Java et son API

API = Application Programming Interface

Tout ce qui relie un programme avec son environnement + tout ce qui n'est pas défini directement dans le langage.

Développer avec Java =>

- connaître le langage
- connaître l'API

  JDK 1.0 : 117 classes et 40 interfaces

  JDK 1.1 : 272 classes, 87 interfaces

  JDK 1.2 : 1525 classes et interfaces

  (env. 10 000 méthodes)

L'API est strictement le même sur toutes les plateformes: interface graphique, accès aux fichiers et au réseau, gestion des processus, etc.

Des API de second niveau sont en cours de définition: p.ex. API pour bases de données relationnelles.

Il existe des paquets destinés à des tâches particulières, non standardisés (p.ex. Collections, Finance, Dessin 3D, etc.)

Le développeur doit constituer sa boîte à outils

Environnement Java: les différents paquets (JDK 1.0.2)

java.lang (21 classes, 2 interfaces)

java.io (23, 3)

java.util (10, 2)

java.net (11, 3)

java.awt (42, 2)

java.awt.image (9, 3)

java.awt.peer (0, 22)

java.applet (1, 3)

Total: 117 classes, 40 interfaces

Evolution de l'environnement: les différents packages (JDK 1.1)

java.applet (1, 3)
java.awt (51, 7)
+ **java.awt.datatransfer** (3, 2)
+ **java.awt.event** (19, 11)
java.awt.image (11, 3)
+ **java.beans** (15, 6)
java.io (44, 8)
java.lang (24, 2)
+ **java.lang.reflect** (5, 1)
+ **java.math** (2)
java.net (14, 3)
+ **java.rmi** (2, 3)
+ **java.rmi.dgc** (2, 1)
+ **java.rmi.registry** (1, 2)
+ **java.rmi.server** (10, 7)
java.security (12, 5)
+ **java.security.acl** (0, 5)
+ **java.security.interfaces** (0, 5)
+ **java.sql** (6, 8)
+ **java.text** (17, 1)
java.util (19, 3)
java.util.zip (14, 1)

Total: 272 classes, 87 interfaces

JDK 1.2 (Java 2)

java.applet (1, 3)
java.awt (64, 14) +
java.awt.color (5, 0)
java.awt.datatransfer (4, 3)
+ java.awt.dnd (4, 15)
java.awt.event (19, 11)
+ java.awt.font (15, 2)
+ java.awt.geom(30, 1)
+ java.awt.im (3, 1)
java.awt.image (11, 3)
+ java.awt.image.renderable
+ java.awt.print
package java.beans (15, 6)
+ java.beans.beancontext
java.io (44, 8)
java.lang (24, 2)
+ java.lang.ref
java.lang.reflect (5, 1)
java.math (2)
java.net (14, 3)
java.rmi (2, 3)
+ java.rmi.activation
java.rmi.dgc (2, 1)
java.rmi.registry (1, 2)
java.rmi.server (10, 7)
java.security (12, 5)
java.security.acl (0, 5)
+ java.security.cert
java.security.interfaces (0, 5)
+ java.security.spec
java.sql (6, 8)
java.text (17, 1)
java.util (19, 3)
+ java.util.jar
java.util.zip (14, 1)

Packages essentiels

Package	Description
java.applet	créer une applet et la faire communiquer dans son contexte d'exécution
java.awt	créer des interfaces avec l'utilisateur pour peindre et afficher des images
java.beans	pour le développement de composants logiciels JavaBeans
java.io	gestion des entrées/sorties à travers des canaux de données, sérialisation et systèmes de gestion de fichiers
java.lang	classes fondamentales pour le langage
java.math	fonctions habituelles (sin, cos, …), opérations en entier et décimal d'une précision arbitraire
java.net	gestion du réseau et communications
java.rmi	invocation à distance des méthodes (Remote Method Invocation)
java.security	sécurité
java.sql	connexion aux bases de données)
java.text	manipulation des textes, des dates, des nombres et des messages d'une manière indépendante des langues

Package	Description
java.util	structures de données des utilitaires pour la manipulation des dates, un modèle d'événement, etc.
javax.accessibility	Définit un contrat entre les composants interface-utilisateur et des technologies d'assistance à l'utilisateur (loupe, vocaliseur, …)
javax.swing	*composants légers* (entièrement écrits en Java, pour la gestion des interfaces, indépendants de la plate-forme.
org.omg.CORBA	correspondance entre l'API CORBA de l'OMG et le langage de programmation Java.

Maîtriser l'environnement

- Approche par thèmes:

- comment créer une applette ?
- comment dessiner, écrire ?
- comment animer une applette ?
- comment interagir avec la souris, le clavier ?
- comment gérer une fenêtre et son contenu ?
- comment gérer des fichiers ?
- comment communiquer sur un réseau ?

- Approche par concepts

- quels sont les concepts sous-jacents ?
- quelles sont leurs relations
- comment interagissent-ils

- Outils de butinage

- tous les paquets sont décrits sous forme hypertexte
- documentation interactive

Classe java.awt.Rectangle

```
public  class  java.awt.Rectangle
    extends  java.lang.Object
{
        // Fields
    public int height;
    public int width;
    public int x;
    public int y;
```

On considère des rectangles dont les côtés sont // aux axes x et y et dont les coordonnées et dimensions sont entières

```
        // Constructors
    public Rectangle();
    public Rectangle(Dimension  d);
    public Rectangle(int  width, int  height);
    public Rectangle(int  x, int  y, int  width,
int height);
    public Rectangle(Point  p);
    public Rectangle(Point  p, Dimension  d);
```

La définition d'un rectangle peut utiliser les notions de Point et de Dimension. On trouvera dans la documentation l'effet exact de chaque constructeur

Rectangle (suite)

```
        // Methods
```

On peut changer la taille d'une rectangle de différentes manières

```
    public void resize(int  width, int  height);
    public void add(int  newx, int  newy);
    public void add(Point  pt);
    public void add(Rectangle  r);
    public void grow(int  h, int  v);
```

… le déplacer

```
    public void move(int  x, int  y);
    public void translate(int  dx, int  dy);
```

… changer sa forme = déplacer et changer sa taille

```
    public void reshape(int  x, int  y, int  width,
int height);
```

Il y a une notion d'"être à l'intérieur"

```
    public boolean inside(int  x, int  y);
```

Il y a des notions de comparaison, d'intersection et d'union de 2 rectangles

```
    public boolean equals(Object  obj);
    public Rectangle intersection(Rectangle  r);
    public boolean intersects(Rectangle  r);
    public Rectangle union(Rectangle  r);
```

Un rect. peut être vide (?)

```
    public boolean isEmpty();
```

et comme tout objet:

```
    public int hashCode();
    public String toString();
```

Classe Rectangle - concepts

Ce qu'il faut retenir:
On considère des rectangles dont les côtés sont // aux axes x et y et dont les coordonnées et dimensions sont entières

La définition d'un rectangle peut utiliser les notions de Point et de Dimension.

On peut changer la taille d'une rectangle de différentes manières, on peut aussi le déplacer.

Un point peut être "à l'intérieur" d'un rectangle

On peut comparer deux rectangles, tester s'ils s'intersectent, calculer leur intersection et leur union

Un rectangle peut être vide

Ce qu'il ne faut pas retenir:
Les noms et paramètres de chaque méthode, on les retrouvera toujours dans la documentation interactive.