux: la classe Scanner. Elle permet une analyse et extraction de données au sein d'un flux quelconque d'une manière aisée. En particulier, elle permet de récupérer une valeur saisie au clavier à la manière de la fonction scanf en C. Dorénavant, plus besoin de se développer une classe soi-même pour faire cette tâche.

Par exemple, le code ci-dessous permet à l'utilisateur de lire un nombre à partir du flux System.in

```
Scanner sc = new Scanner (System.in); int i = sc.nextInt();
```

Ou bien Le scanner peut aussi utiliser des délimiteurs autres que des espaces …

```
//Extraction de plusieurs items séparés par le délimiteurs  R&T à partir de la string:
String chaine = "1 R&T 2 R&T FA R&T FI R&T MASTER-DRI";
Scanner s = new Scanner (chaine).useDelimiter("\\s*R&T\\s*");
System.out.println (s.nextInt());
System.out.println (s.nextInt());
System.out.println (s.next());
System.out.println (s.next());
s.close();
```

Résultats de sortie:

```
1
2
FA
FI
MASTER-DRI
```

Saisir au clavier une chaîne avec un Scanner …

```
import java.util.Scanner;   //Classe permettant d'utiliser un Scanner

public class SaisieClavier
{
  public static void main (String [] args){
   Récupération et affichage d'une chaîne de caractère
   //Declaration d'un nouvel objet Scanner
   System.out.println ("Entrez votre nom :");
   Scanner saisie = new Scanner (System.in);
   //Recuperation de la chaine de caractère
   String nom = saisie.nextLine();
   System.out.println("Bonjour " + nom);
  }
}
```

Le Scanner (suite)

-6-

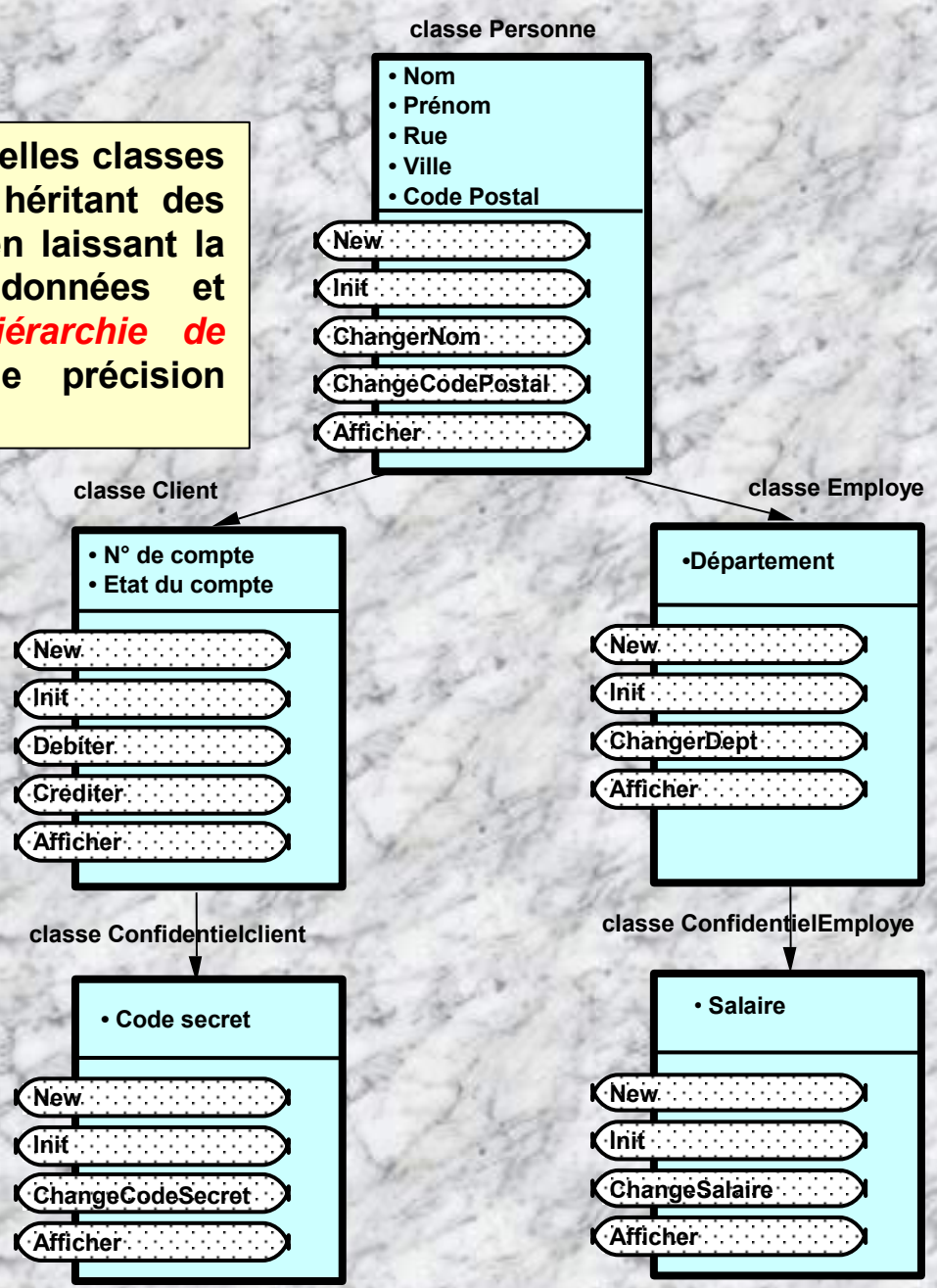
Les classes et objets -Notions avancées-

Java est un langage POO. Comme tous les langages orientés objet, Java offre les notions de classe , d'objet , d'héritage, de polymorphisme, de droits d'accès aux données et méthodes.

Creusons ces nouvelles notions …

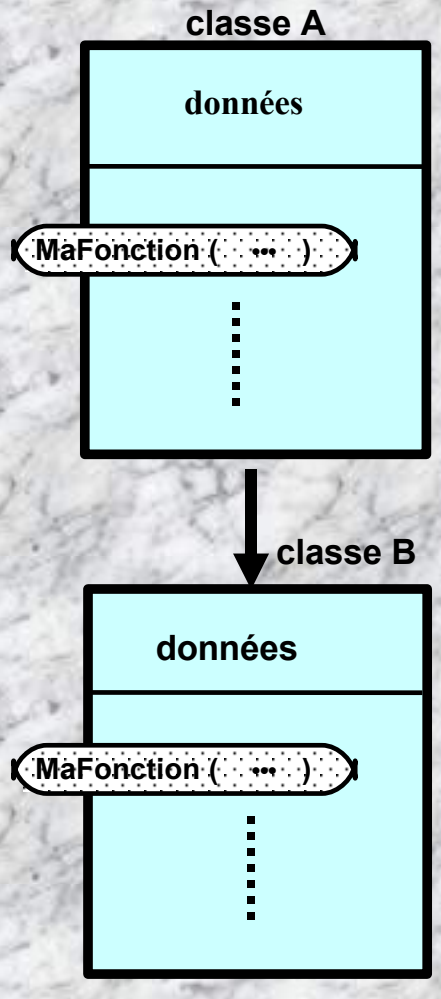
L'héritage

L'héritage permet de construire de nouvelles classes dérivant d'une classe préexistante et héritant des données et méthodes de celle-ci tout en laissant la possibilité d'ajouter de nouvelles données et fonctions. L'ensemble forme une *hiérarchie de classes*. Chaque niveau amène une précision croissante.



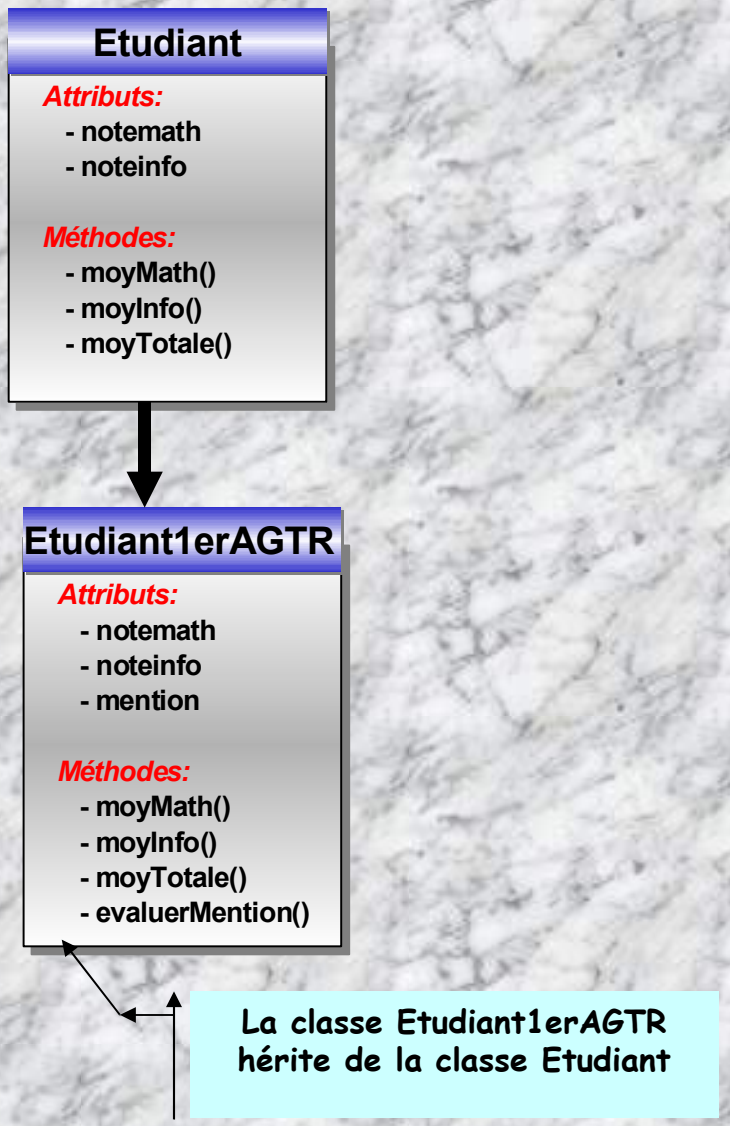
Le polymorphisme

Consiste à donner un même nom à une méthode qui est ensuite partagée à plusieurs niveaux d'une même hiérarchie de classes, chaque classe dans la hiérarchie exécutant cette fonction d'une manière qui lui est propre.



Implémentation de l'héritage

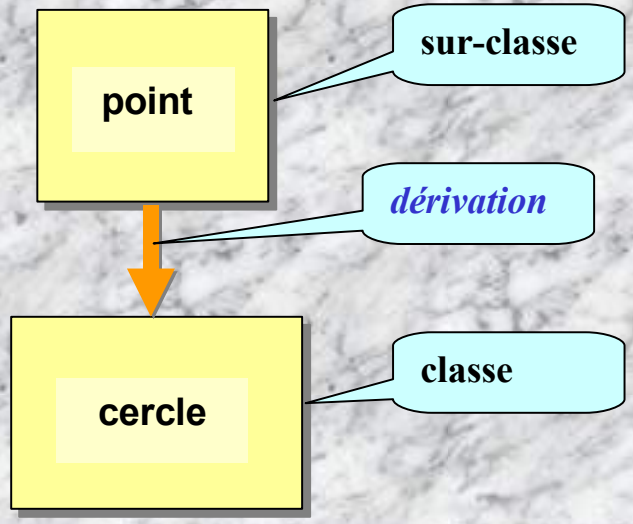
```
class Etudiant {
    public float notemath;
    public float noteinfo;
    public void moyMath (float unenote) {
    ...
    }
    public void moyInfo (float unenote) {
    ...
    }
    public float moyTotale () {
    ...
    }
 }// fin classe Etudiant

public class Etudiant1erCycle extend Etudiant {
    public String mention;
    public String evaluerMention() {
    ...
    }
    public static void main (String[ ] args)  {
      Etudiant Pierre, Paul, Jacques;
    ...
    }
}// fin classe Etudiant1erCycle
```



L'héritage - Définitions

- Une classe est toujours construite à partir d'autre classe dont elle est dérivée. Une classe dérivée est une sous-classe d'une sur-classe.
- La déclaration de la dérivation d'une classe de base se fait de la façon suivante :

```
[Modificateur] class Nom-classe  extends Sur-classe  [implements Interface {interface}]
    {
        <Corps de la classe>
    }
```




```
class point  {
                    float x, y;
                    setX(int)
                    setY(int)
                }
```

```
class cercle extends point
 {  cercle ();
    deplace (int a, int b);
    fillcolor (int couleur);
     ....
 }
```

c1, instanciation de la classe cercle:

```
cercle c1 = new cercle();
```

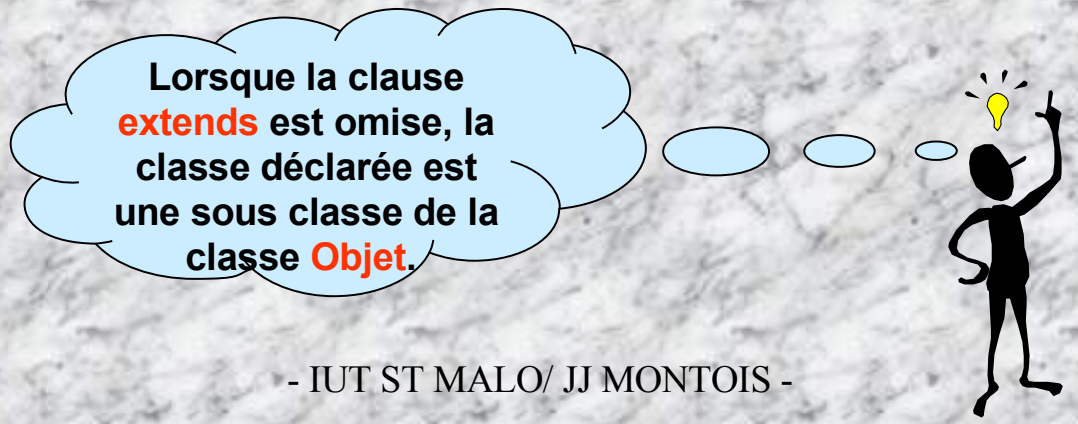
… hérite des attributs et méthodes suivantes:

```
x, y;
setX(), setY(),
deplace(…)
fillcolor(…)
```

Exemple:

```
public class Menu
 {
   ...
 }

class SousMenu extends Menu implements Printable
{
   public void print()
   {
      ...
   }
}
```

⚠ **Toute classe est une sous-classe d'une classe, la sur-classe la plus élevée est la classe Objet.**

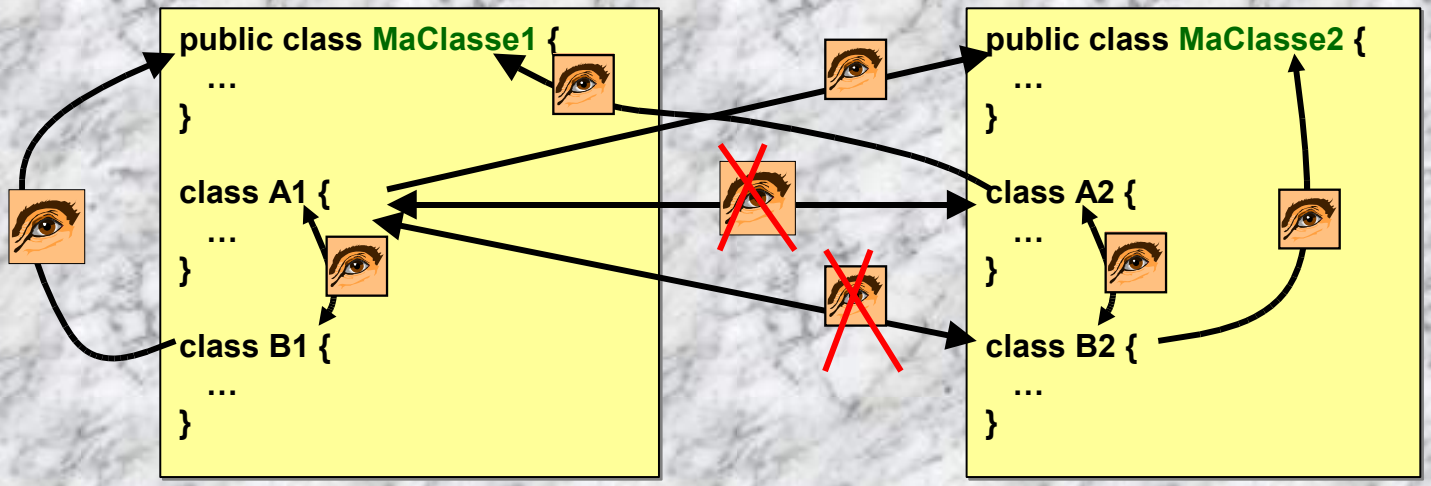


modificateur d'accès

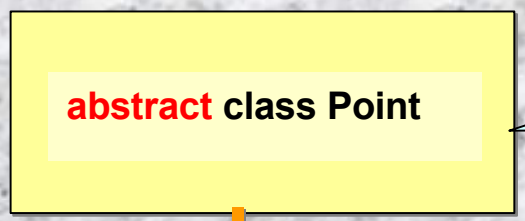
☞ Une classe posséde des droits d'accès spécifiés en suffixe par un modificateur d'accès:

Modificateur	Commentaires
rien	Classe visible par toutes les classes du même paquetage mais invisible dans un autre paquetage
public	Classe visible par toutes les classes, dans tous les paquetages : public MaClasse { ... }
final	Classe ne peut être dérivée : final Color couleurPoint = new Color(0,0,0) ;
abstract	La classe contient des variables, constantes et méthodes de base. C'est le socle d'une dérivation d'autres classes. Elle ne peut être instanciée, elle doit être sous-classée

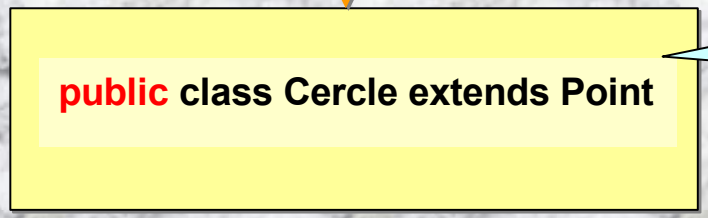
- Une classe final ne peut être dérivée.
- Une variable final ne peut être modifiée.
- Une méthode final ne peut être redéfinie!



Un fichier *.java peut contenir plusieurs classes, cependant, une seule classe au plus peut-être publique



La classe point ne peut être instancié (abstract). Elle sert à créer un concept initial qui va s'enrichir par dérivation. C'est le socle d'une famille de classes



La classe cercle est un ensemble de points … D'autres classes pourront utiliser cette construction (public) à partir d'instanciations ou dérivations.

Exemple:

```
abstract class Point  {
                    float x, y;
                    setX(int)
                    setY(int)
                  }
```

```
class Cercle extends Point
 {  cercle ();
    deplace (int a, int b);
    fillcolor (int couleur);
     ....
 }
```

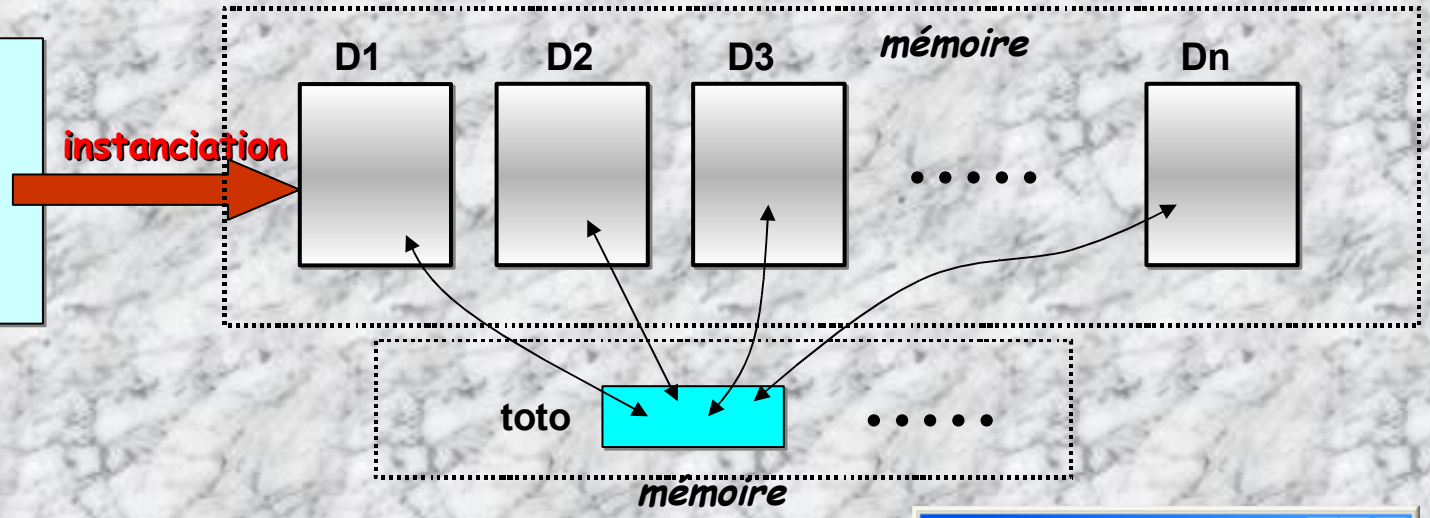
~~Point p1;~~  *Non, Point est asbrait, il ne peut être instancié!*

Cercle c1, c2, c3;  *Oui!*

Variable static, méthode static

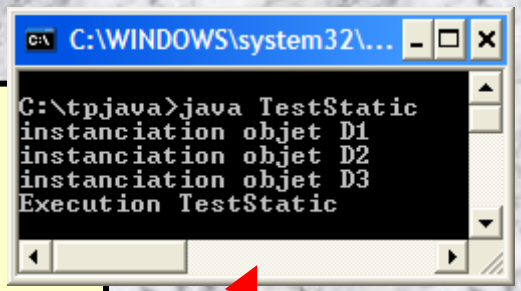
- Une variable déclarée static dans une classe n'est plus une variable instanciée, elle n'appartient qu'à la classe, pas aux objets créés qui peuvent cependant y avoir accès.
- Une variable static n'est copiée qu'une seule fois en RAM, elle est commune à tous les objets créés

```
class UneClasse {
   static int toto;
     ...
     ...
}
```




```
public class UneClasse {
 static int toto=1;
  UneClasse ()
   {  toto++; }
}
```

```
public class TestStatic {
  UneClasse D1, D2, D3;
 public static void main ()
  {
    System.out.println (« instanciation objet D » + toto );
  }
}
```



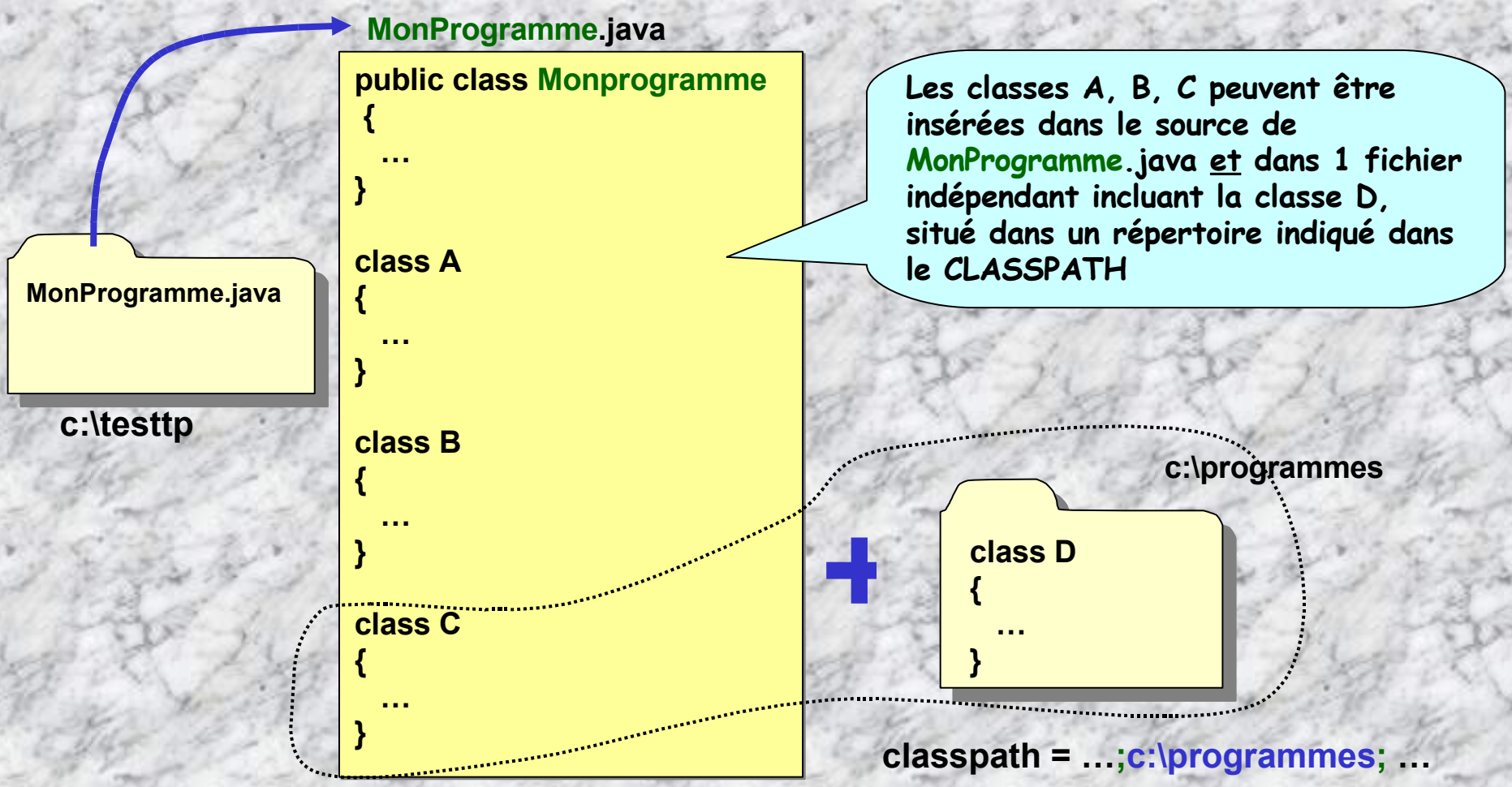
- Une méthode static peut être appelée sans référence objet:

  maMethode() écrite sans objet en préfixe signifie en fait this.maMethode();

- Une méthode static comme exit() appartenant à la classe System peut-être appelée comme suit: System.exit(); ou bien tout simplement exit();

Plusieurs classes dans un fichier *.java

☞ Il peut y avoir plusieurs classes dans un fichier java mais une seule classe public par fichier et son nom doit être le même que le fichier source.



Règle de portée dans un fichier *.java

☞ **Les règles de portée déterminent l'endroit où est reconnue une variable au sein d'un programme**

Variable globale:	Variable reconnue tout au long du programme
Variable locale:	Variable reconnue uniquement dans le bloc de code où elle est délarée

Règle générale

Une variable déclarée dans un bloc de code n'est visible que dans ce bloc et dans les blocs qui y sont imbriqués

```
class Portee
{
  int x = 0;
  void methode1()
   { int y;
     y = x;    //y vaut 0
   }
void methode2()
   { int z = 1;
     z = y;   //incorrect!
   }
```

x est visible dans toute la classe y compris les méthodes

y est locale à methode1

y est invisible de methode2

Fonctions et méthodes

- Les fonctions ont la même syntaxe qu'en C.
- Les arguments d'un type de base (int, double...) sont passés par valeur alors que les objets sont passés par référence.
- En Java, les pointeurs n'existent pas.
- Les fonctions appartenant à un objet sont appelées des méthodes.

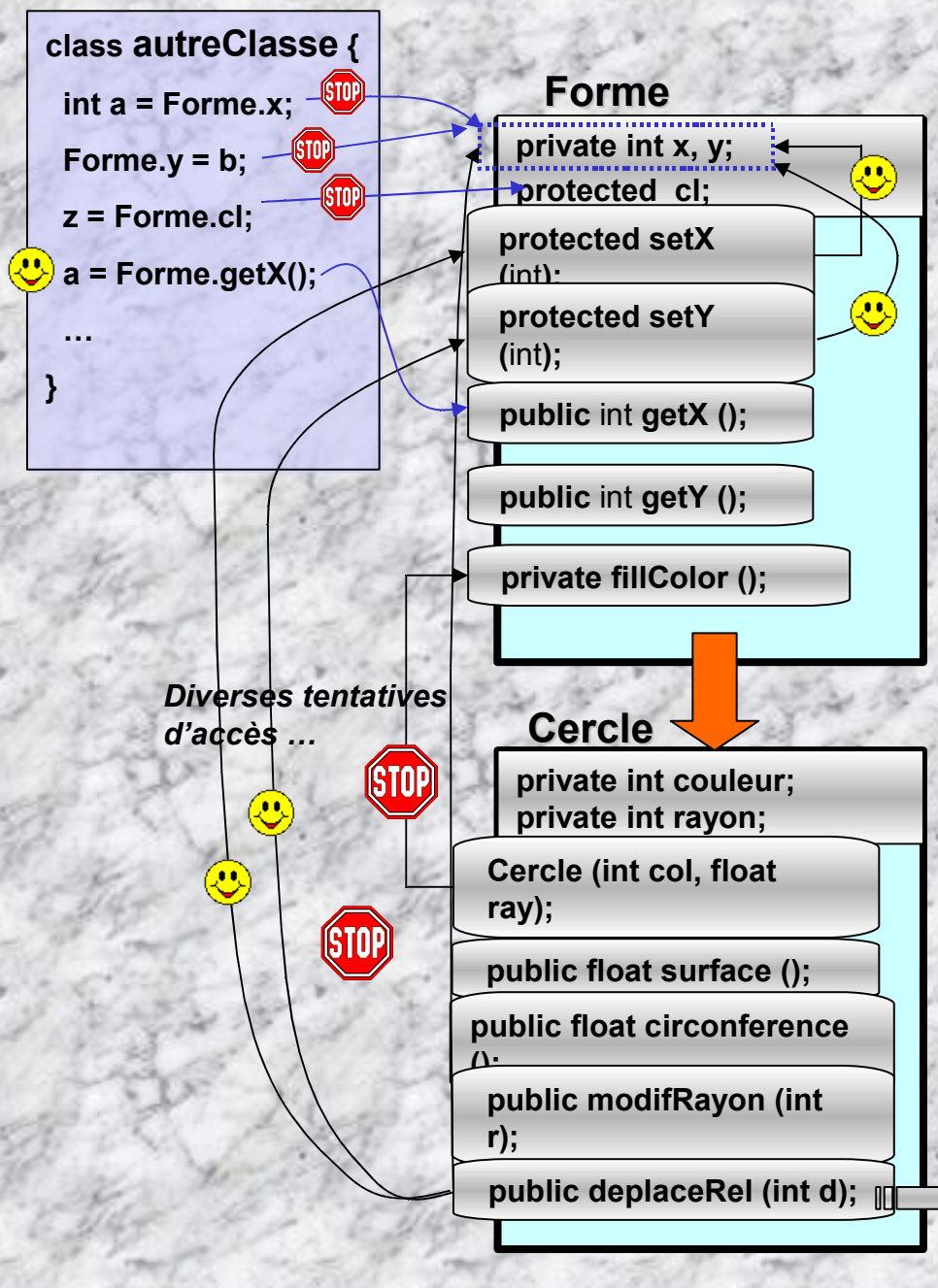
Une déclaration de méthode est de la forme suivante:

```
[Modificateur] type_retourné nomMethode ( arguments, ...)
    {
      <Corps de la méthode>
    }
```

Accessibilité d 'une méthode (ou d 'une variable membre)

Notion de droits d'accès

public	Méthode accessible à l'extérieur de la classe
private	Méthode visible uniquement dans la classe où elle est définie
protected	Méthode visible uniquement dans la classe où elle est définie et dans ses sous-classes



✓ Les attributs x, y (private) ne peuvent être accédés hors de la classe Forme.

✓ L'attribut cl (protected) ne peut être accédé que par les classes dérivées

* Après l'instanciation de l'objet c1:

Cercle c1 = new Cercle (5, 1.2);

c1 hérite des ressources attributs + méthodes suivantes:

couleur, rayon, cl;

- setX() et setY();
- getX() et getY();
- surface ();
- circonference();
- modifrayon();
- deplaceRel();

mais …

```
int a = c1.getX() + d;
int b = c1.getY() + d;
c1.setX(a); c1.setY(b);
```

On récapitule …

Accessibilité d'une méthode, d'une variable:	par défaut	public	private	protected
Depuis la même classe	Oui	Oui	Oui	Oui
Depuis une classe fille du même paquetage	Oui	Oui	Non	Oui
Depuis une classe n'appartenant pas au même paquetage	Non	Oui	Non	Non
Depuis une classe du même paquetage	Oui	Oui	Non	Oui
Depuis une sous classe n'appartenant pas au même paquetage	Non	Oui	Non	Oui

Le mot clé super

Exécuter le constructeur de la classe mère

```
abstract class Forme {
int posx, posy;
…
public Forme (int x, int y) {
  posx=x; posy=y;
 }
 …
}//fin classe
```

```
class Cercle extend Forme {
 int rayon;
 …
 public Cercle (int x, int y, int r) {
 super (x,y);
 rayon = r;
 …
 }//fin classe
```

recherche le constructeur dans la hiérarchie

Exécuter une méthode de la classe mère

```
abstract class Forme {
int posx, posy;
…
public Afficher () { …}
 …
}//fin classe
```

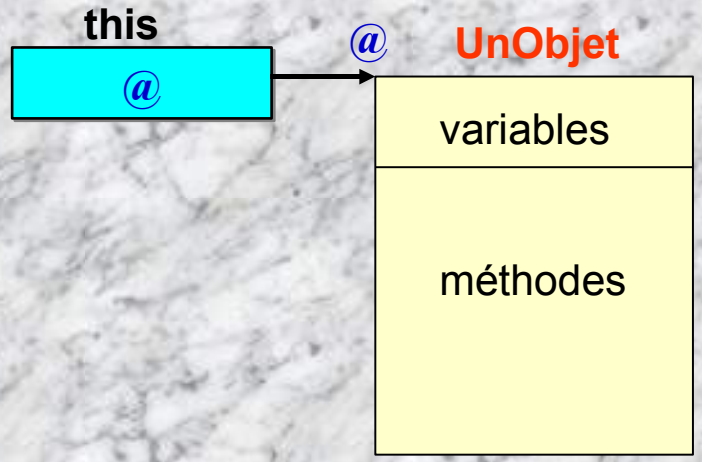
```
class Cercle extend Forme {
 int rayon;
 …
 public Afficher () { …}

 super.Afficher ();
    …
 }//fin classe
```

recherche Affiche dans la hiérarchie

Le mot clé this – Référence de l'instance courante

this est un « pointeur » qui permet de référencer l'objet sur lequel s'exécute une manipulation de variables ou de méthodes

1ère utilisation:



Invoquer this provoque l'exécution du constructeur correspondant

```
class Cercle {
 int x, y; //centre du cercle
 int rayon;
 public Cercle (int ox, int oy) { x=ox; y=oy;}
 public Cercle (int ox, int oy, int r) {
   this(ox, oy);
   rayon = r;
 }
  …
} //fin class
```

Le 2ième constructeur appelle le 1ier constructeur de l'objet référencé par this

2ème utilisation:

this en préfixe permet de référencer l'objet auquel la variable ou méthode utilisée appartient.

```
public class Point {
    public int x = 0;
    public int y = 0;

    // le constructeur!
    public Point (int x, int y) {
     this.x = x;
     this.y = y;
    }

}//finclass
```

Faux si l'on écrit ...!

```
...
x = x;
y = y;
```

Ou bien écrire ...

```
public class Point {
    public int x = 0;
    public int y = 0;

    // le constructeur!
    public Point (int a, int b) {
     x = a;
     y = b;
    }
}//finclass
```

Opérateur instanceof – Catégoriser un objet dans une classe spécifique

- L'opérateur instanceof permet de vérifier si une référence désigne un objet d'une certaine classe ou d'une certaine interface.

- (*objet* instanceof Object) est toujours égal à true si *objet* est différent de null (toute classe hérite de Object et donc tout objet est une instance d'une classe dérivée de Object).

```
class ClasseTruc { … }
class ClasseMachin extends ClasseTruc { … }

class ClasseBidule {
  void methodeBidule () {
    ClasseTruc objet1 = new ClasseMachin ();
    ClasseTruc objet2 = new ClasseTruc ();
    Object objet3 = new ClasseTruc ();
    Object objet4 = null;
    boolean res1 = objet1 instanceof ClasseMachin ; // true
    boolean res2 = objet1 instanceof ClasseTruc ; // true
    boolean res3 = objet2 instanceof ClasseMachin ; // false
    boolean res4 = objet3 instanceof ClasseTruc ; // true
    boolean res5 = objet4 instanceof Object; // false
  }
}
```

Le masquage des variables (tiré de O. Dedieu):

```
class A {
   int x;
   void m() {...}
}
class B extends A{
   int x;
   void m() {...}
}
class C extends B {
   int x, a;
   void m() {...}
   void test() {
       a = super.x;          // a reçoit la valeur de la variable x de la classe B
       a = super.super.x;    // Syntax error
       a = ((B)this).x;      // a reçoit la valeur de la variable x de la classe B
       a = ((A)this).x;      // a reçoit la valeur de la variable x de la classe A
       super.m();            // Appel à la méthode m de la classe B
       super.super.m();      // Syntax error
       ((B)this).m();        // Appel à la méthode m de la classe C (et non B)
   }
}
```

Structure complète d'un programme JAVA

```
//Début du fichier MaClasse.java

    // Eventuelle déclaration de package
    package nomPackage;

    import nomClasse;
    import nomPackage.nomClasse;
    import nomPackage.*;

    //Déclarations des classes et des interfaces du fichier
    public class MaClasse
    {
      // Corps de MaClasse
    }

    class AutreClasse
    {
      // Corps de AutreClasse
    }

    interface NouvelleInterface
    {
      // Corps de NouvelleInterface
    }

    ...
// Fin du fichier MaClasse.java
```

Importer une classe sans package

Importer une classe d'un package

Importer toutes les classes d'un package

Une seule classe ou interface déclarée public, portant le même nom que le fichier

Exemple récapitulatif: il s'agit d 'un crayon ...

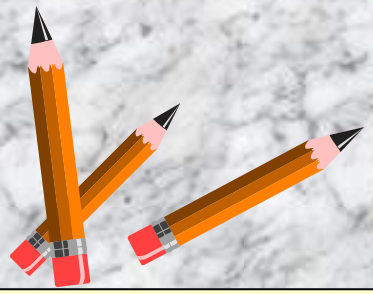
crayon

```
protected int longueur;
protected int diametre;

public Crayon ();
public Crayon (int , int );
public int quelleLongueur();
public int quelDiametre();
public void changeLongueur(int);
```

crayonCouleur

```
protected String couleur;

public CrayonCouleur ( );
public CrayonCouleur (int, int, String );
public CrayonCouleur(String );
public String quelleCouleur();
public void changeCouleur(String);
```

Classe de base ...



Classe de base

```
class Crayon {
   protected int longueur = 100;    // en mm
   protected int diametre = 5;      // en mm
   public Crayon () { … }           // crée un crayon standard
   public Crayon (int l, int d)   // crée un crayon de longueur l et de diametre d
    {
      longueur = l;
      diametre = d;
     }
public int quelleLongueur() { return longueur;}
public int quelDiametre() { return diametre; }
public void changeLongueur(int nouvelleLongueur) {longueur = nouvelleLongueur;}

}
```

Variables membres

Les constructeurs

Les méthodes

Etendons pour avoir des crayons de couleurs...

Classe dérivée

Les constructeurs renvoient l 'initialisation aux constructeurs de la classe de base

```
class CrayonCouleur extends Crayon {
protected String couleur = "gris";
public CrayonCouleur ( ) { super(); }
public CrayonCouleur (int l, int d, String c)
  {
    super( l , d);    // init longueur et diametre par le constructeur de la "superclass"
    couleur = c;
  }

public CrayonCouleur(String c)
  {
    super();
    couleur = c;
  }

public String quelleCouleur() { return couleur; }

public void changeCouleur(String nouvelleCouleur) { couleur = nouvelleCouleur;}

}
```

D'autres méthodes

Exercices – Jouer aux dés

1.1 - Créer une classe score permettant la gestion du score d'un joueur. La classe permettra de créer, pour un nom de joueur indiqué, un compteur (leScore) cumulant le nbr de points acquis. Quatre méthodes seront définies:

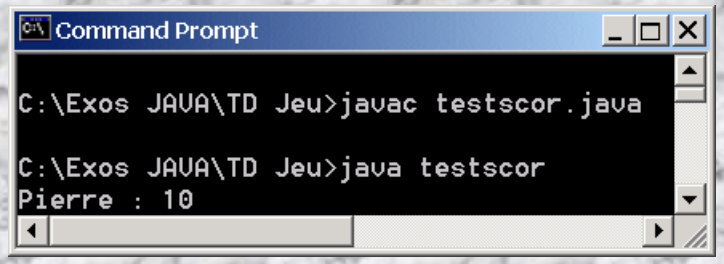
getScore () pour lire le score du joueur,

setScore() pour définir le score initial,

ajoute() pour cumuler des points au score,

affiche() pour afficher le score.

Ecrire un programme simple de test (TestScor permettant d'utiliser la classe score avec l'initialisation suivante: Nom: Pierre, score initial:10, score maximal:200



TD Jeu

testscor.class

score.class

```
public class score {

 // Nom du joueur.
 public String nom;

 //Score maximal
 int scoMax;
 private int leScore;

/*
 * Constructeur par défaut :
 * nom:"inconnu" ,
 * score initial:0 ,
 * score maximal:100 .
 */
 public score() { … }

 //Constructeur avec initialisation du nom.
 public score(String nom) { … }

 // méthode de lecture du score
 public int getScore(){ …}

 //méthode d'écriture du score
 public void setScore (int sco) { … }

 //méthode pour ajouter des points au score.
 public void ajoute (int points) { …}

 //méthode d'affichage du score
 public void affiche () { … }

}
```

Corrigé – 1.1 Jouer aux dés -

```
public class score {
 public String nom;  //Nom du joueur.
 int scoMax;     //Score maximal
 private int leScore;

 public score () {
  nom="inconnu";
  leScore=0;
  scoMax=100;
 }

  public score (String nom) {
  this.nom=nom;
  leScore=0;
  scoMax=100;
 }

public int getScore (){  return leScore; }

 public void setScore (int sco) {
  if (sco>scoMax) leScore=scoMax;
  else if (sco<0) leScore=0;
  else leScore=sco;
 }

  public void ajoute (int points) { setScore(leScore+points); }

  public void affiche () { System.out.println("Score"+nom+" : "+leScore); }
}
```

Constructeur par défaut :nom:"inconnu", score initial:0 , score maximal:100 .

Constructeur avec initialisation du nom.

méthode de lecture du score

méthode d'écriture du score

méthode pour ajouter des points au score.

méthode d'affichage du score

Testons la classe Score …

```
public class testscor {
 public static void main(String args[]) {
  score sc;
  sc = new score();
  sc.nom="Pierre";
  sc.setScore(10);
  sc.scoMax=200;
  sc.affiche();
```

Créer un objet score sc

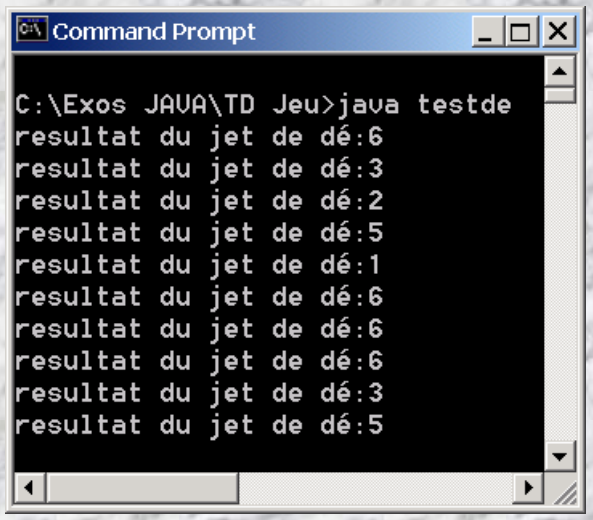
Initialiser les champs de l'objet sc

Afficher les champs de l'objet sc

Exercices (suite)

1.2 - Objet dé

Créer une classe **de** qui fournira la méthode **tirage**() renvoyant un nombre entier entre 1 et 6 tiré au hasard. Pour effectuer le tirage au hasard, utiliser la méthode **Math.random()** et les méthodes de la classe Double de java.lang. Ecrire un petit programme de test **testde** de la classe **de** qui affiche 10 lancés de dé.



Corrigé – 1.2 Objet dé -

```
public class de {
 int tirage(){
  Double D=new Double (6*Math.random()+1);
  int resultat=D.intValue();
  return resultat;
 }

void affiche()
{ System.out.println("résultat du jet de dé:" +
tirage());}

};
```

Création d'un entier double aléatoire entre 1 et 6

Traduction en entier aléatoire entre 1 et 6

Affichage d'un jeté de dé

Création d'un programme de test de la class de

```
public class testde {
 public static void main (String args[]) {
   de unde = new de();
  for (int i=0;i<10;i++) unde.affiche();
}
 }
```

Déclaration, et création en mémoire d'un dé

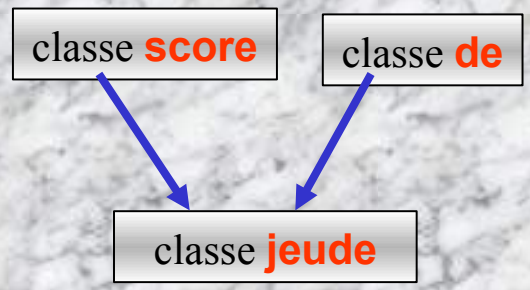
Affiche le tirage aleatoire de 10 jetés

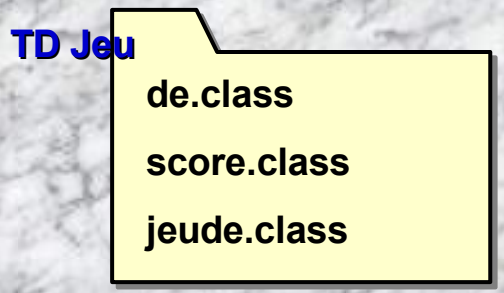
1.3 - Simulation d'un jeu de dé

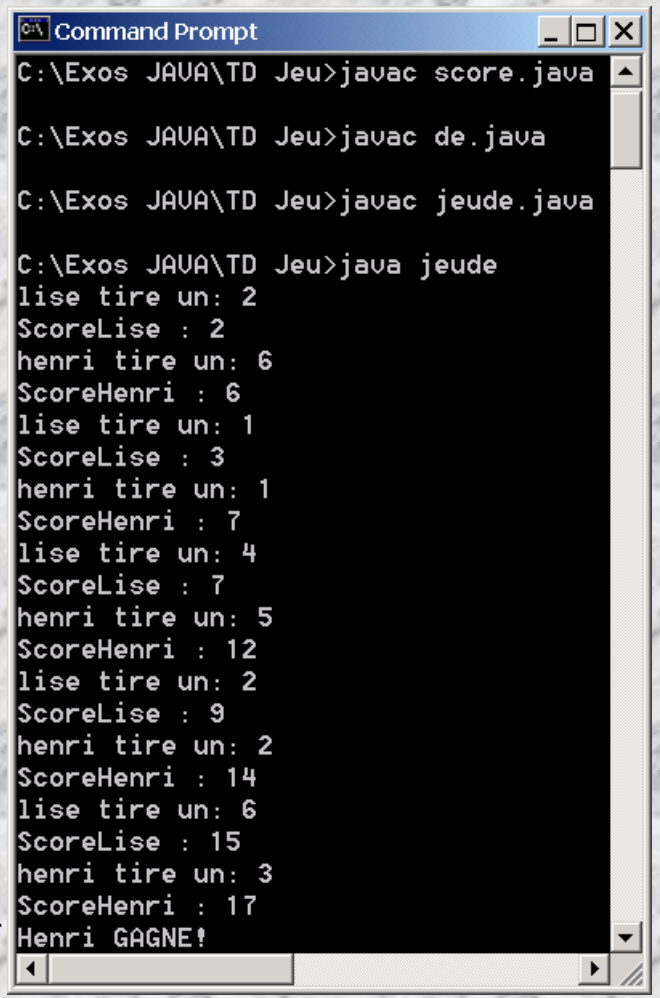
Lise et Henri jouent aux dés. A chaque tour de jeu chacun lance un dé et marque le nombre de points marqués par le dés. En utilisant les classes score et dé déjà construites, écrire les deux programmes **jeude.java** de simulation suivants :

a- Simuler 5 tours et afficher le nom du gagnant (celui qui a le plus de points).

b- on ajoute la règle : le 1ier qui atteint 21 points a gagné; simuler une partie et afficher le nom du gagnant.







Corrigé:

– 1.3 Simulation d'un jeu de dé

```
public class jeude {
 score henri, lise;
 de mon_de;

 jeude() {
  henri=new score("Henri");
  lise=new score("Lise");
  mon_de=new de();
 }

 void jouer (int n){
 int l,h;
  for (int i=1; i<=n; i++) {
   l=mon_de.tirage();
   lise.ajoute(l);
   System.out.println (lise+"tire un: "+ l);
   h=mon_de.tirage();
   henri.ajoute(h);
   System.out.println (henri+"tire un: "+ h);
  }

  if (lise.getScore()>henri.getScore())
   System.out.println("Lise GAGNE!");
  else if (lise.getScore()<henri.getScore())
   System.out.println("Henri GAGNE!");
  else System.out.println("Match nul.");
 }

 public static void main (String args[]) {
    jeude monjeu=new jeude();
    monjeu.jouer(5);
 }
}//finclass
```

Créer les compteurs des 2 joueurs

Créer un dé

Procéder à N tirage des 2 joueurs

indiquer le gagnant

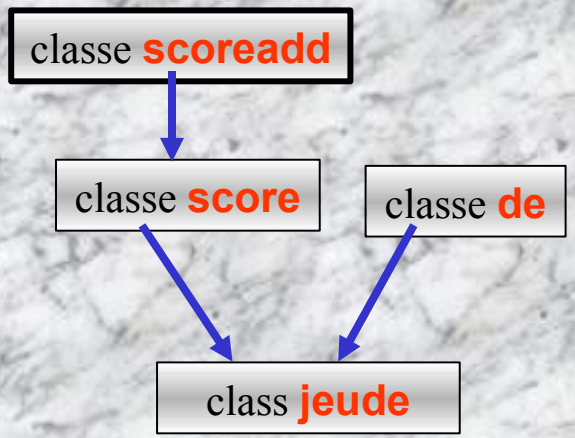
Créer un jeu

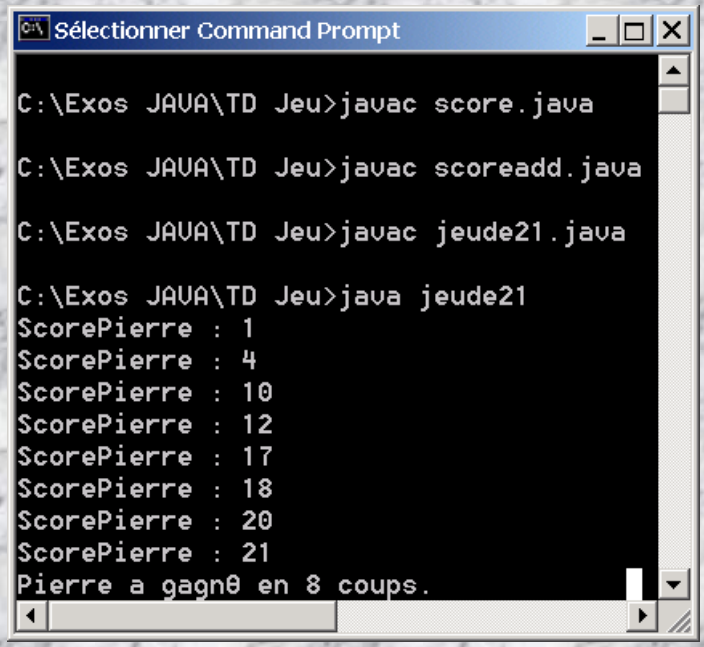
La partie se fait en 5 tirages

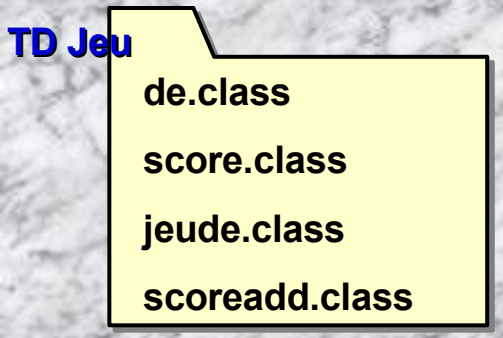
1.4 - Simulation d'un jeu de dé (suite …)

NOTA: Pour les besoins du programme, nous voudrions pouvoir gérer le score d'un joueur, mais aussi le nombre d'essais qui ont été effectués pour obtenir ce score. On dispose déjà d'une classe score pour la gestion du score proprement dit, mais celle-ci ne prend pas en compte le nombre d'essais. Pour éviter de refaire le travail effectué sur la classe score, il faut créer une classe dérivée de la classe score : celle-ci aura d'une part tous les champs (non privés) et toutes les méthodes (non privées) de la classe score (héritage) et d'autre part de nouveaux champs et de nouvelles méthodes.

```
public class scoreadd extends score {
        private int nbEssais;
    }
```







1.4 - Simulation d'un jeu de dé (suite …)

```
public class scoreadd extends score {
 private int nbEssais;

 public scoreadd () {
  super();
  nbEssais=0;
 }

  public scoreadd (String nom) {
  super(nom);
  nbEssais=0;
 }

 public int getNbEssais () {
  return nbEssais;
 }

 public void setScore (int sco) {
  super.setScore(sco);
  nbEssais++;
 }
}
```

Constructeur sans initialisation du nom

Constructeur avec initialisation du nom

lecture du nombre d'essais

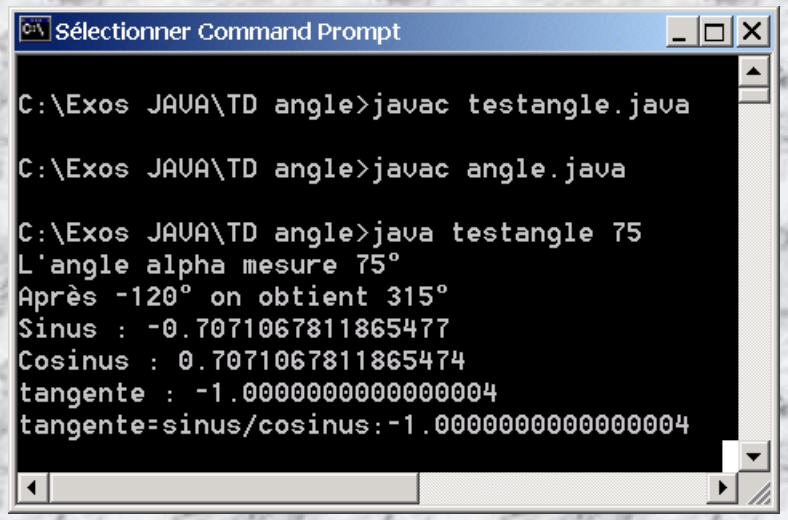
inscription d'un score; surcharge de la méthode setScore héritée

programme simulant le jeu consistant à lancer des dés jusqu'à obtenir 21 points

```
public class jeude21 {
 public static void main(String args[]) {
  int tirage;
  de mon_de= new de();

 scoreadd pierre=new scoreadd ("Pierre");
  while (pierre.getScore()<21) {
   tirage= mon_de.tirage();
   pierre.ajoute(tirage);
   pierre.affiche();
  }
  System.out.println("Pierre gagne en "+pierre.getNbEssais()+" coups.");
 }
}
```

2. Calculs d'angles

Créer une classe angle initialisée par une mesure d'angle en degrés (entre 0 et 360 degrés). Ecrire des méthodes d'addition, de soustraction et de multiplication par un entier [veiller à rester dans l'intervalle 0-360: méthode recadre()]. Ajouter une méthode de conversion en radians (180 degrés = pi radians), puis des méthodes de calcul du sinus, du cosinus et de la tangente (package java.Math). Ecrire un petit programme de test de la classe angle et de ses méthodes.




```
import java.text.DecimalFormat;

public class angle {
 double valeur;
 public angle() { valeur=0;}
 public angle(double ini) { valeur=ini; recadre(); }

 //méthode qui ramène la valeur entre 0 et 360°
 public void recadre() { … }

 // mise en forme pour affichage
 public String toString() {
  DecimalFormat f=new DecimalFormat();
  return f.format(valeur) + '°';
 }

 //méthode de conversion en radians
 public double toRadian() { … }
 //addition
 public void ajouter(angle a) { … }
 //multiplication par un entier
 public void multiplier(int n) { … }
 //fonctions trigo.
 public double sinus() { … }
 public double cosinus() { … }
 public double tangente() { … }
}
```

Corrigé – 2 calcul d'angle -

```
import java.text.DecimalFormat;

public class angle {
 double valeur;
 public angle() { valeur=0; }

 public angle (double ini) {
  valeur=ini;
  recadre();
 }

  public void recadre () {
  if (valeur<0.0)       do {  valeur=valeur+360; } while (valeur<0);
  if (valeur>=360.0) do { valeur=valeur-360;  } while (valeur>=360);
 }

  public String toString () {
  DecimalFormat f = new DecimalFormat();
  return f.format(valeur)+ 'ø';
 }

  public double toRadian () { return valeur*Math.PI/180.0; }

  public void ajouter (angle a) {
  valeur=valeur+a.valeur;
  recadre();
 }

  public void multiplier (int n) {
  valeur=valeur*n;
  recadre();
 }

  public double sinus () { return Math.sin(toRadian()); }

  public double cosinus () { return Math.cos(toRadian()); }

  public double tangente () { return Math.tan(toRadian()); }
}
```

méthode qui ramène la valeur entre 0 et 360

mise en forme pour affichage

méthode de conversion en radians

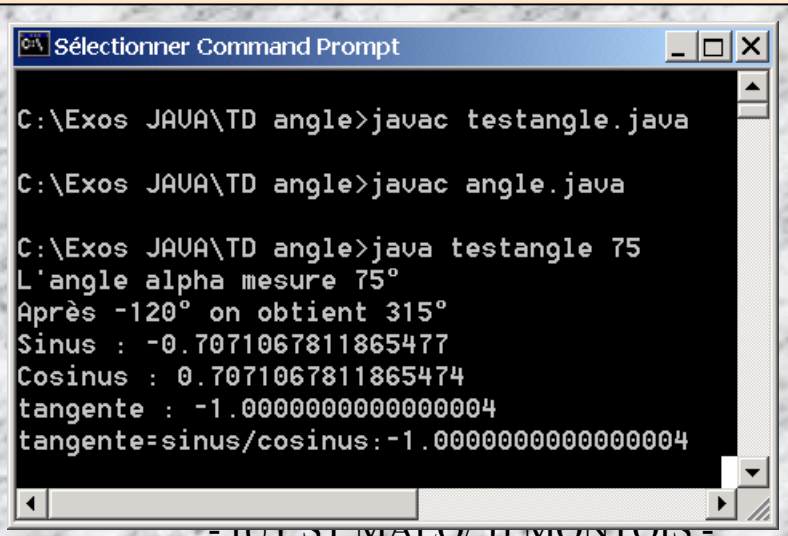
addition d'angle

multiplication par un entier

Fonctions trigonométriques

Tester l'objet angle …

```
public class TestAngle{
 public static void main (String args[]) {
  angle alpha=new angle(Integer.parseInt(args[0]));
  System.out.println("L'angle alpha mesure "+alpha.toString());
  alpha.ajouter(new angle(-120));
  System.out.println("Aprés -120° on obtient "+alpha.toString());
  System.out.println("Sinus : "+alpha.sinus());
  System.out.println("Cosinus : "+alpha.cosinus());
  System.out.println("tangente : "+alpha.tangente());

System.out.println("tangente=sinus/cosinus:"+alpha.sinus()/alpha.cosinus());
```



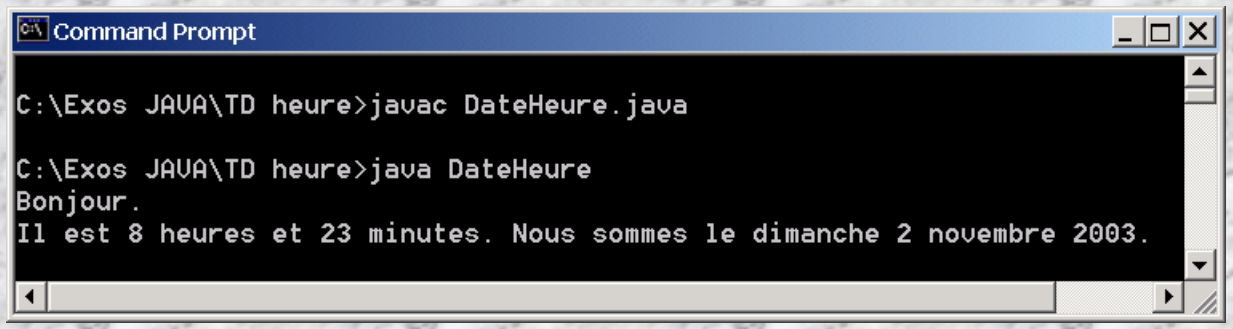
3 – Affichage de Date et heure . Le package java.util contient la classe **Calendar** permettant de gérer les dates et les heures. En utilisant cette classe, et manipulant les constantes: **HOUR_OF_DAY** , **MINUTE,MONTH, DAY_OF_MONTH,YEAR,**

DAY_OF_WEEK, ainsi que les méthodes getInstance(), get, … Ecrire un programme qui dit bonjour et donne l'heure et la date.

On utilisera 2 tableaux:

public static String MOIS[] = {"janvier","fevrier","mars", "avril", "mai", "juin", "juillet", "aout", "septembre", "octobre", "novembre", "decembre"};
public static String JOUR[] = {"dimanche","lundi","mardi", "mercredi", "jeudi", "vendredi", "samedi"};

Si l'heure est <= 12H écrire Bonjour; si > et <= 18H écrire Bon après-midi; si >18H écrire Bonsoir



Méthode simpliste …

```
import java.util.Date;
class TestDate {
public static void main(String args[])
{
   Date Maintenant = new Date();
   System.out.println(Maintenant);
  }
}
```

4 – Calculs de Suites mathématiques

1- Créer une classe suite représentant une suite récurrente. Elle contient un champ représentant le premier terme et une méthode abstraite permettant de calculer un terme à partir du précédent. Ecrire les méthodes permettant de calculer le terme de rang n et la somme des termes de 0 à n.

2- Créer une classe dérivée de la précédente qui représente les suites géométriques (on obtient un terme à partir du précédent en multipliant par une constante). Ecrire un programme qui calcule le terme de rang 20 et la somme des termes de 0 à 20.

Corrigé – 3 Affichage de Date et heure -

```
import java.util.*;
class HeureDate {
 public static String MOIS[]={"janvier","fevrier","mars", "avril", "mai", "juin", "juillet", "aout", "septembre", "octobre",
                              "novembre", "decembre"};

 public static String JOUR[]={"dimanche","lundi","mardi",  "mercredi", "jeudi", "vendredi", "samedi"};

 public static void main (String[] arguments) {

  Calendar aujourdhui = Calendar.getInstance(); // Lecture de la date et de l'heure
  int heure   = aujourdhui .get (Calendar.HOUR_OF_DAY);
  int minute = aujourdhui.get (Calendar.MINUTE);
  int mois    = aujourdhui.get (Calendar.MONTH);
  int jour    = aujourdhui.get (Calendar.DAY_OF_MONTH);
  int an      = aujourdhui.get (Calendar.YEAR);
  int we      = aujourdhui.get (Calendar.DAY_OF_WEEK);

  if (heure < 12) System.out.println("Bonjour!"); // affiche message de bienvenue selon l'heure
  else if (heure < 18) System.out.println("Bon après-midi!");
  else System.out.println("Bonsoir!");

  System.out.print("Il est: ");
  //affiche l'heure-minute
  System.out.print((heure > 12) ? (heure - 12) : heure );
  System.out.print(" heures ");
  if(minute != 0) {
   System.out.print("et " + minute);
   System.out.print( (minute != 1) ? " minutes." : " minute.");
  }
  // affiche la date
  System.out.println(" Nous sommes le "+JOUR[we-1]+" "+jour+" "+MOIS[mois]+" "+an+".");
 }
}
```

Récupère l'heure-minute-date

Affichge de l'heure-minute-date

Corrigé – 4 Calcul de suites mathématiques -

5. Calculs de segment

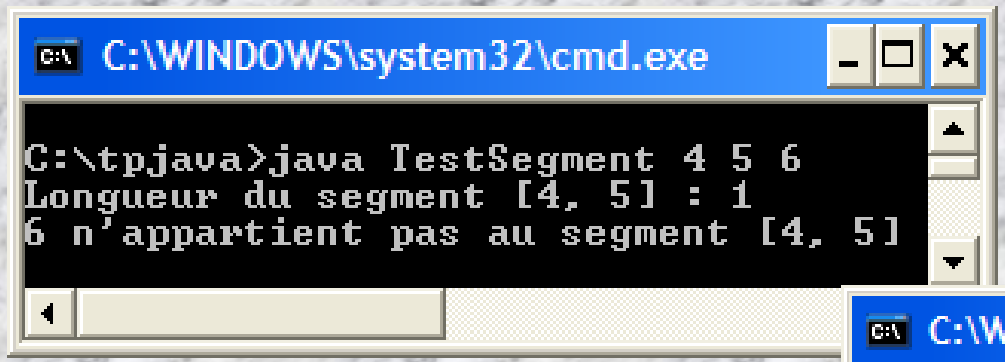
La classe Segment comprend:

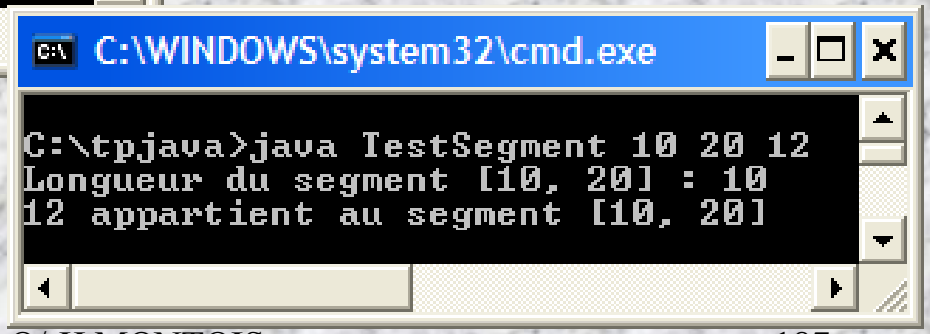
Deux variables privées de type Point, **extr1** et **extr2**, représentant les coordonnées (entières) des extrémités d'un segment dans un repère; où extr2>= extr1.

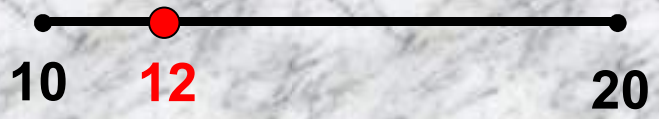
- Un constructeur Segment (…) qui définit les 2 extrémités
- la méthode **longueur**() retourne la longueur (entier) du segment.
- la méthode **appartient(),** indique si un point de coordonnées entières donnée en argument appartient ou non au segment
- une méthode qui redéfinit la méthode public String **toString**() de façon à écrire un segment d'extrémités A(4,14) et B(12,-35) sous la forme : *segment [A(4,14) ; B(12,-35)]*

une classe **TestSegment** permettant de tester la classe **Segment.**

La méthode **main** doit récupérer trois paramètres entiers : *origine*, *extrémité* du segment et *coordonnée* du point à tester.







Corrigé – 5 calcul de segment -

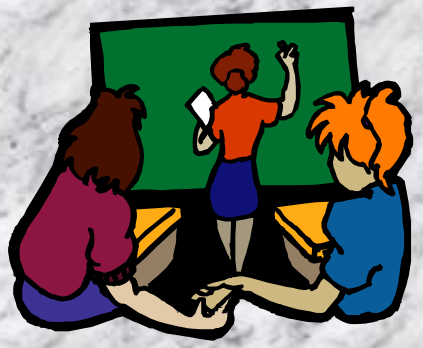
```
class Segment {
  int extr1, extr2;
  Segment (int e1, int e2) { extr1 = e1; extr2 = e2; }
  void classe ()
   {
    if (extr1 > extr2)
   { int tampon;
     tampon = extr1;
     extr1 = extr2;
     extr2 =tampon;
    }
   }
  int longueur() { classe(); return extr2 - extr1; }

  boolean appartient (int x) { return (x - extr1) * (x - extr2) <= 0; }

  public String toString() { classe(); return "segment [" + extr1 + ", " + extr2 + "]"; }
}

class TestSegment {
  public static void main (String args[])
   {
    Segment s = new Segment (Integer.parseInt(args[0]),  Integer.parseInt (args[1]));
    System.out.println("Longueur du " + s + " : " + s.longueur());
    int  point = Integer.parseInt(args[2]);
    if (s.appartient(point)) System.out.println(point + " appartient au " + s);
    else                     System.out.println(point + " n'appartient pas au " + s);
   }
}
```

Classe extr1, et extr2 tel que extr1 <= extr2

Crée un segment selon les arguments de ligne

-7-

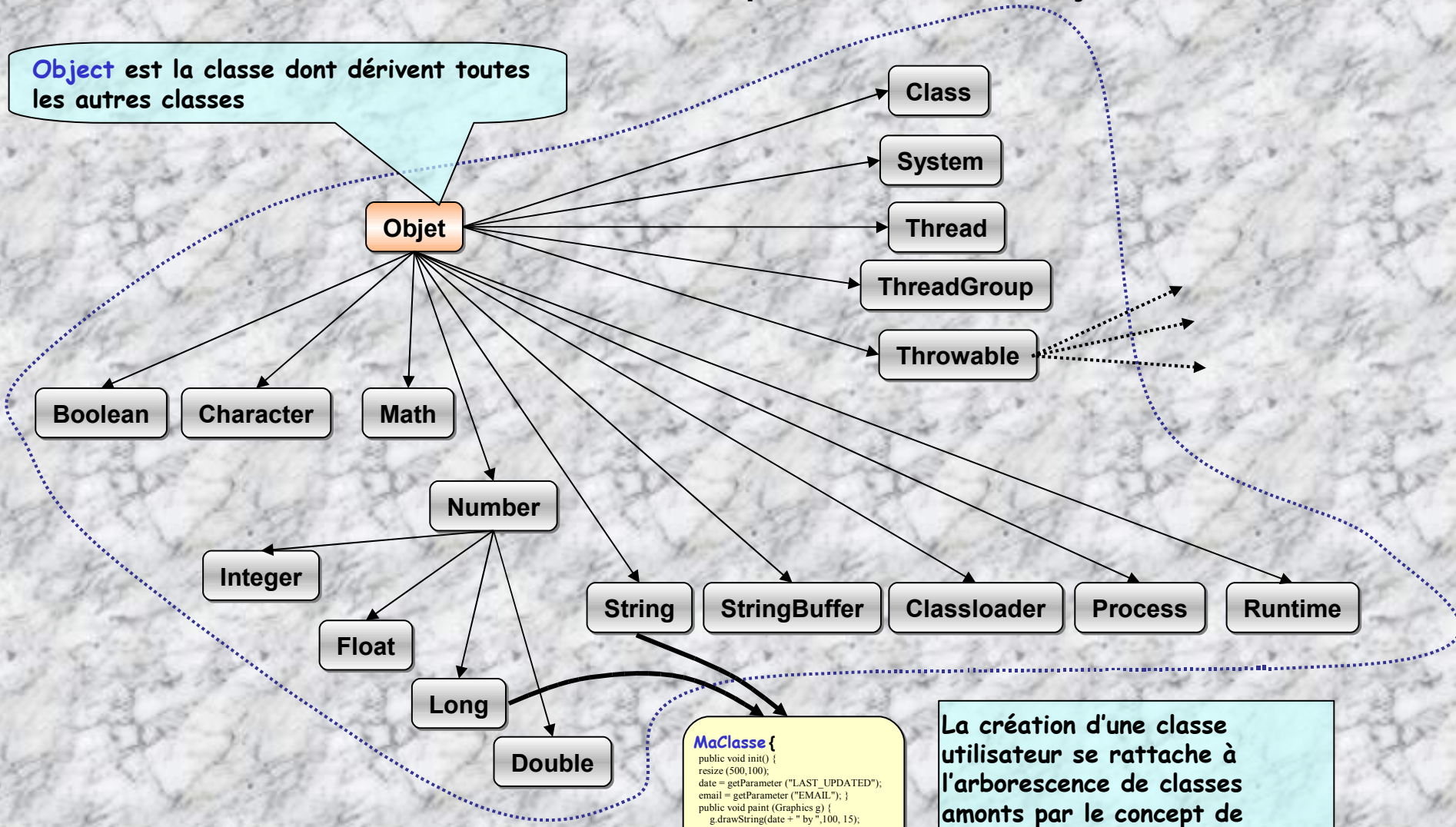
Parlons de familles de classes … sous JAVA



La création d'une classe utilisateur n'est pas isolée; elle s'insère dans une arborescence de classe du langage java, à une place dérivée d'une classe java amont. Chaque classe en amont de votre classe, apporte un «savoir-faire » selon l'ensemble des méthodes, variables, et constantes qu'elle contient.

Objet, mère de toutes les classes …

Schéma des dérivations des classes JAVA à partir de la classe Objet



Classe Object

Object est la classe de base de Java. Tous les objets créés héritent des méthodes suivantes :

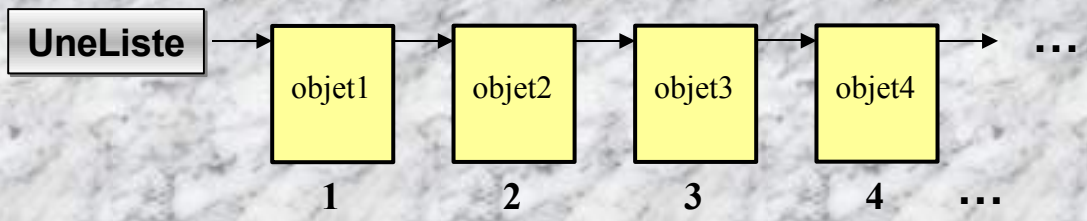
```
public class Object
{
  public final Class getClass ();
  public String toString() ;
  public boolean equals(Object obj);
  public int hashCode() ;
  protected Object clone () throws CloneNotSupportedException ;
  public final void wait();
  public final void wait(long milli_secondes) ;
  public final void wait(long milli_secondes, int nano_secondes) ;
  public final void notify() IllegalMonitorStateException;
  public final void notifyAll() throws IllegalMonitorStateException ;
  protected void finalize() throws Throwable
}
```

Programmation d'objets dynamiques

Java propose deux classes (import java.util.*) pour la manipulation et le stockage dynamique d'objets de types quelconques:

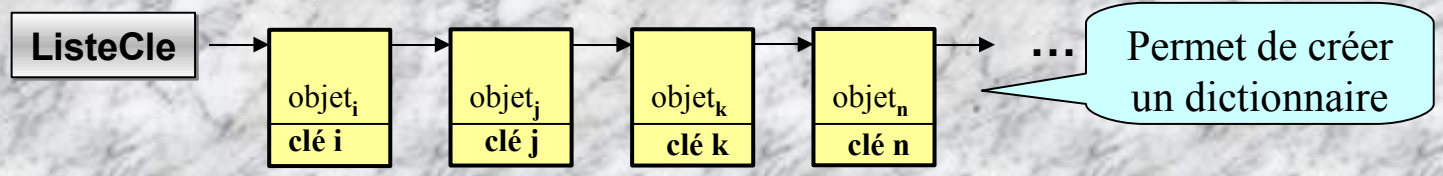
✓**Classe Vector pour la gestion de liste d'objets-Indexation selon la position ordinale**

Vector UneListe = new Vector();



✓**Classe Hashtable pour la gestion de listes d'objets-Indexation selon le contenu (clé)**

Hashtable ListeCle = new Hashtable();



Classe Vector

```
java.lang.Object |
                +--java.util.AbstractCollection |
                                +java.util.AbstractList |
                                                +--java.util.Vector
```

Création d'une bibliothèque indexée numériquement

```
import java.util.*; //package pour utiliser la classe Vector
 public class Bibliotheque  {
 private Vector liste;
 public Bibliotheque()   { liste = new Vector (); }
 public void ajouteUnLivre() { liste.addElement(new Livre ()); }
 public void afficheLesLivres()     {
                int nbLivres = liste.size();
                if (nbLivres > 0) {
                                Livre tmp;
                                for (int i = 0; i < nbLivres; i ++) {
                                                tmp = (Livre)liste.elementAt(i);
                                                tmp.afficheUnLivre();
                                }
                }
                else System.out.println("Il n'y a pas de livres dans la liste");
 }
}
```

Création d'une liste pour stocker les livres qui sont des objets

Définition des méthodes de gestion de la liste

Utilisation de la bibliothèque indexée numériquement ...

```
public class GestionBibliotheque  {

 public static void main (String [] argument)
 { byte choix = 0 ;
   Bibliotheque B = new Bibliotheque ();
   String prénom, nom;
   do  {
      System.out.println ("1. Ajoute un livre");
      System.out.println ("2. Affiche la bibliothèque");
      System.out.println ("3. Sortir");
      System.out.print ("Votre choix : ");
      try { choix  = System.in.read(); }
      catch (Exception e) { System.out.println("Erreur: "+ e.toString());}

      switch (choix) {
        case 1 : B.ajouteUnLivre(); break;
        case 2 : B.afficheLesLivres(); break;
        case 3 : System.exit(0) ;
        default : System.out.println("Option inexistante ");
       }
    } while ( choix != 3);
 }//fin main
}
```

Création d'un objet Bibliothèque

Affiche le menu

Saisie de l'item

Exécution de l'action associée

Quelques méthodes de manipulation d'objets vector (API SUN) …

void **add** (int index, Object element) : *Insert the specified element at the specified position in this Vector.*
boolean **add** (Object o) : *Appends the specified element to the end of this Vector.*
void **addElement** (Object obj) : *Adds the specified component to the end of this vector, increasing its size by one.*
void **clear** () : *Removes all of the elements from this Vector.*
Object **elementAt** (int index) : *Returns the component at the specified index.*
boolean **equals** (Object o) : *Compares the specified Object with this Vector for equality.*
Object **firstElement**() : *Returns the first component (the item at index 0) of this vector.*
int **indexOf** (Object elem) : *Searches for the first occurence of the given argument, testing for equality.*
void **insertElementAt** (Object obj, int index) :*Inserts the specified object in this vector at the specified index.*
boolean **isEmpty**() : *Tests if this vector has no components.*
Object **lastElement** () : *Returns the last component of the vector.*
int **lastIndexOf**(Object elem) : *Returns the index of the last occurrence of the specified object in this vector.*
Object **remove** (int index) : *Removes the element at the specified position in this Vector.*
boolean **remove** (Object o) : *Removes the first occurrence of the specified element in this Vector.*
boolean **removeElement**(Object obj) :*Removes the first (lowest-indexed) occurrence of the argument from this vector.*
void **removeElementAt**(int index) : *Deletes the component at the specified index.*
void **setElementAt**(Object obj, int index) :*Sets the component at the specified index of this vector.*
void **setSize** (int newSize) :*Sets the size of this vector.*
int **size**() :*Returns the number of components in this vector.*
…

Classe Hashtable

Utilisation d'une clé qui peut-être une String, un int, un objet …

* Création d'un dictionnaire d'auteurs indexé par clé

```
import java.util.*;
 public class Dictionnaire  {
  private Hashtable ListeAuteurs;
  public Dictionnaire ()   { ListeAuteurs = new Hashtable();}
  private String creerUneCle(Auteur e) {
                String tmp;
                tmp = (e.quelPrénom()).charAt(0)+ e.quelNom();
                tmp.toUpperCase();
                return tmp;
  }

  <  Méthodes de gestion du dictionnaire … >

}
```

D'abord créer une clé. Par exemple une String de lettres majuscules, composée de la 1ière lettre du prénom suivi du nom de l'auteur

Les méthodes de gestion du dictionnaire d'auteurs …

```
public void ajouteUnAuteur() {
            Auteur nouveau = new Auteur();
            String cle = creerUneCle(nouveau);
            if (ListeAuteurs.get(cle) == null) ListeAuteurs.put(cle, nouveau);
            else System.out.println("Auteur déjà saisi !");
 }

 public void rechercheUnAuteur(String p, String n) {
            String cle = creerUneCle(p, n);
            Auteur aClasse = (Auteur) ListeAuteurs.get(cle);
            if (aClasse != null)  aClasse.afficheUnAuteur();
              else System.out.println(p + " " + n + " est inconnu ! ");
 }
 public void modifieUnAuteur(String p, String n) {
            String cle = creerUneCle(p, n);
            if (ListeAuteurs.get(cle) != null) {
                  Auteur aModifie = new Auteur(p, n) ;
                ListeAuteurs.put(clé, aModifie);
            }
            else System.out.println(p + " " + n + " est inconnu ! ");
 }

    …
```

Suite …

```
public void supprimeUnAuteur(String p, String n) {
                String cle = creerUneCle(p, n);
               Auteur aClasse = (Auteur) ListeAuteurs.get(cle);
                if (aClasse != null)  {
                  ListeAuteurs.remove(cle);
                  System.out.println(p + " " + n + " a été supprimé ");
                 }
                 else System.out.println(p + " " + n + " est inconnu ! ");
 }

 public void afficheLesAuteurs()  {
                 if(ListeAuteurs.size() != 0) {
                   Enumeration enumAuteur = ListeAuteurs.keys();
                   while (enumAuteur.hasMoreElements()) {
                           String cle = (String) enumAuteur.nextElement();
                           Auteur aClasse = (Auteur) ListeAuteurs.get(cle);
                           aClasse.afficheUnAuteur();
                   }
                 }
               else System.out.println("Pas d'auteur dans cette liste");
             }
```

Quelques méthodes de manipulation d'objets Hashtable (API SUN) …

void **clear**() *Clears this hashtable so that it contains no keys.*

boolean **contains**(Object value) *Tests if some key maps into the specified value in this hashtable.*

boolean **containsKey**(Object key) *Tests if the specified object is a key in this hashtable.*

boolean **containsValue**(Object value) *Returns true if this Hashtable maps one or more keys to this value.*

boolean **equals**(Object o) *Compares the specified Object with this Map for equality*

Object **get**(Object key) *Returns the value to which the specified key is mapped in this hashtable.*

Object **put**(Object key, Object value) *Maps the specified key to the specified value in this hashtable.*

int **hashCode**() *Returns the hash code value for this Map as per the definition in the Map interface.*

boolean **isEmpty**() *Tests if this hashtable maps no keys to values.*

Object **remove**(Object key) *Removes the key (and its corresponding value) from this hashtable.*

int **size**() *Returns the number of keys in this hashtable.*

String **toString**() *Returns a string representation of this Hashtable object in the form of a set of entries, enclosed in braces and separated by the ASCII characters ", " (comma and space).*

…

Classes très utiles:

java.util.StringTokenizer

Permet d'énumérer à partir d'une chaîne de caractères str un ensemble de sous-chaînes séparées par des délimiteurs

Exemple: "Je suis une image = d'une ville spéciale : totoville / d'un pays imaginaire : iutland"

Délimiteurs = { =, :, / } ⇒ **5 chaînes extraites:**

Je suis une image
d'une ville spéciale
totoville
d'un pays imaginaire
iutland

Constructeurs:

public StringTokenizer (String **str,** String **delim, boolean returnTokens)**
public StringTokenizer (String **str,** String **delim)**
public StringTokenizer (String **str)**

- ✓**Permet de spécifier la chaîne str dans laquelle on recherche des sous-chaînes.**
- ✓**Les souschaînes sont séparées par des délimiteurs pouvant être n'importe quel caractère de delim.**
- ✓**Si returnTokens est true, l'énumération rendra les sous-chaînes et les délimiteurs.**
- ✓**Par défaut, le délimiteur est un espace et returnTokens est égal à false.**

Méthodes de la classe StringTokeniser

Renvoient *true* si la chaîne str a encore des souschaînes à énumérer

```
public boolean hasMoreElements ()

public boolean hasMoreTokens ()
```

Renvoient la sous-chaîne suivante de str (ou le délimiteur si returnTokens est *true*), et avance la position de recherche dans str au caractère suivant la sous-chaîne renvoyée. La troisième méthode permet de remplacer les délimiteurs recherchés.

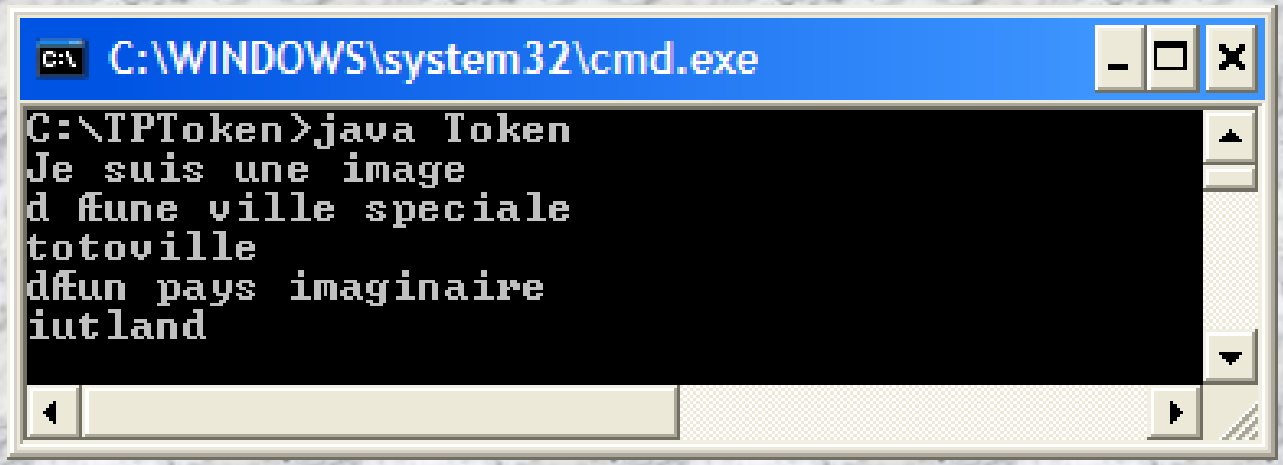
```
public Object  nextElement ()  throws  NoSuchElementException
public String nextToken ()       throws  NoSuchElementException
public String nextToken (String delim) throws  NoSuchElementException
```

Renvoie le nombre de sous-chaînes restant à énumérer

```
public int countTokens ()
```

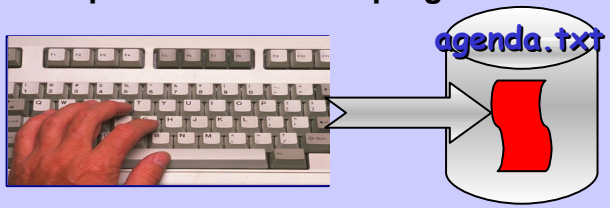
Exemple:

```
import java.util.StringTokenizer;

class Token {
    public static void main (String args[])
        {
          String s = "Je suis une image=d'une ville spéciale:totoville/d'un pays imaginaire:iutland";
          StringTokenizer st = new StringTokenizer (s , "=:/" );
          while ( st.hasMoreTokens() ) { System.out.println(st.nextToken()); }
        }
};
```



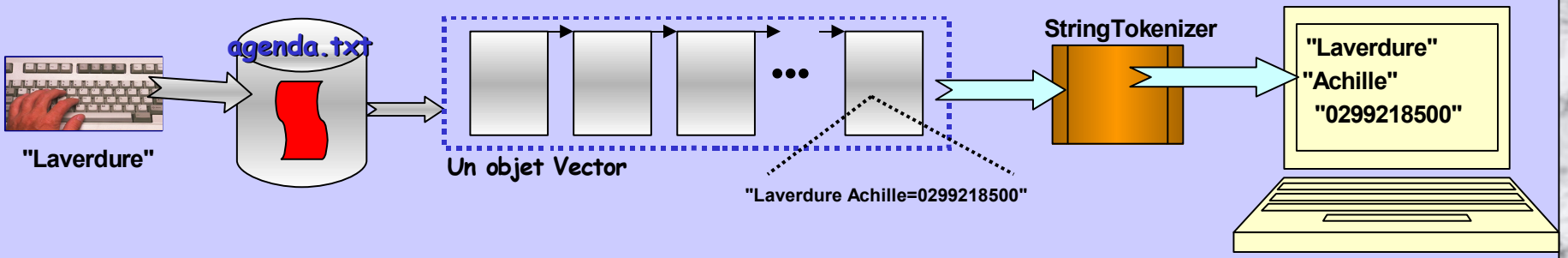
Exercice – Agenda téléphonique très simple. (Etudier au préalable la gestion de fichiers sur disque dur)

1°) Ecrire et tester le programme permettant de saisir au clavier des chaîne de caractères ayant la structure suivante: <nom> <prénom>=<numéro de téléphone> (Ex: Laverdure Achille=0299218500). Puis stocker successivement les chaînes dans un fichier sur disque dur appelé agenda.txt. A l'aide de l'éditeur notepad, vérifier que votre fichier est bien lisible après exécution du programme.



"Laverdure Achille=0299218500"

2°) Ecrire et tester le programme permettant de lire le fichier agenda.txt, et d'afficher le numéro de téléphone d'une personne dont on donne le nom en ligne de commande (java <nom>); gérer le cas des homonymies.
Méthode: On chargera d'abord toute les chaînes du fichier dans un objet e type Vector, puis à l'aide d'un objet de type StringTokenizer on opérera une extraction des champs utiles (nom, numéro tel, …) qu'on affichera à l'écran.



3°) Ecrire et tester le programme permettant de fusionner les deux programmes ci-dessus à travers l'élaboration d'un menu simple :

1 – Créer un Agenda
2 – Recherche Numéro
3 – Ajouter un Nom

Exercice (corrigé) – Agenda téléphonique très simple.

Exercice (corrigé) – Agenda téléphonique très simple.

-8-

Les exceptions sous JAVA

Sentence …

Le problème, ce n'est pas qu'il y en ait un, c'est surtout que l'on ne sache pas le traiter!

C'est de moi, si, si !

Un certain nombre de problèmes (on dit exceptions) peuvent survenir au «runtime ». On peut cependant prévoir de les traiter à l'aide d'une surveillance judicieusement placée à certains endroits du programme. Ainsi, il n'y a pas blocage de l'exécution du programme.

Traitement local de l'erreur

La gestion d'erreur par exceptions permet d'écrire de manière sécurisée et claire un programme, en focalisant sur le segment de codes susceptibles de produire une erreur (TRY), et en isolant localement, le traitement d'erreur (CATCH) des d'instructions du segment surveillé.

```
try
        { Instruction(s) surveillées }

catch (ClasseException exceptionInterceptee)
        { Instruction(s) traitant l'erreur }
```

Il existe plusieurs types de ClasseException:
IOException, AWTException, ClassNotFoundException, etc etc

Exception interceptée dans le bloc surveillé. C'est un objet!

- try-catch obligatoirement utilisé dans les opérations d'E/S system.in.read() et gestion de fichiers.
- Les packages java.lang, java.util, java.io, java.net, java.awt définissent de nombreuses Error et Exception (consulter les API SUN!)

Quelques exceptions du paquetage java.lang …

ArithmeticException *Thrown when an exceptional arithmetic condition has occurred.*

ArrayIndexOutOfBoundsException *Thrown to indicate that an array has been accessed with an illegal index.*

ArrayStoreException *Thrown to indicate that an attempt has been made to store the wrong type of object into an array of objects.*

ClassNotFoundException *Thrown when an application tries to load in a class through its string name using: The forName method in class Class.*

Exception *The class Exception and its subclasses are a form of Throwable that indicates conditions that a reasonable application might want to catch.*

IllegalArgumentException *Thrown to indicate that a method has been passed an illegal or inappropriate argument.*

IndexOutOfBoundsException *Thrown to indicate that an index of some sort (such as to an array, to a string, or to a vector) is out of range.*

NegativeArraySizeException *Thrown if an application tries to create an array with negative size.*

NoSuchMethodException *Thrown when a particular method cannot be found.*

NullPointerException *Thrown when an application attempts to use null in a case where an object is required.*

NumberFormatException *Thrown to indicate that the application has attempted to convert a string to one of the numeric types, but that the string does not have the appropriate format.*

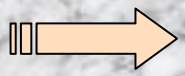
RuntimeException *RuntimeException is the superclass of those exceptions that can be thrown during the normal operation of the Java Virtual Machine.*

StringIndexOutOfBoundsException *Thrown by the charAt method in class String and by other String methods to indicate that an index is either negative or greater than or equal to the size of the string.*

☞**Les exceptions sont des classes.**

☞**La levée d'une exception instancie un objet.**

Classe Exception du paquetage java.lang …



```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
```

- Constructeurs:
  - ✓ **Exception**() *Construit une Exception sans spécifier de message.*
  - ✓ **Exception** (String **s**) *Construit une Exception en spécifier de message.*

- Méthodes héritées de la classe **java.lang.Throwable:**

  fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

- Méthodes héritées de la classe **java.lang.Object**:

  clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait,

Une structure plus complète …

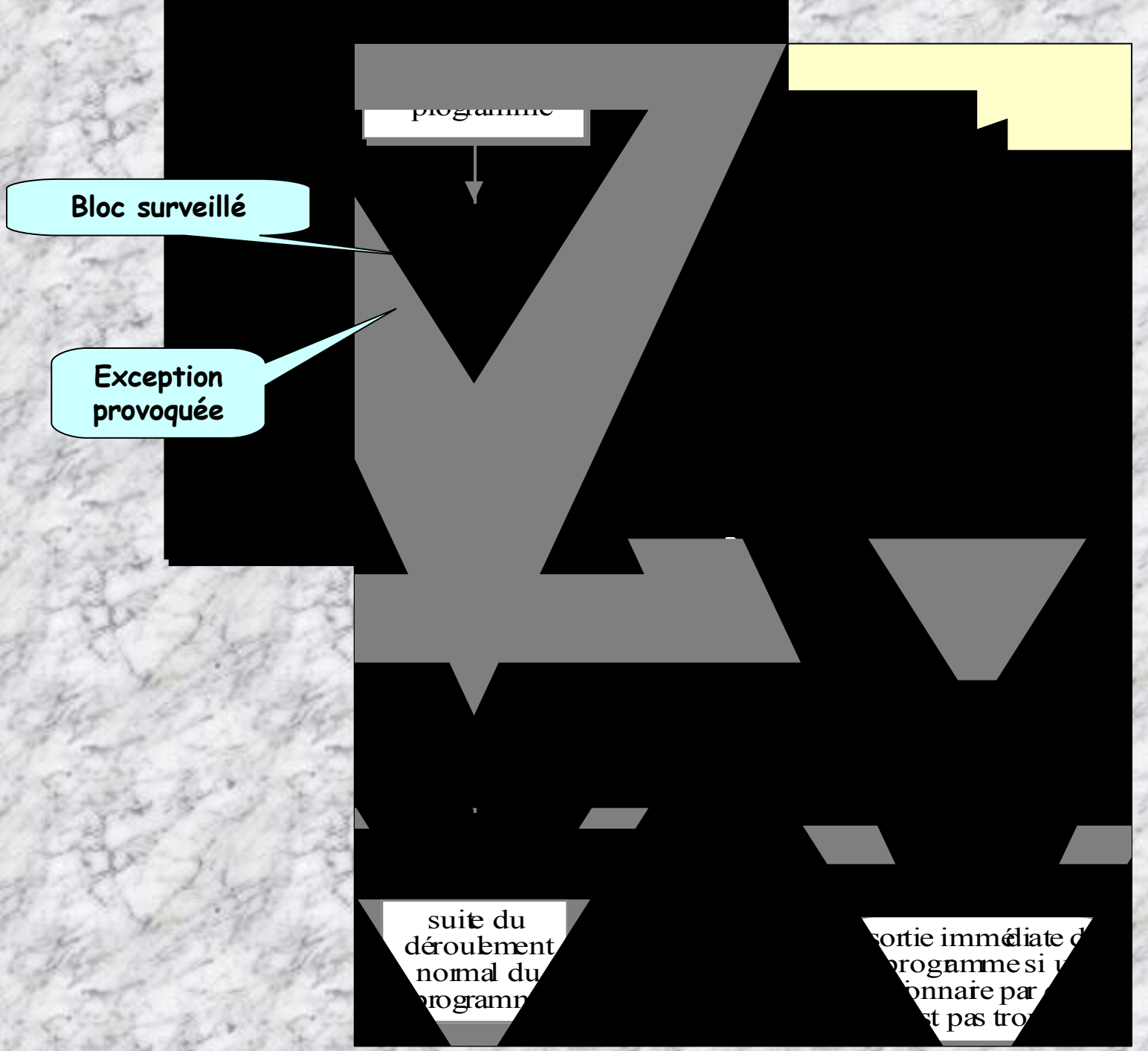
```
…
try
    { bloc d'instruction(s) surveillées; acquisition de ressources }

catch (ClasseException1 exceptionInterceptee) { Instruction(s) traitant l'erreur1 }
catch (ClasseException2 exceptionInterceptee) { Instruction(s) traitant l'erreur2 }
catch (ClasseException3 exceptionInterceptee) { Instruction(s) traitant l'erreur3 }
…
finally  { bloc toujours exécuté avec ou sans exception; libération de ressources}
```

finally toujours exécuté, sert à récupérer les ressources avec ou sans exception - (*optionnel*)

☞ **On peut lever une exception avec l'instruction throw (e) laquelle spécifie un objet e à lancer**

```
{
  …
 IOException e = new IOException("Fichier non trouvé");
  …
 throw (e);
}
```

```
catch (IOException e )
   { instructions(s) traitant l'erreur e }
```


Bloc surveillé
Exception provoquée
suite du déroulement normal du
sortie immédiate d
programme si
t pas tro

☞ **Quand une exception de classe ClasseException est déclenchée dans le bloc try, le contrôle passe au premier bloc catch qui traite la classe d'exception ClasseException. Ce catch reçoit en paramètre l'exception déclenchée:**

catch (ClasseException *exceptionInterceptee*).

☞ **Si aucun des catch n'est capable d'intercepter l'exception, le contrôle est rendu au premier catch capable d'intercepter une exception de classe ClasseException, parmi les méthodes mémorisées dans la pile d'exécution et exécutant un try-catch.**

☞ **Si aucun catch n'est rencontré, la JVM indique l'exception qui est survenue et arrête le thread dans laquelle elle est survenue (blocage complet du programme!).**

☞ **Le bloc instructions d'un catch peut éventuellement redéclencher l'exception interceptée exceptionInterceptee pour la propager dans la pile d'exécution, grâce à l'instruction:**

throw *exceptionInterceptee*;

☞ **Le bloc d'instructions du dernier catch peut être optionnellement suivi de l'instruction finally, suivi lui aussi d'un bloc d'instructions spécifiant les instructions qu'il faut toujours exécuter à la suite du bloc try si aucune exception n'a été déclenchée ou à la suite du traitement d'un catch.**

Exemple de traitement local d'une exception déclenchée grâce à throw dans un bloc try et interceptée par un des catch qui suivent : '

```
class UneClasse
{
  ...
  void methode ()
  {
     try
     {
       ...
      throw new Exception ();
     }
     catch (Exception e)
     {
       System.out.println("Erreur: "+ e.toString());
         // Traitement de l'exception
     }
  }
}
```

Levée d'une exception

Traitement de l'exception

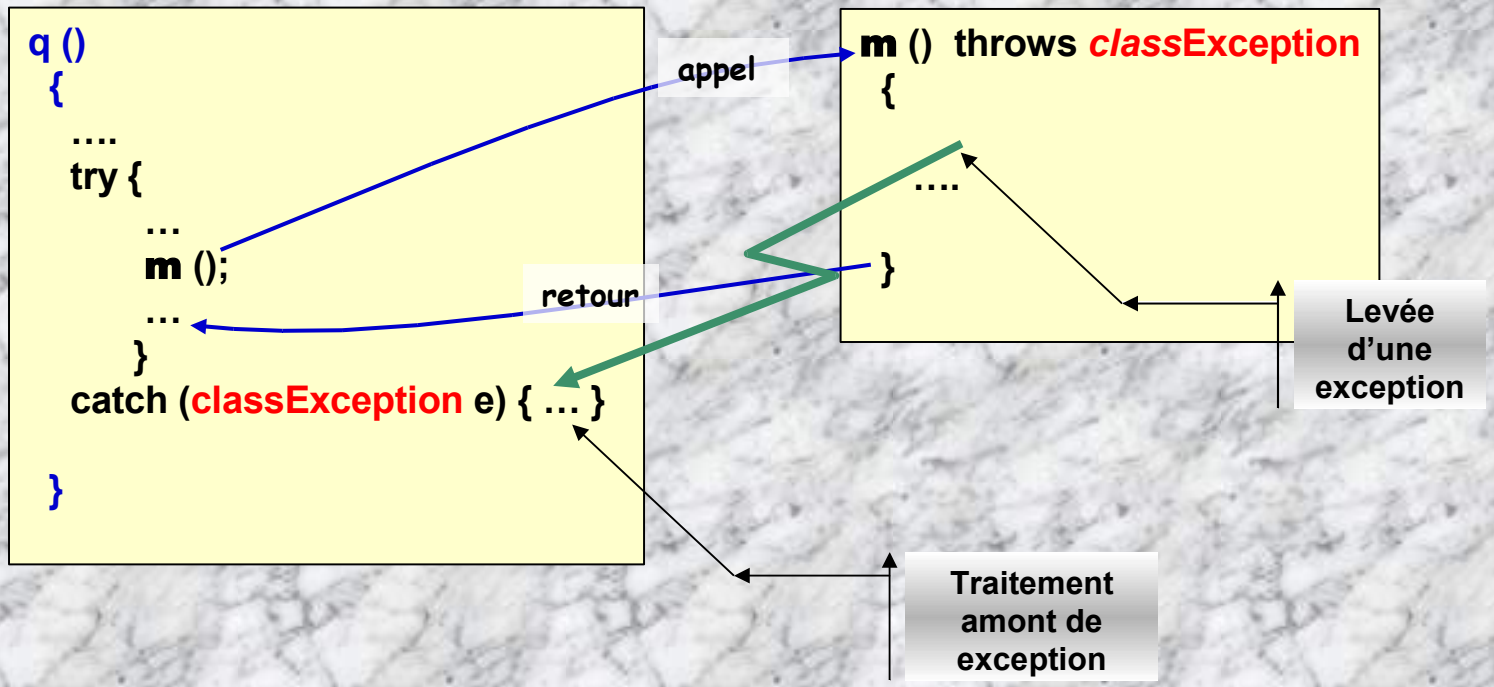
Type de l'exception

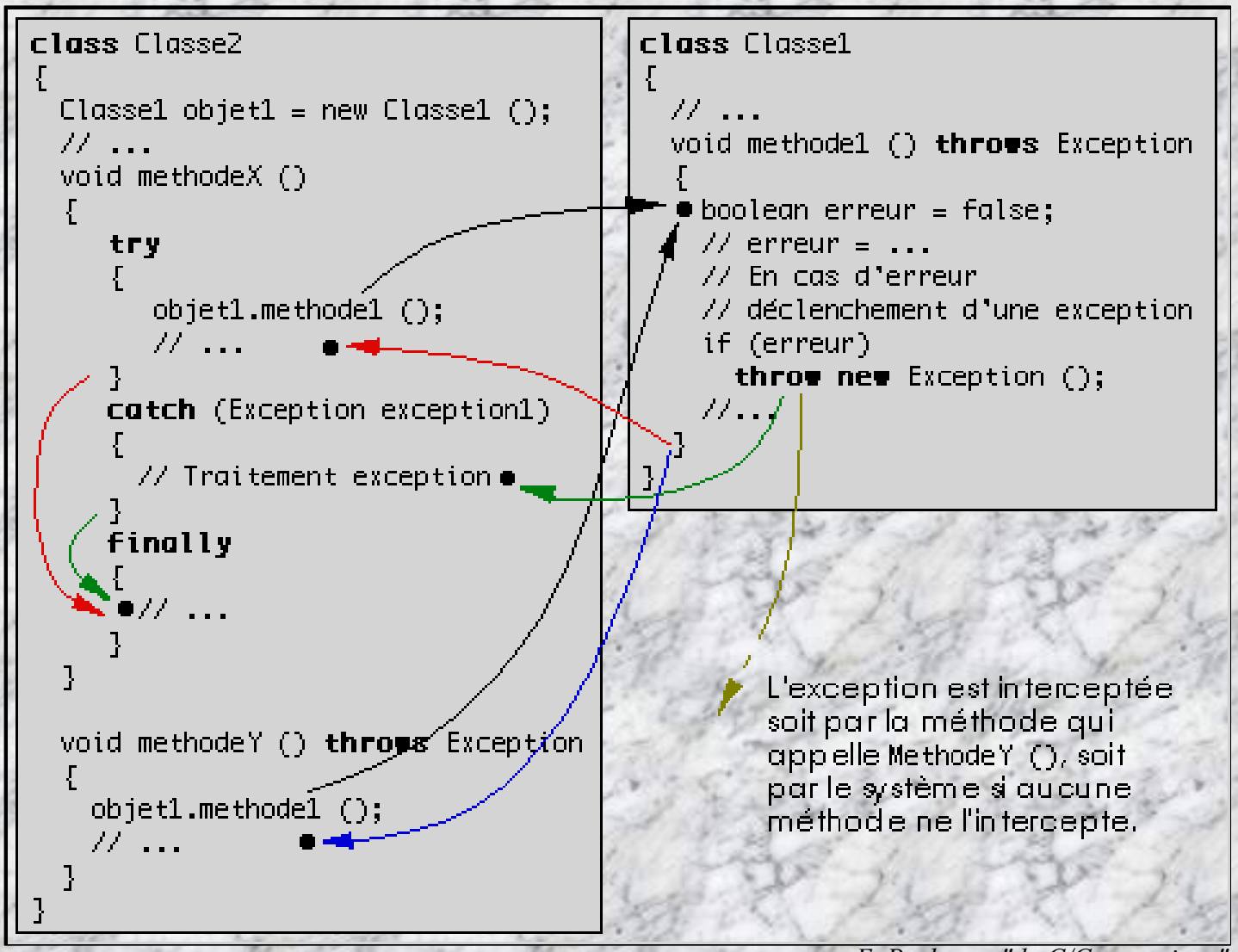
Autre exemple …

```
class Classe2
      {
       Classe1 objet1 = new Classe1 ();
          ...
       void methodeX ()
       {
         try
         {
            objet1.methode1 ();
            // ...
         }
         catch (ClasseException exception1)
         {  // Que faire en cas de problème ? }

          // ... Eventuellement d'autres catch (...)
         finally
         {
           // Le bloc finally est optionnel. Que faire après que le bloc try ou qu'un bloc catch aient été exécutés ?
         }
       }

       void methodeY () throws ClasseException
       {
         objet1.methode1 ();
           ...
       }
      }
```

Traitement amont ou différé de l'erreur

Contrairement au traitement local dès que l'erreur est captée au sein d'une structure try – catch, on peut propager l'indication d'exception au moyen de la directive throws associée avec la méthode m () susceptible de provoquer une erreur. Ainsi, l'exception sera-t-elle traitée en amont, par la méthode appelante.

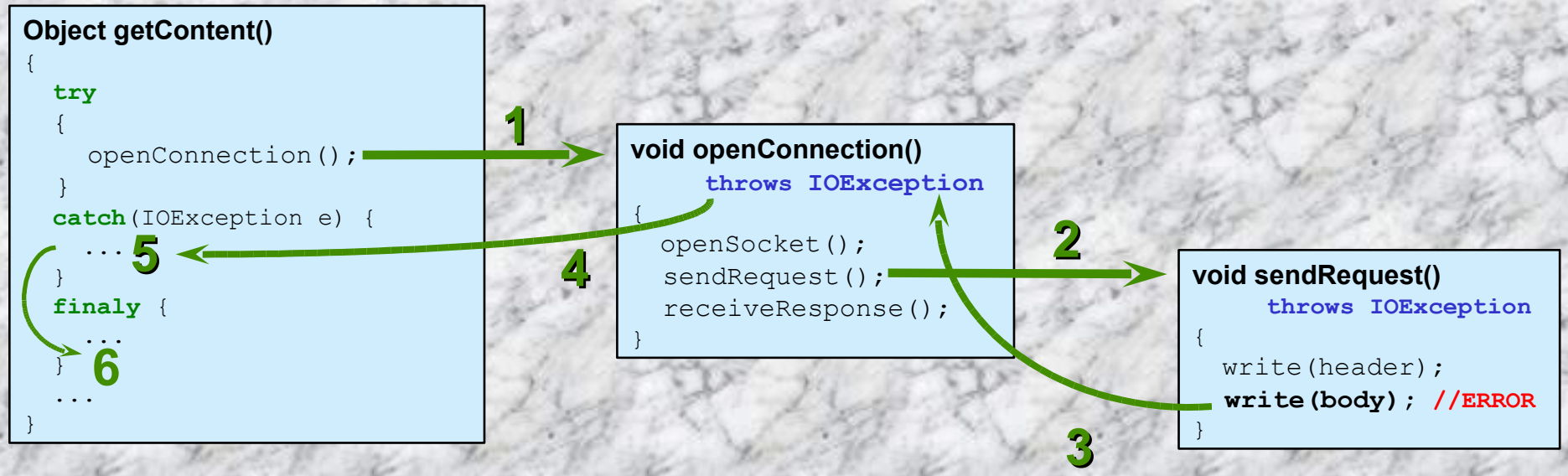




E. Puybaret, "du C/C++ au java"

(Tiré de O. Dedieu)

« Ce sont des instances de classes dérivant de java.lang.Exception
La levée d'une exception provoque une remontée dans l'appel des méthodes jusqu'à ce qu'un bloc catch acceptant cette exception soit trouvé. Si aucun bloc catch n'est trouvé, l'exception est capturée par l'interpréteur et le programme s'arrête. »



L'appel à une méthode pouvant lever une exception doit :

- soit être contenu dans un bloc try/catch
- soit être situé dans une méthode propageant (throws) cette classe d'exception

Centraliser les traitements d'exceptions …

```
class UneClasse
        {
         private void declencheUneException () throws Exception
         { throw new Exception (); }

         private void methode1 () throws Exception
         { declencheUneException (); }

         private void methode2 () throws Exception
          {
            methode1 ();
            declencheUneException ();
          }

         private void methode3 () throws Exception
          { methode1 ();  }

         public void methodePrincipale ()
          {
            try
              {
                 methode2 ();
                 methode3 ();
              }
            catch (Exception exception)
              {  // codes à exécuter en cas d'exception  }
          }
        }
```

Quand une exception est déclenchée, le système recherche dans la pile d'exécution la première méthode qui traite cette exception dans un bloc catch. L'exemple, permet de centraliser les traitements d'exception dans la méthode methodePrincipale () au lieu de traiter toutes les exceptions qui peuvent survenir dans chacune des méthodes methodeI () où pourrait survenir une exception.

-9-

Les Interfaces sous JAVA

Les Interfaces

- Une interface est une spécification formelle de classe.
- Une interface permet de définir ce que les classes dérivées (les sous-classes) doivent offrir comme méthodes en leur laissant la responsabilité de l'implémentation (pas de contenu).
- Il est possible de définir plusieurs implémentations d'une même Interface.
- Une interface peut hériter (extends) d 'une autre interface
- Une interface est une classe dont toutes les méthodes (vide) sont abstraites (abstract) et les attributs sont final.

il est inutile de spécifier les clauses final et abstract lorsque la classe est déclarée comme interface

```
interface Bipede {
[final] int nbPieds = 2;
[abstract] void UneMethode ();
}
```

Pour hériter d'une interface, il faut l'implémenter.

```
class Homme implements Bipede {
…
}
```

Où il est question d 'appareils électriques ...

- ✓ **On spécifie formellement l'interface – appareilElectrique.**
- ✓ **Deux spécifications de méthodes sont prévues: estEnclenche () , alimente (…)**
- ✓ **Les implémentations utilisateur à venir devront définir le comportement de ces 2 méthodes.**

```
public interface appareilElectrique
{
  public boolean estEnclenche();
  public void alimente (boolean alim);
}
```

teste si l'appareil est enclenche

on appelle cette methode avec *true* lorsque l'on branche l'appareil dans une source de courant active, ou *false* si la source est inactive

Implémentation – radio

radio est une implémentation de l'interface appareilElectrique

```
class radio implements appareilElectrique
{
   final static int freqMiseEnRoute = 1007;    // 100.7 MHz
   int freq;
   boolean allumee = false;
   public boolean estEnclenche() { return allumee; }

   public void alimente(boolean a)
   {
      allumee = a;
      if (allumee)  freq = freqMiseEnRoute;
    }

   public boolean changeFreq (int freq)
   {
      if (!allumee)  this.freq = freq;
      return allumee;
    }
}//fin class radio
```

on ne peut changer de frequence que si la radio est en marche. Retourne true si le changement a ete effectue

Implémentation - lampe

lampe est une implémentation de l'interface appareilElectrique

```
class lampe implements appareilElectrique
{
    boolean allumee = false;
    public boolean estEnclenche ( ) { return allumee; }
    public void alimente (boolean alim) { allumee = alim;}
}
```

Implémentation - rallongeElectrique

```
class rallongeElectrique implements appareilElectrique {
   appareilElectrique appareilBranche = null;
   boolean conduit = false;    // indique si l'interrupteur est ferme (conduit=true)
   boolean estAlimente = false;
public boolean estEnclenche ( ) { return conduit; }

public void alimente (boolean alim)
   {
      if (alim)
      {    // on branche la rallonge dans une prise
         if (!estAlimente && conduit) on();
      }  else   if (estAlimente && conduit) off();  // on debranche
     estAlimente = alim;
   }

 public void enclenche()
 {  if (!conduit && estAlimente) on();
   conduit = true;
 }

public void declenche()
 {  if (conduit && estAlimente) off();
   conduit = false;
 }

private void on ( )
 { if (appareilBranche != null) appareilBranche.alimente(true); }

private void off ( )
 { if (appareilBranche != null) appareilBranche.alimente(false); }

public boolean branche(appareilElectrique app)
 {  if (appareilBranche != null)  return false; // il y a deja un appareil b
   appareilBranche = app;
   return true;
 }
}//fin classe rallongeElectrique
```

rallongeElectrique est une implémentation de l'interface appareilElectrique

gestion de l'interrupteur

gestion de l'appareil branche

branche un nouvel appareil a la rallonge

Exemple d'utilisation - testElectrique

```
class testElectrique {

public static void main (String args[]) {
  rallongeElectrique rallonge1, rallonge2;
  rallonge1 = new rallongeElectrique();
  rallonge2 = new rallongeElectrique();
  radio radioSalon = new radio();
  lampe lampeCuisine = new lampe();

  rallonge1.alimente(true);
  rallonge2.alimente(true);

  rallonge1.branche(radioSalon);
  rallonge2.branche(lampeCuisine);
  System.out.println("la radio du salon est "+ (radioSalon.estEnclenche() ? "allumee" : "eteinte"));
  System.out.println("on appuie sur l'interrupteur de la rallonge 1");
  rallonge1.enclenche();
    System.out.println("la radio du salon est maintenant "+ (radioSalon.estEnclenche() ? "allumee" :
"eteinte"));
 }//fin main
}//fin class testElectrique
```

Créer les rallonges, la radio, la lampe

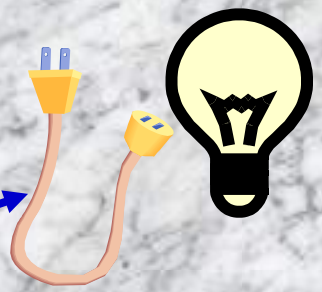
Imaginer que l'on branche les rallonges dans 2 prises

Brancher des appareils dans ces 2 rallonges

Exécution: java testElectrique ↵

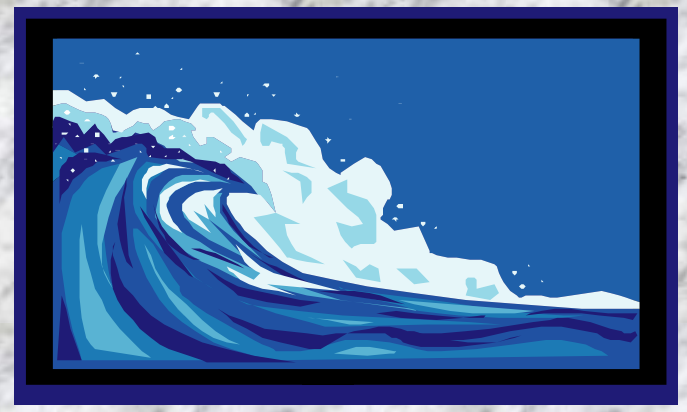
on branche …

```
la radio du salon est eteinte

on appuie sur l'interrupteur de la rallonge
1

la radio du salon est maintenant allumee
```



-10-

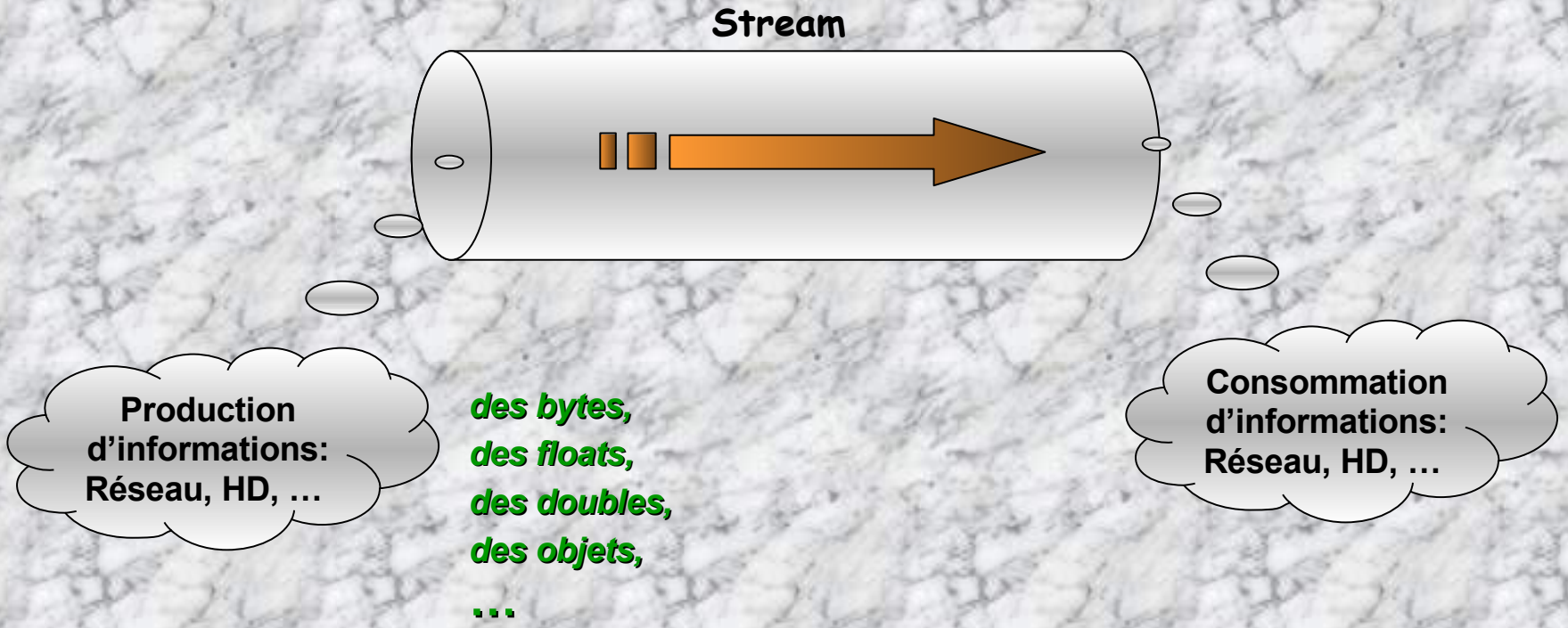
Les Flux (STREAM) sous JAVA



Flux, reflux et golfstream ….

En JAVA, les « vieux » flux d'E/S: stdin, stdout, ont été considérablement généralisés, et spécialisés. Les flux de données sont devenus des classes. On peut donc créer autant d'objet flux que l'on souhaite, afin d'établir des tubes de communication d'E/S

LES FLUX (STREAM)



- ✓ Les flux sont spécialisés, ils ne possèdent pas les mêmes méthodes, et les mêmes types manipulés.
- ✓ Certains flux sont rustiques, d'autres plus riches en méthodes et types.

Les FLUX de base …

```
java.lang.Object  |
                  +--java.io.OutputStream

java.lang.Object  |
                  +--java.io.InputStream
```

- abstract class InputStream
- abstract class OutputStream

- ✓ Ces 2 classes *abstract* qui représentent les socles des flux, ne sont pas instanciables! Elles sont les classes mères de toutes les autres classes définissant les flux spécifiques: FileOutputStream, BufferedOutputStream, DataOutputStream, BufferedInputStream, DataInputStream, …
- ✓ Les méthodes de gestion de ces 2 flux de base sont peu nombreuses; on ne peut y lire ou écrire que des bytes. La plupart du temps, InputStream, et OutputStream seront « wrappée » avec un flux plus riche en types de données, et en méthodes.

Les méthodes des flux de base InputStream, et OutputStream ...

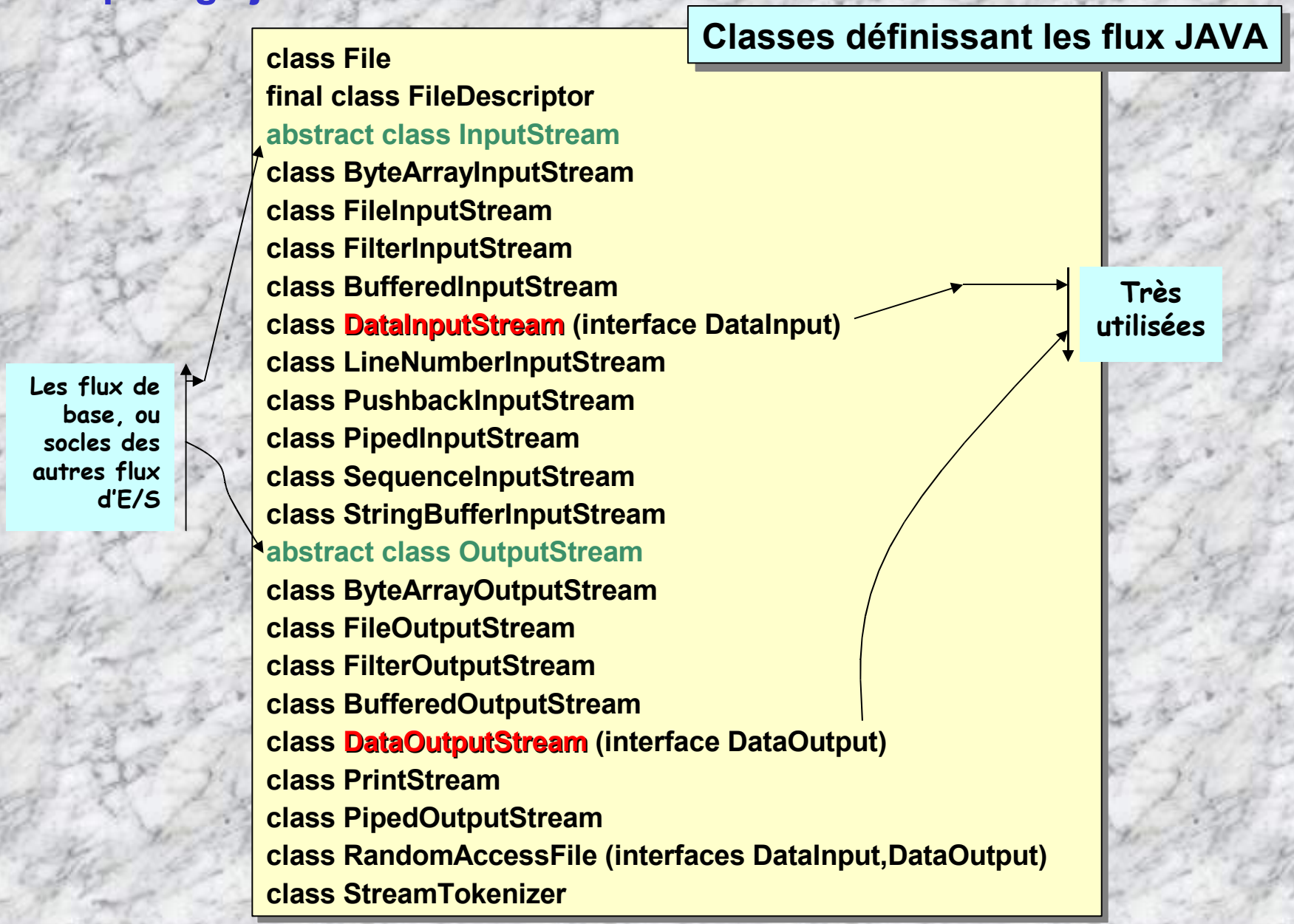
InputStream

int **available** ()	Returns the number of bytes that can be read (or skipped over) from this input stream without blocking by the next caller of a method for this input stream.
void **close** ()	Closes this input stream and releases any system resources associated with the stream.
void **mark** (**int** readlimit)	Marks the current position in this input stream.
boolean **markSupported** ()	Tests if this input stream supports the mark and reset methods.
abstract int **read**()	Reads the next byte of data from the input stream.
int **read** (**byte[] b**)	Reads some number of bytes from the input stream and stores them into the buffer array b.
int **read** (**byte[]** b, **int** off, **int** len)	Reads up to len bytes of data from the input stream into an array of bytes.
void **reset**()	Repositions this stream to the position at the time the mark method was last called on this input stream.
long **skip** (**long** n)	Skips over and discards n bytes of data from this input stream.

OutputStream

void **close** ()	Closes this output stream and releases any system resources associated with this stream.
void **flush** ()	Flushes this output stream and forces any buffered output bytes to be written out.
void **write** (byte[] b)	Writes b.length bytes from the specified byte array to this output stream.
void **write** (byte[] b, int off, int len)	Writes len bytes from the specified byte array starting at offset off to this output stream.
abstract void **write** (int b)	Writes the specified byte to this output stream.

Paquetage java.io des entrées/sorties



- **Interfaces**
  - ✓ Interface DataInput
  - ✓ Interface DataOutput
  - ✓ Interface FilenameFilter

- **Exceptions**
  - ✓ class EOFException
  - ✓ class FileNotFoundException
  - ✓ class IOException
  - ✓ class InterruptedIOException
  - ✓ class UTFDataFormatException

Ce qui signifie que l'on peut définir une nouvelle implémentation de DataInputStream, DataOutputStream, …

Jeu d'exceptions. Choisir une exception pour sécuriser son programme en cas d'usage de flux: DataInputStream, DataOutputStream, …

```
try { //utilisation d'un flux File
   }
catch (FileNotFoundException e) {
      //traite exception
   }
```

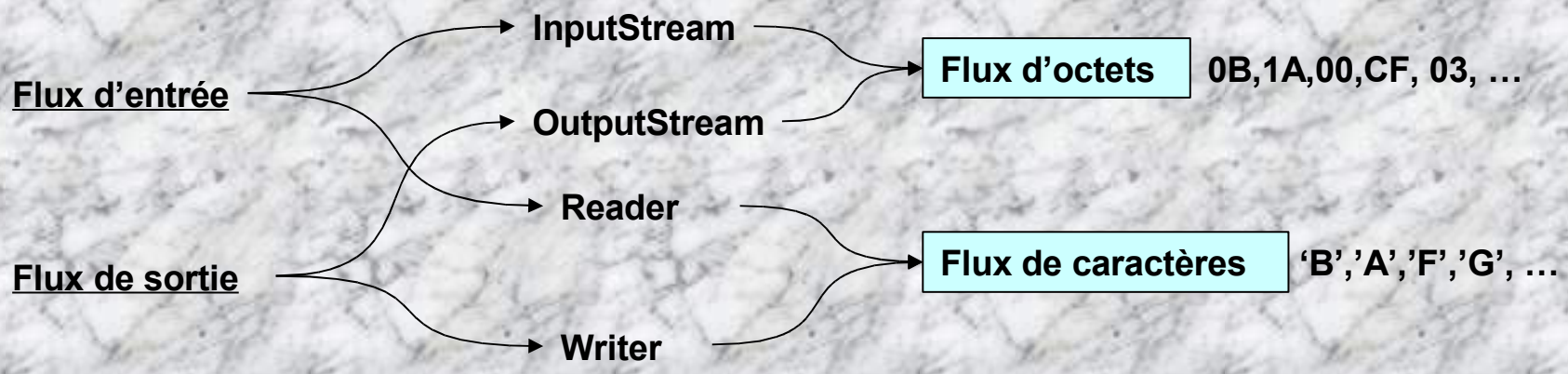
☞ Java distingue les flux binaire et les flux texte.

Flux binaire:
L'information est transmise sans modification de la mémoire au flux, ou du flux à la mémoire.

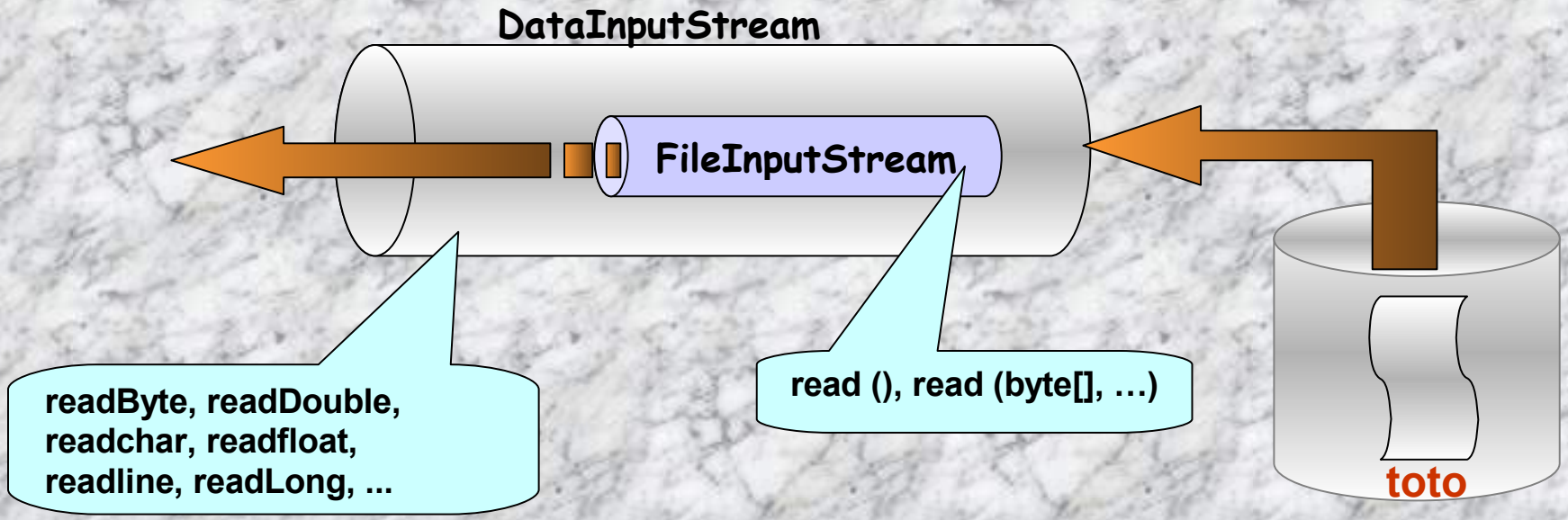
Flux texte:
L'information subit une transformation –formatage- afin que le flux reçoive ou transmette en définitive une suite de caractères.

⚠ **Les flux DataInput(Ouput)Stream, et FileInput(Ouput)Stream sont des flux binaires**

☞ **Le nom des flux se décompose en un préfixe, et un suffixe; Ex: DataInputStream**



- Le programmeur peut-être amené à inclure la gestion d'un stream «simple » à l'intérieur d'un stream plus riche en méthodes, ou plus spécialisé en types des données.
- On dit que l'on « wrap » le stream au sein d'un stream plus adéquat



Fichiers d'entrée:

```
FileInputStream  fis = new FileInputStream("toto");
DataInputStream  d = new DataInputStream( fis );
```

Les flux sont spécialisés, ils ne possèdent pas les mêmes méthodes, et les mêmes types manipulés. Certains flux sont rustiques, d'autres plus riches en méthodes et types

Java.lang.Object

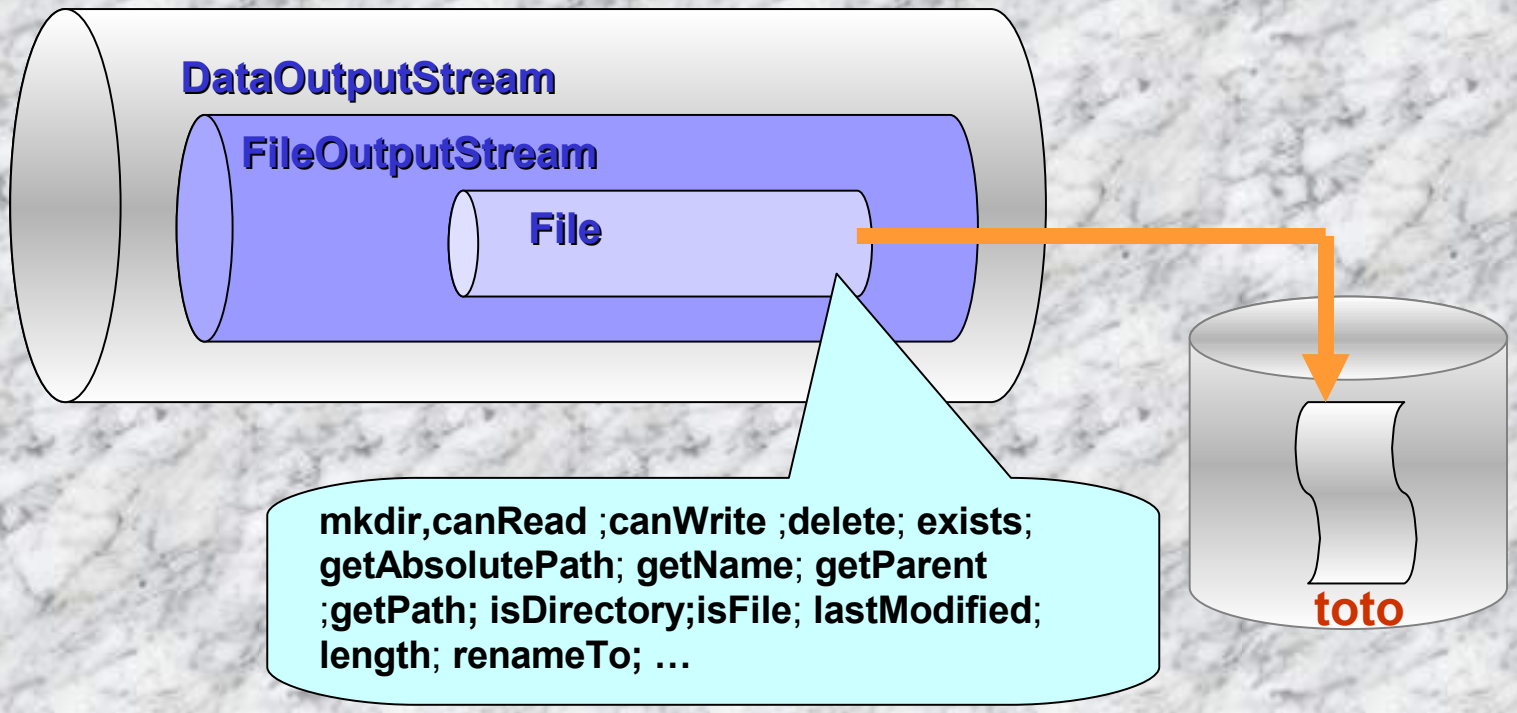
+ -- Java.io.InputStream

+ -- java.io.FileInputStream

int **available()**	Returns nbr of bytes that can be read from the file input stream without blocking.
void close()	Closes this file input stream and releases any system resources associated with the stream.
protected void finalize()	Ensures that the close method of this file input stream is called when there are no more references to it.
FileDescriptorget FD()	Returns the FileDescriptor object that represents the connection to the actual file in the file system being used by this FileInputStream.
int read()	Reads a byte of data from this input stream.
int read (byte[] b)	Reads up to b.length bytes of data from this input stream into an array of bytes.
int read(byte[] b, int off, int len)	Reads up to len bytes of data from this input stream into an array of bytes.
long skip(long n)	Skips over and discards n bytes of data from the input stream.

Fichiers de sortie:

```
File f = new File("toto");
FileOutputStream fos = new FileOutputStream(f);
DataOutputStream d = new DataOutputStream(fos);
```



Java.lang.Object

+ -- Java.io.InputStream

+ -- **java.io.FileInputStream**

Ne lit que des bytes

int available ()	Returns the nbr of bytes that can be read from this file input stream without blocking.
void close()	Closes this file input stream and releases any system resources associated with the stream.
protected void finalize()	Ensures that the close method of this file input stream is called when there are no more references to it.
FileDescriptor getFD()	Returns the FileDescriptor object that represents the connection to the actual file in the file system being used by this FileInputStream.
int read()	Reads a byte of data from this input stream.
int read (byte[] *b*)	Reads up to b.length bytes of data from this input stream into an array of bytes.
int read (byte[] *b*, int *off*, int *len*)	Reads up to len bytes of data from this input stream into an array of bytes.
long skip (long *n*)	Skips over and discards n bytes of data from the input stream.

Java.lang.Object

+ -- Java.io.InputStream

+ -- java.io.FilterInputStream

+ -- **java.io.DataInputStream**

Lit des bytes, char, float, boolean, int, …

Extrait doc Sun)

int **read (byte[] b)**	Reads some number of bytes from the contained input stream and stores them into the buffer array b.
int **read (byte[] b, int off, int len)**	Reads up to len bytes of data from the contained input stream into an array of bytes.
boolean **readBoolean ()**	See the general contract of the readBoolean method of DataInput.
byte **readByte ()**	See the general contract of the readByte method of DataInput.
char **readChar ()**	See the general contract of the readChar method of DataInput.
double **readDouble ()**	See the general contract of the readDouble method of DataInput.
float **readFloat ()**	See the general contract of the readFloat method of DataInput.
void **readFully (byte[] b)**	See the general contract of the readFully method of DataInput.
void **readFully (byte[] b, int off, int len)**	See the general contract of the readFully method of DataInput.
int **readInt ()**	See the general contract of the readInt method of DataInput.

String **readLine ()** **Deprecated.** *This method does not properly convert bytes to characters. As of JDK 1.1, the preferred way to read lines of text is via the BufferedReader.readLine() method. Programs that use the DataInputStream class to read lines can be converted to use the BufferedReader class by replacing code of the form:*

DataInputStream d = new DataInputStream(in); with: **BufferedReader d = new BufferedReader(new InputStreamReader(in));**

long **readLong ()**	See the general contract of the readLong method of DataInput.
short **readShort ()**	See the general contract of the readShort method of DataInput.
int **readUnsignedByte ()**	See the general contract of the readUnsignedByte method of DataInput.
int **readUnsignedShort ()**	See the general contract of the readUnsignedShort method of DataInput.
String **readUTF ()**	See the general contract of the readUTF method of DataInput.
static String **readUTF** (DataInput in)	Reads from the stream in a representation of a Unicode character string encoded in Java modified UTF-8 format; this string of characters is then returned as a String.
int **skipBytes (int n)**	See the general contract of the skipBytes method of DataInput.

Java.lang.Object

+ -- Java.io.OutputStream

+ -- java.io.FilterOutputStream

+ -- **java.io.DataOutputStream**

Ecrit des bytes, char, float, boolean, int, …

Extrait doc Sun)

void **flush()** Flushes this data output stream.

int **size()** Returns the current value of the counter written, the number of bytes written to this data output stream so far.

void **write (byte[] b, int off, int len)** Writes len bytes from the specified byte array starting at offset off to the underlying output stream.

void **write (int b)** Writes the specified byte (the low eight bits of the argument b) to the underlying output stream.

void **writeBoolean (boolean v)** Writes a boolean to the underlying output stream as a 1-byte value.

void **writeByte (int v)** Writes out a byte to the underlying output stream as a 1-byte value.

void **writeBytes (String s)** Writes out the string to the underlying output stream as a sequence of bytes.

void **writeChar (int v)** Writes a char to the underlying output stream as a 2-byte value, high byte first.

void **writeChars (String s)** Writes a string to the underlying output stream as a sequence of characters.

void **writeDouble (double v)** Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first

void **writeFloat (float v)** Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first.

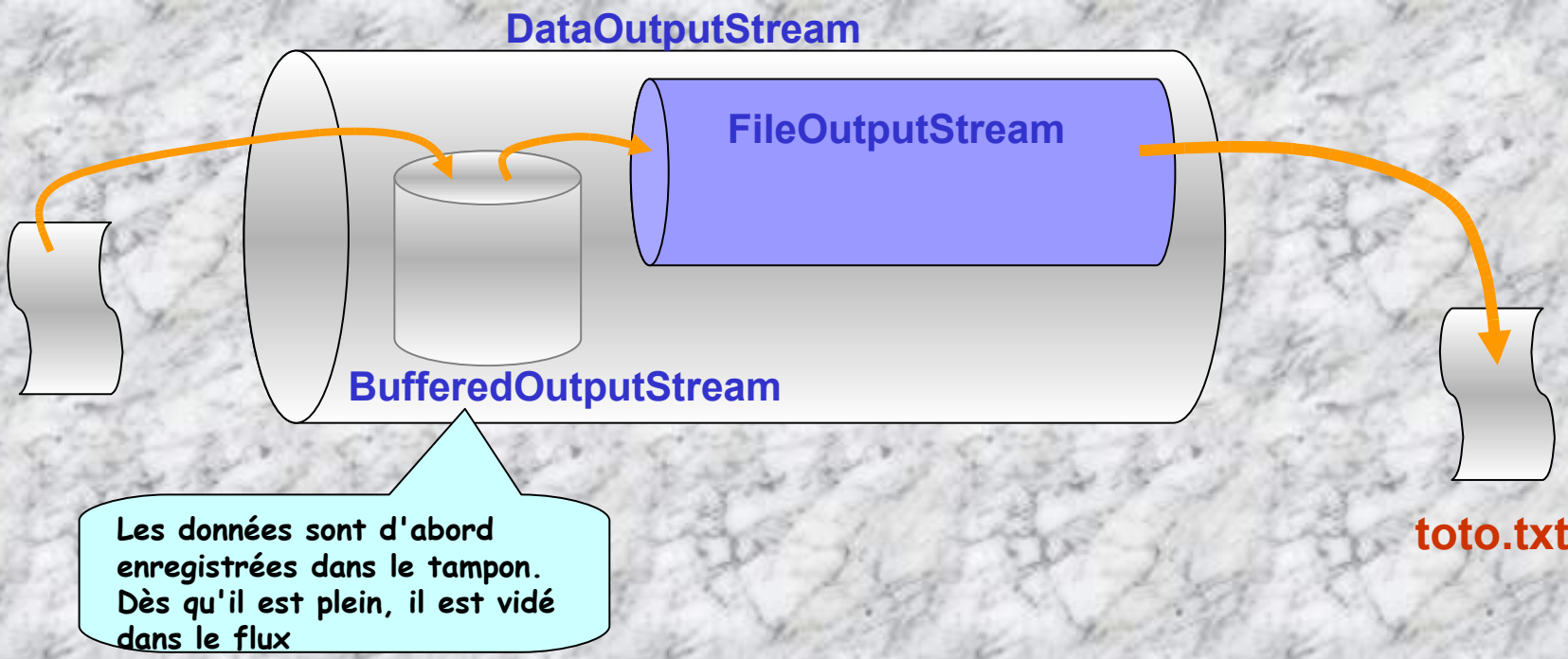
void **writeInt (int v)** Writes an int to the underlying output stream as four bytes, high byte first.

void **writeLong (long v)** Writes a long to the underlying output stream as eight bytes, high byte first.

void **writeShort (int v)** Writes a short to the underlying output stream as two bytes, high byte first.

void **writeUTF (String** str) Writes a string to the underlying output stream using UTF-8 encoding in a machine-independent manner.

On peut aussi doter un flux d'un tampon afin d'optimiser les échanges:



Mise en œuvre ...

```
DataOutputStream sortie = new DataOutputStream
                          (new BufferedOutputStream
                          (new FileOutputStream (toto.txt)));
```

flux standards: Ces flux de base, sont gérés par la classe System

```
public final class java.lang.System extends java.lang.Object {
    // Variables
    public static PrintStream err;
    public static InputStream in;
    public static PrintStream out;
    // Méthodes
    public static void arraycopy (Object src, int src_pos, Object dst, int dst_pos, int length);
    public static long currentTimeMillis ();
    public static void exit (int status);
    public static void gc();
    public static Properties getProperties ();
    public static String getProperty (String key);
    public static String getProperty (String key, String def);
    public static SecurityManager getSecurityManager ();
    public static void load (String filename);
    public static void loadLibrary (String libname);
    public static void runFinalization ();
    public static void setProperties (Properties props);
    public static void setSecurityManager (SecurityManager s);
    }
```

Exemple: System.out.println (System.in.getClass().getName());

résultat : `java.io.BufferedInputStream`

Exemples d'utilisation des flots standards:

Flux standard de sortie

```
import java.io.*;
public class lecture {

static public void main (String[] args)
{
   int x = 0;
   System.out.print("saisie :");
   System.out.flush();

try
 {
    x=System.in.read();
   System.out.println ( System.in.available() + " car. sont en attente." );
 }
catch (IOException e) { System.out.println(e.getMessage());}

    System.out.println("int lu :"  +  x);
    System.out.println("car. lu :"  +  (char) x);

}
```

fin de saisie avec CTRL-D ou CTRL-Z	fin de saisie avec CR

`c:\>java lecture`
`saisie :ABCD3 car. sont en attente.`
`int lu :65`
`car. lu :A`
	`C:\>java lecture`
	`saisie :ABCD`
	`4car. sont en attente.`
	`int lu :65`
	`car. lu :A`

Encore un exemple ...

```
import java.io.*;

public class majuscule2 {
static public void main (String[] args)
{
  int x = 0;
  System.out.print("saisie :");
  System.out.flush();
try {
  do
  {
     x = System.in.read ();
     System.out.println ( Character.toUpperCase((char) x) );
  }while (System.in.available() > 0);

  System.out.println( System.in.available() + " car. sont en attente." );

} catch (IOException e)  System.out.println(e.getMessage());

} //main
} // class
```

fin de saisie avec CTRL-D ou CTRL-Z	fin de saisie avec CR
C:\>java majuscule2	C:\>java majuscule2
saisie :abcdeA B C D E 0 car. sont en attente.	saisie :abcde A B C D E 0 car. sont en attente.

classe File

- ✓ **Permet de manipuler, créer des fichiers, répertoires sur le disque dur …**
- ✓ **Flux souvent « wrapé » avec DataInputStream ou DataOutputStream pour enrichir les opérations d'E/S de données**

```
public class java.io.File extends java.lang.Object {

public final static String pathSeparator;
public final static char pathSeparatorChar;
public final static String separator;
public final static char separatorChar;

public File(File dir, String name);
public File(String path);
public File(String path, String name);

public boolean canRead ();
public boolean canWrite ();
public boolean delete ();
public boolean equals (Object obj);
public boolean exists ();
public String getAbsolutePath ();
public String getName ();
public String getParent ();
public String getPath ();
public int hashCode ();
public boolean isAbsolute ();
public boolean isDirectory ();
public boolean isFile ();
public long lastModified ();
public long length ();
public String[] list ();
public String[] list (FilenameFilter filter);
public boolean mkdir ();
public boolean mkdirs ();
public boolean renameTo (File dest);
public String toString ();
}
```

Les variables

Les constructeurs

Les méthodes

Constructeurs

File (File *dir*, String *name*) : ***dir* est le dossier du fichier, *name* est son nom.**

File (String *name*) : ***name* est le nom du fichier.**

Méthodes principales

boolean **exists()** :**indique si le fichier existe.**

String **getAbsolutePath()** :**renvoie le chemin complet d'accès au fichier sous forme absolue**

String **getName()** :**renvoie le nom du fichier.**

boolean **isDirectory()** :**indique s'il s'agit d'un dossier**

boolean **isFile()** :**indique s'il s'agit d'un vrai fichier**

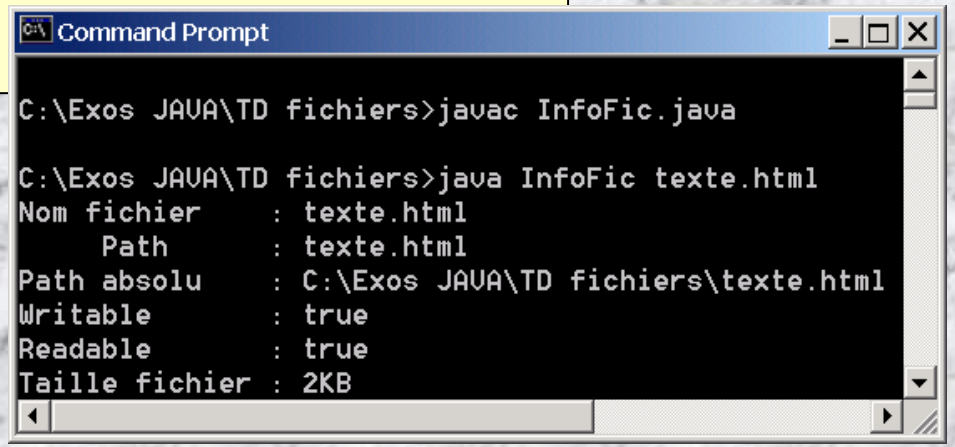
int **length()** :**renvoie la taille du fichier**

String**[]** **list()** :**renvoie la liste des fichiers d'un dossier sous forme de tableau de chaînes.**

...

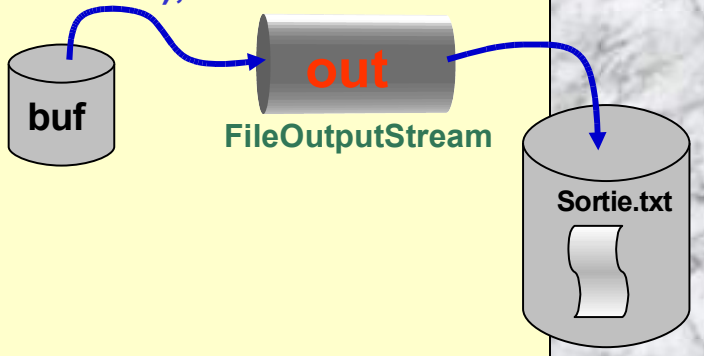
Obtenir des informations sur un fichier …

```
Import java.io.*;

class InfoFic {
  public static void main (String args[]) {

File file = new File(args[0]);
    if (file.exists()) {
      System.out.println("Nom fichier  : " + file.getName());
      System.out.println("     Path        : " + file.getPath());
      System.out.println("Path absolu  : " + file.getAbsolutePath());
      System.out.println("Writable       : " + file.canWrite());
      System.out.println("Readable      : " + file.canRead());
      System.out.println("Taille fichier : " + (file.length() / 1024) + "KB");
    }
    else System.out.println( "Fichier inconnu!");
  }
}
```



Ecrire dans un fichier …

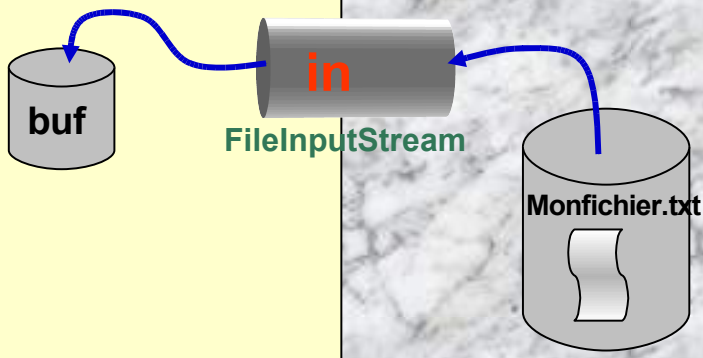
```
class EcritureFichier {
  public static void main (String args[]) {
    // lire le clavier
    byte[ ] buf = new byte[64];
    try {
      System.in.read (buf, 0, 64);
    }
    catch (Exception e) {
      System.out.println("Erreur: " + e.toString());
    }

    // Sortir les données dans le fichier
    try {
          FileOutputStream out = new FileOutputStream( "Sortie.txt");
          out.write(buf);
        }
    catch (Exception e) {
         System.out.println ("Erreur: " + e.toString());
       }
  }
}
```



(d 'après JAVA par M. Morisson, Ed S&SM)

Lire dans un fichier …

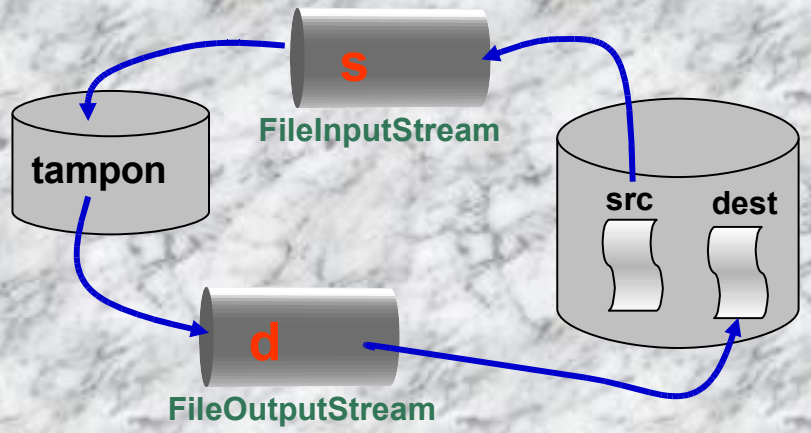
```
class LectureFichier {
  public static void main (String args[]) {
    byte buf[] = new byte[64];
    try {
      FileInputStream in = new FileInputStream( "Monfichier.txt");
      in.read(buf, 0, 64);
    }
    catch (Exception e) {
      System.out.println("Erreur: " + e.toString());
    }
    String s = new String(buf, 0);
    System.out.println(s);
  }
}
```



(d 'après JAVA par M. Morisson, Ed S&SM)

Exemple de copie de fichiers

```
public class CopieFichier {

public static void copie (String src  ,String dest)
{
  FileInputStream s = null;
  FileOutputStream d = null;
  try
  {
   FichierLecture  Source = new FichierLecture(src);
   FichierEcriture  Destination = new FichierEcriture(dest);
   s = new FileInputStream(Source);
   d = new FileOutputStream(Destination);
   byte tampon[] = new byte[1024];
   int lu=0;
   do {
      lu=s.read(tampon);
      if (lu!=-1) d.write(tampon,0,lu);
    }while (lu!=-1);
  }// try
   catch (IOException e) {System.err.println(e.getMessage());}
   finally
   { if (s != null)
     {try {s.close();} catch (IOException e){}}
    if (d != null)
     {try {d.close();} catch (IOException e){}}
   } //finfinally
} //fincopie
```




```
public static void main (String args[])
{
  if (args.length!=2)
   {
     System.err.println("java CopieFichier <src> <dest>");
     return;
   }
copie ( args[0], args[1] );
}//finmain
}//finCopieFichier
```

Quelques autres opérations sur les fichiers …

Créer un fichier

```
try {
    File fich = new File ("filename");
   boolean succes = fich.createNewFile ();  // Créer le fichier s'il n'existe pas
   if (succes) { // Le fichier n'existe pas et a été créé }  else {  // Le fichier existe déjà  }
} catch (IOException e) {  //Traitement du pb à la création fichier }
```

Déplacer un fichier (répertoire) :

```
File fich = new File ("filename");          // Fichier (ou répertoire) à déplacer
File dir = new File ("directoryname");   // Répertoire de Destination
boolean succes = fich.renameTo(new File(dir,  fich.getName()));    // Déplacer le fichier vers le nouveau répertoire
if (!succes) {  // Le fichier n'a pas été déplacé correctement  }
```

Renommer un fichier (répertoire) :

```
File fich = new File("oldname");     // Fichier (ou répertoire) avec un nom à modifier
File fich2 = new File("newname");   // Fichier (ou répertoire) avec un nouveau nom
boolean succes = fich.renameTo( fich2 );  // Renommer le fichier (ou répertoire)
if (!succes) {   //Le fichier n'a pas été renommé correctement  }
```

Flux sérialisable - Ecrire un objet dans un flux

- La sérialisation consiste à pouvoir prendre un objet en mémoire et à en sauvegarder l'état sur un flux de données connecté à un fichier, ou le réseau. Ce concept permettra aussi de reconstruire, ultérieurement, l'objet en mémoire à l'identique de ce qu'il pouvait être initialement. On peut stocker des objets différents dans un flux « serializable »
- Un flux est sérialisable si l'on peut y insérer ou extraire des objets quelconques. On peut se servir de ce dispositif pour stocker des objets dans un fichier , ou bien transmettre par réseau des informations empaquetées sous la forme d'objet.
- La sérialisation peut donc être considérée comme une forme de persistance des données.
- Pour qu'un objet soit sérialisable, il faut que la classe qui lui a donné naissance par instanciation soit libellée comme suit:

  Ou que la classe utilisée soit sérializable …

```
class UneClasse implement Serializable
 {
    …

 }
```

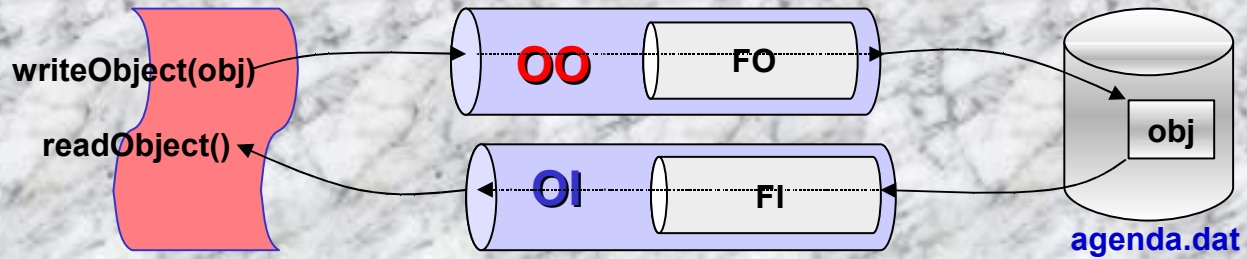
- On utilise les 2 méthodes de l'objet flux sérialisable utilisé:

```
private void writeObject (java.io.ObjectOutputStream out)  throws IOException

private void readObject (java.io.ObjectInputStream in)  throws IOException, ClassNotFoundException;
```

```
class UneClasse implement Serializable
 {
  …

}
```

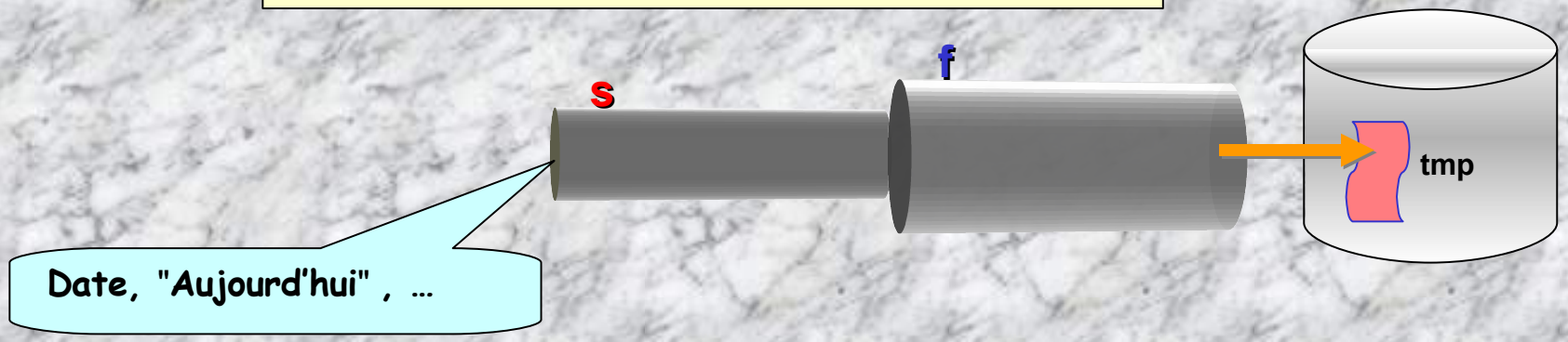
```
class StockeObjet {
 UneClasse Unobjet= new UneClasse();
 public StockeObjet () {
  try{ FileOutputStream FO = FileOutputStream ("agenda.dat");
       ObjectOutputStream OO= new ObjectOutputStream (FO);
       OO.writeObject(Unobjet);
        OO.close()
         …
      }
   catch(IOException e) {"Erreur stockage" }
 }
}
```

```
class DeStockeObjet {
 UneClasse Autreobjet new UneClasse();
public DeStockeObjet () {
  try{ FileInputStream FI = FileInputStream ("agenda.dat");
       ObjectInputStream OI= new ObjectInputStream (FI);
       Autreobjet=(UneClasse)OI.readObject ();
        OI.close()
         …
     }
   catch(IOException e) {"Erreur déstockage" }
 }
}
```



Ecrire des objets et des données primitives (int, char, long, …) sérialisable dans un flux est facile …

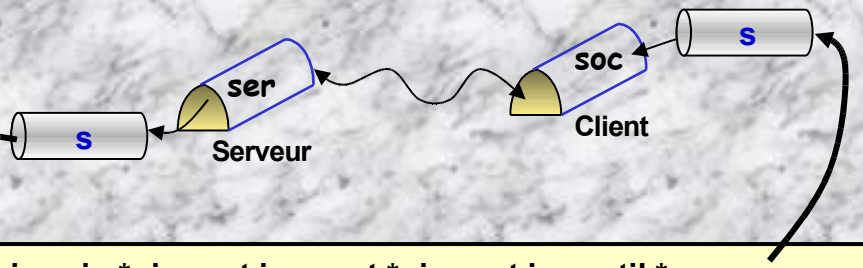
```
// Sérialiser la date courante dans un fichier:

FileOutputStream f = new FileOutputStream("tmp");
ObjectOutput s = new ObjectOutputStream(f);
s.writeObject("Aujourd'hui");
s.writeObject(new Date());
s.flush();
```



Une sérialisation d'objet par le réseau au moyen de socket …

L'exemple ci-dessous montre comment utiliser les socket pour transmettre et recevoir des objets. La classe Serveur reçoit des objets à l'aide d'un readObject(), tandis qu'un Client transmet des objets à l'aide de writeObjet; le Serveur doit démarrer le 1er et attendre que writeObject transmette les données.

(La programmation réseau sera étudiée dans le fascicule N°2)




```
import java.io.*; import java.net.*; import java.util.*;
public class Serveur {
  public static void main (String args[]) {
    //Crée le serversocket et utilise ses flux associés
   // pour recevoir les objets sérialisés
    ServerSocket ser = null; //socket de connexion
    Socket soc = null;         //socket de réception
    String str = null;
    Date d = null;
     try { ser = new ServerSocket (8020);
           //Attente d'une connexion sur le socket.
           soc = ser.accept();
           InputStream o = soc.getInputStream();
          ObjectInput s = new ObjectInputStream(o);
           str = (String) s.readObject ();
           d = (Date) s.readObject ();
           s.close();
           //éditer l'objet reçu
           System.out.println (str);
           System.out.println (d);
     } catch (Exception e) {
            System.out.println (e.getMessage());
            System.out.println("Erreur de serialization");
            System.exit(1);
     }
   }
}
```

```
import java.io.*; import java.net.*; import java.util.*;
public class Client {
  public static void main(String args[]) {

   try {
    //Crée un socket
      Socket soc = new Socket(InetAddress.getLocalHost(), 8020);
   //Sérialise la date d'aujourd'hui dans le outputstream
   //associé au socket
          OutputStream o = soc.getOutputStream();
          ObjectOutput s = new ObjectOutputStream(o);

           s.writeObject ("Date d'aujourd'hui:");
           s.writeObject (new Date());
           s.flush ();
           s.close ();
    } catch (Exception e) {
           System.out.println (e.getMessage());
           System.out.println ("Erreur de serialization");
           System.exit(1);
     }
  }
}
```

La directive transient et la sérialisation …

- Le mot-clé transient est lié à la sérialisation des classes Java
- transient permet d'interdire la sérialisation de certaines variables d'une classe.

classe à sérialiser

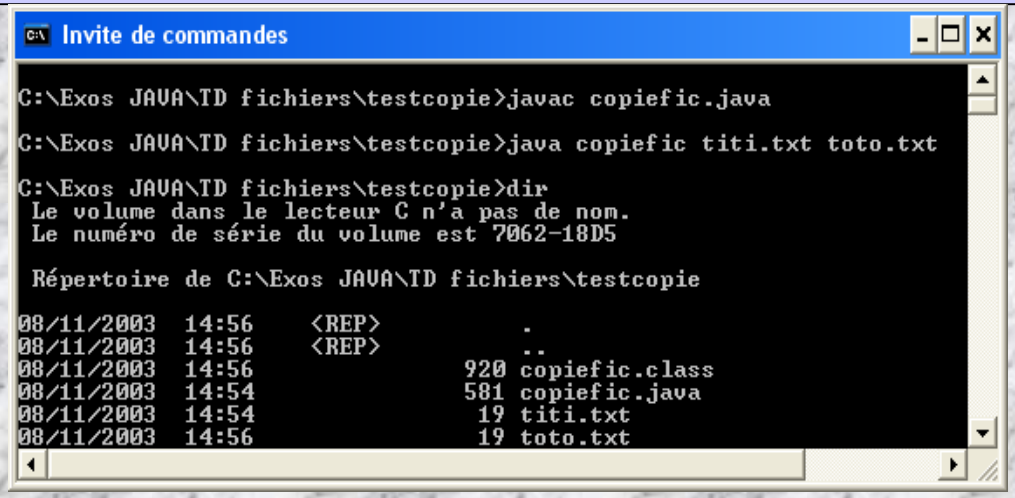
```
class MaClasse  implements java.io.Serializable {
public transient int var1 = 10; //entier transient;  non sauvegardé lors d'une sérialisation
public int var2 = 20;           //entier normal; valeur sauvegardée après sérialisation
}
```

- ✓ Si une instance de cette classe est sérialisée dans un flux, la variable 'var1' ne sera pas sauvegardée,.Lors de la désérialisation elle prendra la valeur 0, malgré la présence de la valeur 10 par défaut. L'attribution d'une valeur par défaut se fait lors de l'instanciation de l'objet
- ✓ La directive transient trouve des applications dès lors qu'une donnée sensible ne doit en aucun cas apparaître dans un fichier. Un mot de passe par exemple. Mais transient peut également permettre de "remettre à zéro" certaines valeurs.

EXERCICES:

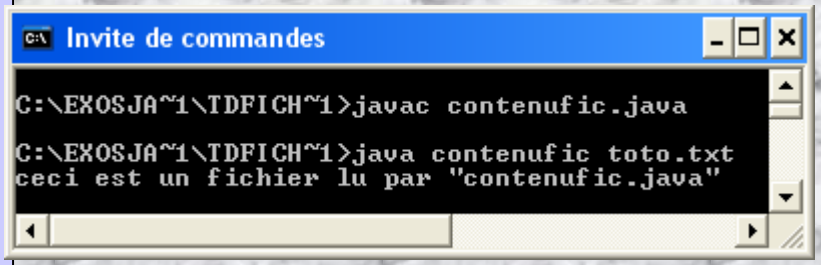
1 – Copie de fichiers

Ecrire un programme qui copie le contenu d'un fichier à l'écran (par défaut) ou dans un autre fichier. Les flots d'entrée et de sortie seront représentés par les variables in de type InputStream et out de type OutputStream. Le programme exigera au moins un paramètre qui sera le nom d'un fichier associé au flot d'entrée. Si un second paramètre est présent il représentera le nom du fichier associé au flot de sortie, sinon on utilisera la console écran représentée par System.out.



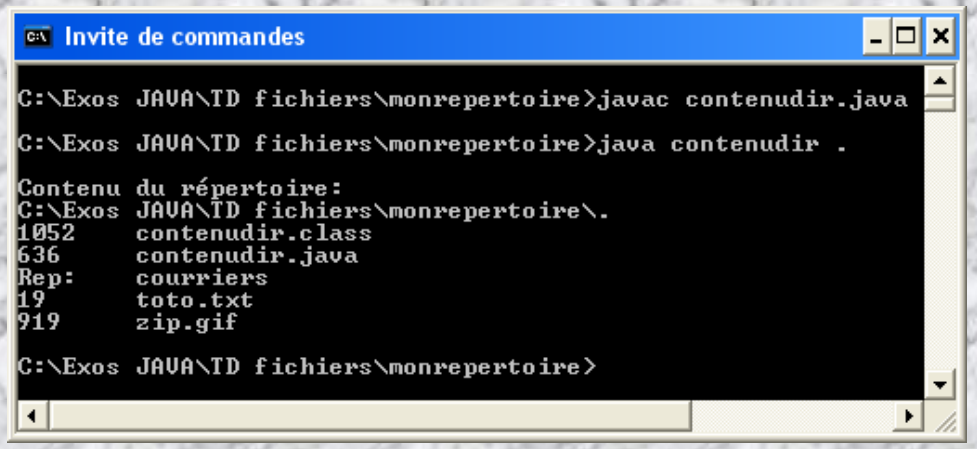
2 - Afficher le contenu d'un fichier

Ecrire un programme qui affiche le contenu du fichier texte dont le nom est passé en paramètres. NOTA: Pour gérer les fichiers de type texte utiliser des objets des classes InputStreamReader et OutputStreamWriter. Ces objets sont respectivement associés à des flots d'octets de type InputStream et OutputStream, ils ont pour tâche de convertir les octets en caractères. Pour pouvoir lire des lignes entières de texte utiliser un objet de la classe BufferedReader associé à un InputStreamReader. Cet objet utilise un tampon (buffer) qui minimise le nombre d'accès au fichier. Utiliser la méthode **readLine()** de la classe BufferedReader qui renvoie une chaîne de caractères ou null lorsque la fin du flot est atteinte.



3 – Contenu d'un répertoire

Ecrire le programme **contenudir** qui affiche le contenu d'un dossier



4 – Créer un fichier stockant sur disque dur des objets

Ecrire l'application **Fiche** permettant de saisir, puis stocker sur le disque dur des informations de type coordonnées d'une personne. Ces informations seront packagées dans des objets distincts/ personne.

4.1) Créer une interface de saisie permettant de rentrer le nom, le prénom, le N° de téléphone, l'@ mail, la profession d'une personne. Insérer ces informations dans un objet, en y ajoutant sa date de création. Tester la viabilité de cette interface.

4.2) Ajouter à l'interface définie en 4.1 la partie stockage des objets sur le disque dur, dans un fichier **<nom fichier>.agd** où agd est le type des fichiers agenda créés. Tester la viabilité du programme créé.

4.3) Créer une interface d'affichage permettant d'afficher le nom d'une personne recherchée, dans un fichier sélectionné de type **agd**

4.4) Fusionner les programmes 4.2 et 4.3 en construisant une nouvelle application permettant, soit de créer, et saisir des informations, soit de rechercher une personne en donnant son nom complet ou un groupe de personnes en indiquant les 1ière lettres du patronyme.

Exercice1 (corrigé) – Copie de fichiers

```
import java.io.*;
public class copiefic {

 public static void main (String args[]) {
  InputStream in=System.in;
  OutputStream out=System.out;
  int c=0;

  if (args.length==0) {
   System.out.println("syntaxe: java copiefic <nomfic1>
[<nomfic2]");
   System.exit(1);
  }
  try {
   if (args.length>0) in=new FileInputStream(args[0]);
   if (args.length>1) out=new FileOutputStream(args[1]);
   while ((c=in.read())!=-1) out.write(c);
   in.close();
   out.close();
  } catch (IOException e) { System.out.println(e.toString());}
 } //finmain
} //finclass
```

Exercice2 (corrigé) - Afficher le contenu d'un fichier

```
import java.io.*;
public class contenufic {
 public static void main (String args[]) {
  if (args.length==0) {
   System.out.println("syntaxe: java <nomfichier>");
   System.exit(1);
  }
  try {
   InputStream ficin=new FileInputStream(args[0]);
   InputStreamReader ficinr=new InputStreamReader(ficin);
   BufferedReader ficinbuf=new BufferedReader(ficinr);
   String ligne;
   while ((ligne=ficinbuf.readLine())!=null)
    System.out.println(ligne);
   ficinbuf.close();
  }
  catch (Exception e) {
   System.out.println(e.toString());
  }
 }
}
```

Exercice3 (corrigé) – Contenu d'un répertoire

```
//Afficher le contenu d'un repertoire
import java.io.*;
public class contenudir {
 public static void main(String[] args) {
  if (args.length == 0) {
        System.out.println("Syntaxe: java <path repertoire>");
        System.exit(1);
       }
  File f=new File(args[0]);
  System.out.println("\nContenu du r,pertoire:");
  System.out.println(f.getAbsolutePath());
  String[] liste=f.list();
  for (int i=0; i<liste.length; i++) {
    File ff=new File(liste[i]);
    if (ff.isDirectory()) System.out.println("Rep: \t"+liste[i]);
     else System.out.println(""+ff.length()+" \t"+liste[i]);
    }
  }
}
```

Exercice4 (corrigé) – Créer un fichier stockant sur disque dur des objets

Exercice4 (corrigé) – Créer un fichier stockant sur disque dur des objets (suite)

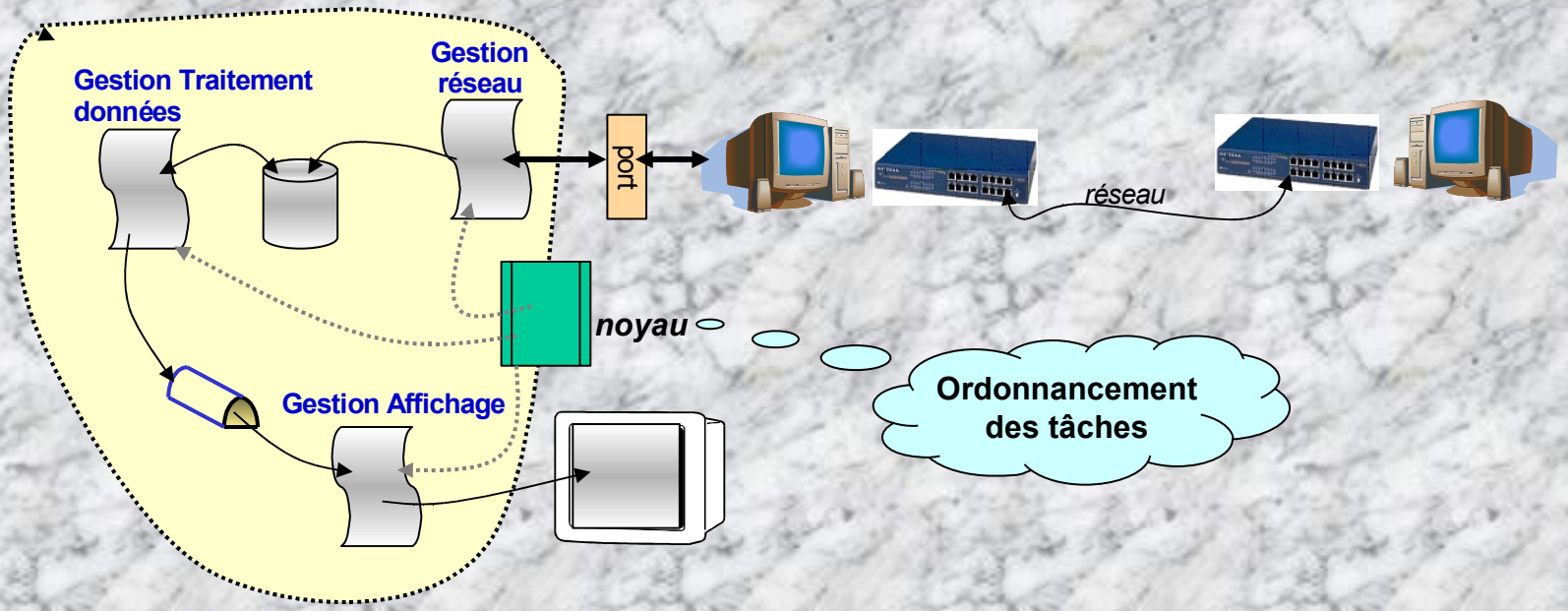
-11-

La programmation concurrente sous JAVA

-Les threads-

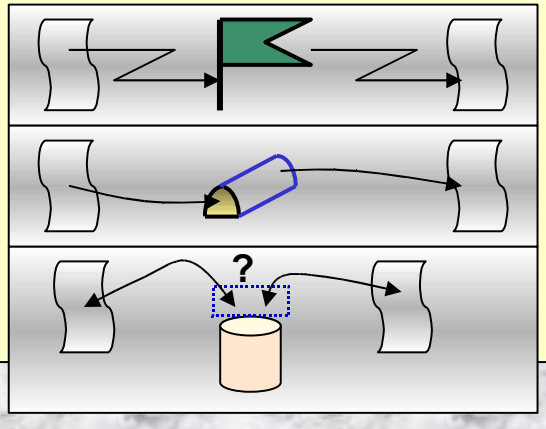
Dans ce chapitre, on étudie l'aptitude de JAVA au multiprocessing. C'est à dire sa capacité de concevoir une application mutitâches, où l'on a répartie dans plusieurs classes les activités procédurales.

Une application informatique peut-être découpée en plusieurs tâches indépendantes -thread- dont le contrôle et l'ordonnancement sont gérés par un « scheduler », ou « noyau ». Chacune d'entre elles prend en charge une fonction de commande ou de contrôle spécifique.

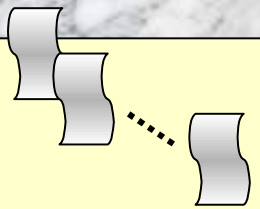


Un noyau doit pouvoir gérer une application multi-thread selon 3 mécanismes fondamentaux:

- ✓ **La synchronisation entre threads**
- ✓ **La communication entre threads**
- ✓ **L'exclusion mutuelle sur ressource partagé**



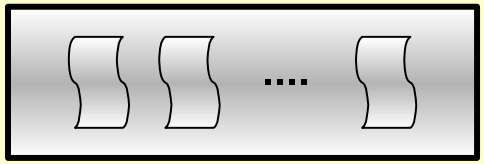
Les threads JAVA



- Les **threads** sont des processus indépendants. Aussi appelés Tâches

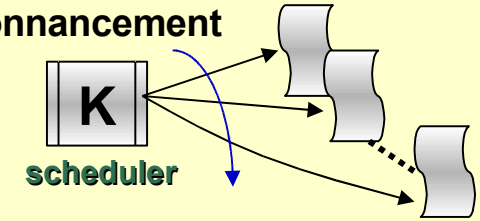
- Les **threads** permettent d'exécuter plusieurs programmes indépendants les uns des autres. Ceci permet une exécution parallèle de différentes tâches de façon autonome.

- Les **threads** permettent de construire des applications multitâche, avec des concepts issus des techniques du parallélisme.



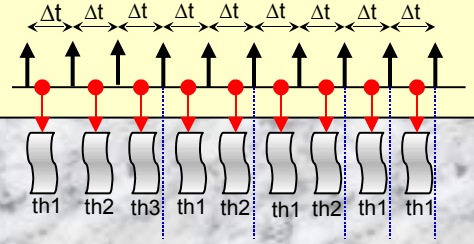
- Les **threads** JAVA s'apparentent aux threads UNIX
  (processus légers)

- Les **threads** JAVA sont gérés par un scheduler (noyau) non préemptif. SUN ne précise pas le mécanisme d'ordonnancement



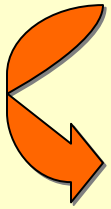
- Les **threads** JAVA sont définis par le paquetage java.lang.Thread et l'interface Runnable

- Les **threads** JAVA sont ordonnancés en temps partagés



- Lors de la compilation de source JAVA, un ordonnanceur (scheduler, noyau) simpliste est implémenté dans l'application.

- L'implémentation et le comportement de l'ordonnanceur de processus (scheduler) n'est pas spécifié par Sun..



Les différentes machines virtuelles Java n'auront pas forcément le même comportement

- Une JVM peut se comporter comme un noyau temps réel (pas de timeslicing) ou comme un noyau préemptif

- Les Threads peuvent partager des données  Problème d 'exclusion mutuelle!

- Les Threads peuvent exécuter des activités à certains instants

Il faut synchroniser les sections critiques: méthodes et blocs de codes synchronisés,

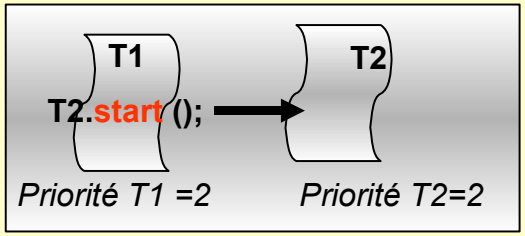
PRIORITES DES THREADS

☞ **Chaque thread possède une priorité comprise entre 1 (plus petite priorité) et 10 (plus grande priorité)**

Thread.MIN_PRIORITY ≤ thread JAVA ≤ **Thread.MAX_PRIORITY**

☞ **Par défaut, un thread reçoit la priorité Thread.NORM_PRIORITY (de valeur = 5)**

☞ **Chaque thread hérite de la priorité du thread qui l'a créé.**

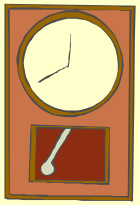


PLANIFICATION DES THREADS

☞**JAVA intègre un exécutif permettant de planifier l'ordre dans lequel s'exécute les différents threads**
composants une application multitâche.

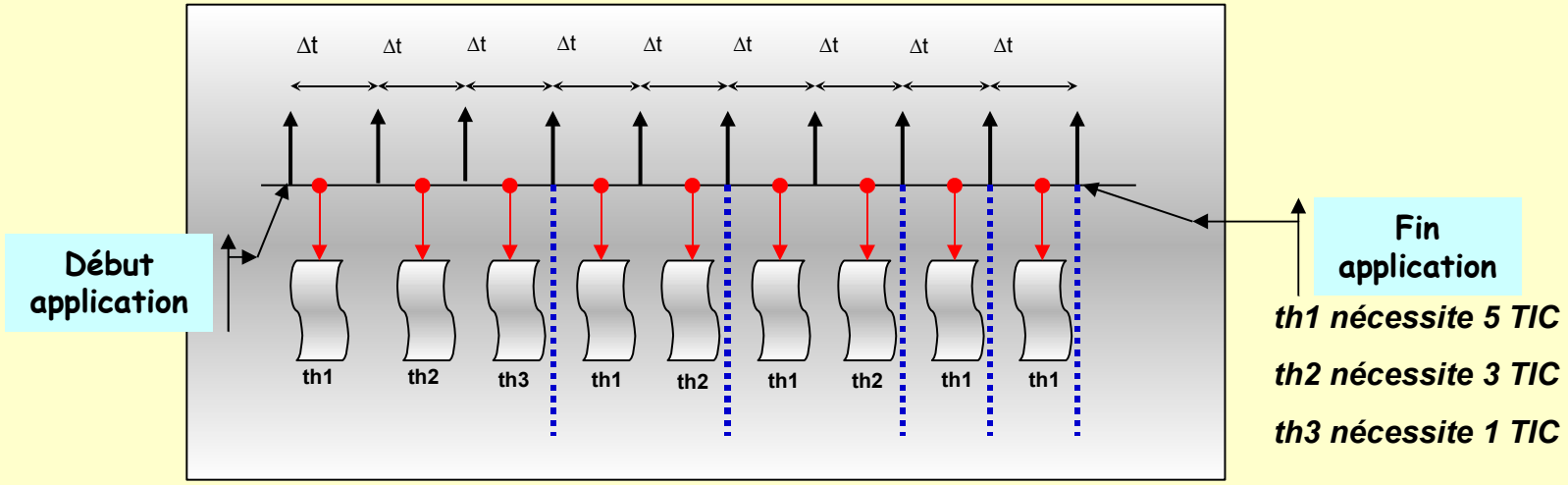
☞**La tâche de l'ordonnanceur consiste à sélectionner le thread de plus haute priorité pour lui affecter**
le CPU.

☞**Les plateformes JAVA ne sont pas toutes identiques au niveau du principe d'ordonnancement, certaines utilisent le principe de « temps partagé », d'autres plus rustiques se limite simplement à aider la commutation de threads sur la demande explicite de ceux-ci.**



Plateforme JAVA avec découpage de temps (Time-sharing):

Chaque thread reçoit une tranche de temps (un TIC horloge)identique durant laquelle il effectue son traitement. A l'expiration de ce TIC, même si le thread n'a pas terminé son travail, l'ordonnanceur JAVA préempte le CPU au profit d'un autre thread de priorité identique. Processus appelé « Round-Robin »



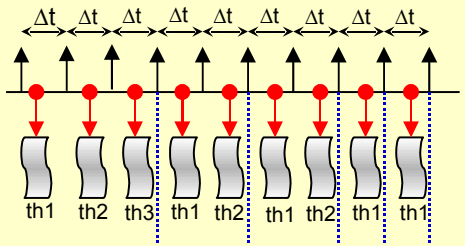
Plateforme JAVA sans découpage de temps:

Chaque thread d'un groupe de threads de priorités identiques s'exécute jusqu'à son achèvement, ou bien qu'il ne quitte volontairement l'état actif pour rentrer dans un état suspendu ou d'arrêt définitif.

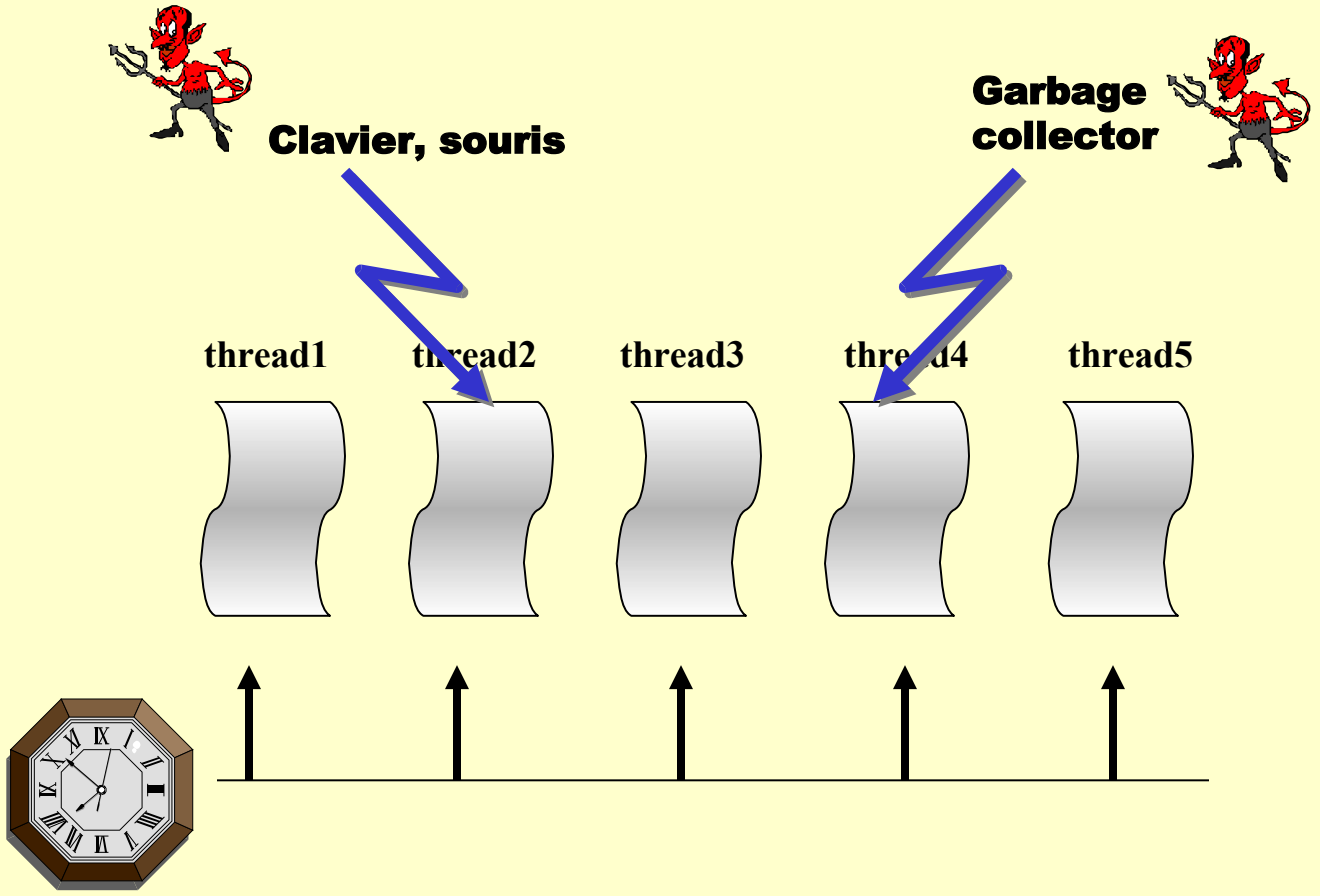
Δt : TIC horloge (Time Interrupt Clock)

Temps partagé:

Allocation d'une tranche de temps Δt pour l'exécution de chaque thread.



☞Parmi les threads, certains ne sont pas lancés par l 'utilisateur, ce sont des processus deamon: thread captant les évènements clavier, souris, … thread garbage collector, ...



✧ **deameon** : « **disk and execution monitor ».** Un démon est un processus s'exécutant en tache de fond. Il se lance au démarrage d'un système d'exploitation et attend la validation des certaines conditions pour effectuer une ou plusieurs actions. La plupart des démons sont transparents pour les utilisateurs.

Création d 'un Thread

On peut créer un thread à partir d 'une dérivation du paquetage java.lang.Thread

```
import java.lang.Thread
public class UnTread  extends Thread
 {
   public void run
     {
       …
     {
}
```

Instancier un Thread

```
class MonProgramme
 {
public static void main ( String argv[] )
  {
   UnThread  T1 = new UnThread();
   T1.start ();          //démarrer un thread
   T1.stop ();           //arrêter un thread
   T1.suspend ();        //suspendre un thread
   T1.resume ();         //reprendre l 'exécution d 'un thread
   …
  }
}
```

Les principales méthodes d 'un Thread

Le noyau JAVA est un automate logiciel

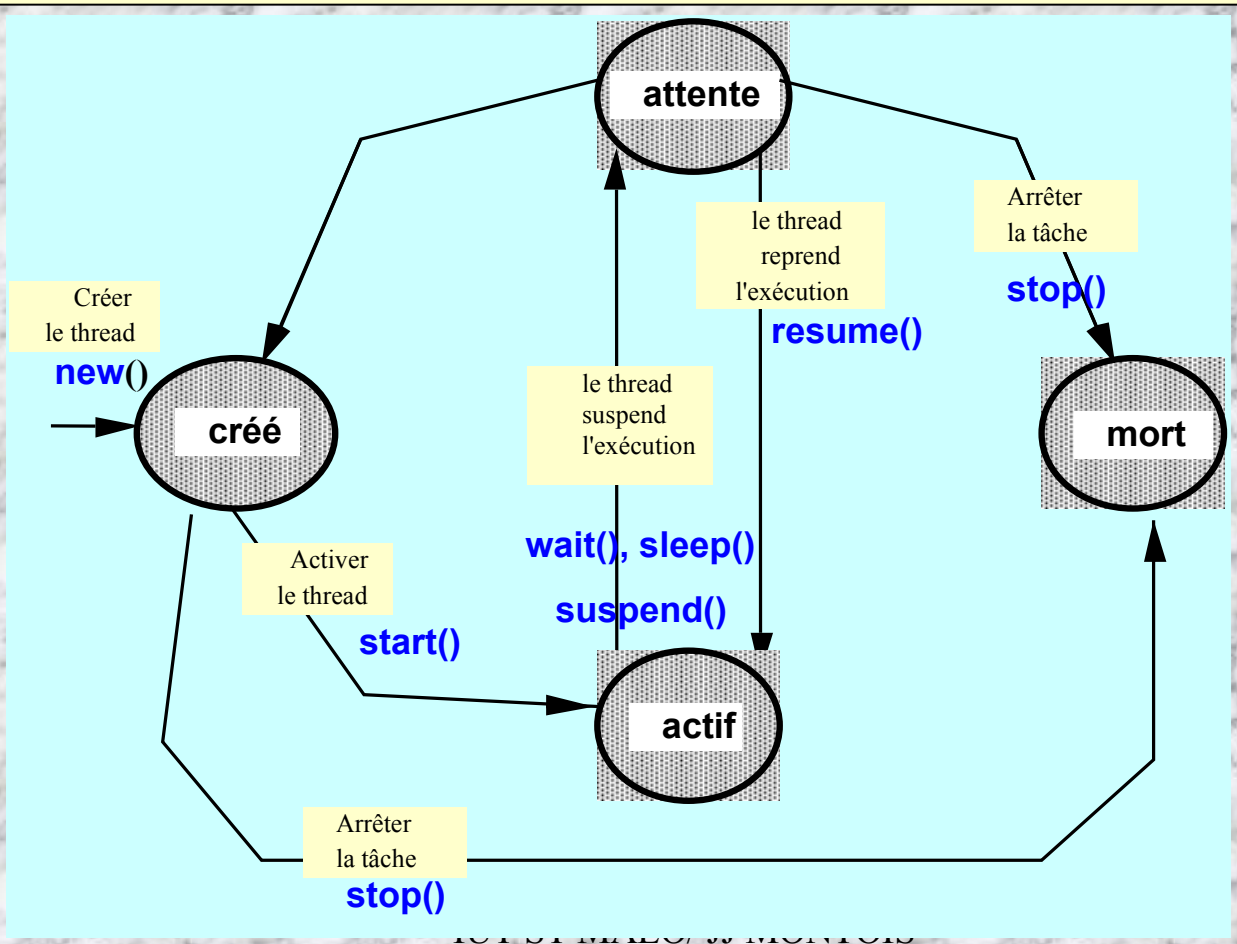
L'automate comprend 4 états, ce sont les états que peuvent prendre un thread.

ATTENTE: Le thread attend un évènement: restitution de ressource, synchronisation, échéance, …

CREE: Le thread est démarré, il peut déployer ses activités si sa priorité est la plus haute.

ACTIF: Le thread de plus haute priorité est en exécution, le CPU lui est alloué

MORT: Le thread n'est plus nécessaire, il ne participe plus à l'activité.



D ’autres méthodes de gestion des threads ...

destroy()	**Arrêt brutal du thread**
interrupt()	**Permet d'interrompre les différentes méthodes d'attente en appelant une exception**
sleep()	**Met en veille le thread (pour l’animation)**
stop()	**Arrêt non brutal du thread**
suspend()	**Arrêt d'un thread en se gardant la possibilité de le redémarrer par la méthode resume()**
resume()	**Reprise de l’activité du thread**
wait()	**Met le thread en attente une certaine durée**
yield()	**Donne le contrôle au schéduleur (non préemptif)**

La classe java.lang.Thread (Extrait de la doc SUN) :

activeCount()	Returns the current number of active threads in this thread group.
checkAccess()	Determines if the currently running thread has permission to modify this thread.
countStackFrames()	Counts the number of stack frames in this thread.
currentThread()	Returns a reference to the currently executing thread object.
destroy()	Destroys this thread, without any cleanup.
dumpStack()	Prints a stack trace of the current thread.
Enumerate(Thread[])	Copies into the specified array every active thread in this thread group and its subgroups.
getName()	Returns this thread's name.
getPriority()	Returns this thread's priority.
getThreadGroup()	Returns this thread's thread group. interrupt() Interrupts this thread.
interrupted()	Tests is the current thread has been interrupted.
isAlive()	Tests if this thread is alive.
isDaemon()	Tests if this thread is a daemon thread.
isInterrupted()	Tests if the current thread has been interrupted.
join()	Waits for this thread to die.
join(long)	Waits at most millis milliseconds for this thread to die.
join(long, int)	Waits at most millis milliseconds plus nanos nanoseconds for this thread to die.
resume()	Resumes a suspended thread.
run()	If this thread was constructed using a separate Runnable run object, then that Runnable object's run method is called; otherwise, this method does nothing and returns.

(Suite …)

setDaemon(boolean)	Marks this thread as either a daemon thread or a user thread.
setName(String)	Changes the name of this thread to be equal to the argument name.
setPriority(int)	Changes the priority of this thread.
sleep(long)	Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.
sleep(long, int)	Causes the currently executing thread to sleep (cease execution) for the specified number of milliseconds plus the specified number of nanoseconds.
start()	Causes this thread to begin execution; the Java Virtual Machine calls the run method of this thread.
stop()	Forces the thread to stop executing. stop(Throwable) Forces the thread to stop executing.
suspend()	Suspends this thread.
toString()	Returns a string representation of this thread, including the thread's name, priority, and thread group.
yield()	Causes the currently executing thread object to temporarily pause and allow other threads to execute.

Exemple, Thread et interface Runnable: *DIP Genève, Alexandre Maret & Jacques Guyot*

```
class afficheur extends Thread
{
   public afficheur (String  s) {
      super(s);
   }

   public void run()
   {
      while (true)
      {
         System.out.println("je suis le processus"  + getName());
         Thread.yield();    // passe le controle
      }
   }
}
```

constructeur permettant de nommer le processus

affiche son nom puis passe le contrôle au suivant

Exécution ??

```
class thread12
{
public static void main(String args[])
 {
       afficheur thread1 = new afficheur("1");
       afficheur thread2 = new afficheur("2");

       thread1.start();
       thread2.start();

       while (true)
      {
          System.out.println("je suis la tache principale !");
          Thread.yield();
       }
  }
}
```

Rend le CPU

```
je suis la tache principale !
je suis le processus 1
je suis le processus 2
je suis la tache principale !
je suis le processus 1
je suis le processus 2
je suis la tache principale !
je suis le processus 1
je suis le processus 2
je suis la tache principale !
je suis le processus 1
je suis le processus 2
je suis la tache principale !
je suis le processus 1
je suis le processus 2
je suis la tache principale !
je suis le processus 1
je suis le processus 2
je suis la tache principale !
je suis le processus 1
je suis le processus 2
je suis la tache principale !
je suis le processus 1
je suis le processus 2
```

Course de robots (*d'après JF Macary & C. Nicolas; Programmation JAVA*)

```
class CourseRobot
{
    static Robot rob1, rob2;
   public static void main (String arg[])
   {
    rob1 = new Robot(10, "r1");
    rob2 = new Robot(5,  "r2");
    rob1.run();
    rob2.run();
   }

class Robot
{ int vitesse;
  String nom;
  public Robot (int v, String n) { vitesse = v; nom = n;}
}
public void sleep (int tempo) { for (in i = 0; i<tempo; i++);}

public void run ()
   {
     for (int i=0; i < 10; i++)
       { System.out.print (nom);
         sleep (1000/vitesse);
       }
     System.out.println(": "+ nom + " est arrivé!");
   }//finrun
}//finCourseRobot
```

Lancement des robots

Approche séquentielle du parallélisme

Question: quel robot arrivera le premier? cela était-il prévisible? Imaginer le résultat de l'exécution de ce programme. Peut-on vraiment parler de parallélisme?

Approche concurrente du parallélisme

```
class CourseRobot
{
 static Robot rob1, rob2;
 public static void main (String arg[]) throws InterruptedException
  {
    rob1 = new Robot(10, "r1");
    rob2 = new Robot(5,  "r2");
    rob1.start();
    rob2.start();
    rob1.join();
    rob2.join();
  }
class Robot extends Thread
{ int vitesse;
  String nom;
  public Robot (int v, String n) { vitesse = v; nom = n;}
}
public void run ()
   {
     for (int i=0; i < 10; i++)
       { System.out.print (nom);
         try {
         sleep (1000/vitesse);
         }catch (Exception e) {;}
       }
     System.out.println(": "+ nom + " est arrivé!");
   }//finrun
}//finCourseRobot
```

Lancement robots

Attendre la mort des threads


```
r1 r2 r2 r2 r2 r2 r1 r2 r2 r2 r2 r2
r2 est arrivé!
r1 r1 r1 r1 r1 r1 r1 r1
r1 est arrivé!
```

une petite pause culturelle …

A propos de robots, concernant les fameuses 3 lois de la robotique édictées par Isaac Asimov dès les années 1958. Lois qui ont perturbées le fonctionnement de HAL dans « 2001 l'odyssée des étoiles », on avait, en effet, trafiqué ses directives. Voir « 2010 » …

Asimov crée une éthique du robot …

PREMIERE LOI
Un robot ne peut nuire à un être humain ni laisser sans assistance un être humain en danger.

DEUXIEME LOI
Un robot doit obéir aux ordres qui lui sont donnés par les êtres humains, sauf quand de tels ordres sont incompatibles avec la Première loi

TROISIEME LOI
Un robot doit protéger sa propre existence tant que cette protection n'est pas incompatible avec à la Première ou à la Deuxième loi.

Auxquelles ont ajoute aujourd'hui la Zéroième Loi:

Un robot ne peut blesser l'humanité, ou par son inaction, permettre que l'humanité soit blessée.

Loi citée par R. Giskard Reventlov et R. Daneel Olivaw dans « Les robots et l'Empire » Isaac Asimov, J'ai lu, 1985

Interface Runnable

☞ D 'une façon directe, on crée des thread à partir de la dérivation de la class thread.

Problème:

- Créer des threads dérivant d 'une autre classe, notre classe application par exemple?
- La méthode Thread.stop() est qualifiée de final. On ne peut donc pas la remplacer.

Solution:

Implémenter l'interface Runnable !

```
public class MonThread  extends MaSurClass implements Runnable
 {
   public void run
    {
      ...
    {
}
```

```
class afficheurRunnable implements Runnable {
   boolean cont = true;
   String nomProcessus;
   Thread th;

   public afficheurRunnable(String s) {
      nomProcessus = s;
   }
   public void start() {
      if (th == null) {
         th = new Thread(this,nomProcessus);
         th.start();
      }
   }
   public void stop() {
      if (th != null) {
         th.stop();
         th = null;
      }
   }
   public void run() {
      while (cont) {
         System.out.println("je suis le processus "+th.getName());
         Thread.yield();    // passe le controle
      }
   }
}
```

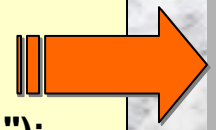
constructeur permettant de nommer le processus

On définit la méthode start

On définit la méthode stop

On définit la méthode run qui donne le corps de la tâche

```
class runnable12 {

   public static void main(String args[]) {

      afficheurRunnable run1 = new afficheurRunnable("1");
      afficheurRunnable run2 = new afficheurRunnable("2");

      run1.start();
      run2.start();

      while (true) {
         System.out.println("je suis la tache principale !");
         try {
            Thread.sleep(20);
         } catch (InterruptedException e) { }
      }
   }
}
```



Exécution ??

```
je suis la tache principale !
je suis le processus 1
je suis le processus 2
je suis le processus 1
je suis le processus 2
je suis le processus 1
je suis le processus 2
je suis le processus 1
je suis le processus 2
je suis la tache principale !
je suis le processus 1
je suis le processus 2
je suis le processus 1
je suis le processus 2
je suis le processus 1
```

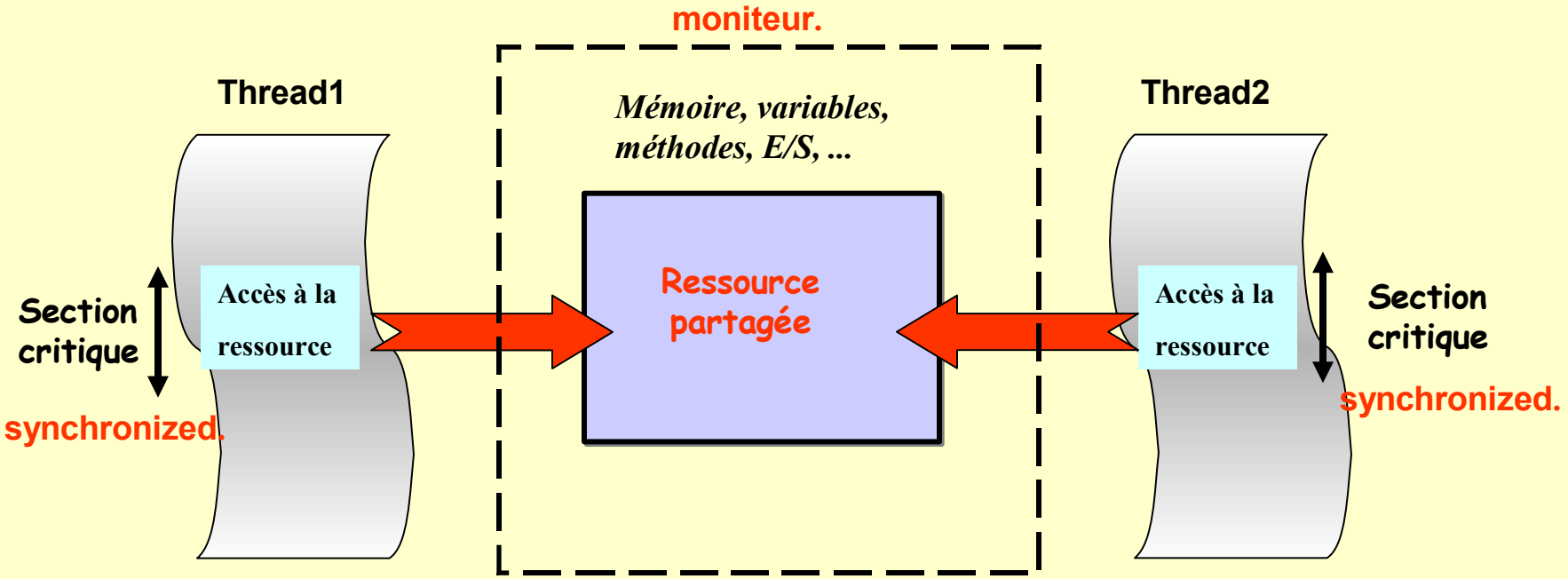
Alors, faire dériver de la classe java.lang.Thread, ou implémenter l'interface Runnable ?

	Avantages	Inconvénients
extends java.lang.Thread	Chaque thread a ses données qui lui sont propres.	On ne peut hériter d'une autre classe.
Implements java.lang.Runnable	L'héritage reste possible. On peut implémenter autant d'interfaces souhaitées.	Les données de la classe sont partagés par tous les threads qui y sont basés. Parfois, cela peut-être souhaité.

```
public class thread2 implements Runnable {
  thread2()
  {
    Thread t = new Thread (this);
    t.start();
  }
  public void run()
  { while (true)
    {
      try
      { Thread.sleep(3000); }
     catch (InterruptedException ie) { }
      System.out.println ("R&T ST MALO !");
    }
  }
  public static void main(String[] args)
    {  thread2 essai = new thread2(); }
}
```

```
public class thread1 {
  static class MonThread extends Thread {
        public void run()
          { while (true)
                {
          try
            { Thread.sleep(3000); }
                  catch (InterruptedException ie) { }

          System.out.println ("R&T ST MALO  !");
       }
          }
}//finclass

  public static void main(String[] args)
    {
      MonThread t = new MonThread();
      t.start();
    }
}//finclass
```

Partage de ressources – Exclusion mutuelle sur ressource partagée



☞ Le mécanisme d'exclusion mutuelle présent dans Java est le moniteur.

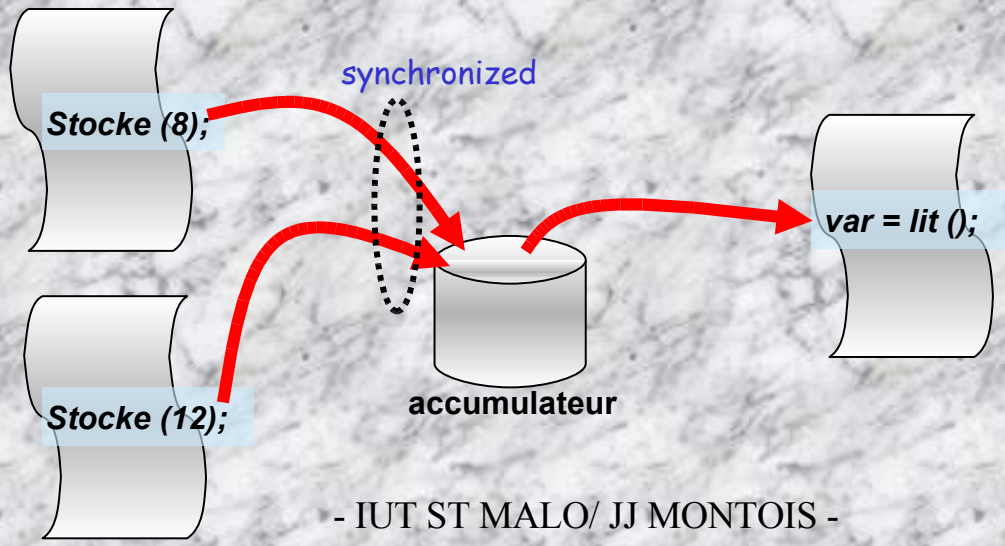
☞ Pour créer un moniteur, nous devons utiliser le mot-clé synchronized.

☞ On applique le mot-clé synchronized à une méthode ou un bloc de code pour sécuriser une section critique

Exemple: Exclusion mutuelle: *DIP Genève, Alexandre Maret & Jacques Guyot*

Si le moniteur est déjà occupé, les processus suivants seront mis en attente. L'ordre de réveil des processus n'est pas déterministe.

```
class mutexAcc
{
  int accumulateur = 0;

  public synchronized void stocke (int val )  { accumulateur += val;}

  public int lit() { return accumulateur; }
}
```

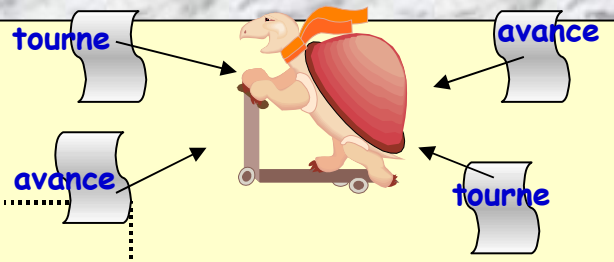


Sécurisation de blocs de code

```
synchronized void methode1() {
    // section critique...
}

void methode2() {
    synchronized ( this ) {
        // section critique...
    }
}
```

L'utilisation de méthodes synchronisées trop longues peut créer une baisse d'efficacité.
Avec Java, il est possible de placer n'importe quel bloc dans un moniteur, ce qui permet ainsi de réduire la longueur des sections critiques.

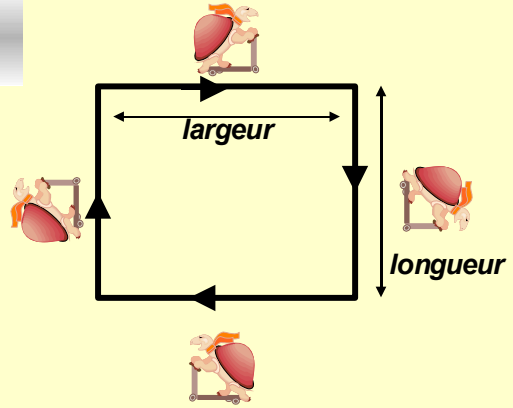
```
public class Tortue {
   private Point pos;
   private int angle;

   public Tortue (int x,int y,int angle) { // … }

   public synchronized void tourne (int degres) { angle += degres; }

   public synchronized void avance (int distance) {
      pos.x += (int) ((double)distance*Math.cos((double)angle));
      pos.y += (int) ((double)distance*Math.sin((double)angle));
   }

   public int angle () { return angle; }
   public synchronized Point pos () { return new Point (pos.x,pos.y); }
}//finclass
```



Plusieurs tâches donnent des ordres de mouvement d'un Logo Tortue

```
public class mutexLogo {
public int lectureEntier () { //lecture d'un nombre entier  }

   public static void carre (Tortue tortue) {
      int largeur    = lectureEntier();
      int longueur = lectureEntier();
      synchronized (tortue) {
         tortue.tourne(90);tortue.avance(largeur);
         tortue.tourne(90);tortue.avance(longueur);
         tortue.tourne(90);tortue.avance(largeur);
         tortue.tourne(90);tortue.avance(longueur);
      }
      // autres taches...
   }
}
```

Synchronisation du bloc



Sécurisation des variable de classes (*DIP Genève, Alexandre Maret & Jacques Guyot*

Pour sécuriser l'accès à une variable de classe, il faut créer un moniteur commun à toutes les instances de la classe. La méthode getClass() retourne la classe de l'instance dans laquelle on l'appelle. On peut maintenant créer un moniteur qui utilise le résultat de getClass() comme "verrou".

```
class mutexStatic {
   private int accumulateur = 0;
    private static int acces = 0;

    public synchronized void stocke(int val) {
        accumulateur += val;
        synchronized (getClass()) { acces += 1; }
    }

    public int lit() {
        synchronized (getClass()) { acces += 1; }
        return accumulateur;
    }

    public int acces() { return acces; }
}
```

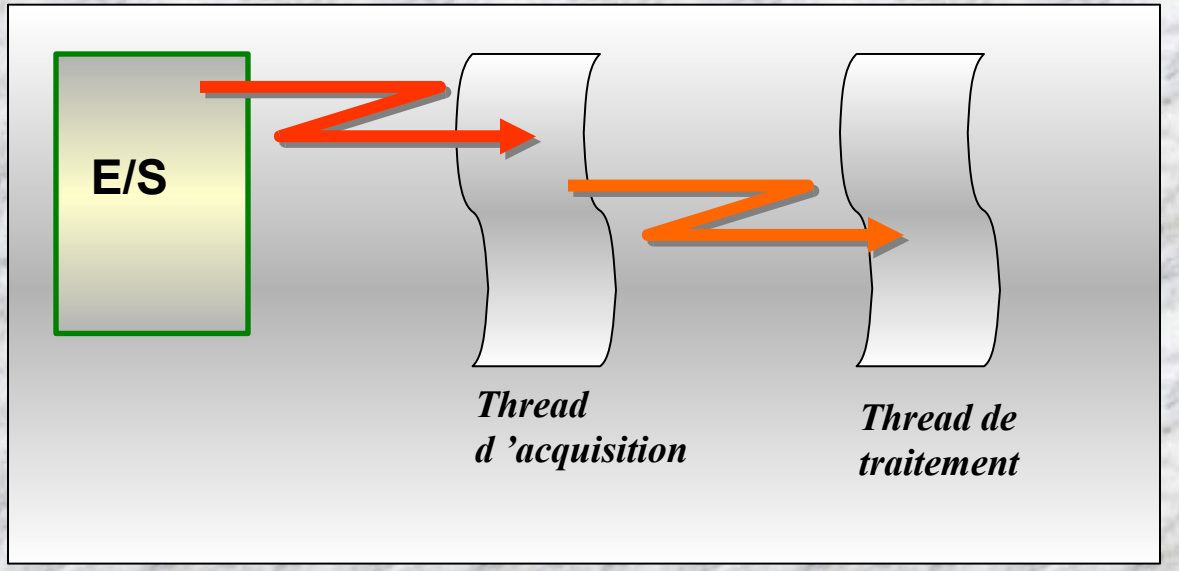
classe accumulateur qui incrémente **acces** chaque fois que l'on accède à **stocke** ou à **lit**. La variable **acces** est une variable de classe (déclarée public), elle est donc partagée par les différentes instances de cette classe. La méthode **getClass()** retourne un objet de type Class avec lequel on crée un nouveau moniteur.

Synchronisation en Threads

Il y a nécessitée de synchroniser des threads dans certaines activités:

- **Accès aux mêmes ressources**
- **Enchaînement d'activités**

☞ **La synchronisation de Threads utilise des signaux modélisés par les méthodes synchronized : wait ( ) , notify ( ) , notifyAll ( ) .**



Exemple: producteurs-consommateurs.

Soit un tampon borné de n objets, un thread producteur et un thread consommateur.

Application des tampons circulaires FIFO :

- **Télécommunications**
- **Communication asynchrones entre deux threads de vitesses différentes**
- **Architecture client-serveur**
- **….**

Exemple - tamponCirc.java

Le constructeur crée un tampon de taille éléments

On transmet des objets

```
class tamponCirc {
  private Object tampon[];
   private int taille;
   private int en, hors, nMess;

   public tamponCirc (int taille) {
      tampon = new Object[taille];
      this.taille = taille;
      en = 0;     //index dans la fifo
      hors = 0;  //index hors la fifo
      nMess = 0; //nbr de mesg ds la fifo
   }

   public synchronized void depose (Object obj) {
      while (nMess == taille) {    // si plein
         try { wait();              // attends non-plein
               } catch (InterruptedException e) { … }
      }
      tampon[en] = obj;
      nMess++;
      en = (en + 1) % taille;
      notify();              // envoie un signal non-vide
   }

   public synchronized Object preleve () {
      while (nMess == 0) {    // si vide
        try { wait();          // attends non-vide
              } catch (InterruptedException e) { … }
      }
      Object obj = tampon[hors];
      tampon[hors] = null;    // supprime la ref a l'objet
      nMess--;
      hors = (hors + 1) % taille;
      notify();              // envoie un signal non-plein
      return obj;
   }
}
```

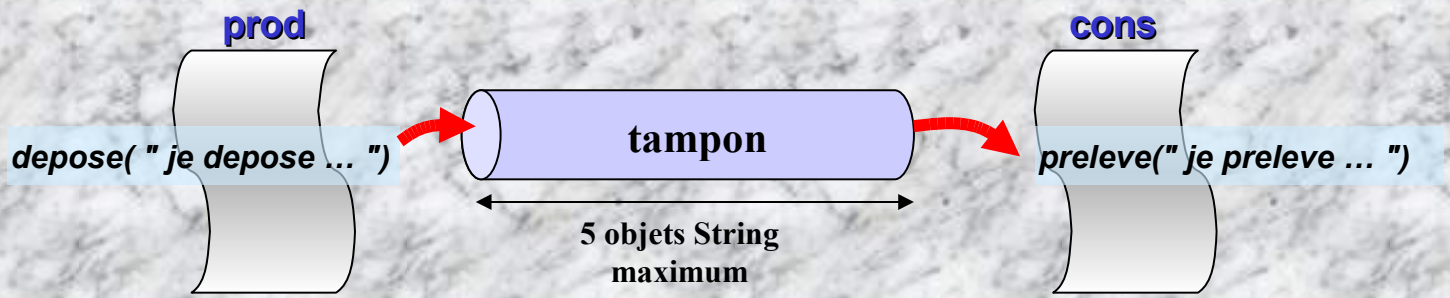
Exemple - utiliseTampon.java

```
class producteur extends Thread {

   private tamponCirc tampon;
   private int val = 0;

   public producteur (tamponCirc tampon) {
      this.tampon = tampon;
   }

   public void run() {
      while (true) {
         System.out.println("je depose "+val);
         tampon.depose(new Integer(val++));  //conversion int en objet  car tampon contient que des objets

         try {
            Thread.sleep((int)(Math.random()*100));    // attend jusqu'a 100 ms
         } catch (InterruptedException e) {}
      }
   }
} //fin class producteur

class consommateur extends Thread {

   private tamponCirc tampon;

   public consommateur (tamponCirc tampon) {
      this.tampon = tampon;
   }

   public void run() {
      while (true) {
         System.out.println("je preleve "+((Integer)tampon.preleve()).toString());
         try {
            Thread.sleep((int)(Math.random()*200));    // attends jusqu'a 200 ms
         } catch (InterruptedException e) {}
      }
   }
} //fin class consommateur
```

```
class utiliseTampon {
public static void main(String args[]) {

      tamponCirc tampon = new tamponCirc(5);
      producteur prod = new producteur(tampon);
      consommateur cons = new consommateur( tampon );

      prod.start();
      cons.start();
      try {
         Thread.sleep(30000);    // s'execute pendant 30 secondes
      } catch (InterruptedException e) { … }
   }
}
```

Exécution ...

```
...
je depose 165
je depose 166
je preleve 161
je depose 167
je preleve 162
je depose 168
je preleve 163
je depose 169
je preleve 164
je depose 170
je preleve 165
je depose 171
je preleve 166
je preleve 167
...
```

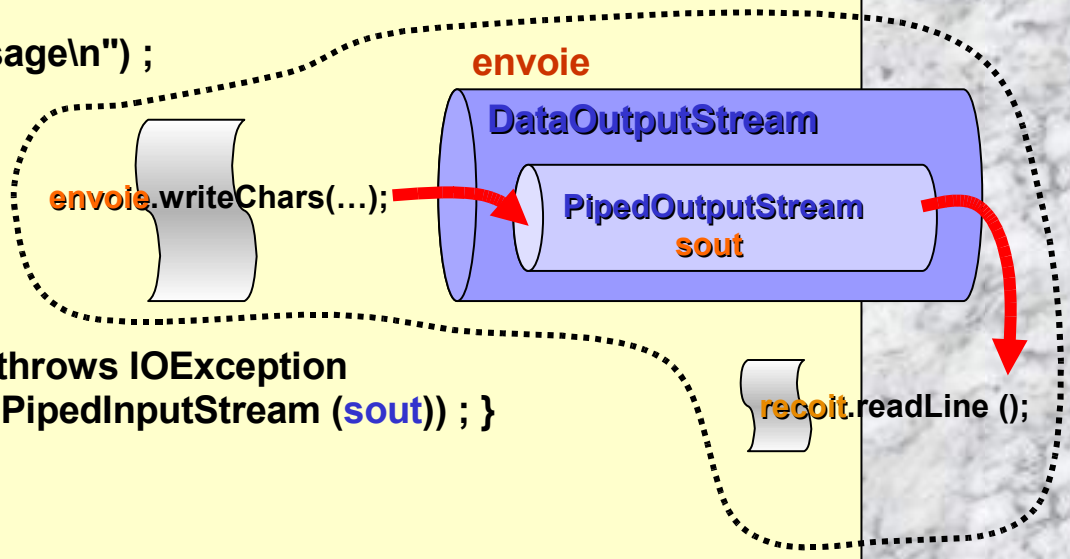


Communication entre threads de données primitives.

Les flux PipedInputStream / PipedOutputStream permettent d'établir une connexion entre 2 threads

```
public PipedInputStream();
public PipedInputStream(PipedOutputStream src);
public PipedOutputStream();
public PipedOutputStream(PipedInputStream snk);
```

```
import java.io.* ;
public class PipeEntreThread {
public static void main (String args []) throws IOException
 {
   PipedOutputStream sout = new PipedOutputStream () ;
   DataOutputStream envoie = new DataOutputStream (sout) ;
   MonThread thrd = new MonThread (sout) ; //affectation d'un tube de communication
   thrd.start () ;
   envoie.writeChars ("j'envoie un message\n") ;
 }
}

class MonThread extends Thread
{
  DataInputStream recoit ;
  MonThread (PipedOutputStream sout) throws IOException
   { recoit = new DataInputStream (new PipedInputStream (sout)) ; }

public void run ()
 {
  try { // On affiche ce qu'on recoit du pipe
  System.out.println (recoit.readLine ()) ;
 } catch (IOException e) { }
}
}//fin main
```



Erreur 404

Page en construction …

(livraison prévue dans le courant de l'année universitaire 2006-2007)

-12-

La programmation d'Interface Homme-Machine (IHM)

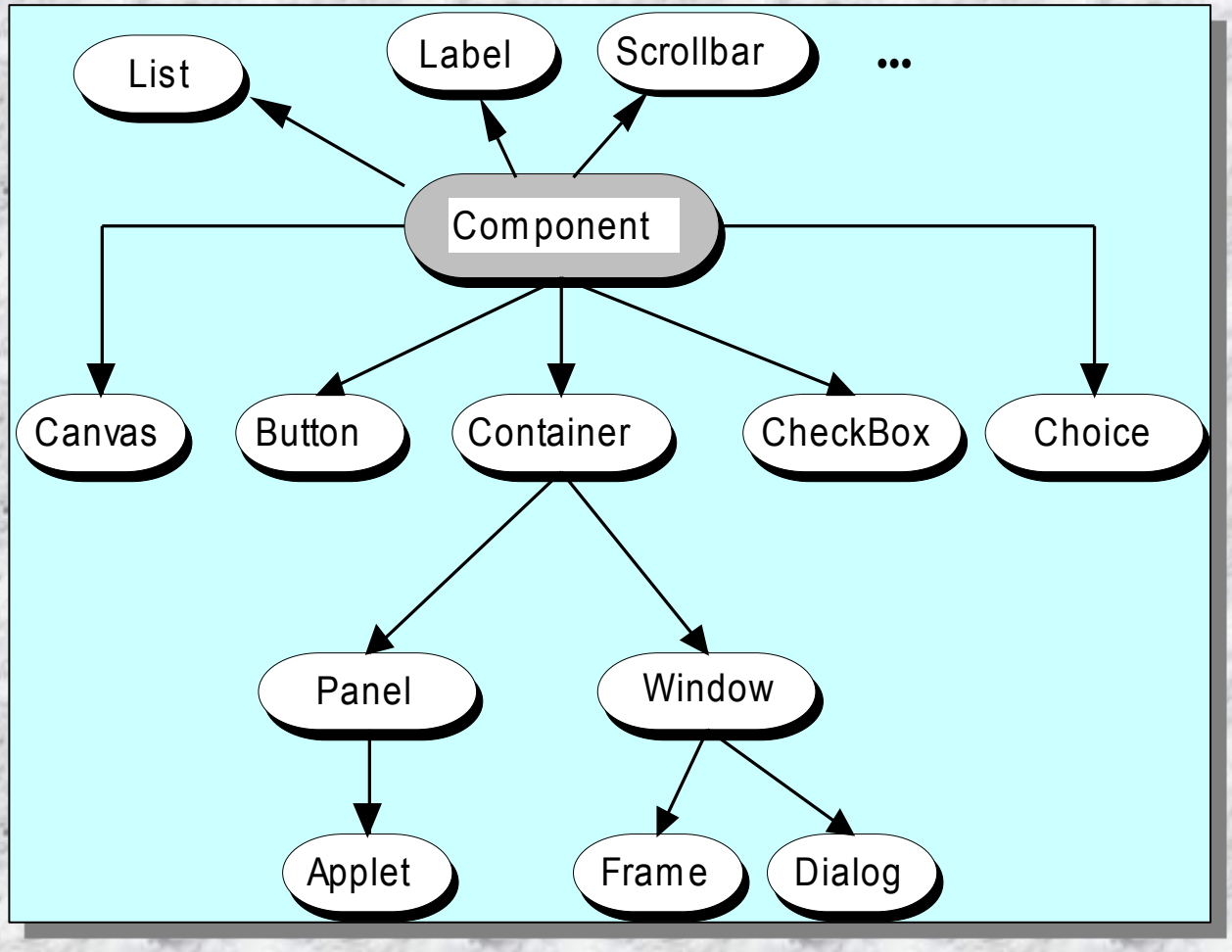
- Gérer les événements - Modèle évènementiel de JAVA -

2 APIs pour concevoir des IHM:
- Les AWT (Abstract Windows Toolkit)
- Les Swing

(seuls les AWT seront présentés)

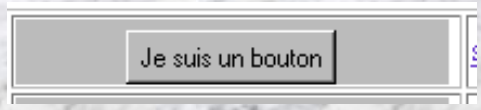
Les AWT (Abstract Windows Toolkit)

Les AWT sont des classes utilisées pour la conception d'interface homme-machine ou IHM.



Définitions et l'action de quelques-unes des méthodes associées.

1.Button : dit, dessine moi un bouton ...

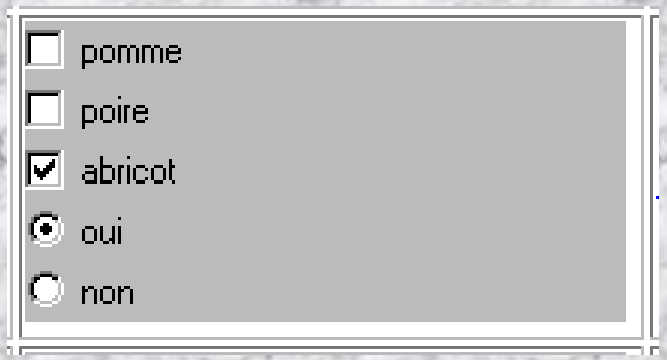



```
import java.awt.Button;
public class ButtonExemple extends java.applet.Applet
{
        public void init()
        {
            add(new Button("Je suis un bouton"));
        }
}
```

Les principales méthodes du Button:

Button ();	Créer un bouton sans label
Button (String txt);	Créer un bouton avec un label
String getLabel ();	Obtenir le label du bouton
void setLabel ();	Changer le label du bouton

1.Checkbox : Boutons à cocher. Il peuvent être indépendant entre eux, (ex: pomme, poire, abricot) ou non (ex:oui OU non).




```
import java.awt.Checkbox;
import java.awt.CheckboxGroup;
import java.awt.GridLayout;
public class CheckboxExemple extends java.applet.Applet
{
 public void init()
 {
    setLayout(new GridLayout(5,1)); // mise en page
                                    // voir  les layouts
    add(new Checkbox("pomme"));
    add(new Checkbox("poire"));
    add(new Checkbox("abricot", null, true)); // déja cliqué
    CheckboxGroup groupe = new CheckboxGroup();
    add(new Checkbox("oui", groupe, true));
    add(new Checkbox("non", groupe, false));
 }
}
```

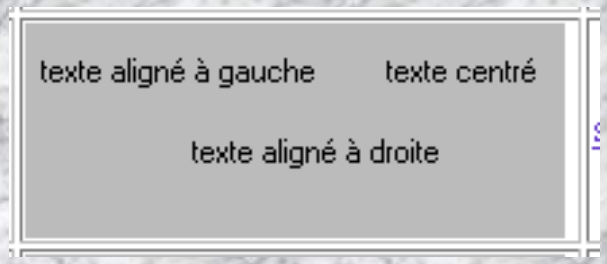
Les principales méthodes du Checkbox

Checkbox ();	Créer une boîte à cocher (non cochée)
Checkbox (String *txt*);	Créer une boîte à cocher (non cochée) affichant un texte *txt*
String getLabel ();	Obtenir le label du bouton
void setLabel ();	Changer le label du bouton

1. Choice : permet de selectionner un objet dans une liste.

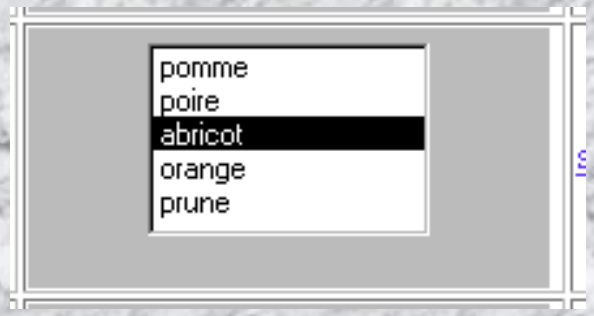

```
import java.awt.Choice;
public class ChoiceExemple extends java.applet.Applet
{
 public void init()
 {
  Choice c = new Choice();
  c.addItem("pomme");
  c.addItem("poire");
  c.addItem("abricot");
  c.addItem("prune");
  c.addItem("fraise");
   add(c);
 }
}
```

1. Label : affiche un texte



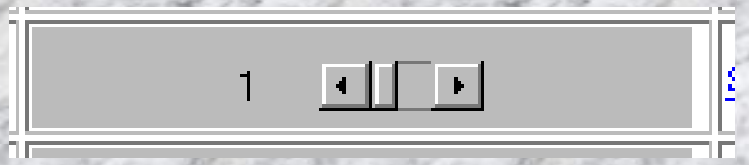

```
import java.awt.Label;
public class LabelExemple extends java.applet.Applet
{
   public void init()
   {
     add(new Label("texte aligné à gauche"));
     add(new Label("texte centré", Label.CENTER));
     add(new Label("texte aligné à droite", Label.RIGHT));
   }
}
```

1. List : affiche une liste




```
import java.awt.List;
public class ListExemple extends java.applet.Applet
{
  public void init()
  {
   List lst = new List(4, false);
   lst.addItem("pomme");
   lst.addItem("poire");
   lst.addItem("abricot");
   lst.addItem("orange");
   lst.addItem("prune");
   add(lst);
  }
}
```

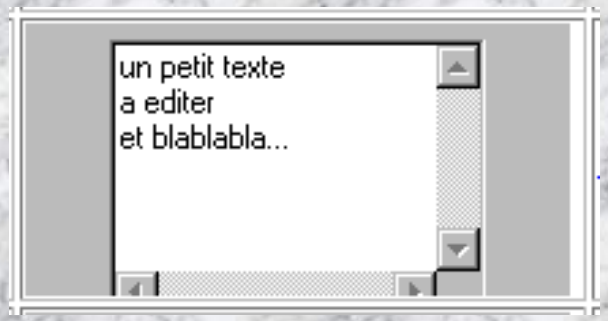
1. Scrollbar: afficher une barre de défilement



On peut aussi utiliser un xxxlistener à partir de la version 1.1.x de JAVA

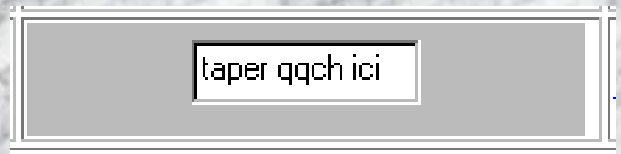
```
public class ScrollbarExemple extends java.applet.Applet
{
  Label  valeur;
  Scrollbar sb;
  public void init()
  {
   valeur = new Label("1");
   add(valeur);
   sb = new Scrollbar(Scrollbar.HORIZONTAL,
                      // direction
                      1,       // valeur initiale
                      1,       // epaisseur de la barre
                      1,       // valeur minimale
                      50);     //valeur maximale
   add(sb);
  }
  public boolean handleEvent (Event e )
  {
        if(e.target  instanceof  Scrollbar)
             valeur.setText (String.valueOf( sb.getValue() ));
        return true;
  }
}
```

1. TextArea: Fenêtre d'édition




```
import java.awt.TextArea;
public class TextAreaExemple extends java.applet.Applet
{
 public void init()
 {
   TextArea ta = new TextArea("un petit texte\na editer\net  blablabla...",5,20);
   add(ta);
 }
}
```

1. TextField: Tampon d'édition



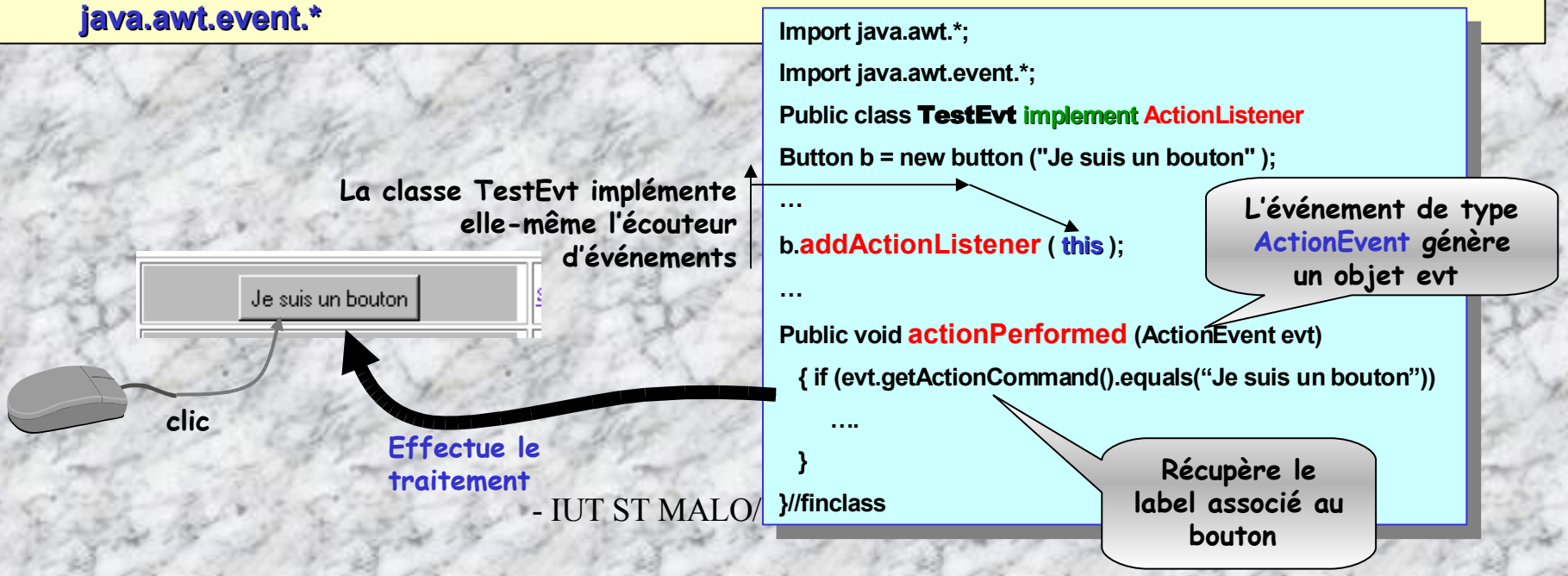

```
import java.awt.TextField;
public class TextFieldExemple extends java.applet.Applet
{
   public void init()
   {
      TextField tf = new TextField("taper qqch ici");
      add(tf);
   }
}
```

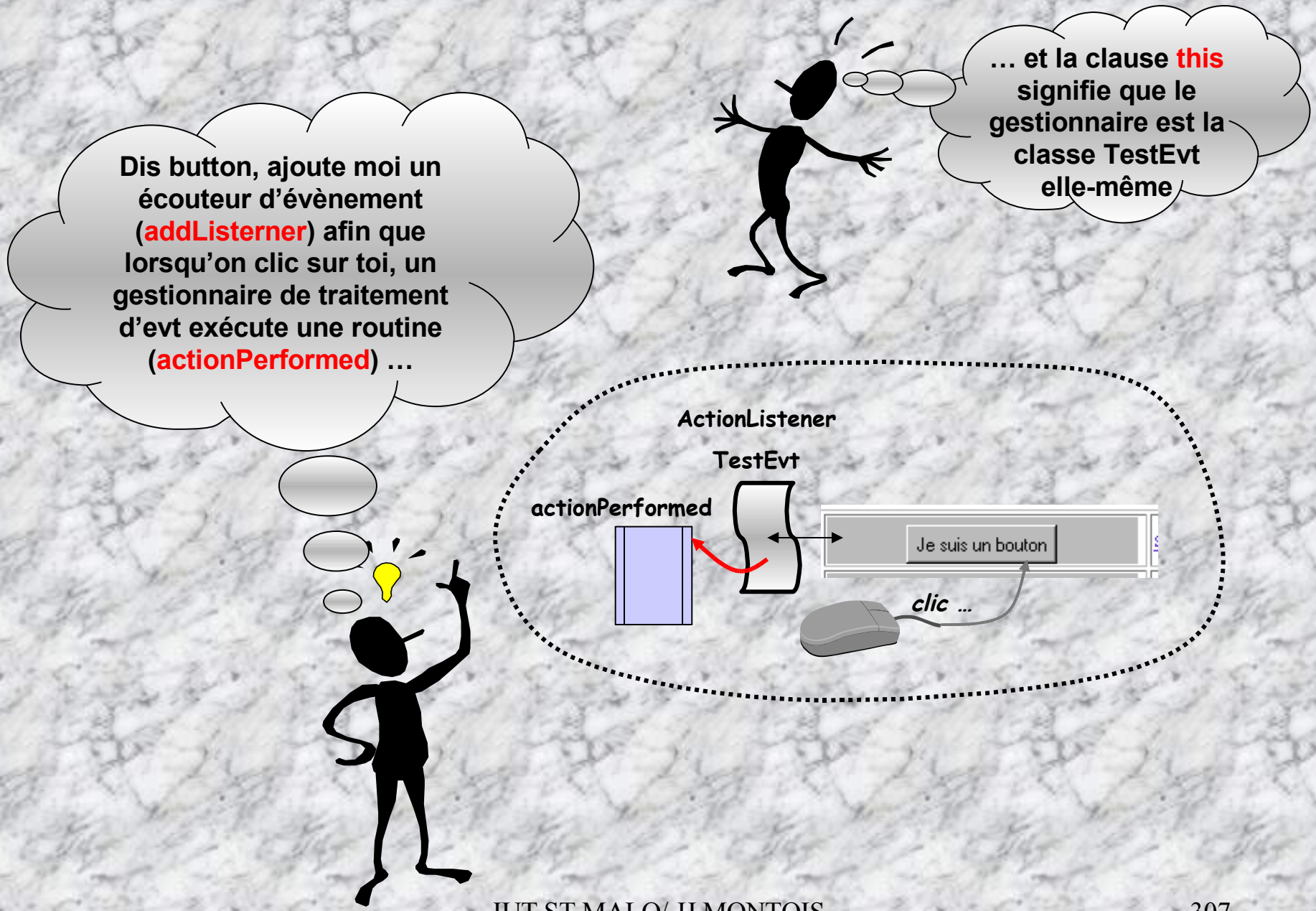
Gérer les événements-Définition du modèle évènementiel Listener (> JAVA 1.1.x)

Prise en compte d'évènement:

- Clic sur un objet graphique (button, choice, list, ouverture/fermeture de fenêtre, …)
- Comportements clavier, souris ..

1. La gestion d'évènements appliqués sur un composant est effectuée par l'intermédiaire d'un objet spécifique (ex: **ActionListener**) qui « écoute » les évènements [**ActionEvent**] associés à ce composant et lance le traitement prévu [**actionPerformed()**]
2. Un composant objet qui doit subir un évènement doit s'inscrire auprès d'un « écouteur » [**addActionListener (@ de l'objet listener)**]. Si celle-ci est la présente classe, cet argument est this
3. Pour manipuler les évènements, Il faut importer (import) les packages : **java.awt.*** et **java.awt.event.***




Dis button, ajoute moi un écouteur d'évènement (addListerner) afin que lorsqu'on clic sur toi, un gestionnaire de traitement d'evt exécute une routine (actionPerformed) …
… et la clause this signifie que le gestionnaire est la classe TestEvt elle-même
ActionListener
TestEvt
actionPerformed
Je suis un bouton
clic …

Méthodes utiles associées aux événements evt de la classe ActionEvent :

String **evt.getActionCommand ()**	**Retourne la commande de l'évènement sous forme d'une string établie préalablement par un setActionCommand(). Dans le cas d'un Button, si cette méthode n'a pas été exécutée, c'est le label du Button qui est retourné. Ex:** `if (evt.getActionCommand().equals("Cdebouton"))`

Object **evt.getSource()**	**Retourne une référence (type Objet) sur l'objet qui a généré l'évènement. Cette référence permet de connaître la source de l'évènement. Ex:**
	`Object src = evt.getSource () ;`
	`if(src instanceof Button)`
	`  if(((Button)src).getLabel().equals ("avant"))…`

class **evt.getSource().getClass()**	**Retourne la classe de l'objet qui a généré l'évènement. Sert à exploiter les propriétés de l'objet source, surtout quand le récepteur est susceptible d'écouter des evt issus de sources diverses**

Quelques « écouteurs » XXXListener à l'écoute des évènements xxxEvent …

Il existe de nombreux « écouteurs » spécialisés xxxListener d'évènements xxxEvent :

ActionListener  :ActionEvent spécifique effectué sur un composant button (bouton pressé puis relaché), List (2xclic sur un élément), TextField (zone texte validée par <CR>), MenuItem (menu choisi).
Méthode: actionPerformed ()

ItemListener :Evt quand un élément de liste (Choice, List, Checkbox) est sélectionné/désélectionné.
Méthode: itemStateChanged ()

KeyListener : KeyEvent généré quand un utilisateur saisit du texte au clavier.
Méthodes: keyPressed () , keyReleased (), keyTyped ()

MouseListener :MouseEvent généré par le cliquage de la souris
Méthodes: mousePressed (), mouseReleased (), mouseClicked (), mouse [Entered, Exited]()

MouseMotionListener :MouseEvent généré quand la souris se déplace sur un composant.
Méthodes: mouseDragged (), mouseMoved ()

WindowListener :WindowsEvent généré lors d'une action de gestion fenêtre.
Méthodes: windowsActived (), windowsClosed (), windowsDeactived(), windowsOpened(), etc

Implantation d'un gestionnaire d'évènement:


```
class TraiteEvt implement xxxListener, yyyListener {
  méthode de traitement: actionPerformed(actionEvent e), mouseClicked(WindowEvent e), …
   ….
   ….
}
```

On peut utiliser plusieurs xxxListener dans la classe utilisateur …
public class Enfant extends Parent **implements MouseListener, KeyListener { ..... }**

---

1. **Les composants créés doivent indiquer les événements qui les intéressent et la classe dans laquelle se trouve les méthodes de gestion de ces événements.**

3. **La délégation par un composant de l'écoute des évènements à un listener se fait par l'une des méthodes suivantes:**

addxxxListener (*référence objet listener***)**

- **addActionListener (…)**
- **addAdjustmentListener (…)**
- **addFocusListener (…)**
- **addItemListener (…)**
- **addKeyListener (…)**
- **addMouseListener (…)**
- **addMouseMotionListener (…)**
- **addWindowListener (…)**

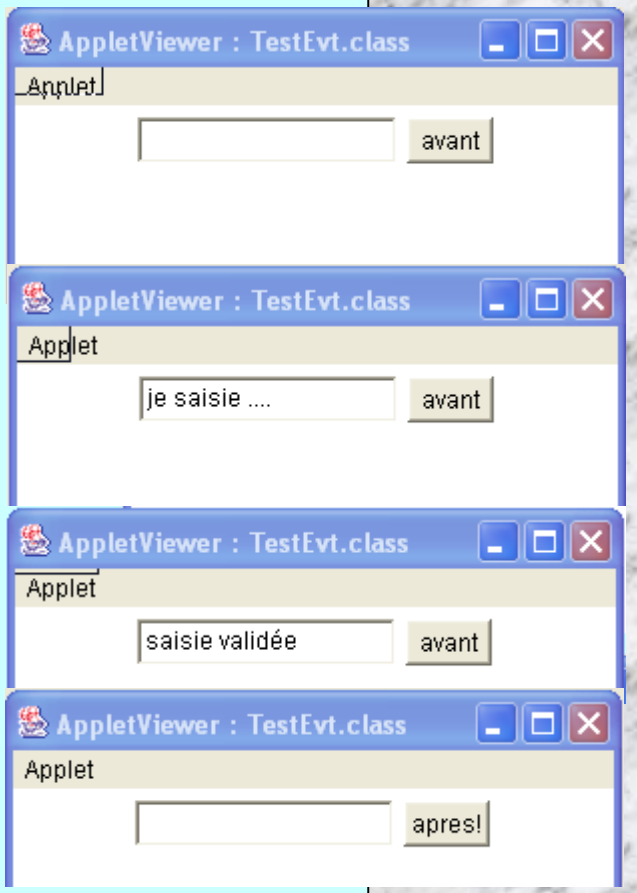
```
Import java.awt.*;
Import java.awt.event.*;
public class TestEvt
Button b = new button ("Je suis un bouton" );
…
b.addActionListener ( new TraitEvt() );
…
}//finclass

class TraiteEvt implement ActionListener
   public void actionPerformed (ActionEvent evt)
   {
      <Traitement de l'évènement; exploitation de evt>
        ….
   }
```

Un TextField et un Button écoutés par le même Listener …

```
import java.applet.* ;
import java.awt.* ;
import java.awt.event.* ;

public class TestEvt extends Applet
{
Button b = new Button ("avant") ;
TextField t = new TextField (15) ;

public void init ()
 {
   add (t) ; add (b) ;
   t.addActionListener ( new Evenement () ) ;
   b.addActionListener (new Evenement () ) ;
 }
}//finclass

class Evenement implements ActionListener
{
 public void actionPerformed (ActionEvent evt)
  {
    Object src = evt.getSource () ;
    if (src instanceof TextField) ((TextField) src).setText ("saisie validée");
    else if (src instanceof Button) if ( ((Button) src).getLabel ().equals ("avant"))
                                        ((Button) src).setLabel ("apres!") ;
                                    else ( (Button) src).setLabel ("avant") ;
 }
}
```

J'associe 1 comportement à 2 sources d'évènements

Je crée 1 comportement



Application qui définie plusieurs LISTENERs destinés à 3 objets: 2 Button, 1 TextField produisants des evts

```
import java.awt.*; import java.awt.event.*;

class FenetreCpt extends Frame {
 int compteur;
 Button bIncr= new Button("+");
 Button bDecr= new Button("-");
 Button bQuit= new Button("quit");
 TextField afficheCpt = new TextField(5);
```

3 écouteurs …

```
class ActionIncr implements ActionListener {
  public synchronized void actionPerformed (ActionEvent e) {compteur ++; afficherCpt();}
  };
class ActionDecr implements ActionListener {
  public synchronized void actionPerformed (ActionEvent e) {compteur --; afficherCpt();}
  };
class ActionQuit implements ActionListener {
  public synchronized void actionPerformed (ActionEvent e) {System.exit(0);}
  };
```

Je crée 3 comportements …

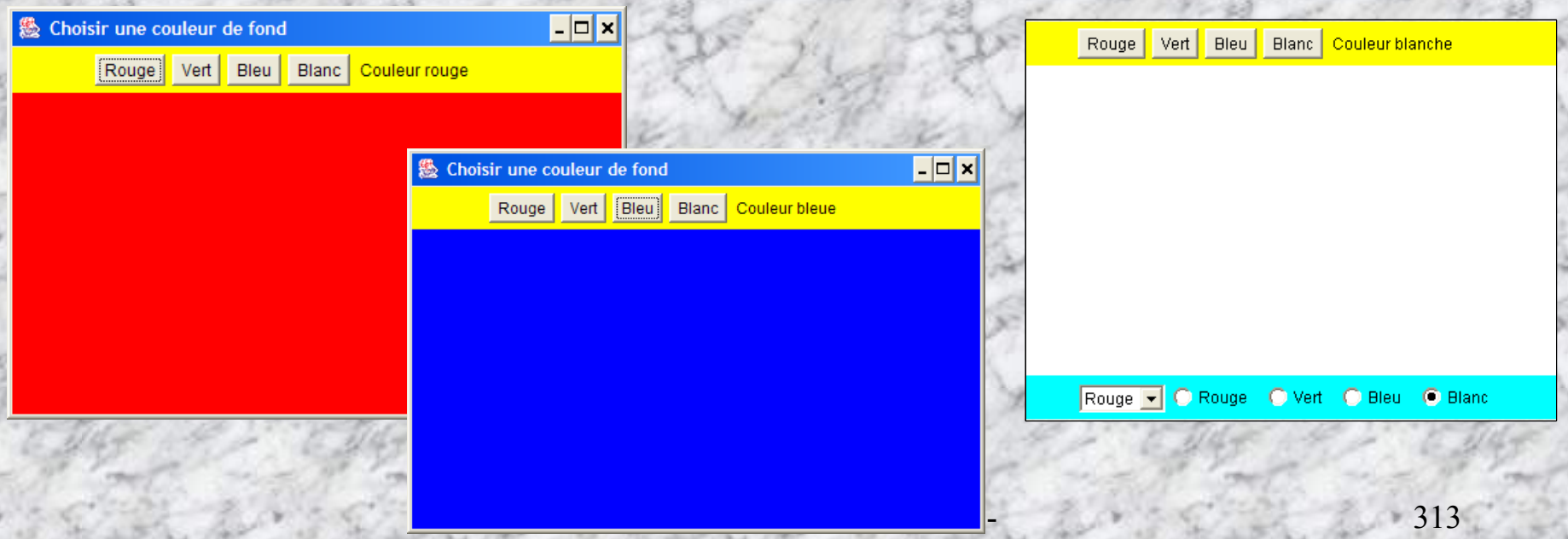
```
 void afficherCpt() { afficheCpt.setText(String.valueOf(compteur));}

 public FenetreCpt (String nom) { // constructeur
  super("compteur " + nom); compteur=0;
  setLayout(new FlowLayout());
  add(bIncr); add(bDecr); add(afficheCpt); add(bQuit);

  bIncr.addActionListener (new ActionIncr());
  bDecr.addActionListener (new ActionDecr());
  bQuit.addActionListener (new ActionQuit());
  pack(); setVisible(true);
  afficherCpt();
  }
 }//fin class

public class TestLISTENER {
  static public void main (String argv[]) {new FenetreCpt("Test LISTENER");}
}//fin class
```

3 composants s'enregistrant à des écouteurs spécifiques …

J'associe les comportements à des objets sources d'évènements

Exercice1-Evènements: **Expérimenter la gestion de quelques événements provoqués par l'utilisateur-**
Il s'agit de mettre en œuvre les mécanismes concernant les composants (button, …), les gestionnaires de répartition (layout, …), et les classes d'événements (xxxListeners, …). A cet effet, créer une application chargée de choisir la couleur de fond d'écran de différentes manières (rouge, vert bleu, blanc).
1°) créer une classe **ChoixCouleur1** dont le panneau du haut contient 4 boutons associés à 4 couleurs par exemple rouge, vert, bleu, blanc, puis un label où sera affichée la couleur choisie. Mettre en place un listener permettant d'afficher la couleur choisie à chaque clic de souris.
2°) Prolonger l'exercice en créant une classe **ChoixCouleur2,** qui en plus présentera sur son panneau du bas, une liste déroulante avec les 4 couleurs, et une série de 4 boutons radios associés aux 4 couleurs. Il s'agit ici de pouvoir en plus changer la couleur de fond de la fenêtre en sélectionnant une couleur à l'aide d'un bouton radio ou dans une liste déroulante.
3°) Ajouter un listener de clavier afin que les actions d'affichage de couleurs puissent être commandées indifféremment par la souris ou le clavier.

Méthodes utiles:
setText, setBackground



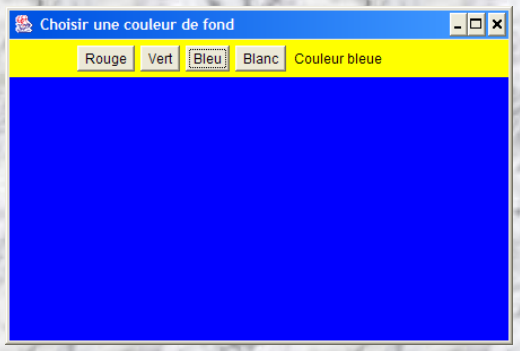
Squelette …

```
import java.awt.*;
import java.awt.event.*;
public class ChoixCouleur1 extends Frame implements ActionListener{
Label l;
Button rouge, vert, bleu, blanc;
Panel p;

public ChoixCouleur1 (String titre) {
super (titre);
  //Au départ, choisir le fond  noir
  //Création d'un panel (panneau) où s'affichera la couleur choisie
  //Au départ choisir un fond jaune dans le panneau
  //Créer les boutons rouge, vert, bleu, blanc
  //Créer le label "Couleur Sélectionnée"
  //Associer aux boutons rouge, vert, bleue, blanc le listener présent dans  la classe ChoixCouleur
  //Ajouter dans le panneau les boutons rouge, vert, bleu, blanc, le label
  //Ajouter le panneau dans le container graphique support
 }//finconstructeur

public void actionPerformed (ActionEvent evt) {
  //Récupérer l'Object, source de l'évènement
  //caster l'Object en type Button
  //Selon le bouton actionné, colorer le panneau avec la couleur associée
  //Ecrire le label « Couleur rouge » ou « Couleur bleue », …
 }

public static void main (String arg[]) {
  ChoixCouleur1 f = new ChoixCouleur1 ("Choisir une couleur de fond");
  //Régler le panneau selon les points (200,100) et (450, 300);
  //Rendre le panneau Visible
 }
}//finclass
```

Exercice1-Evènements(Corrigé) **Expérimenter la gestion de quelques événements provoqués par l'utilisateur-**

```
import java.awt.*;
import java.awt.event.*;
public class ChoixCouleur1 extends Frame implements ActionListener{
Label l;
Button rouge, vert, bleu, blanc;
Panel p;

public ChoixCouleur1 (String titre) {
super (titre);
setBackground (Color.black);
 p =new Panel();
 p.setBackground (Color.yellow);
 rouge = new Button ("Rouge");
 vert    = new Button ("Vert");
 bleu   = new Button ("Bleu");
 blanc = new Button ("Blanc");
 l = new Label ("Couleur sélectionnée");
 rouge.addActionListener(this);
 vert.addActionListener(this);
 bleu.addActionListener(this);
 blanc.addActionListener(this);

 p.add (rouge);
 p.add (vert);
 p.add (bleu);
 p.add (blanc);
 p.add (l);
 add ("North", p);

}//finconstructeur
```

constructeur de la fenêtre



Construction du panneau du haut,
Mise en couleur du panneau,
construction des boutons,
Spécification du label
Ajout des écouteurs sur chacun des boutons

on place les boutons sur le Panel p, puis p sur l'instance de fenêtre ChoixCouleur1, et en haut (North).

```
public void actionPerformed (ActionEvent evt) {
 Object obj = evt.getSource();
 Toolkit.getDefaultToolkit().beep(); // pour "bipper"
Button b= (Button)obj;
 if  (b==rouge) {
   setBackground(Color.red);
   l.setText("Couleur rouge");
   }
  else if  (b==bleu) {
   setBackground(Color.blue);
   l.setText("Couleur bleue");
   }
  else if (b==vert) {
   setBackground(Color.green);
   l.setText("Couleur verte");
   }
  else {
   setBackground(Color.white);
   l.setText("Couleur blanche");
   }
}

public static void main (String arg[]) {
 ChoixCouleur1 f = new ChoixCouleur1("Choisir une couleur de fond");
 f.setBounds(200, 100, 450, 300);
 f.setVisible(true);
 }

}//finclass
```

Méthode appelée dès l'apparition de l'évènement evt

Récupération de l'objet ayant reçu l'évènement?

Quel est l'objet Button ayant reçu l'évènement?

construction d'une instance de panneau, puis affichage et écoute des 4 boutons dans l'attente d'un clic utilisateur

Rajoutons des boutons radio, et une liste déroulante …

Le constructeur où s'effectue la construction et la mise en place du panneau …

```
public class ChoixCouleur2 extends Frame implements ActionListener, ItemListener {
Panel ph, pb;
Label l;
Button rouge, vert, bleu, blanc;
Choice ch ;
CheckboxGroup gr ;
Checkbox rad1, rad2, rad3, rad4;
public ChoixCouleur2 (String titre) {
 super (titre);
 setBackground(Color.black);
 ph =new Panel();
 ph.setBackground(Color.yellow);
 pb =new Panel();
 pb.setBackground(Color.cyan);
 rouge = new Button("Rouge");
 vert = new Button("Vert");
 bleu = new Button("Bleu");
 blanc= new Button("Blanc");
 l = new Label ("Couleur sélectionnée");
 ch = new Choice();
 ch.add("Rouge");
 ch.add("Vert");
 ch.add("Bleu");
 ch.add("Blanc");
 ch.select(2);
 gr = new CheckboxGroup();
 rad1 = new Checkbox("Rouge", false, gr);
 rad2 = new Checkbox("Vert", false , gr);
 rad3 = new Checkbox("Bleu", false , gr);
 rad4 = new Checkbox("Blanc", false , gr);
```

définition des composants

construction des panneaux et mise en couleur

construction des boutons et du label

construction de la liste déroulante

construction des boutons radio

```
// pose des écouteurs d'action sur les boutons
 rouge.addActionListener(this);
 vert.addActionListener(this);
 bleu.addActionListener(this);
 blanc.addActionListener(this);
// pose des écouteurs d'items sur les boutons radio et la liste
 ch.addItemListener(this);
 rad1.addItemListener(this);
 rad2.addItemListener(this);
 rad3.addItemListener(this);
 rad4.addItemListener(this);

// on place les boutons et le label sur le panneau du haut ph
 ph.add(rouge);
 ph.add(vert);
 ph.add(bleu);
 ph.add(blanc);
 ph.add(l);
 add("North", ph);
// on place la liste et la liste sur le panneau du bas pb
 pb.add(ch);
 pb.add(rad1);
 pb.add(rad2);
 pb.add(rad3);
 pb.add(rad4);
 add("South", pb);
 }
```

Méthode appelée dès l'apparition de l'évènement evt

Récupération de l'objet ayant reçu l'évènement?

```
public void actionPerformed (ActionEvent evt) {
  Object obj = evt.getSource();
 // on procéde au changement de couleur en fonction de cet objet
 if ( str=="Rouge")  {
   setBackground(Color.red);
   l.setText("Couleur rouge");
   }
  else if  (str=="Bleu") {
   setBackground(Color.blue);
   l.setText("Couleur bleue");
   }
  else if ( str=="Vert") {
   setBackground(Color.green);
   l.setText("Couleur verte");
   }
  else {
   setBackground(Color.white);
   l.setText("Couleur blanche");
   }

 }
public void itemStateChanged (ItemEvent evt) {
// obj est l'objet qui a reçu evt, de type sélection d'un item
 Object obj = evt.getSource();
/** on récupère l'item sélectionné avec la méthode getItem() qui interroge
l'événement
* et on "cast" pour obtenir le résultat sous forme de chaine.*/
 String s = (String) (evt.getItem());
 changeCoul( s );
 }
public void changeCoul(String str) {
   }
// ici on teste le nom de l'objet émetteur de l'événemrnt
public void changeCouleur(Object objet) {
  if ( (objet==rouge) ) {
   setBackground(Color.red);
```

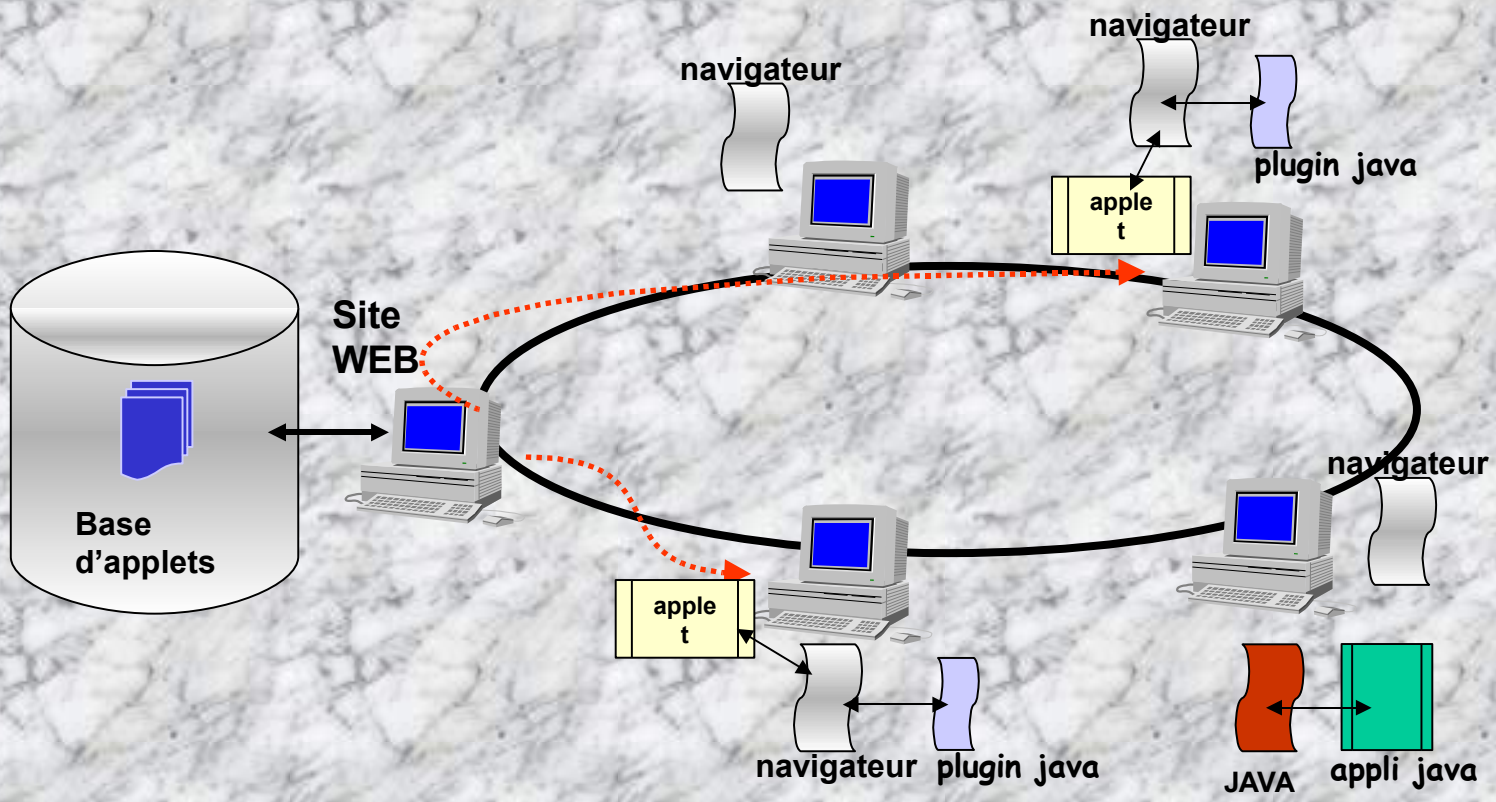
-13-

La programmation de codes téléchargeables.

APPLETs – MIDLETs …

Les APPLETs

- Les applets sont téléchargées d'un site WEB ou localement. Elles ont besoin d'un navigateur compatible JAVA pour s'exécuter (plugin java) sur la machine cible
- Les applications sont exécutables localement à l'aide de l'interpréteur java.exe



Différence entre une applet JAVA et une application JAVA.

- Une applet JAVA est le code téléchargeable d'une classe s'insèrant dans une balise html :

  < applet CODE= "Hello.class" > ..... < /applet >

- Une application JAVA doit contenir obligatoirement la fonction main ()

Une applet JAVA

```
import java.awt.*;
public class Hello extends java.applet.Applet
 {
   public void paint (Graphics g)
    {
      g.drawString("hello world",5,20);
    }
 }
```

Une application JAVA

```
import java.io.*;
class Hello
 {
   public static void main (String args[])
    {
      System.out.println (" salut le monde! ");
    }
 }
```

L'écriture d'Applets

Une applet est un programme comme un autre mais il doit être exécuté dans une page html à l'aide d'un « plugin » java du navigateur utilisé. Il y a une partie déclaration et une partie code.

A - Partie déclaration

Java possède un nombre défini de package (composés d'interfaces ou de classes).

⚠ **Pour utiliser ces packages, il suffit de les importer dans le source avec l'instruction import.**

java.applet	importé par défaut dans tout programme.
java.awt	graphisme et gui.
java.awt.image	les images.
java.awt.peer.	
java.io	entrees/sorties.
java.lang	classes liées au langage (fonctions mathématiques ...)
java.net	réseaux.
java.util	structures de données (token, vecteur...).

Exemple:

```
import java.awt.Graphics;
```

Le joker * peut-être utilisé
import java.awt.*

B - Partie code

Une applet est une instance de la classe Applet. Cette classe est définie public.

Exemple : **public class MonApplet extends java.applet.Applet**

```
import java.awt.*
public class MonApplet extends java.applet.Applet
{
   ...
}
```

Les méthodes principales de la classe Applet:

init()	méthode appelée lors du chargement de l'applet.
start()	méthode appelée lors du commencement de l'exécution de l'applet.
stop()	méthode appelée lors de l'interruption de l'applet.
run()	méthode appelée lors d'un Thread.
paint()	méthode appelée pour l'affichage.
update()	méthode appelée lors de l'appel de la fonction repaint().
action()	gestion des événements.

Schéma général d'une applet

```
import java.applet.*;
import java.awt.*;

public class <NomApplet> extends Applet {

public void init() {
   <Initialisations diverses>
   <Démarrage de processus – run(s)>
}
public void start() {
      <Démarrer l'applet; appelée chaque fois que l'on entre dans le document la contenant >
}
public void paint (Graphics g) {
      <Afficher les objets graphiques dans le panneau graphique g>
}
public void stop() {
      <Arrêter l'applet; appelée chaque fois que l'on sort du document la contenant. >
}
public void destroy() {
      <Relâcher les ressources, l'applet libère la mémoire>
}
```

Cycle de vie d'une applet: init() → start() → paint() → stop() → destroy()

APPLET ET SECURITE …

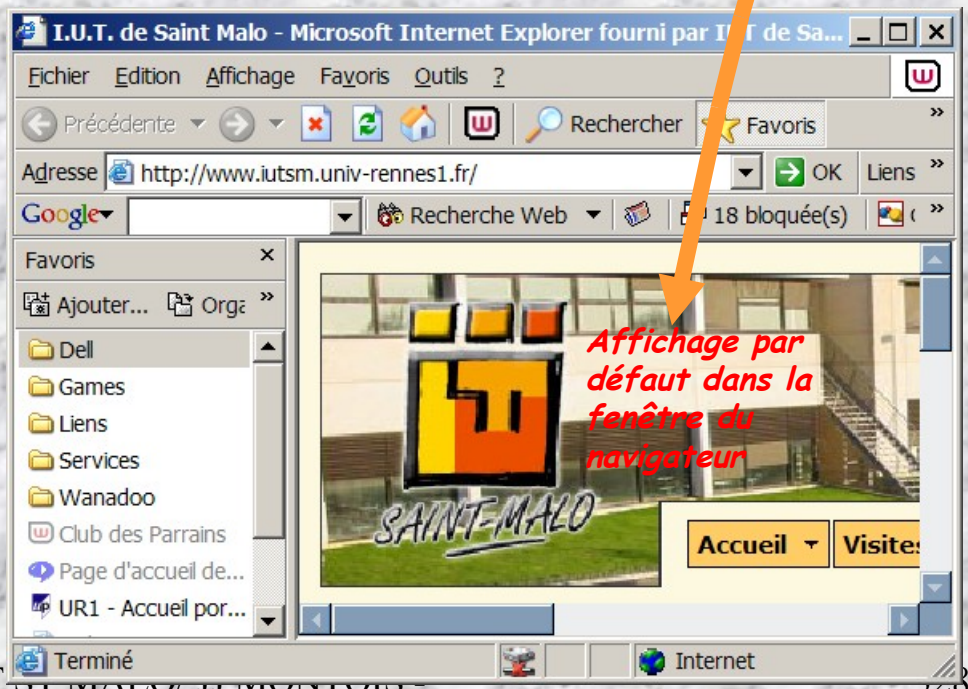
L'exécution d'une applet chargée via le réseau ne doit pas mettre en péril la sécurité. Aussi, un certain nombre de manipulations sont interdites à l'applet lorsqu'elle est chargée dans la mémoire du système :

- ✓ **Accès au système de fichier local**
- ✓ **Lancement de tache au moyen de exec()**
- ✓ **Chargement de librairies ou définition de méthodes natives**
- ✓ **Accès au System.getProperty() donnant des informations sur l'utilisateur ou la machine locale.**
- ✓ **Modification des propriétés système**
- ✓ **Accès à un autre groupe de thread.**
- ✓ **Changement de ClassLoader, SocketImplFactory, SecurityManager, ContentHandlerFactory, URLStreamHandlerFactory**
- ✓ **Ouvrir un connexion réseau vers une autre machine que celle dont elle provient**
- ✓ **Accepter des connexions.**

EXEMPLE d'applet: le sempiternel 1ier programme …

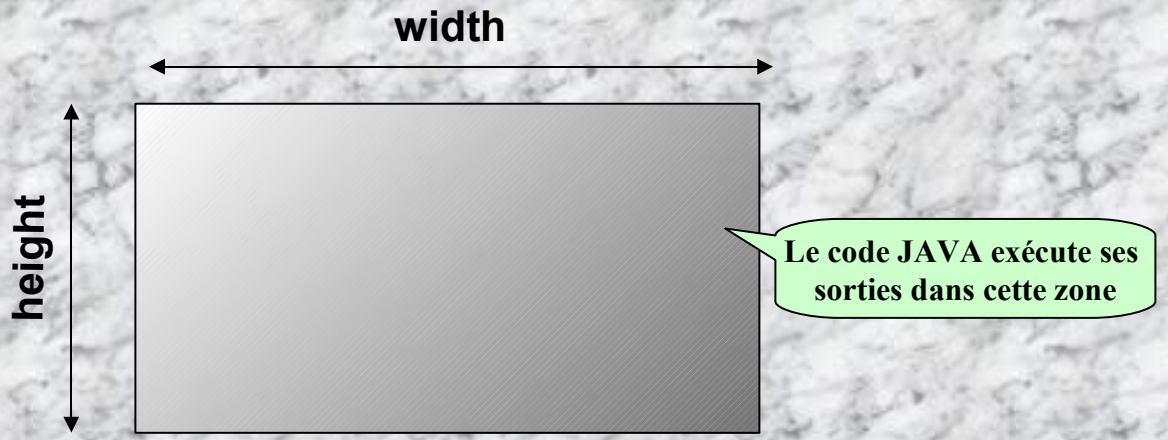
```
import java.awt.Graphics;
public class HelloWorld extends java.applet.Applet
{
   public void paint (Graphics g)
    {
      g.drawString("hello world", 5, 25);
    }
}
```

g: Fenêtre graphique du navigateur par défaut



Insérer une applet dans une page HTML

```
<APPLET code="x.class"  width=20 height=20>   </APPLET>
```



L'objet peut être:
→ **Une image animée**
→ **Un extrait de texte HTML**
→ **Tout programme.**

- **Lorsque la balise est rencontrée par le navigateur, le programme JAVA présent sur le serveur est chargé dans le répertoire cache de l'ordinateur.**
- **Le programme Java est chargé comme un fichier class correspondant aux pcodes générés par le compilateur Java.**
- **Le code *.class peut être présent localement sur votre disque, dans le répertoire de la page HTML pour des activités de tests, ou sur le serveur.**
- **Le navigateur garde l'applet présent dans le cache même si on modifie la classe Java qu'il doit lire. Aussi, lorsqu'on développe des applets, il faut configurer le navigateur pour vider le cache.**

Les bibliothèques à charger ou package

```
import java.io.*;
import java.applet.Applet;

public class hello extends Applets
{
   <code source>
}
```

```
import java.io.*;
import java.applet.Applet;

public class hello extends Applets
{
  public void init ()
   { rezise (250, 300); }
}
```

La fonction resize redéfinit la taille de la zone réservée à l'APPLET, par sa largeur et sa hauteur

Sortie graphique dans g

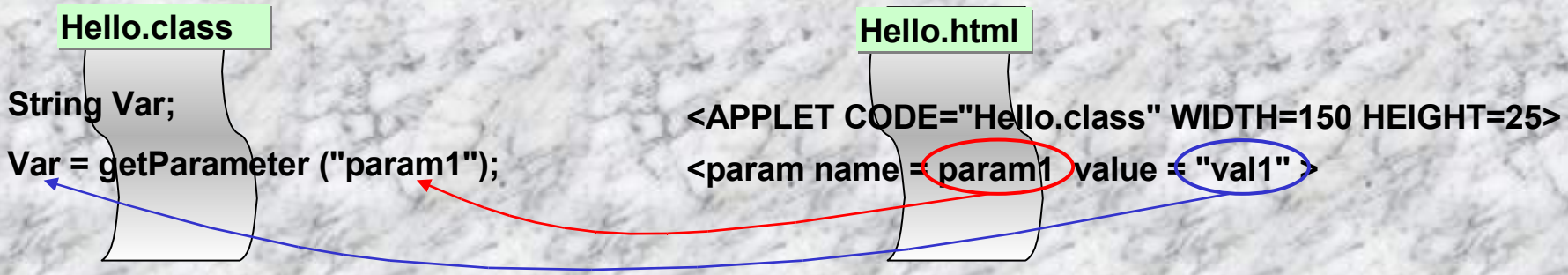
```
import java.io.*;
import java.applet.Applet;
import java.awt.Graphics;
public class hello extends Applets
{
  public void init ()
   { rezise (250, 300); }
 public void paint (Graphics g)
    { g.drawstring "Hello world", 10, 50);}
}
```

☞ Le code appelé peut comprendre des paramètres définis dans la balise html, comme suit :

```
<APPLET CODE="Hello.class" WIDTH=150 HEIGHT=25>
<param name=param1 value = "val1" >
<param name=param2 value = "val2" >
   …

</APPLET>
```

Usage de la méthode getParameter (String *nomvariable*) dans le source java de l'applet qui renvoie la valeur de *nomvariable* sous forme d'une String

```
String var = getParameter (« nomvariable" ) ;
```



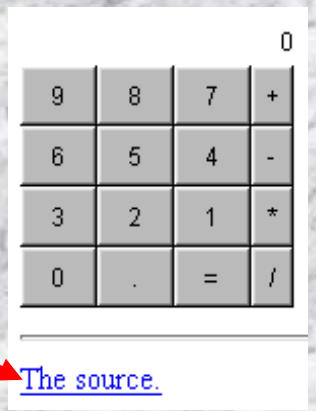
Exemple – Soit le code html suivant:

```
<applet code="Clock.class" width=50 height=50> <param name=Couleur value="vert"> </applet>
```

… alors, un appel getParameter("Couleur") retourne la valeur "vert".

☞ **Pour connaître le source de l'applet, on « ancre » un lien dans la page html avec la balise:**

```
<A href = "Hello.java">  The source </A>
```



L'interface Applet

La méthode getParameterInfo permet de définir les paramètres qui seront passés à l'Applet. Elle retourne un tableau de chaînes de caractères. Le tableau renvoyé est un tableau de 3 chaînes comprenant :

1. **le nom de l'argument**
2. **le type de l'argument**
3. **la description de l'argument**

Par exemple, cette méthode s'utilise de la façon suivante :

```
public String[][] getParameterInfo ()
  {
    String tableau[][] =
        {
          {"largeur",  "int", "largeur de l'applet"},
          {"hauteur", "int", "longueur de l'applet"},
          {"URL",      "url", "adresse du fichier GIF indiquant la présence de l'Applet"}
        };
  return tableau ;
  }
```

Pour initialiser les différentes valeurs déclarées dans la table *tableau* on utilise la méthode getParameter avec en argument le nom de l'argument :

```
variable = getParameter ("largeur" ) ;
```

Si le paramètre n'est pas trouvé la fonction getParameter retrourne la valeur *null*.

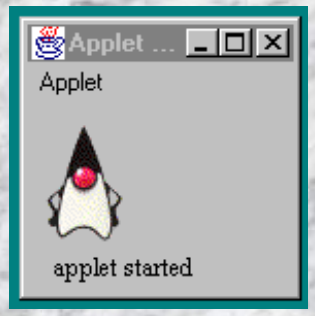
1ier exemple (Sun): Afficher l'image T1.gif. Cette image se trouve dans le répertoire java\demo\.

```
import java.applet.*;
import java.awt.*;
public class exemple1 extends Applet
 {
   Image image;
   public void init() { image=getImage (getDocumentBase(),"java/demo/T1.gif "); }
   public void paint (Graphics g) { g.drawImage(image,0,0,this); }
 }
```

Remarques :

1. **Les imports peuvent être mis avec ou sans caractère ***
2. **Le programme ci-dessus sera inclus dans un fichier exemple1.java et le fichier résultant de la compilation sera exemple1.class**
3. **Les méthodes init() et paint() sont exécutées séquentiellement.**
4. **drawImage est exécutée aux coordonnées 0,0 du rectangle réservé par l'Applet dans la balise HTML**

Exécution :



2ième exemple (Sun): Afficher 8 images différentes qui change à chaque clic de souris sur l'image.

```
import java.applet.*;
import java.awt.*;
public class exemple2 extends Applet
// exemple2.class sera le code appelé par le navigateur
{
  Image buf[];   // buf est un tableau à remplir avec les noms des différentes images.
  int i=0, j = 1;
  int a=0, b=0;
  public void init()
   {
    buf = new Image[7];
    for (i = 0; i<7; i++)
     {
       buf[i] = getImage (getCodeBase(), "T" + j + ".gif");
       j++;
     }
    i=0;
   }
 public void paint (Graphics g) { g.drawImage ( buf[i], 0, 0, this); }
 public boolean mouseUp (Event e, int x, int y)
  {
    repaint();
    i++;
    if (i == 7) i = 0;
    return true;
  }
}
```

Remarques :

La méthode init() se contente de déclarer un tableau buf contenant 8 images remplies par les fichiers T1.gif ... T7.gif.

getCodeBase() est l'URL de l'applet, l'image se trouve dans le sous-répertoire courant de cette URL.

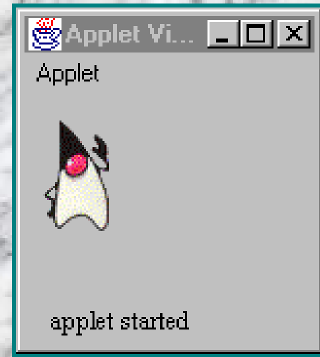
La méthode paint() dessinera l'une des 8 images.

A chaque clic de la souris, repaint() provoque un nouvel appel à la méthode paint()

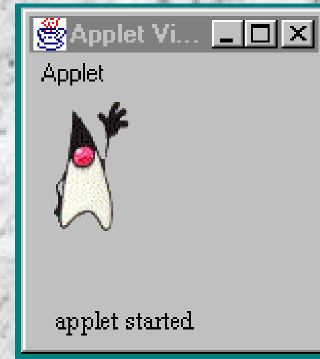
<u>Exécution</u> : (cliquez sur l'icône pour suivre l'exécution) :

'clic'

'clic'



'clic'



'clic'

'clic'

'clic'

'clic'

'clic'

DIFFERENTES METHODES DE GESTION DE TEXTE, SON, IMAGE.

Méthodes disponibles pour récupérer des informations sur le document HTML contenant l'applet :

public URL getDocumentBase() : retourne l'URL de base du document contenant l'applet
public URL getCodeBase() : retourne l'URL de base de l'applet.

Méthodes pour récupérer simplement des images et des sons :

public Image getImage (URL url)
public Image getImage (URL url, String nom)
public AudioClip getAudioClip (URL url)
public AudioClip getAudioClip (URL url, String nom)

Méthodes de la classe java.applet.AudioClip permettant de"manipuler" les sons récupérés :

public abstract void play()
public abstract void loop()
public abstract void stop()

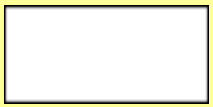
Méthodes disponibles pour jouer directement les sons :

public void play (URL url)
public void play (URL url, String nom)

Pour les amateurs de graphismes 2D, les principales méthodes …

```
import java.awt.*;
import java.awt.geom.*;
import javax.swing.*;
```

```
Double g.draw (new Line2D.Double(x, y+rectHeight-1, x + rectWidth, y));
```



```
Double g.setStroke(stroke);
g.draw (new Rectangle2D.Double(x, y, rectWidth, rectHeight));
```

```
Double g.setStroke(dashed);
g.draw (new RoundRectangle2D.Double(x, y, rectWidth, rectHeight, 10, 10));
```

```
Double g.setStroke (wideStroke);
g.draw (new Arc2D.Double(x, y, rectWidth, rectHeight, 90, 135, Arc2D.OPEN));
```

```
Double g.setStroke(stroke);
g.draw (new Ellipse2D.Double(x, y, rectWidth, rectHeight));
```

```
int x1Points[] = {x, x+rectWidth, x, x+rectWidth};
int y1Points[] = {y, y+rectHeight, y+rectHeight, y};
GeneralPath polygon = new GeneralPath(GeneralPath.WIND_EVEN_ODD, x1Points.length);
polygon.moveTo(x1Points[0], y1Points[0]);
for (int index = 1; index < x1Points.length; index++) { polygon.lineTo(x1Points[index],
y1Points[index]); };
polygon.closePath();
g.draw(polygon);
```

Suite ...

	int x2Points[] = {x, x+rectWidth, x, x+rectWidth};
	int y2Points[] = {y, y+rectHeight, y+rectHeight, y};
	GeneralPath polyline = new **GeneralPath**(GeneralPath.WIND_EVEN_ODD, x2Points.length);
	polyline.moveTo (x2Points[0], y2Points[0]);
	for (int index = 1; index < x2Points.length; index++)
	{ polyline.lineTo(x2Points[index], y2Points[index]); };
	g.draw(polyline);

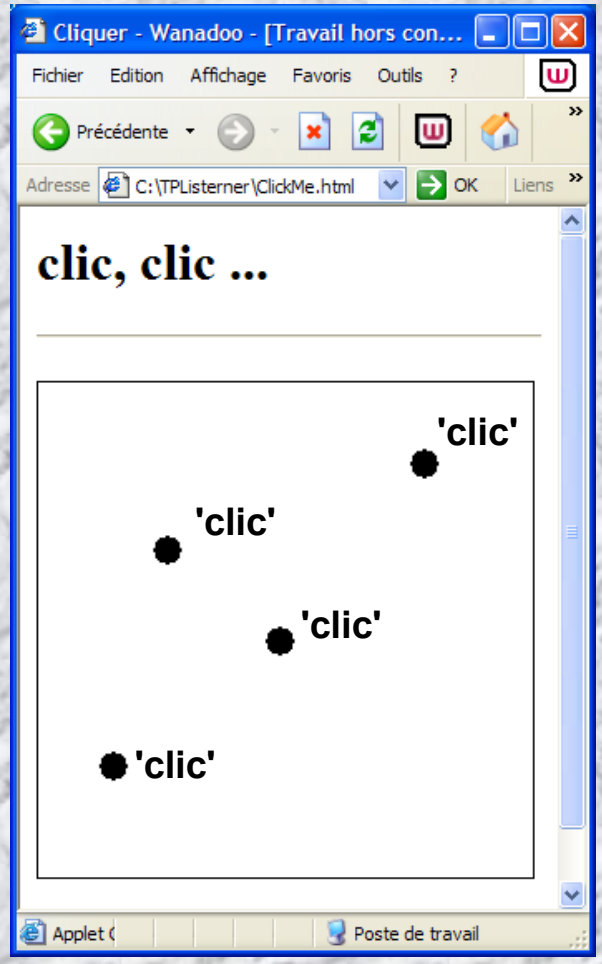
	Double (red) g.setPaint(red); g.**fill** (new Rectangle2D.Double(x, y, rectWidth, rectHeight));
	Double g.setPaint(redtowhite); g.**fill** (new RoundRectangle2D.Double(x, y, rectWidth, rectHeight, 10, 10));
	g.setPaint(red); g.**fill** (new Arc2D.Double(x, y, rectWidth, rectHeight, 90, 135, Arc2D.OPEN));
	Double g.setPaint (redtowhite); g.**fill** (new Ellipse2D.Double(x, y, rectWidth, rectHeight));

	GeneralPath int x3Points[] = {x, x+rectWidth, x, x+rectWidth};
	int y3Points[] = {y, y+rectHeight, y+rectHeight, y};
	GeneralPath filledPolygon = new GeneralPath(GeneralPath.WIND_EVEN_ODD, x3Points.length);
	filledPolygon.moveTo(x3Points[0], y3Points[0]);
	for (int index = 1; index < x3Points.length; index++)
	{ filledPolygon.lineTo(x3Points[index], y3Points[index]); };
	filledPolygon.closePath();
	g.setPaint(red); g.fill(filledPolygon);

Exercice 1-Applet: - Expérimenter la gestion d'évènements avec une Applet –

Ecrire une applet ClickMe.java qui permet de tracer une zone graphique rectangulaire dans laquelle, un spot rouge est affiché à l'endroit (x,y) où un clic de souris est pressé. On implémentera un gestionnaire MouseListerner situé dans ClickMe, et particulièrement la méthode mousePressed (…). Une classe Spot.java sera conçue séparément de la classe ClickMe, son rôle est de stocker la position (x, y) du clic de souris, là où sera affiché le spot circulaire rouge. A la création d'un objet Spot, x=y= -1. Les classes ClickMe et Spot seront placées dans le même répertoire, un fichier html sera conçu pour charger l'applet clickMe.class avec un navigateur quelconque.

```
public class ClickMe extends Applet implements MouseListener {
   private Spot spot = null;
   private static final int RAYON = 7;
   public void init() { <Chargement du gestionnaire MouseListerner>   }

   public void paint(Graphics g) {
      <Dessiner une zone rectangulaire à fond blanc>
      <Dessiner le spot et le remplir couleur rouge>
                  }
   }
   public void mousePressed(MouseEvent event) {
        <Créer un objet spot (rayon 7) de type Spot >
           }
      <Récupérer dans l'objet spot, les coordonnées x, y du clic >
      repaint();
   }
   //Autres actions non implémentées
   public void mouseClicked(MouseEvent event) {}
   public void mouseReleased(MouseEvent event) {}
   public void mouseEntered(MouseEvent event) {}
   public void mouseExited(MouseEvent event) {}
}
```



ClickMe.class
Spot.class
ClickMe.html

Importer les bibliothèques suivantes:
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

Exercice1-Applet: (Corrigé) - Expérimenter la gestion d'évènements avec une Applet –

```
public class ClickMe extends Applet implements MouseListener {
   private Spot spot = null;
   private static final int RADIUS = 7;
   public void init () {
      addMouseListener(this);
   }
   public void paint (Graphics g) {
      // dessine un cadre noir et un fond blanc
      g.setColor (Color.white);
      g.fillRect (0, 0, getSize().width - 1, getSize().height - 1);
      g.setColor (Color.black);
      g.drawRect (0, 0, getSize().width - 1, getSize().height - 1);
      // dessinne un spot
      g.setColor(Color.red);
      if (spot != null) {
         g.fillOval (spot.x - RADIUS, spot.y - RADIUS, RADIUS * 2, RADIUS * 2);
      }
   }
   public void mousePressed (MouseEvent event) {
      if (spot == null) {
         spot = new Spot(RADIUS);
      }
      spot.x = event.getX();
      spot.y = event.getY();
      repaint();
   }
   public void mouseClicked(MouseEvent event) {}
   public void mouseReleased(MouseEvent event) {}
   public void mouseEntered(MouseEvent event) {}
   public void mouseExited(MouseEvent event) {}
}
```

```
public class Spot {
   public int size;
   public int x, y;
   public Spot(int intSize) {
      size = intSize;
      x = -1;
      y = -1;
   }
}
```

Exercice1-Applet: (Corrigé) - Expérimenter la gestion d'évènements avec une Applet –

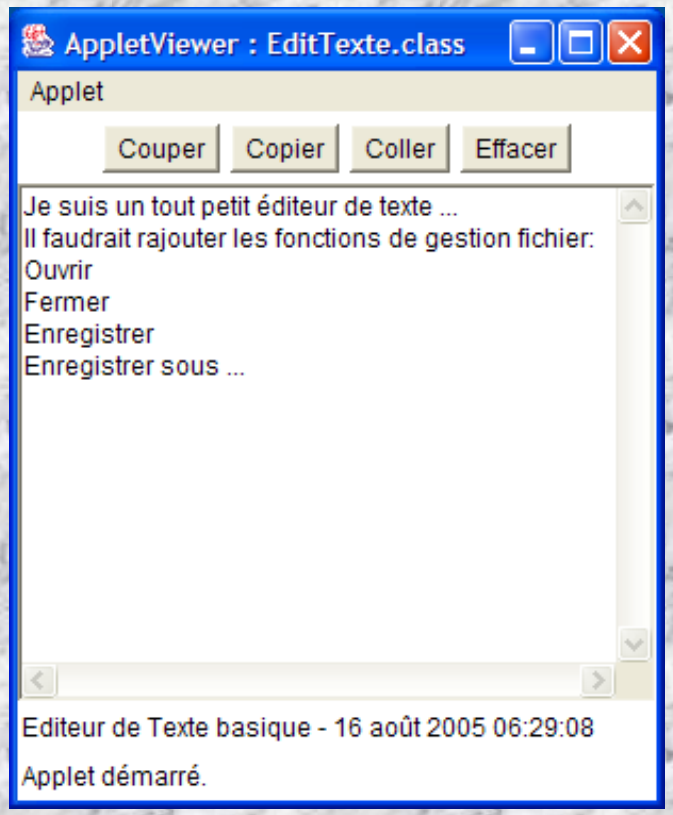
```
<HTML>
        <HEAD>
    <TITLE> Cliquer</TITLE>
        </HEAD>
        <BODY>
    <H1>clic, clic ...</H1>
                    <HR>
                    <P>
<APPLET CODE="ClickMe.class" WIDTH="300" HEIGHT="300">     </APPLET>
                    </P>
                    <HR>
        </BODY>
</HTML>
```

Exercice2-Applet: - Construire un éditeur de texte simpliste

Ecrire une applet EditTexte.java qui permet de définir une zone texte, associée avec un panel de boutons Copier-Couper-Coller-Effacer permettant quelques fonctions simples de manipulation de texte. Insérer à la fin le Label « Editeur de Texte basique » suivi de la date courante. Revoir les classes TextAera, Panel, layout, Button, Event, ainsi que les méthodes utiles associées.

Importer les bibliothèques:

```
import java.applet.Applet;
import java.awt.*;
import java.util.Date;
```




```
public class EditTexte extends Applet
{
 private TextArea texte  = new TextArea ();  //Zone de texte
   …

 public void init ()
 {
   <Choix d'un layout BorderLayout >
   <Création des boutons Couper-Copier-Coller-Effacer >
    <Création d'un Panel>
    <Insertion des boutons dans le panel créé>
    <Centrage dans le Panel de la barre des boutons et de la zone de texte>
    <Affichage et centrage de la Date>
 }

 // Méthode appelée quand on clique sur un bouton
 public boolean action (Event event, Object eventArg)
 {
   if ("Couper".equals (eventArg))
     { <compléter>  }

   else if ("Copier".equals (eventArg))
       <compléter>

   else if ("Coller".equals (eventArg))
     <compléter>

   else if ("Effacer".equals (eventArg))
     <compléter>

   return true;
 }
}
```

Exercice 2 Applet (Corrigé): - Construire un éditeur de texte simpliste

Le source JAVA …

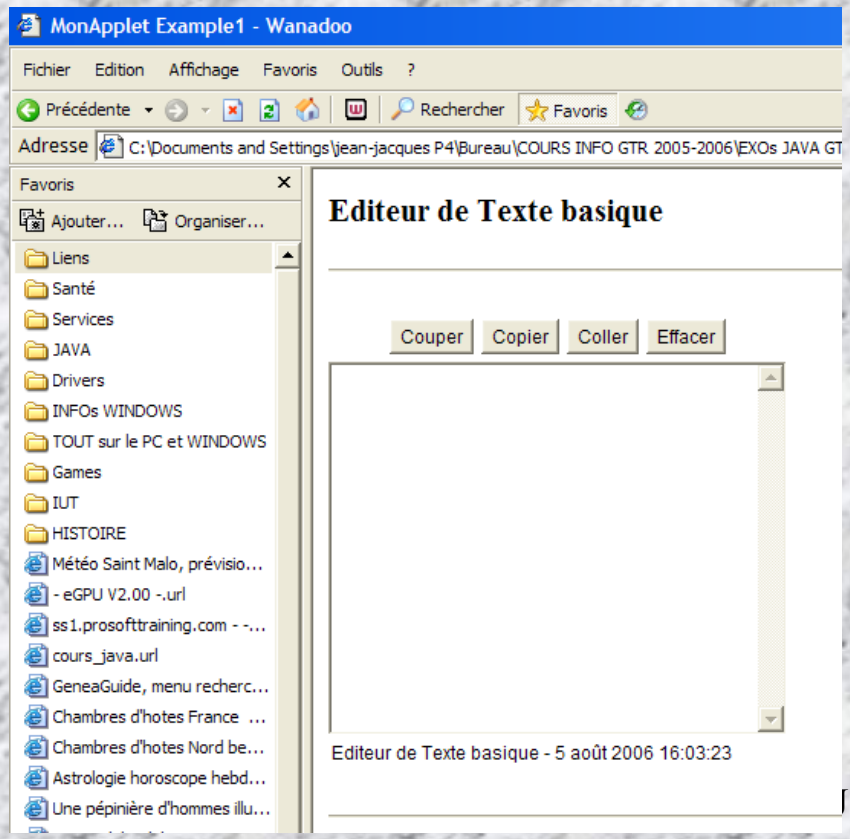
```
public class EditTexte extends Applet {
  private TextArea texte      = new TextArea ();
  private String   texteCopie = "";

  public void init ()
  {
    setLayout (new BorderLayout ());  // Choix d'un layout BorderLayout (FlowLayout par défaut)
    Panel panel = new Panel ();  // Création d'une barre de boutons avec les commandes
    panel.add (new Button ("Couper"));  // Couper/Copier/Coller/Effacer (layout FlowLayout par défaut)
    panel.add (new Button ("Copier"));
    panel.add (new Button ("Coller"));
    panel.add (new Button ("Effacer"));
    add ("North", panel);    // Ajout en haut de la barre de bouton
    add ("Center", texte);  // Ajout au centre de la zone de saisie
     add ("South", new Label (  "Editeur de Texte basique - " + new Date ().toLocaleString ()));  // Ajout en bas d'un label
  }

  public boolean action (Event event, Object eventArg) // Méthode appelée quand on clique sur un bouton
  {
    if ("Couper".equals (eventArg))
    {
      // Simulation d'une action Copier/Effacer
      postEvent (new Event (this, Event.ACTION_EVENT, "Copier"));
      postEvent (new Event (this, Event.ACTION_EVENT, "Effacer"));
    }
    else if ("Copier".equals (eventArg))

      texteCopie = texte.getSelectedText (); // Récupération du texte sélectionné
    else if ("Coller".equals (eventArg))
      // Remplacement de la sélection par texteCopie
      texte.replaceText (texteCopie, texte.getSelectionStart (), texte.getSelectionEnd ());
    else if ("Effacer".equals (eventArg))
      // Remplacement de la sélection par rien
      texte.replaceText ("", texte.getSelectionStart (), texte.getSelectionEnd ());
    return true;
  }
}
```

Exercice 2 : (Corrigé-suite) - Construire un éditeur de texte simpliste

Le fichier html …

```
<HTML>
  <HEAD> <TITLE>MonApplet Exemple1</TITLE> </HEAD>
  <BODY>
    <H1>Editeur de Texte basique</H1>
    <HR>
     <P>
       <APPLET CODE="EditTexte.class" WIDTH="300" HEIGHT="300">     </APPLET>
     </P>
     <HR>
  </BODY>
</HTML>
```

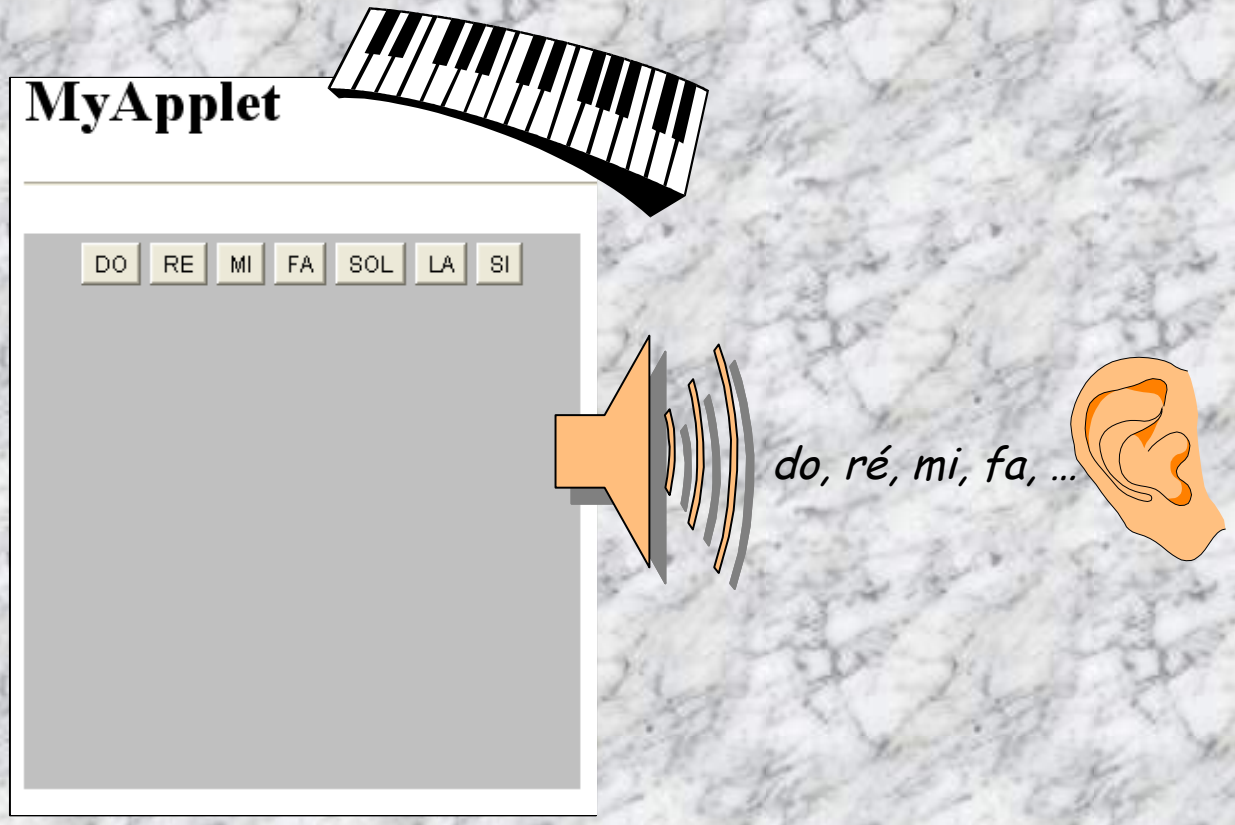
Résultat …



Exercice 3 (Applet): - Gestion de sons –

Ecrivons une applet qui permet de jouer les 7 notes de musique associées chacune à un bouton labellisé do, ré, mi, … Les sons sont déclenchés par un click de souris.
On suppose que les fichiers do.au , re.au, …, si.au sont dans le même dossier que l'applet.
[Etudier les méthodes: getAudioClip(…), getCodebase(…), play(…). Revoir la gestion des évènements]

JDK >1.2 autorise la création et jouer des clips audio au sein d'applets ou d'applications. Le clips peut-être dans l'un des formats suivants: AIFF, AU , WAV, MIDI (fichier de type 0 ou type 1), RMF
Le dispositif son peut gérer des données audio 8 et 16 bits avec un taux d'échantillonnage virtuellement quelconque.Les fichiers audio avec le JDK sont réglés à une vitesse d'échantillonnage de 22 kHz en 16-bit stereo. Si le matériel ne supporte pas des données de 16-bitou la stereo, 8-bit ou l'audio mono est fournie.



Exercice 3 (Applet): (Corrigé) - Gestion de sons-

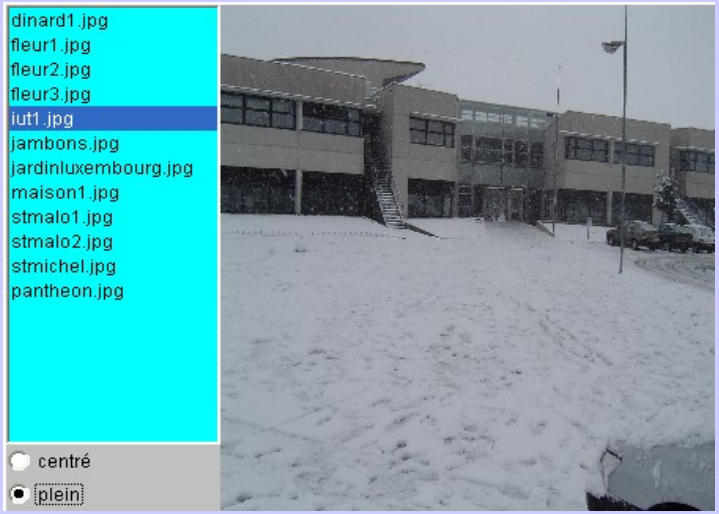
```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class Son1 extends Applet implements ActionListener {
   AudioClip Do; AudioClip re; AudioClip mi; AudioClip fa; AudioClip sol; AudioClip la; AudioClip si;

  Button bdo=new Button("DO"); Button bre=new Button("RE"); Button bmi=new Button("MI");
  Button bfa=new Button("FA"); Button bsol=new Button("SOL"); Button bla=new Button("LA");
  Button bsi=new Button("SI");

  public void init() {
   Do = getAudioClip (getCodeBase(),"do.au");
   re = getAudioClip (getCodeBase(),"re.au");
   mi = getAudioClip (getCodeBase(),"mi.au");
   fa = getAudioClip (getCodeBase(),"fa.au");
   sol = getAudioClip (getCodeBase(),"sol.au");
   la = getAudioClip (getCodeBase(),"la.au");
   si = getAudioClip (getCodeBase(),"si.au");

 add(bdo); add(bre); add(bmi); add(bfa); add(bsol); add(bla); add(bsi);

   bdo.addActionListener (this);
   bre.addActionListener (this);
   bmi.addActionListener (this);
   bfa.addActionListener (this);
   bsol.addActionListener (this);
   bla.addActionListener (this);
   bsi.addActionListener (this);

    }
     public void actionPerformed(ActionEvent e) {
    if (e.getSource()==bdo) Do.play();
     else if (e.getSource()==bre) re.play();
     else if (e.getSource()==bmi) mi.play();
     else if (e.getSource()==bfa) fa.play();
     else if (e.getSource()==bsol)sol.play();
     else if (e.getSource()==bla) la.play();
     else if (e.getSource()==bsi) si.play();
    }
    // public void stop() { if (bruitFond) si.stop(); } }
}
```

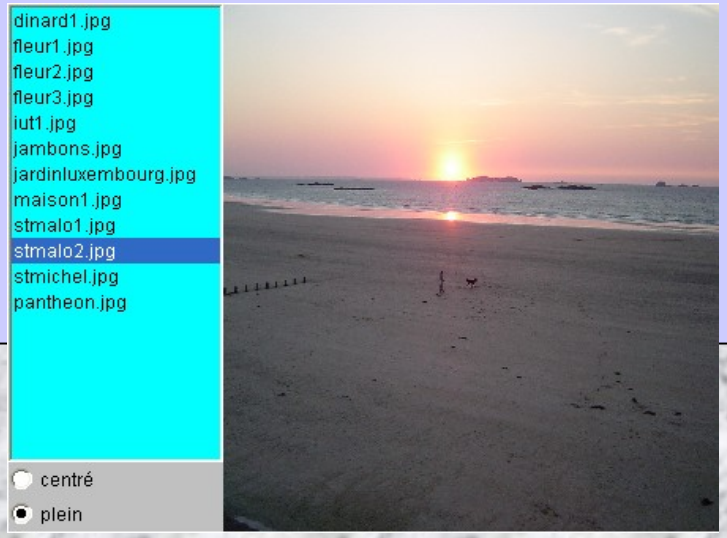
Exercice 4 (Applet): -**Visionner des images – Usage d'un Canvas**

Ecrire une applet visio.java permettant de visionner des images de types jpg ou gif . On suppose que les 12 fichiers images listés dans la fenêtre sont placé dans le même dossier que l'applet visio.class.

Le fichier visio.html est tel que:

```
<HTML>
<HEAD>
 <TITLE>Visionneuse d'images</TITLE>
</HEAD>
<BODY>
 <H1>Visionneuse d'images</H1>
 <HR>
 <DIV ALIGN=center>
 <APPLET CODE="Visio.class" WIDTH="450" HEIGHT="350">
 <PARAM NAME="img1" VALUE="dinard1.jpg">
 <PARAM NAME="img2" VALUE="fleur1.jpg">
 <PARAM NAME="img3" VALUE="fleur2.jpg">
 <PARAM NAME="img4" VALUE="fleur3.jpg">
 <PARAM NAME="img5" VALUE="iut1.jpg">
 <PARAM NAME="img6" VALUE="jambons.jpg">
 <PARAM NAME="img7" VALUE="jardinluxembourg.jpg">
 <PARAM NAME="img8" VALUE="maison1.jpg">
 <PARAM NAME="img9" VALUE="stmalo1.jpg">
 <PARAM NAME="img10" VALUE="stmalo2.jpg">
 <PARAM NAME="img11" VALUE="stmichel.jpg">
 <PARAM NAME="img12" VALUE="pantheon.jpg">
 </APPLET>
 </DIV>
 <HR>
</BODY>
</HTML>
```





Insérer dans la classe les bibliothèques suivantes:

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
```

Exercice 4 (suite): -Visionner des images –

On utilisera une zone image particulière pour l'insertion des différentes images prévues. On prévoira 2 mode de visualisation; le mode centré: on regarde le centre de l'image; le mode plein: on visionne toute l'image. A cet effet, on construira une classe Canvas ImaZone afin de spécifier un type zone d'image. Un objet ima instancié d'ImaZone permet de définir une zone dans laquelle on peut: insérer une image setImage (...), régler le mode setMode (…), dessiner l'image selon le mode choisie paint (…).

```
class ImaZone extends Canvas {
 Image img;
 int mode=0;

 ImaZone () {
  img=null;
  setBackground(Color.white);
 }

 void setImage (Image img) {
  this.img=img;
  repaint();
 }

 void setMode (int m) { mode=m; }

 public void paint (Graphics g) {
  if (img!=null) {
   switch (mode) {
    case 0 :
     int x=(getSize().width-img.getWidth(this))/2;
     int y=(getSize().height-img.getHeight(this))/2;
     g.drawImage(img, x, y, this); break;
    case 1 :
     g.drawImage(img,0,0,getSize().width, getSize().height,this);
     break;
   }
  }
 }
```

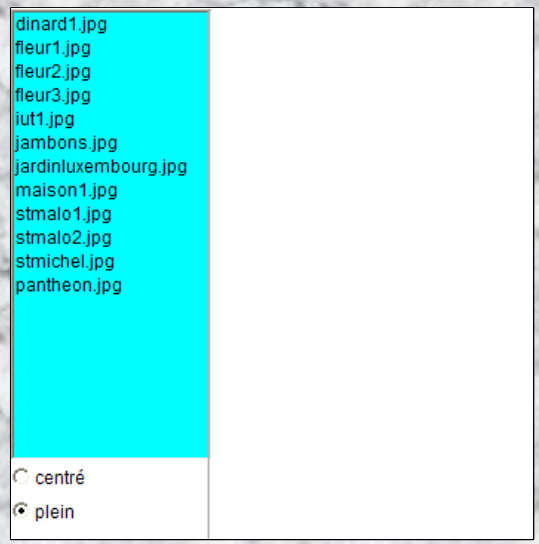
Au départ, la zone image est blanche

On visualise l'image

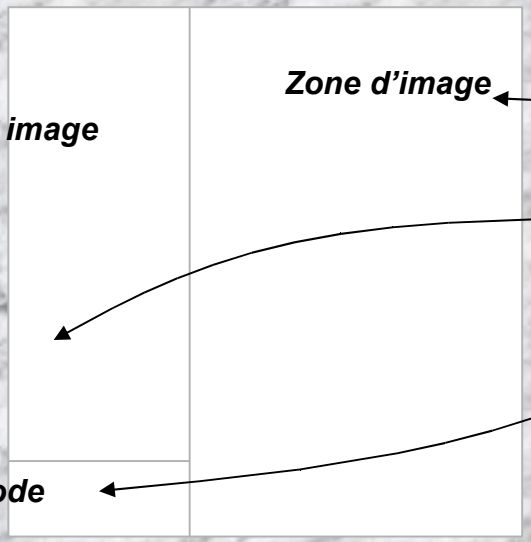
On règle le mode: centré ou plein

On dessine l'image centrée

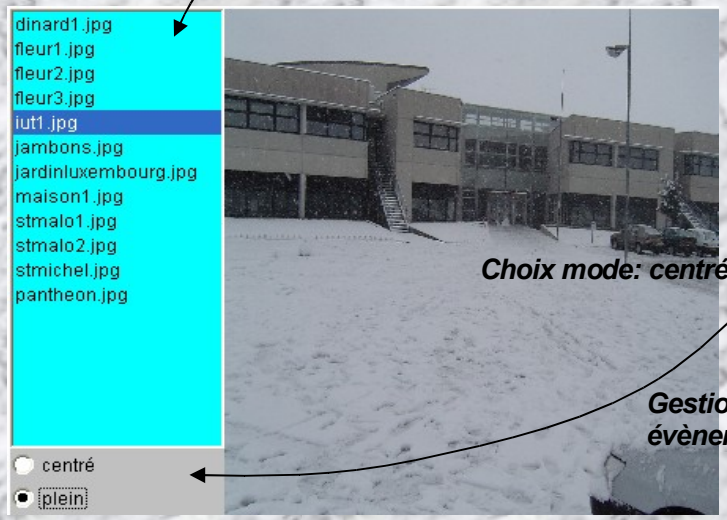
On dessine l'image entière



Exercice 4 (Applet): (Corrigé) -Visionner des images –



Insertion liste d'images



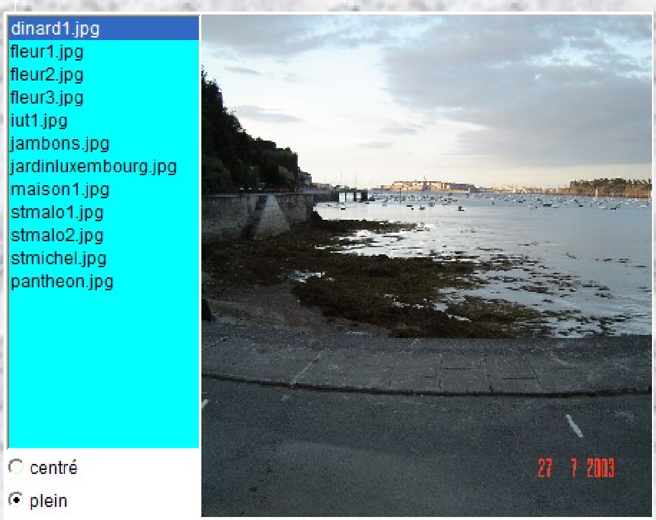
Gestion évènement choix

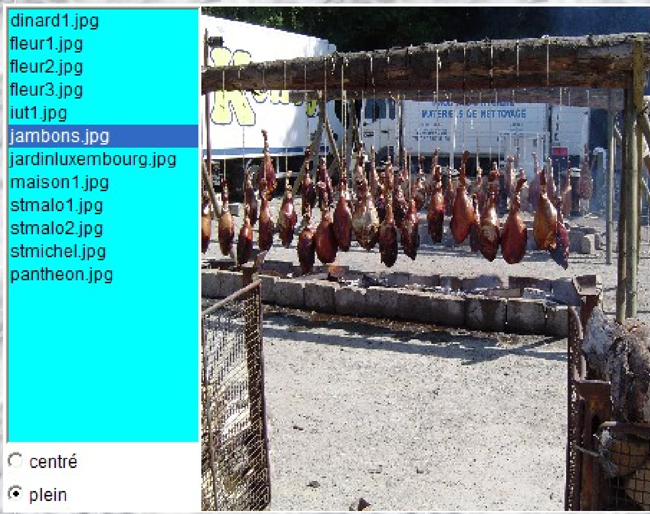
Création d'une zone d'image

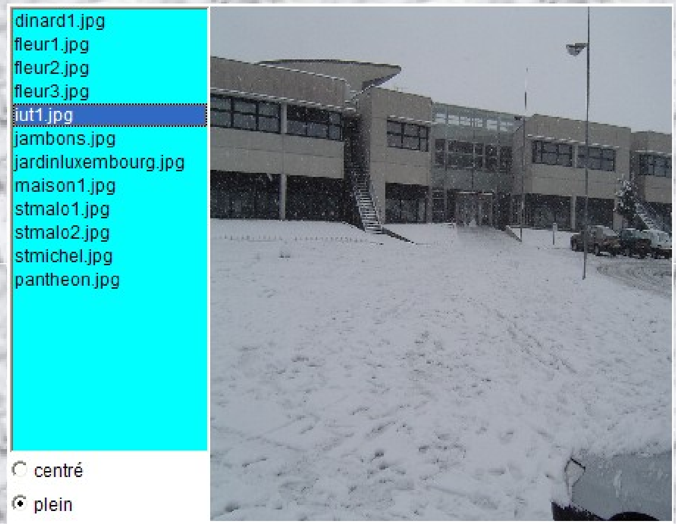
```
public class Visio extends Applet implements ItemListener {
 ImaZone ima = new ImaZone();
 List choix=new List();
 Checkbox mode0,mode1;

 public void init() {
  setLayout(new BorderLayout());
  add( ima, BorderLayout.CENTER);
  Panel gauche=new Panel();
  add(gauche,BorderLayout.WEST);
  gauche.setLayout(new BorderLayout());
  gauche.add(choix,BorderLayout.CENTER);
  choix.addItemListener(this);
  choix.setMultipleMode(false);
  choix.setBackground(Color.cyan);
  //lecture des paramètres et liste
   int i=1;
   String imgName;
   do {
     imgName=getParameter("img"+i);
     if (imgName!=null) choix.add(imgName);
     i++;
   } while (imgName!=null);
  //mode d'affichage
   Panel chk=new Panel();
   gauche.add (chk,BorderLayout.SOUTH);
   CheckboxGroup groupeRadio = new CheckboxGroup();
   mode0 = new Checkbox("centré",groupeRadio,true);
   mode1 = new Checkbox("plein",groupeRadio,false);
   chk.setLayout(new GridLayout(2,1));
   chk.add(mode0); chk.add(mode1);

   mode0.addItemListener(this); mode1.addItemListener(this);
  }
  public void itemStateChanged (ItemEvent e) {
   if (e.getStateChange()==ItemEvent.SELECTED) {
    if (mode0.getState()) ima.setMode(0);
    else ima.setMode(1);
    Image img=getImage(getCodeBase(),choix.getSelectedItem());
    ima.setImage(img);
   }
  }
 //insérer ici la classe canvas Imazone
}//finclassVisio
```

Le fichier visio.html …

```
<HTML>
<HEAD>
 <TITLE>Visionneuse d'images</TITLE>
</HEAD>
<BODY>
 <H1>Visionneuse d'images</H1>
 <HR>
 <DIV ALIGN=center>
 <APPLET CODE="Visio.class" WIDTH="450" HEIGHT="350">
 <PARAM NAME="img1" VALUE="dinard1.jpg">
 <PARAM NAME="img2" VALUE="fleur1.jpg">
 <PARAM NAME="img3" VALUE="fleur2.jpg">
 <PARAM NAME="img4" VALUE="fleur3.jpg">
 <PARAM NAME="img5" VALUE="iut1.jpg">
 <PARAM NAME="img6" VALUE="jambons.jpg">
 <PARAM NAME="img7" VALUE="jardinluxembourg.jpg">
 <PARAM NAME="img8" VALUE="maison1.jpg">
 <PARAM NAME="img9" VALUE="stmalo1.jpg">
 <PARAM NAME="img10" VALUE="stmalo2.jpg">
 <PARAM NAME="img11" VALUE="stmichel.jpg">
 <PARAM NAME="img12" VALUE="pantheon.jpg">
 </APPLET>
 </DIV>
 <HR>
</BODY>
</HTML>
```







Quelques choix …

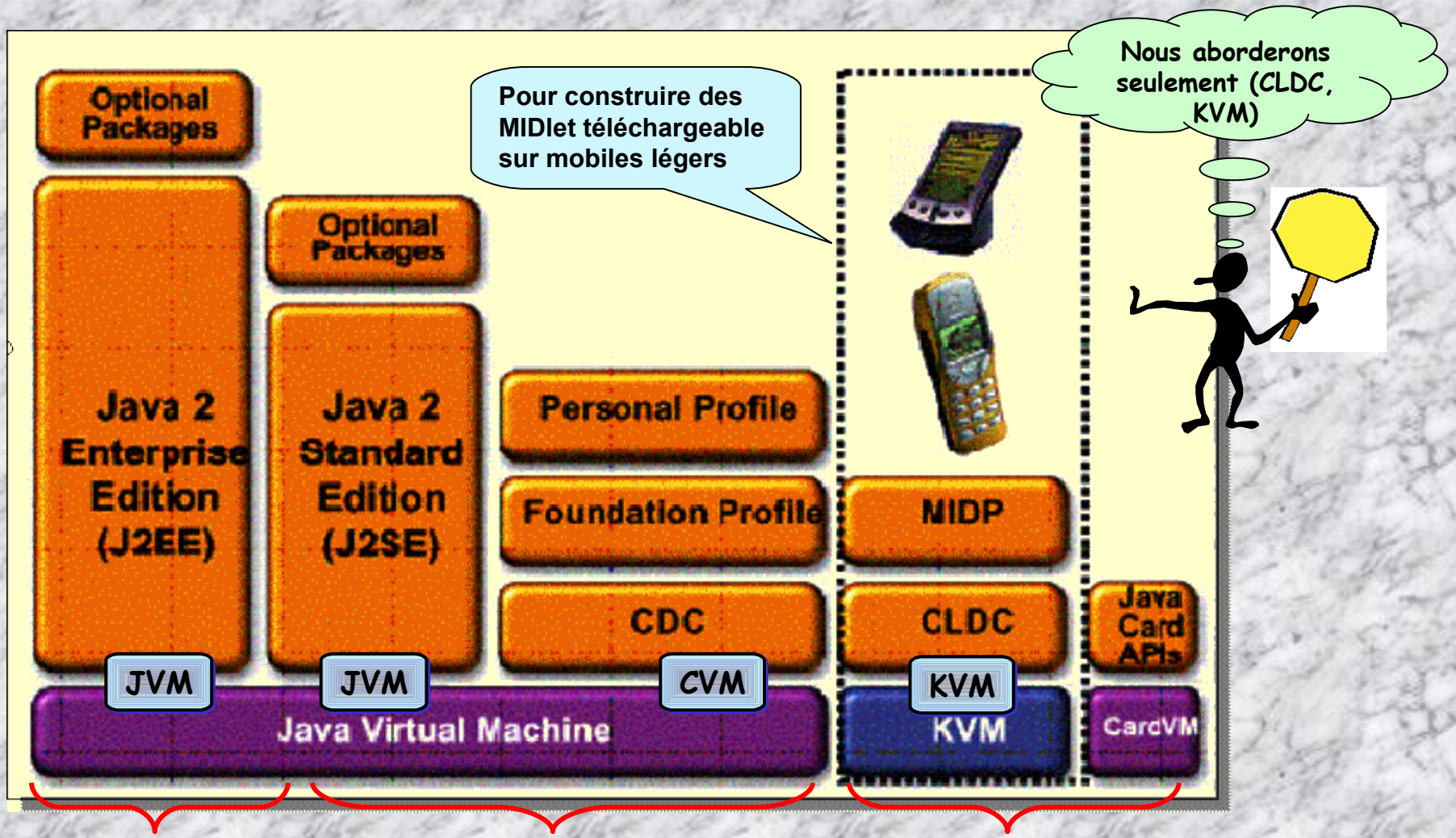
-14-

-Les MIDlets-

L'écriture de codes mobiles

Plateformes J2ME™ & Matériels légers mobiles

Pour concevoir des applications embarquées sur dispositifs mobiles, 2 couples de standards: (CDC, CVM) et (CLDC, KVM)



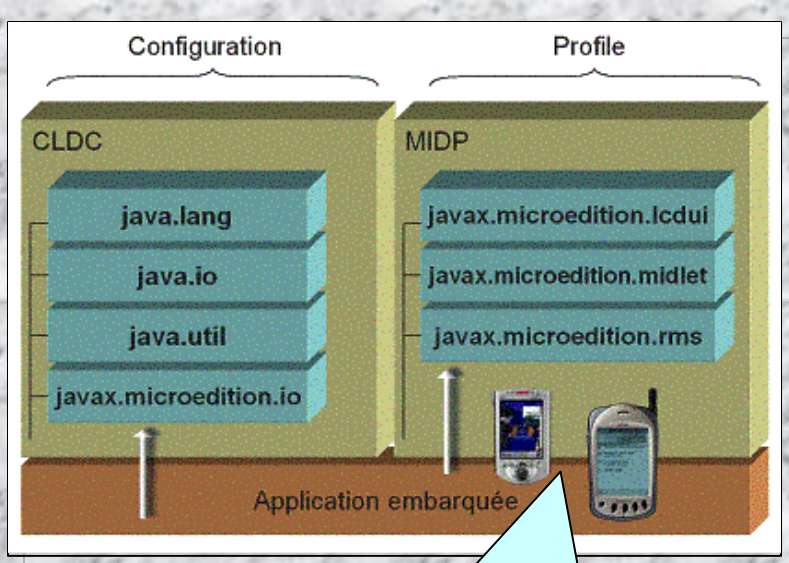
Conception d'applications JAVA embarquées = SDK + CLDC + MIDP

Les profiles : standard MIDP - Ils permettent à une certaine catégorie de terminaux d'utiliser des caractéristiques communes telles que la gestion de l'affichage, des évènements d'entrées/sorties (pointage, clavier, …) ou des mécanismes de persistance (Base de données légère intégrée).

Les configurations : standard CLDC - Elles définissent une plate-forme minimale en terme de services concernant un ou plusieurs profiles donnés.

Les machines virtuelles : En fonction de la cible, la machine virtuelle pourra être allégée afin de consommer plus ou moins de ressources (KVM, CVM, …)

Le système d'exploitation : L'environnement doit s'adapter à l'OS existant (Windows CE, Palm Os, …)

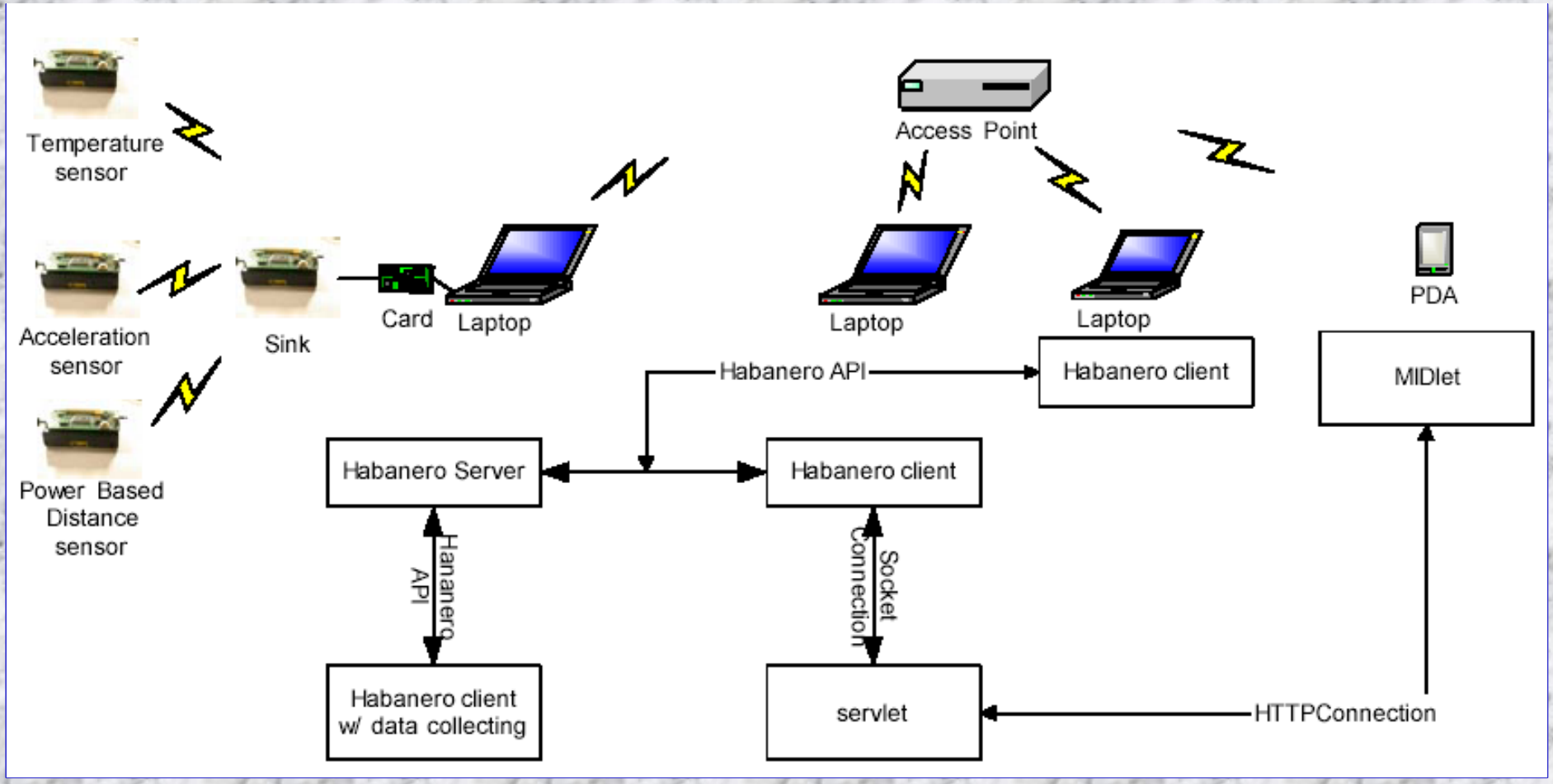


L'architecture J2ME se découpe en plusieurs couches

CLDC (Connected Limited Device Configuration)

Matériels à ressources limitées: téléphones cellulaires, assistants personnels, périphériques légers sans fil (wireless)

- ✓ Minimum de 160Ko à 512Ko de RAM,
- ✓ processeur 16 ou 32 bits, vitesse 16Mhz ou plus
- ✓ Alimentation limitée, prise en charge d'une batterie
- ✓ Connexion réseau non permanente, sans fil.
- ✓ Interface graphique limitée ou inexistante



Monitoring Wireless Sensor Networks by Heterogeneous Collaborative Groupware
Liang Cheng, Tian Lin, Yuecheng Zhang, and Qing Ye Laboratory Of Networking Group

Paquetages JAVA à utiliser dans la conception de MIDlet pour les terminaux mobiles

Mobile Information Device Profile (MIDP), JSR-037

Le vérificateur de Java 2 Standard Edition (J2SE) n'est pas adapté pour des dispositifs contraints matériellement . En effet, il nécessite un minimum de 50 ko d'espace pour stocker le code binaire, et au moins 30-100 ko de RAM au « runtime ». En outre, la puissance du CPU nécessaire pour exécuter le traitement itératif du flux de données avec le vérificateur JDK standard doit être conséquente.

En conclusion, il est nécessaire d'utiliser des classes javax.* adaptées pour construire des applications embarquées sur mobiles.

java.util	Classes utilitaires
java.io	classes des entrées/sorties
java.lang	Classes de base
javax.microedition.io	Classes de connexion distantes
javax.microedition.lcdui	Classes pour les interfaces graphiques
javax.microedition.midlet	Classe dont dérive l'application midlet
javax.microedition.rms	Classes pour le stockage persistant de données en mémoire

Javac du sdk ≥ 1.2

cldc, midp

JAVA & LES MOBILES - MIDLET conçue avec J2ME Wireless Toolkit

- ✓ La structure du code d'un midlet est similaire à celle d'une applet
- ✓ Il n'y a pas de méthode main() et les MIDlets étendent toujours la classe MIDLET.
- ✓ L'interface utilisateur se trouvent dans le package **lcdui**.

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class HelloWorld extends MIDlet implements CommandListener {
 private Command cmdExit;
private Display monEcran;
 private TextBox maZoneTexte = null;

public HelloWorld () {
  monEcran = Display.getDisplay (this);
  cmdExit = new Command ("Sortie", Command.EXIT, 2);
  maZoneTexte = new TextBox ("HelloWorld", "Bonjour le monde", 256, 0);
  maZoneTexte.addCommand (cmdExit);
  maZoneTexte.setCommandListener (this);
 }
 public void startApp() { monEcran.setCurrent (maZoneTexte);  }

 public void pauseApp() { }

 public void destroyApp (boolean unconditional) { }

 public void commandAction (Command maCmd, Displayable d ) {
      if (maCmd == cmdExit) {
        destroyApp (false);
        notifyDestroyed ();
      }
 }
}
```

Package indispensable pour une midlet

Gestionnaire d'évènement assuré par HelloWorld

Le display est associé à l'objet courant

Créer une commande de sortie

Zone d'affichage de texte

Traitement des évènements

CYCLE DE VIE D'UNE MIDLET ET PAQUETAGES …

pauseApp()

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class MaMidlet extends MIDlet implements CommandListener {
  …
public MaMidlet  () { … }

 public void startApp() {  …  }

 public void pauseApp() { … }

 public void destroyApp (boolean unconditional) { … }

 public void commandAction (Command maCmd, Displayable maRepresentation)
    { … }
}
```

Cycle de conception d'une MIDlet selon SUN

① Editer le programme qui affiche sur un mobile: « Hello world MIDlet »

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

 /** Exemple de MIDlet affichant un texte et exécute une commande Exit.
public class HelloMIDlet extends MIDlet implements CommandListener {
    private Command exitCommand; // La commande exit
    private Display display;              // L'écran pour la MIDlet
    public HelloMIDlet() {
      display = Display.getDisplay(this);
      exitCommand = new Command ("Exit", Command.SCREEN, 2);
  }
 /** Démarre la MIDlet Hello en créant une TextBox, et en associant la commande exit et un listener */
    public void startApp() {
       TextBox t = new TextBox("Hello MIDlet", "Test string", 256, 0);
       t.addCommand(exitCommand);
       t.setCommandListener(this);
       display.setCurrent(t);
    }
/** Pause n'est pas implémenté puisqu'il n'y a pas d'activités de background et aucun stockage à clôturer */
    public void pauseApp() {    }
/** La destruction doit nettoyer tous ce qui n'est pas pris en charge par le garbage collector. Rien à nettoyer ici*/
    public void destroyApp(boolean unconditional) {    }
/*Réponse aux commandes ( exit, …)  La commande exit, nettoie et notifie que al MIDlet a été détruite*/
    public void commandAction (Command c, Displayable s) {
        if (c == exitCommand) {
       destroyApp(false);
       notifyDestroyed();
       }
    }
}//finclass
```

Hypothèse: localisation des de classes prévérifiées utilisées dans **lib/midp.jar** ou le s/répertoire **classes**

② Compiler la MIDlet …

Pour compiler tous les programmes conçus pour la plateforme MIDP, exécuter la commande suivante:

```
javac -d classes -classpath classes src/exemple/HelloMIDlet.java
```

③ Préverifier la MIDlet …

Après avoir compilé le programme, il doit être prévérifié par la commande preverify .
Cette commande doit être exécutée à partir du répertoire de la plateforme ad hoc (build/linux, build/solaris, build/win32)

```
preverify -classpath classes -d classes tmpclasses
```

④ MIDlet Descriptor File, Jar File Manifest, OTA listing

Créer un fichier descripteur pour la MIDlet - hello.jad

```
MIDlet-Name: HelloWorld  MIDlet-Version: 1.0.0
MIDlet-Vendor: Sun Microsystems, Inc.
MIDlet-Description: Sample Hello World MIDlet
MIDlet-Info-URL: http://java.sun.com/j2me/
MIDlet-Jar-URL: http://<host>/<path>/hello.jar
MIDlet-Jar-Size:    1020  MicroEdition-Profile:
MIDP-1.0  MicroEdition-Configuration: CLDC-1.0
MIDlet-1: HelloWorld,, HelloMIDlet
```

⑤ Créer un fichier `manifest` pour le fichier jar - hello.mf)

```
MIDlet-Name: HelloWorld  MIDlet-Version: 1.0.0
MIDlet-Vendor: Sun Microsystems, Inc.  MicroEdition-Profile:
MIDP-1.0  MicroEdition-Configuration: CLDC-1.0
MIDlet-1: HelloWorld,, HelloMIDlet
```

⑥ Créer une page HTML qui référence l'application pour un usage OTA - hello.html

```
<html>
  <body>
     <a href="http://<host>/<path>/hello.jad">Hello World <a>
  </body>
</html>
```

De plus, ajoutez un lien (voir ci-dessus) dans une page HTML existante. L'exécutable midp affichera seulement le lien ".jad" et ignorera le reste de la page page. Vérifier la présence de cette page dans le cas d'une application « wireless ».

⑦ « Packager » la MIDlet

Créer un fichier jar intégrant toutes les classes et ressources nécessaires à votre application.

```
cd  classes
jar cmf  hello.mf
```

⑧ Installation de la MIDlet sur un serveur Web

Placer le fichier descripteur, le fichier jar, et la feuille OTA dans un répertoire visible d'un serveur web. Ainsi, votre application pourra être téléchargée via le réseau.

1. Le serveur web doit être configuré pour fournir un type MIME spécifique type pour le fichier descripteur jad et le fichier archive jar.
2. Si Apache est utilisé, ajouter les 2 lignes suivantes au fichier texte **httpd.conf**:

```
AddType text/vnd.sun.j2me.app-descriptor      .jad
AddType application/java-archive              .jar
```

⑨ Exécuter la MIDlet à partir d'une ligne de commande

La commande ci-dessous montre comment utiliser l'exécutable midp avec une URL distante. L'argument -transient impose que le fichier descripteur doit être récupéré, et le fichier jar référencé dans le descripteur doit être temporairement installé et démarré. Lorsque l'application est quittée, le source temporairement installé est ôté de la mémoire.

```
midp -transient  http://<hostname>/<path>/hello.jad
```

⑩ Installer et exécuter la MIDlet en utilisant le GUI

Invoquer l'exécutable midp pour émuler le comportement d'un utilisateur pour récupérer et exécuter une application OTA. La présence ou non d'arguments provoque l'affichage du gestionnaire graphique. Celui-ci peut installer, lister, lancer, actualiser, et ôter les applications.

midp

Lorsque l'objet GUI est affiché par midp :

3. Choisir Install.
4. Entrer l'URL de hello.html, et sélectionner Go.
5. Sélectionner "Hello World" à partir de la liste affichée, et choisir Install.
6. Lorsque la liste des « suites » est affichée, choisir "HelloWorld« , et lancer là.

C'EST TOUT POUR AUJOURD'HUI.