Java - Les bases

Code: java-intro

Originaux

url: http://tecfa.unige.ch/guides/tie/html/java-intro/java-intro.html

url: http://tecfa.unige.ch/guides/tie/pdf/files/java-intro.pdf

Auteurs et version

- Daniel K. Schneider - Vivian Synteta

- Version: 1.2 (modifié le 4/12/00 par VS)

Prérequis

- Petites connaissances en programmation

  Module technique suppl.: java-util (explication des packages et classpath)

Modules

Module d'exercices: act-java-intro

Module technique suivant: java-jsp (Java server pages)

Module technique suivant: java-servl (Java servlets)

Module technique suppl.: java-jhtml (pages Java actives, démodées !!)

Objectifs

- Notions de Java de base
- (pas de "comment insérer un applet dans un page html" !)

1. Table de matières détaillée

2. Les premiers pas avec JAVA

- Buts de ce chapitre:
  - Donner une feeling pour Java
  - Apprendre a éditer, compiler et consulter la doc on-line

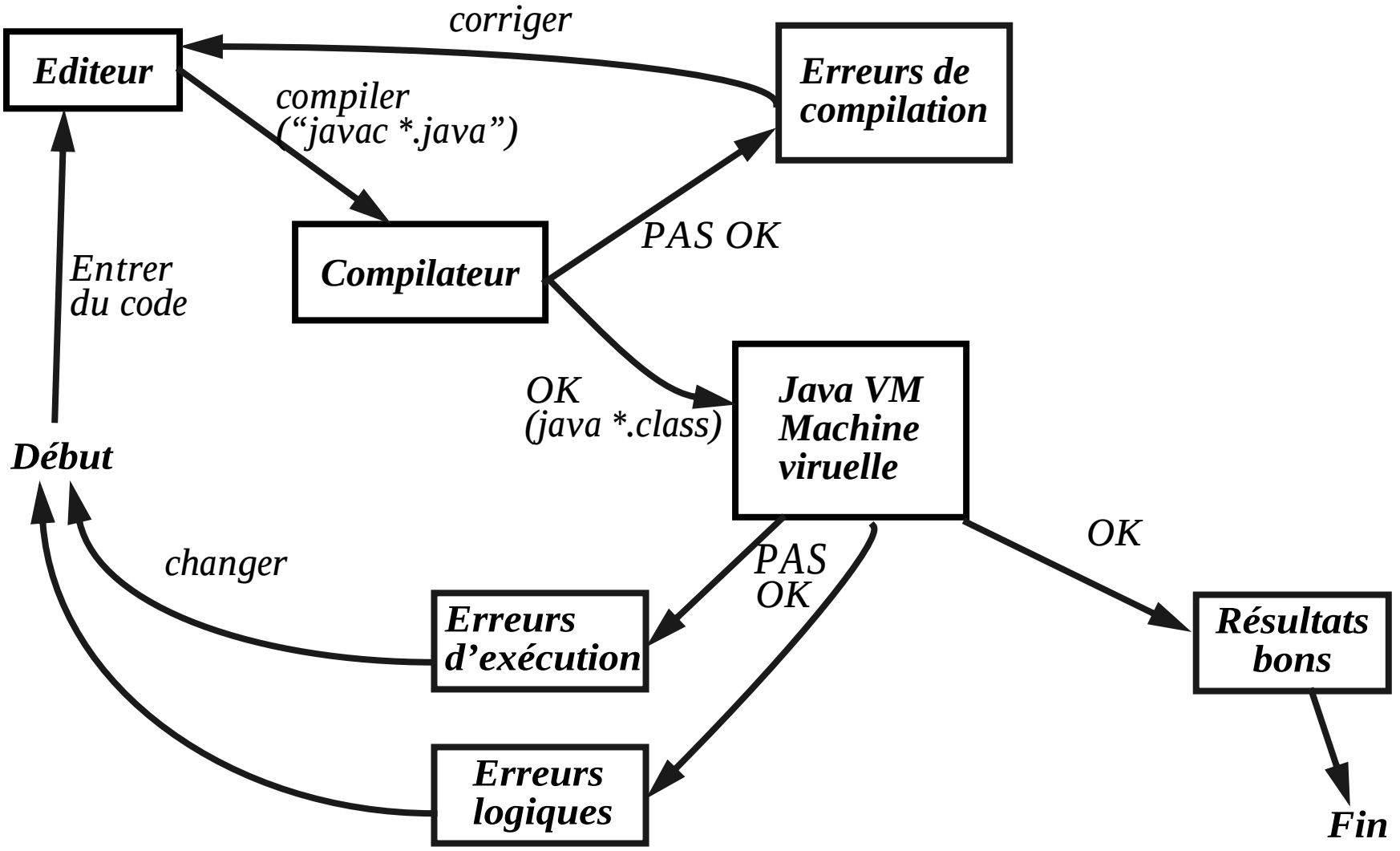
2.1 Caractéristiques et usage du langage Java

- Indépendance de la plateforme
- Fait pour le réseau: applets, servlets, libraries réseau
- Petit langage
  - beaucoup de librairies (packages)
- Strictement typé (“typed”)
- Fait par Sun (mais plusieurs autres implémentations)

Faire des bookmarks

- Page Staf2x
- Page Pointeurs JAVA de TECFA
- Répertoire exemples Staf2x

2.2 Le cycle de développement



2.3 La plus simple application (Hello)

Exemple 2-1: Hello World

http://tecfa.unige.ch/guides/java/staf2x/ex/basics/

```
// Everything in Java is a class
public class Hello {

  // All programs must have main()
  public static void main(String[] args) {
    // Say hello!
    System.out.println("Hello World!");

  } // This marks the end of main()
}   // Marks the end of the Hello class
```

Explications

- L'application est définie par la classe *Hello*
- / / sont des commentaires
- Le fichier doit s'appeler *Hello.java*
- Chaque application doit définir une méthode main
- Compiler / Exécuter:
  - **`javac`** Hello.java
  - **`java`** Hello
- Pour plus d'infos regarder:

  url: Your first cup of Java

3. Anatomie d'un programme Java

- Note: Il ne faut pas essayer de tout comprendre .....

Exemple 3-1: Simple dessin (Le Ring) Java Gently (2nd edition), p.20

```
import java.awt.*;
import java.awt.event.*;
class Ring extends Frame {
/* The Ring program inspired by Rings by J M Bishop Dec 1996
http://tecfa.unige.ch/guides/java/staf2x/ex/basics/Ring.java */
  public static void main (String [ ] args) {
    // Create a graphics frame of Class Ring
    // Set the Size and make it appear as
    // outlined in the paint method.
    Frame monFrame= new Ring ();
    monFrame.setSize (300, 200);
    monFrame.setVisible (true);
   // Add functionality for closing the window
   [coupé ici]
  public void paint (Graphics g) {
    // Draw a yellow ring
    g.setColor (Color.yellow);
    g.drawOval (100,50,50,50);
// Label the drawing
    g.drawString("LA bague !", 110,140);
  }
}
```

A. Packages et classes de l'API

- API = application programmer's interface
- Les packages sont des ensembles de classes qui rajoutent de la fonctionalité au langage
- Il faut explicitement déclarer chaque package qu'on désire utiliser
- Il existe 22 packages dans le JDK 1.1.7 et PLEINS de classes

  url: http://tecfa2.unige.ch/guides/java/jdk1.1/docs/api/packages.html

Importation de l'awt (interface utilisateur)

```
import java.awt.*;
import java.awt.event.*;
```

Définition de la classe principale "Ring"

- Utilisation et élargissement de la classe "système" *Frame* (fenêtre)

```
class Ring extends Frame { }
  .....
```

Instantiation d'un objet Ring

```
Frame monFrame= new Ring ();
monFrame.setSize (300, 200);
monFrame.setVisible (true);
```

- on crée un objet (instantiation) du type Frame avec la classe Ring
- et on utilise des méthodes de la classe Frame pour définir la taille et l'afficher
- ... pour les détails voir plus tard !

Définition de la méthode paint

```
public void paint (Graphics g) { ...}
```

- implémente une méthode paint pour afficher le contenu du Frame;
- la méthode paint nécessite un argument de type Graphics
  - "paint" est appelée automatiquement quand le frame se crée
  - la variable "g" instantie la classe Graphics

Appel à des méthodes de la classe Graphics

```
g.setColor (Color.yellow);
g.drawOval (100,50,50,50);
// Label the drawing
g.drawString("LA bague !", 110,140);
```

4. Classes et méthodes: un premier regard

4.1 Les classes

- Une classe représente une définition abstraite des "propriétés" et "capacités" d'un objet
- classe = structure d'information + méthodes de traitement d'information.
- Un programme Java est définie par un ensemble de classes
- Une fois définie une classe, on peut (et doit) créer des objets concrets.
- Lorsque le programme est exécuté, des objets seront crées et des méthodes seront exécutés.
- Des objets peuvent communiquer entre eux par le biais de méthodes "publiques"

Syntaxe de class

```
[Modificateur] class
NomDeClasse [extends SuperClasse] [implements Interface1[, Interface2]]
  {
    CorpsDeClasse
   }
... détails à suivre plus tard
```

Exemples simples:

```
public class Hello {..... }
public void paint (Graphics g) { .... }
```

A. Les constructeurs

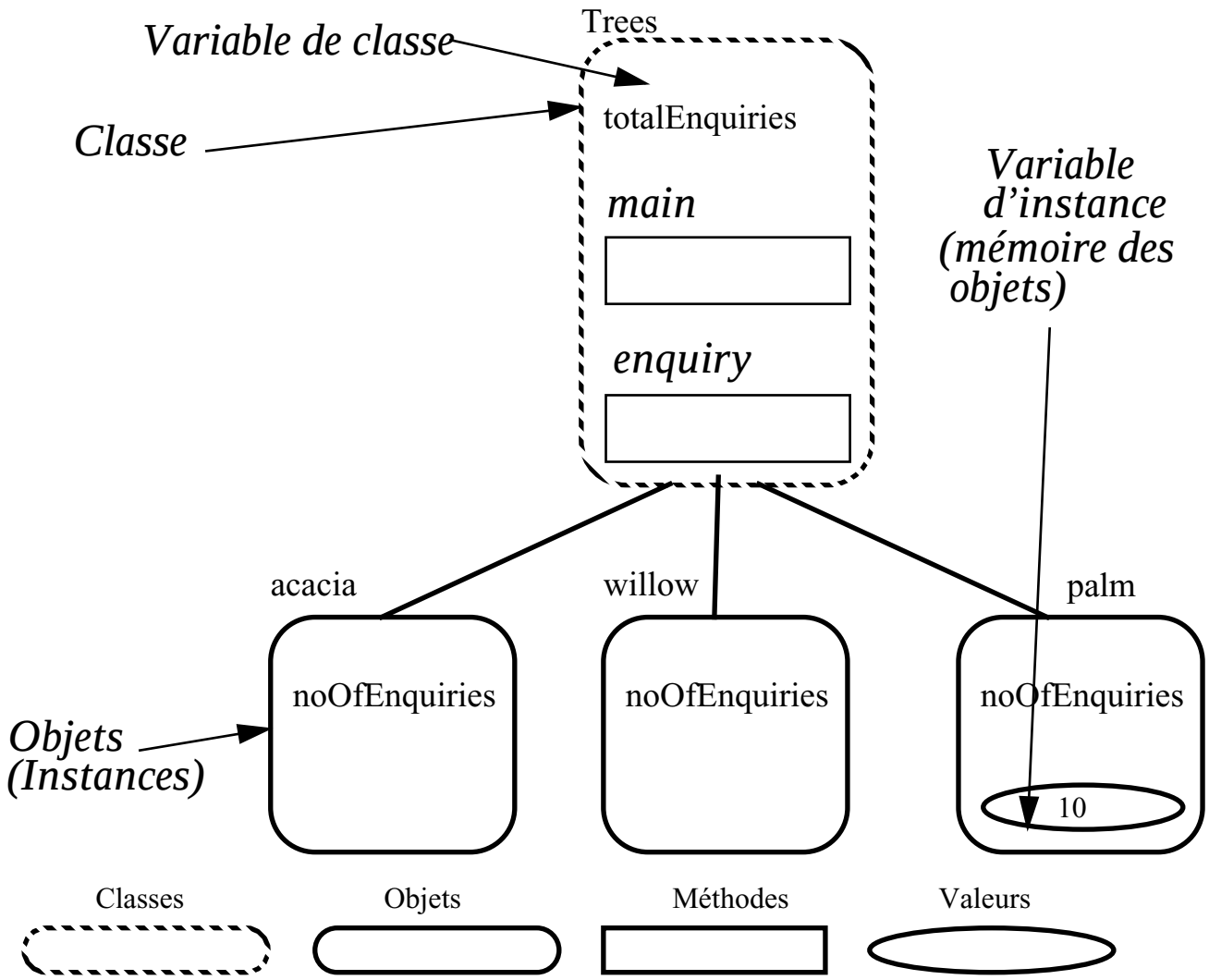
- Ce sont des méthodes spéciales destinées à instantier les classes;
- ils portent le même nom que la classe !
- ils ne retournent aucun type (pas même void);

Exemple 4-1: Classe et constructeur simple

```
// Game est une classe
class Game {
  }
// Game {} est un constructeur
Game MyGame = new Game {}
```

B. Les diagrammes de classe (Voir Java Gently, pp 30, 81ss)

Exemple 4-2: Les arbres



C. Un programme qui implémente nos arbres

```
//http://tecfa.unige.ch/guides/java/staf2x/ex/basics/Trees.java
class Trees {
    static int totalEnquiries;
    int noOfEnquiries;
    public Trees () {
    }
    public Trees (int n) {
     noOfEnquiries = n;
    }
    public static void main (String [ ] args) {
    // on crée 3 arbres
     Trees acacia = new Trees ();
     Trees willow = new Trees ();
     Trees palm = new Trees (10);

     palm.enquiry();
     System.out.println
     ("Number of enquiries for palm trees = " + palm.noOfEnquiries +
      "\nNumber of total enquiries = " + totalEnquiries );
    }
    void enquiry () {
    noOfEnquiries++;
    totalEnquiries++;
    }
}
```

D. Les constructeurs (again)

- On définit deux constructeurs:
  - un qui crée un simple objet de classe Trees
  - un qui permet en plus de donner une valeur à la variable noOfEnquiries

```
   public Trees () {   }

   public Trees (int n) {
noOfEnquiries = n;
   }
```

E. Simple Output

- System.out.println imprime un string
- Notez que l'on peut concatener strings et nombres pour créer un seul string:

```
System.out.println
    ("Number of enquiries for palm trees = " +
     palm.noOfEnquiries +
     "\nNumber of total enquiries = " +
     totalEnquiries
     );
```

F. Une méthode très simple

```
void enquiry () {
 noOfEnquiries++;
 totalEnquiries++;
 }
```

- enquiry incrémente des compteurs à chaque fois que l'on apelle
  `palm.enquiry();`
- appelle (invoque) cette méthode sur l'objet palm qui a été crée
- une méthode est toujours attachée à un objet ou une classe.

4.2 Les définitions de méthodes

- Détails, exemples, explications etc. voir plus tard !

Syntaxe d'une méthode

Syntaxe: [Modificateurs] TypeRetourné NomDeMéthode(ListeDeParamètres)
{ CorpsDeLaMéthode }
Les méthodes ont toujours un "TypeRetourné"

- Le *TypeRetourné* indique le type que la méthode "retourne" quand on l'invoque.
  - sauf s'il s'agit de constructeurs. Dans ce cas seulement, aucun type n'est spécifié (pas même void).
- La *liste des paramètres* s'exprime comme en langage C.
  (Type nom_de_variables, Type nom_de_variables, ...)
- Exemple de *modificateurs*:
  - **`public:`** La méthode peut être utilisée par d'autres méthodes en dehors de la même classe
  - **`private:`** Le contraire (en plus ce sont de méthodes "final")
  - **`abstract:`** Méthodes d'interfaces sans code, mais que des sous-classes doivent implémenter (à voir plus tard)
  - **`static:`** Méthode qui s'applique uniquement à la classe et aux variables de classe (pas aux instances, à voir plus tard)
  - **`protected et final:`** voir plus tard ....

A. Exemples:

• Une méthode publique typique qui ne retourne rien:

```
public void paint (Graphics g) { ..... }
```

• Une méthode publique qui retourne une valeur booléenne:

```
public boolean action(Event event, Object what) { ...}
```

• Une méthode statique typique:

```
public static void main (String [ ] args) {... }
```

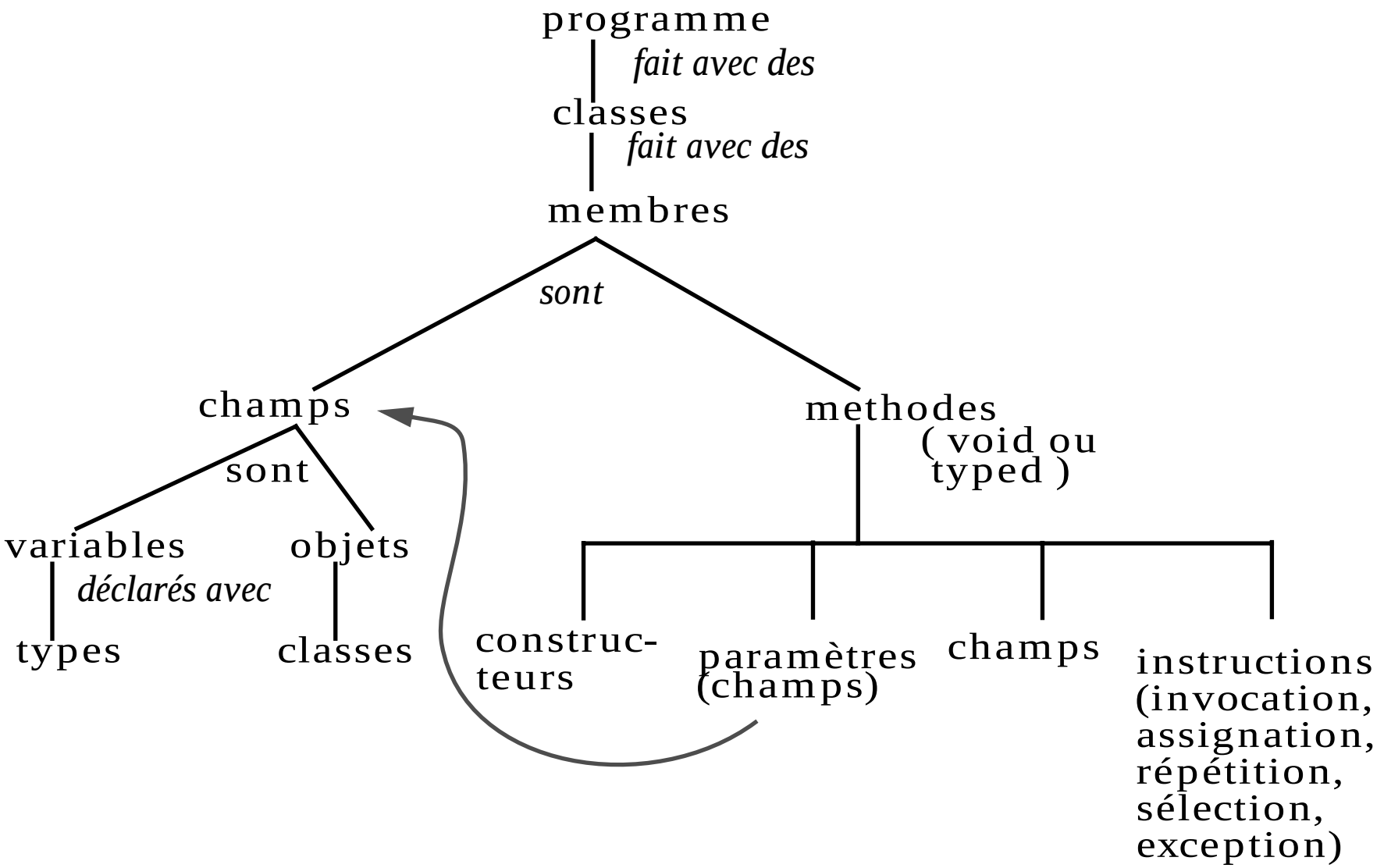
• Une méthode publique pour initialiser un applet

```
public class MyEAIapplet extends Applet { ..}
       public void init() {
```

B. Le langage JAVA est un langage polymorphique:

- Il est possible de faire de la surcharge de nom de méthodes :
  - utiliser un nom de méthode qui existe déjà pour une méthode de la même classe ou d'une superclasse.
- Il existe deux types de surcharge:
  - Surcharge "verticale" : déclarer une méthode de même spécification qu'une méthode héritée, mais en faire une spécification différente.
  - Surcharge "horizontale" : déclarer une méthode qui a le même nom qu'une autre méthode, mais pas la même liste de paramètres.
    (voir: D. "Les constructeurs (again)" [15])

4.3 La structure d'un programme JAVA



4.4 Simple I/O, assignation et simples variables

Exemple 4-3: HelloWorld interactif

http://tecfa.unige.ch/guides/java/staf2x/ex/basics/Greetings.java

```
import java.io.*;
class Greetings {
  /* A simple greetings program    by J M Bishop  Oct 1996
   * -------------------------     Java 1.1  Dec 1997 */
  public static void main (String [] args) throws IOException {
    BufferedReader in = new BufferedReader
          (new InputStreamReader(System.in));
    System.out.println("What is your name?");
    String name = in.readLine();
    System.out.println("Bonjour " + name);
  }
}
```

- Ignorez comment fonctionne le input pour le moment

  Mémorisez la méthode pour lire une ligne:

```
// 1. Définir un input buffer
BufferedReader in = new BufferedReader
       (new InputStreamReader(System.in));
// 2. Utiliser:
String xxxx = in.readLine()
```

5. Algorithmique de base

- Formule: Programme = structure de données + algorithmes
- Les algorithmes sont des *instructions* qui manipulent les données. Dans ce contexte on parle aussi de *structures de contrôle*.

5.1 Instructions les plus importantes

Nom	Fonctionalité	Voir:
assignation	assigner une valeur à une variable	5.3 “Assignation, type cast et expressions simples” [23]
for	"Pour faire"; faire N fois, etc.	5.4 “La répétition I: l’instruction for” [24]
if	"selection"; si alors sinon	5.5 “La sélection (if / else) et les conditionnels” [25]
while	"tant que faire", boucler tant qu’une condition soit remplie	5.7 “La répétition II (while)” [28]
try	"essayer" et si cela ne marche pas faire autre chose	5.8 “Les exceptions simples” [30]
expressions	calculer quelque chose, invoquer une méthodes etc.	5.9 “Classes et méthodes (encore une fois)” [32]

5.2 Les types de données

	Max / ou valeurs possibles	Exemple	Initiatlisation par défaut
Nombres			
byte	127	89	0
short	32767	-3	0
int	2147483647	10	0
long	10e18	10	0
float	10e38	123.456789	0.0
double	10e308	2.1	0.0
Autres			
boolean	true false	true	false
char	A-Z,a-z,!@...	*	""
Vecteurs et Matrices			
xxx []	-	int i []; char tablo [] [];	selon le type
Classes Java			
String	"....."	"Daniel S"	""
pleins d'autres !			

5.3 Assignation, type cast et expressions simples

Syntaxe: assignation simple		Exemples
variable = expression;		
variable	voir aussi 5.2 “Les types de données” [22]	i tree
expression	voir les manuels Java	x+y / 10 "Sylvere M" palm.enquiry() vrai | faux

Exemple 5-1: Simple Calcul

http://tecfa.unige.ch/guides/java/staf2x/ex/basics/SimpleCalculs.java

```
//
public class SimpleCalculs {

    // All programs must have main()
    public static void main(String[] args) {

 int somme = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10;
 float moyenne = somme / 10;
 System.out.println ("La moyenne est égale à " +  moyenne);
 }
    }
```

5.4 La répétition I: l'instruction for

Syntaxe: For-statement		Exemples
for (start; check; update) { body }		
start	initialisations de variables d'itération	int I=0; int Kaspar = 0;
check	condition qui teste s'il faut sortir de la boucle	i<10;
update	mise à jour de variables d'itération	i++; ou i=i+a;
body	instruction ou bloc d'instructions	System.out.println(i + "fois,")

Exemple 5-2: Imprimer des messages un peu répétitifs

http://tecfa.unige.ch/guides/java/staf2x/ex/basics/ForYou.java

```
public class ForYou {

      // All programs must have main()
      public static void main(String[] args) {
    // Say hello!
    for (int i=1; i<11; i++) {
      System.out.println("J'aimerai t'embrasser " + i + " fois\n ");
    }
      }
  }
```

5.5 La sélection (if/else) et les conditionnels

Syntaxe: if		Exemples
if (condition) bloc_d'instructions *else* bloc_d'instructions		
condition	(.....)	(nombre > 1)
bloc_ d'instructions	{instruction; instruction; ....} ou instruction_simple;	{ a = 1 + a; b = 2 + a; }

	instruction simple	for (int i=1) {
		System.out.println
		("forever")
		;
		}

		a = b;

Illustration:

```
if (thesecondstring.length() < thefirststring.length())
      {
        System.out.println("The second string is shorter");
      }
    else
      System.out.println("The second string is NOT shorter");
```

Exemple 5-3: Comparaison de la longeur de 2 strings

```
import java.io.*;
class Condition {
  /* A simple program to say what is the shortest string of two
   * -------------------------    Java 1.1  Oct 1998 */

  public static void main (String [] args) throws IOException
{
    BufferedReader in = new BufferedReader
          (new InputStreamReader(System.in));

    System.out.println("Please: input a string");
    String thefirststring = in.readLine();
    System.out.println("Please: input another string");
    String thesecondstring = in.readLine();
    if (thesecondstring.length() < thefirststring.length())
      {System.out.println("The second string is shorter"); }
    else {
        if (thesecondstring.length() == thefirststring.length())
          { System.out.println("The two strings have the same length"); }
        else {
            System.out.println("The first string is shorter");
          }
      }
  }
}
```

5.6 Lire des nombres

Exemple 5-4: Lire un nombre

http://tecfa.unige.ch/guides/java/staf2x/ex/basics/Sign.java

```
import java.io.*;

class Sign {
  /* A simple program to get the sign of a number
   * -------------------------    Java 1.1  Oct 1998 */
  public static void main (String [] args) throws IOException,
NumberFormatException
{
    BufferedReader in = new BufferedReader
          (new InputStreamReader(System.in));

    System.out.println("Please: input a number");
    Integer thenumber = new Integer(in.readLine());
    if (thenumber.intValue() < 0)
      {
        System.out.println("Negative number");
      }
    else
      {
        System.out.println("Positive number");
      }
  }
}
```

5.7 La répétition II (while)

<table>
<tr><th>Syntaxe: while</th><th>Exemples</th></tr>
<tr><td colspan="2">while (conditions) {
bloc_d'instructions
}</td></tr>
<tr><td></td><td>comme pour le "if"
(voir 5.5 “La sélection (if / else) et les conditionnels” [25]</td></tr>
<tr><td colspan="2">Voici la logique d'un while:</td></tr>
<tr><td>Initialisation_des_conditions
while (conditions) {
bloc_d'instructions
changement_des conditions
}</td><td><pre>a = 0;
while (a < 10) {
  n = n + a;
  a = a + 1;
  }</pre></td></tr>
</table>

Illustration:

```
while (CurrentNumber.length()!=0) {
 TheSum = TheSum + (new Integer(CurrentNumber)).intValue();
 CurrentNumber=in.readLine();
    }
```

Exemple 5-5: Lire une série de nombres

http://tecfa.unige.ch/guides/java/staf2x/ex/basics/Sum.java

```
import java.io.*;
class Sum {
    /* A simple program to get the sum of a list of numbers
     * -------------------------    Java 1.1  Oct 1998 */
    public static void main (String [] args) throws IOException,
NumberFormatException
    {

 BufferedReader in = new BufferedReader
     (new InputStreamReader(System.in));
 System.out.println("Please:");
 System.out.println("Input a list of numbers, pressing enter between each
number.");
 System.out.println("When you have finished, just press enter.");
 int TheSum = 0;
 String CurrentNumber = in.readLine();

 while (CurrentNumber.length()!=0)
     {
     TheSum = TheSum + (new Integer(CurrentNumber)).intValue();
     CurrentNumber=in.readLine();
     }
 System.out.println("The sum of all the numbers is " + TheSum);
    }
}
```

5.8 Les exceptions simples

Syntaxe:		Exemples

try {
bloc_d'instructions
}
catch (Signal_d'exception var) {
bloc_d'instructions
}
catch (Signal_d'exception var) {
bloc_d'instructions }
...		

Signal d'exception	Signaux définies par le système	NumberFormatException
	définis par vous-même	

Illustration:

```
try {
    Integer TheNumber = new Integer(in.readLine());
    System.out.println("The string you typed is a valid java number.");
    }
catch (NumberFormatException e)
    {
    System.out.println("The string you typed is not a valid java number.");
    }
```

Exemple 5-6: Lire un nombre et gérer les exceptions

http://tecfa.unige.ch/guides/java/staf2x/ex/basics/Catch.java

```
import java.io.*;
class Catch {
    /* A simple program to read a string and tell if it is a valid java number
     * -------------------------    Java 1.1  Oct 1998 */
    public static void main (String [] args) throws IOException,
NumberFormatException
    {
 BufferedReader in = new BufferedReader
     (new InputStreamReader(System.in));
 System.out.println("Please: Input a number.");

 try
     {
     Integer TheNumber = new Integer(in.readLine());
     System.out.println("The string you typed is a valid java number.");
     }
 catch (NumberFormatException e)
     {
     System.out.println("The string you typed is not a valid java number.");
     }

    }
}
```

5.9 Classes et méthodes (encore une fois)

- récursion vs. itération

Exemple 5-7: Les palindromes

```
import java.io.*;
class Call {
    /* A simple program to show the way java methods are invoked
     * -------------------------    Java 1.1  Oct 1998 */

    static String TheString;

    // constructor for the class
    Call () throws IOException
    {
     BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
     System.out.println("Please:");
     System.out.println("Input a string");

     TheString = in.readLine();

     System.out.println("The palindrome is:");
     System.out.println(Palindromize1(TheString) + " (using method 1)");
     System.out.println(Palindromize2(TheString) + " (using method 2)");
     System.out.println("Hopefully, we get the same result");
    }

    // Simple Main method
    public static void main (String [] args) throws IOException
    {
     Call TheCall = new Call();
    }
```

```
    String Palindromize1 (String AString)
    {

     if (AString.length() < 2)
         {
          return AString;
         }
     else
         {
// return the palindromized (2-last element) + 1st char
          return Palindromize1(AString.substring(1)).concat(AString.substring(0,1));
         }
    }


    String Palindromize2 (String AString)
    {

     String ThePalindrome = new String();

     for (int i = (AString.length() - 1); i >= 0; i--)
         {
          ThePalindrome = ThePalindrome.concat(AString.substring(i,i + 1));
         }

     return ThePalindrome;
    }

}
```

5.10 Arrays et Tables

Syntaxe: Simples tables	Exemple
int TheTable[] = new int[100];	Définit une table avec 100 éléments, chaque élément doit être un nombre entier.

Exemple 5-8: Lire des nombres et les mettre dans un tableau

http://tecfa.unige.ch/guides/java/staf2x/ex/basics/Table2.java

```
import java.io.*;

class Table2 {
    /* A VERY simple program to show how to use an array
     * -------------------------    Java 1.1  Oct 1998 */

    public static void main (String [] args) throws
                    IOException, NumberFormatException
    {
 BufferedReader in = new BufferedReader
     (new InputStreamReader(System.in));
 System.out.println("Please:");
 System.out.println("Input a list of numbers, pressing enter between each
number.");
 System.out.println("When you have finished, just press enter.");
```

```
 int TheTableLength = 0;
 int TheTable[] = new int[100];
 String CurrentNumber = in.readLine();
 while (CurrentNumber.length()!=0)
     {
     TheTableLength = TheTableLength + 1;
     TheTable[TheTableLength - 1] = (new
                     Integer(CurrentNumber)).intValue();
     CurrentNumber=in.readLine();
     }
 System.out.println("The computer has a very big short term memory (100 elements
max):");
 for (int i = 0; i < (TheTableLength); i++)
     {
     System.out.print(TheTable[i] + " ");
     }
    }
}
```