TD/TP AOO

Vania.Marangozova@imag.fr
Vincent.Danjean@imag.fr

Plan des TDs/TPs

- 10 séances, partagées entre TD et TP

Séance	Type	Sujet
1	TD	Introduction Java
2	TP	Environnement de travail Java
3-4-5-6-7	TD	Lecture,utilisation, écriture de classes élémentaires Java. Hiérarchie de classes.
8-9-10	TP	Utilisation de classes prédéfinies de Java

Introduction à Java

Présentation basée sur présentations de

Ph.Genoud, Ch. Bruley, S. Bouchenak

Origines de Java

- 1990
  - World Wide Web inexistant
  - Début de l'ère de développement frénétique des PCs, popularisation, vulgarisation
  - Projet Oak de SUN Microsystems : langage pour la communication des appareils électroniques de poche et domotique
- 1993
  - mars : le NCSA lance MOSAIC, le premier navigateur internet (protocole http, langage html), le web décolle...
  - été : Oak change d'orientation et s'adapte à la technologie internet
- 1995
  - mai 1995 : annonce officielle de la naissance de la technologie Java (issue de Oak)

La technologie Java

"The network is the computer"

- Un langage de programmation orienté objet
- Une machine virtuelle (JVM)
- Des bibliothèques de classes standards (JAVA API)
  - plus de 1500 classes dans java 1.3, 1.6 est sur le point de sortir
- Ensemble d'outils pour la compilation, l'exécution, le déboguage, la documentation, l'archivage…

Popularité de Java?

- Exécution dans les navigateurs Web
  - Applets : programmes Java qui sont téléchargés et exécutés dans le navigateur de l'utilisateur
- "Write Once Run Everywhere"
  - Les programmes Java n'ont pas besoin d'être recompilés pour être exécutés sur d'autres machines

Popularité de Java : oui mais…

- Les applets
  - A l'origine de nombreux problèmes de sécurité et de nombreux travaux de recherche sur la sécurisation de l'exécution
- Exécution partout… où il y a la machine virtuelle Java
  - La machine virtuelle Java (JVM) doit être installée sur la machine d'exécution
  - La JVM a une implémentation différente selon les architectures
  - La taille de la JVM a évolué de 500K à plusieurs G

Popularité de Java : le marché a été conquis

- Effet de mode et propagation de l'application de Java
  - Java marche dans un environnement distribué
  - La plupart des entreprises aujourd'hui ont besoin d'infrastructures distribuées
  - Java est à la base de nombreuses technologies utilisées :
    - J2EE : serveur d'applications, connexion à des bases de données
    - JMS : service de communication par messages
    - …
  - Lors de la conception d'un nouveau produit il est naturellement écrit en Java pour faciliter l'intégration dans une entreprise qui utilise déjà Java
    - Retour vers un monde homogène

La technologie Java

- Un langage de programmation orienté objet
- Une machine virtuelle (JVM)
- Des bibliothèques de classes standards (JAVA API)
  - plus de 1500 classes dans java 1.3, 1.6 est sur le point de sortir
- Ensemble d'outils pour la compilation, l'exécution, le déboguage, la documentation, l'archivage…

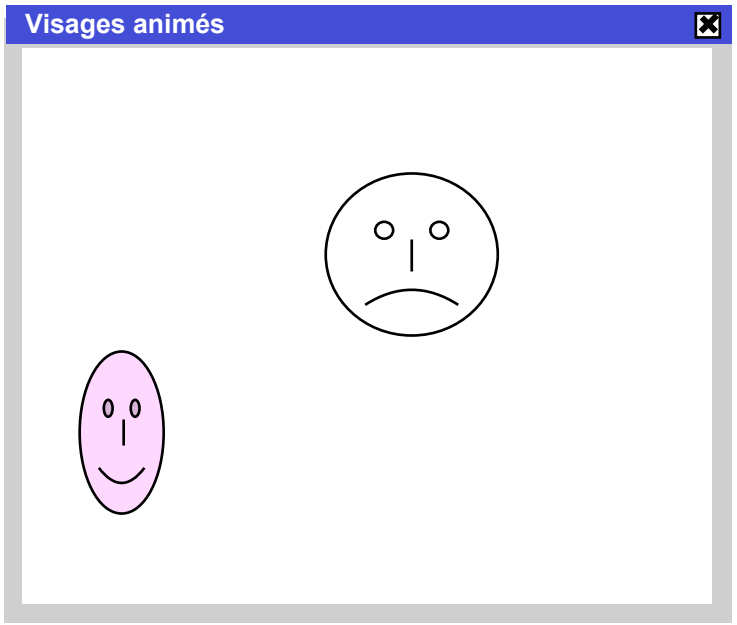
Java : un langage orienté-objet

- Qu'est-ce qu'un objet ?

  - Idée : le monde réel ne marche pas à base de paramètres, procédures et paquetages… Pourquoi ne pas modéliser les phénomènes directement (au plus proche de ce qu'ils sont) ?
  - **Un objet modélise une entité**
    - concrète
      ex: ville, service hospitalier, scanner, véhicule, étudiant
    - abstraite
      ex: réunion, planning, date, commande
  - Un objet est caractérisé par
    - des actions qu'il peut effectuer
    - des propriétés

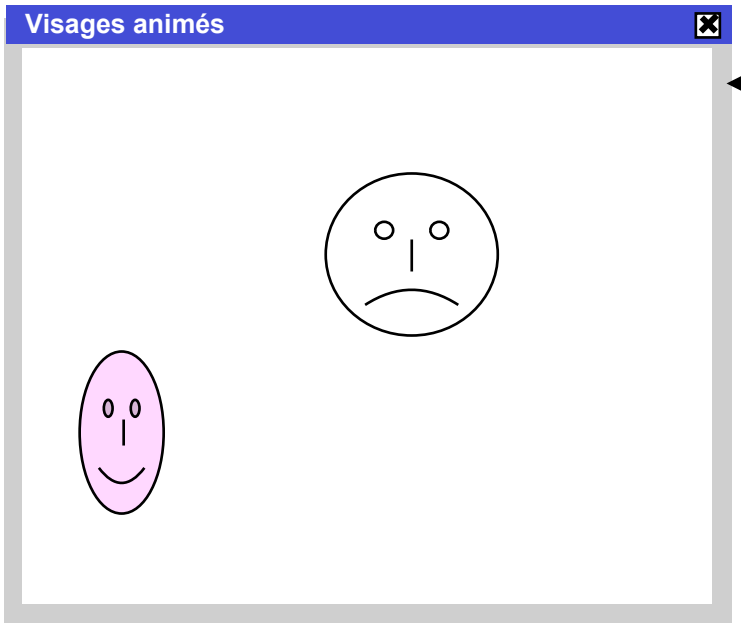
Les Objets

- Exemple : les visages animés
  - Quelles sont les entités composant le jeu?
  - Que doivent savoir faire ces entités ?
  - Quelles sont leurs propriétés ?



Les Objets : Compétences

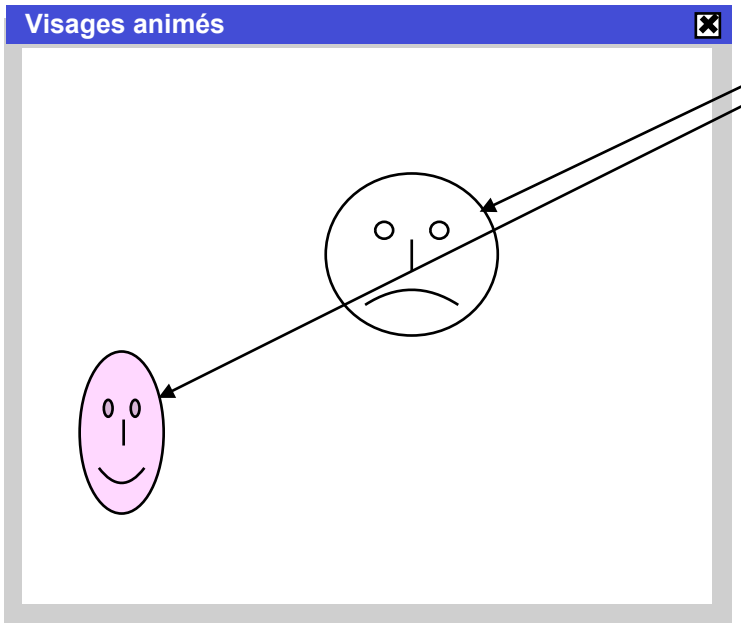
- Exemple : les visages animés
  - Entités, actions, caractéristiques?



Fenêtre
- actions
  - se fermer
  - s'iconifier
  - passer au premier plan
  - ...
- caractéristiques
  - position
  - largeur, hauteur
  - ...

Les Objets : Compétences

- Exemple : les visages animés
  - Entités, actions, caractéristiques?



Visage
- actions
  - sourire
  - faire la tête
  - s'afficher
- caractéristiques
  - couleur
  - position
  - largeur, hauteur
  - ...

Les objets Java

- Une entité est définie par les actions qu'elle peut faire et par ses caractéristiques

- Un objet est défini par ses méthodes et ses attributs

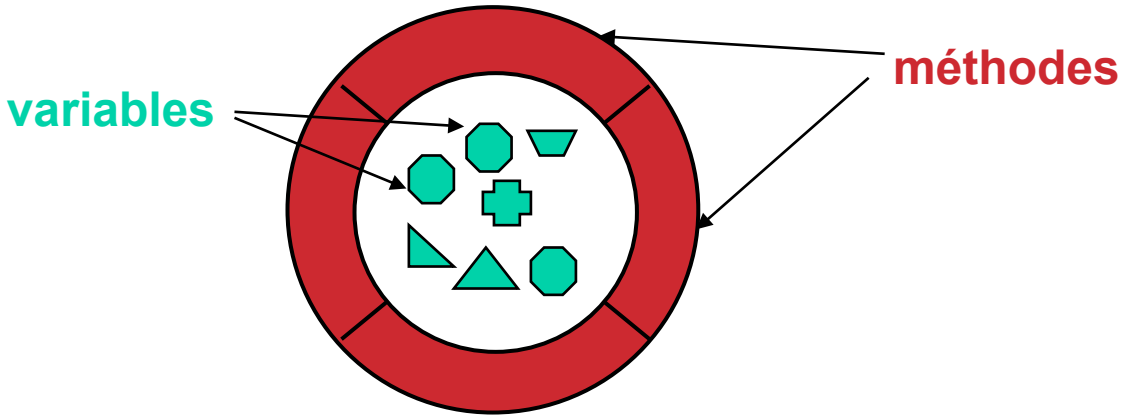
- Attributs
  - les particularités d'un objet à un instant donné
    - ex: patient mesure 1,82 m et pèse 75 Kg
    - Ex: la commande est validée
  - les attributs sont typés et nommés
    - ex: float hauteur; float poids;
  - l'ensemble des attributs = état de l'objet à un instant donné

- Méthodes
  - définissent le comportement de l'objet (ce qu'il peut faire, comment il peut le faire…) et ses réactions aux stimulations externes
    - ex: un étudiant passe un examen, etc…

- Identité de l'objet = permet de distinguer un objet d'un autre, de manière unique

Les objets

- Un objet est le regroupement de données (variables ou attributs) et des traitements (méthodes) associées
- Les données de l'objet (attributs) sont encapsulées par des opérations (méthodes) qui peuvent accéder et modifier ces données
- Les méthodes constituent donc l'interface de l'objet



objet *visage1*

Exemple

```
public class PointCartesien {
   double x,y;

   double abscisse() { return x;}
   double ordonnee() { return y;}

   …

   void positionner(double x, double y) {
       this.x=x; this.y=y;
   }
   void translation(Vecteur v) {
       positionner(x+v.dx(),y+v.dy());
   }
}
```

Les objets et les classes

- Les objets sont des entités d'exécution
- Ils sont crées selon un "moule" qui définit quels attributs et quelles méthodes les objets vont avoir
- Le "moule" pour un objet est une classe

- variable - type
- objet - classe

- La création d'un objet = l'instanciation d'une classe
- L'objet = une instance d'une classe

Exemple

- Instanciation

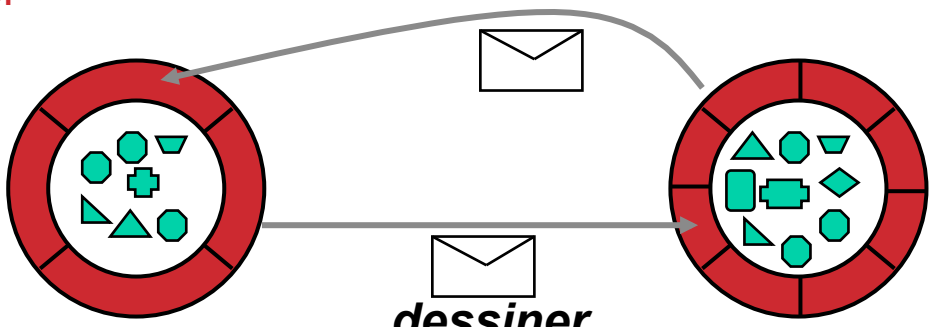
```
PointCartesien p1 = new PountCartesien(10,20);
```

- Constructeurs

```
public class PointCartesien {
    …
    //méthode spéciale : constructeur de la classe
    PointCartesien(double x, double y){
            positionner(x,y);
    }
    …
}
```

Communication entre objets

- Dans un langage procédural
  - Appels de fonctions organisés en modules
  - Points de lancement : ex. fonction principale (main)
- Dans un langage OO
  - Interactions entre objets en utilisant les appels de méthodes
  - Points de lancement : des objets avec des méthodes qui vont être exécutées
- Un appel de méthode
  - objet cible
  - nom de la méthode à déclencher
  - paramètres de cette méthode



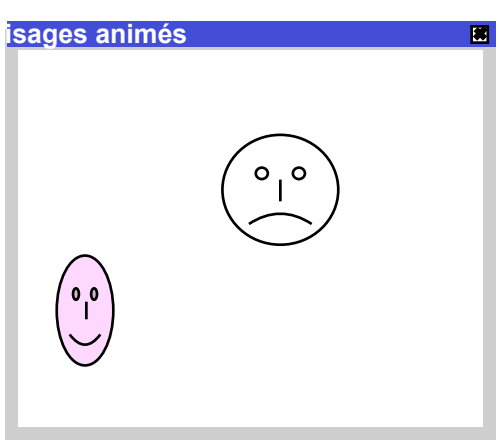
Exemple

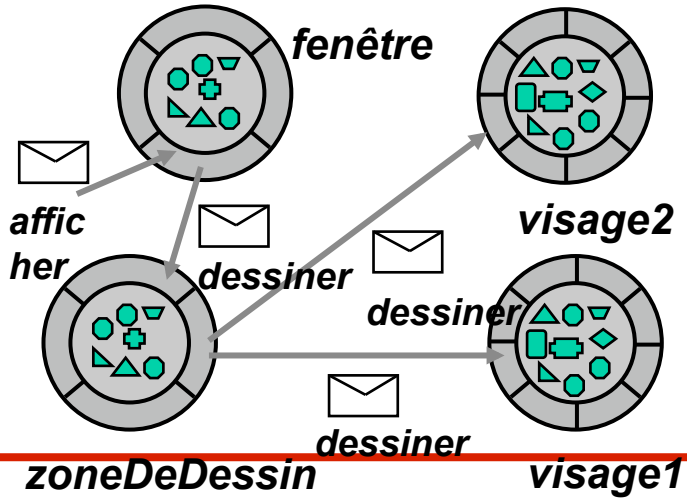
```
class Carré {
   PointCartesien p1,p2,p3,p4;

   Carré(x1:Float, y1:Float, ...) {
       p1.positionner(x1,y1);
   }
}
```

Une application OO

- Une application OO
  - Composée d'un ensemble d'objets
  - Ces objets doivent être crées et chargés en mémoire
  - Le lancement de l'application se fait par appel de méthodes sur ces objets
    - L'exécution des méthodes appelées peut provoquer l'appel d'autres méthodes à d'autres objets ou la création de nouveaux objets



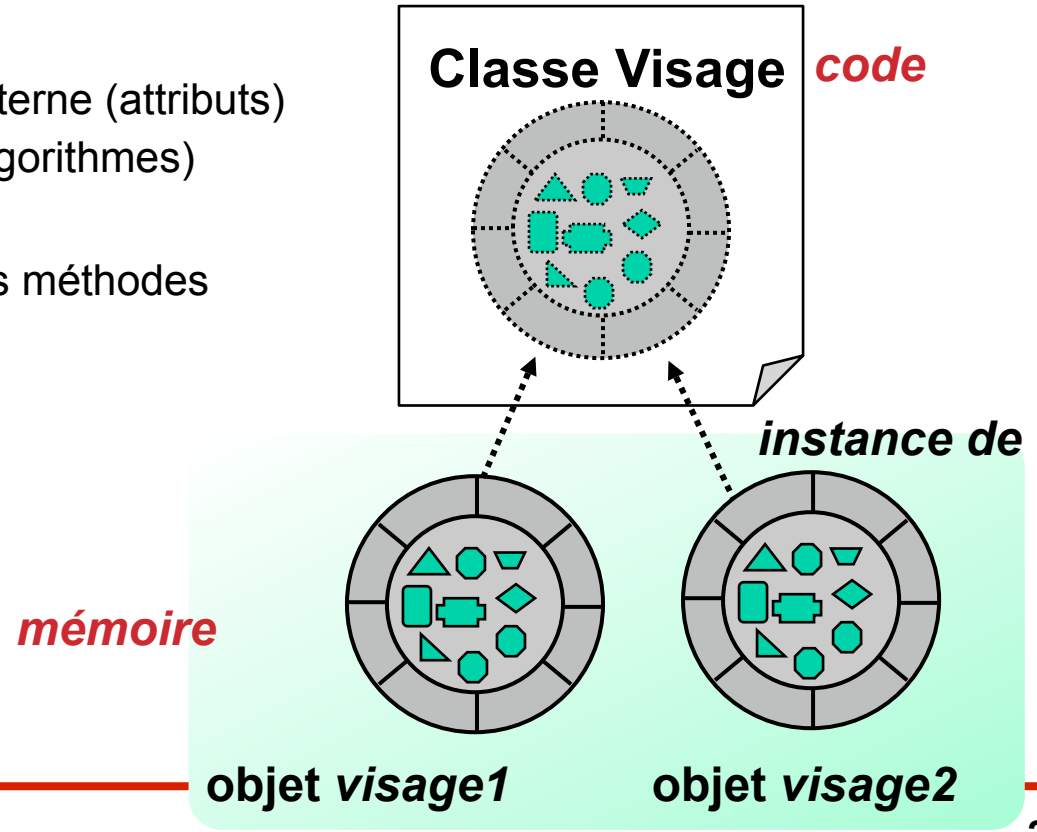


Exemple de programme Java

```
class DemoVisagesAnimes  {
    public static void main(String[] argv)  {
      Frame fenêtre = new Frame("Titre de la fenêtre");
      Dessin dessin = new Dessin();
      fenêtre.add(dessin);
      fenêtre.show();
      Visage v1 = new Visage();
      Visage v2 = new Visage(200,100,200,200,5);
      dessin.ajouterObjet(v1);
      dessin.ajouterObjet(v2);
      v2.pleurer();
      dessin.repaint(10);
    }
  }
```

Les classes Java plus en détail

- Classe
  - Schéma pour des objets
  - Les objets (instances) sont créés (instanciés) à partir des classes
- Structure d'une classe
  - partie privée
    - structure de données interne (attributs)
    - corps des méthodes (algorithmes)
  - partie publique (interface)
    - noms et paramètres des méthodes



Héritage en Java

- Une classe peut hériter d'une classe existante
  - Elle raffine/spécialise cette classe
  - Elle hérite des attributs et méthodes de ses ancêtres/super-classes
  - Elle peut ajouter de nouveaux attributs et/ou de nouvelles méthodes
  - Elle peut modifier ou redéfinir les méthodes héritées

Héritage : exemple

```
class Visage {
   Cercle tête, oG, oD; //centre et diamètre
   Arc bouche; //points, courbure
   Visage() {…}
   dessiner() {…}
}
```

```
class VisageCouleur extends Visage {
        //mêmes attributs tête, Og,Od, bouche
        Couleur c; //nouvel attribut
        VisageCouleur(…, c : Couleur) {
                super(…);
                this.c = c;
        }
        dessiner() { //étendue
                super();
                colorierTête();
        }
        changerCouleur(…) {…} //nouvelle
}
```

Interfaces Java

- Définissent un ensemble de méthodes sans implémentation
  - Les méthodes que l'on pourra appeler
  - Séparation entre fonction et implémentation
  - Les classes implémentent les interfaces i.e fournissent le code correspondant aux méthodes déclarées dans les interfaces

```
interface VisageItf() {
   sourire();
   froncerSourcis();
   tirerLangue();
}
```

```
class Visage() implements VisageItf {
   … //attributs, autres méthodes
   sourire() { …//IMPLEMENTATION}
   froncerSourcis() {… //IMPLEMENTATION}
   tirerLangue() {… //IMPLEMENTATION}
}
```

  - Deux classes différentes peuvent implémenter la même interface de manières différentes : ex dessiner pour un carré et un cercle

Java : un langage

- Offre toutes la fonctionnalité d'un langage de programmation puissant
- Syntaxe familière (proche du C ou du C++)
- Typage fort
  - Beaucoup de vérifications sont faites à la compilation
  - "Un programme Java qui compile est un programme qui est susceptible de fonctionner … comme on veut est une toute autre question"
- Pas de manipulation de pointeurs
  - Les pointeurs sont en effet cachés
- Mémoire
  - Gestion automatique : "garbage collector"
  - malloc() et free() n'existent plus
- Java peut être utilisé pour le prototypage
  - De nombreux environnements offrent des exécuteurs symboliques

Java : un langage

- Pour bien utiliser un langage, il faut maîtriser les paradigmes sous-jacents qu'il met en œuvre.

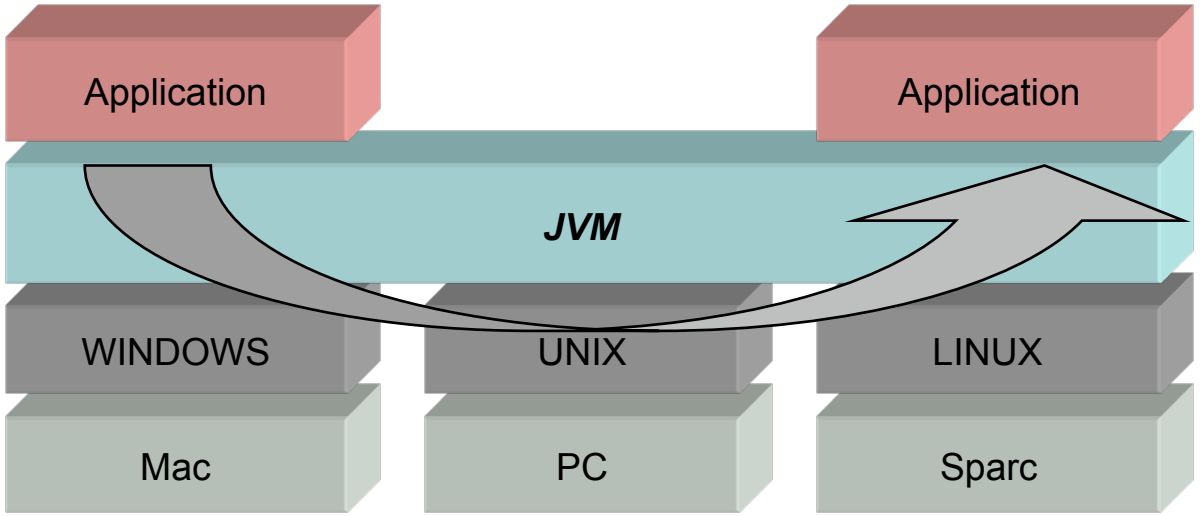
- Maîtriser un langage c'est :
  - Connaître la syntaxe
  - Connaître la sémantique (paradigmes)
  - Connaître le modèle d'exécution
    - Dans le cas de Java, il faut comprendre le fonctionnement de la JVM et savoir ainsi exactement comment un programme va s'exécuter.

- Pour utiliser efficacement un langage, des bibliothèques préexistantes sont indispensables :
  - Apprendre à se servir des bibliothèques par cœur est inutile dans le cadre scolaire,
  - Il faut apprendre à trouver ce dont on a besoin
    - Javadoc

La technologie Java

- Un langage de programmation orienté objet
- Une machine virtuelle (JVM)
- Des bibliothèques de classes standards (JAVA API)
  - plus de 1500 classes dans java 1.3, 1.6 est sur le point de sortir
- Ensemble d'outils pour la compilation, l'exécution, le déboguage, la documentation, l'archivage…

JVM



- Abstraction d'un environnement homogène

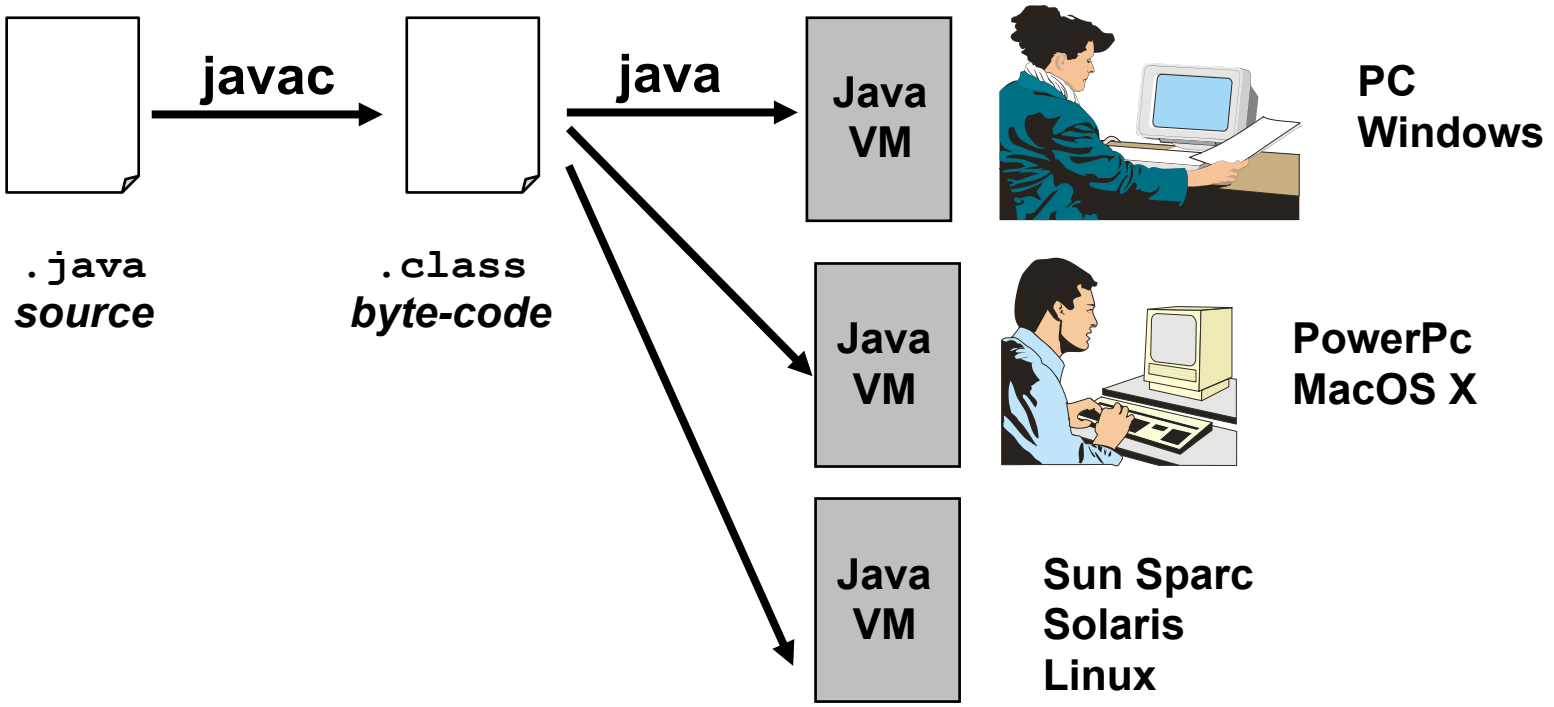
Spécification de la JVM : Objectif

- Définition de l'abstraction d'une machine d'exécution :
  - Effectuer des calculs
    - Ensemble d'instructions
  - Contrôler l'accès à la mémoire
    - Mécanismes de protection
  - Gérer le contrôle des flots d'exécutions
    - Processus & threads
  - Accéder aux dispositifs sous-jacents
    - Abstraction du matériel

La machine virtuelle Java

Exécution d'un programme Java compilé

- byte-code assure la portabilité des programmes Java
  - langage d'une Machine Virtuelle
  - à l'exécution un interpréteur simule cette machine virtuelle



Exécution d'un programme Java

- Java est un langage compilé et puis interprété
  - Java est compilé vers le langage machine de la JVM : le byte-code
    - Que l'on compile sous Linux ou Windows, c'est le même byte-code
  - Le byte-code est interprété par la JVM
  - La JVM est un processus nécessitant un support
    - Par exemple une machine physique

```
// exemple pour désassambleur
public class Test {

   public static void main(String[] args)
   {
      for (int i = 0; i < 10; i++)
         System.out.println("Hello " + i);

   }

}
```

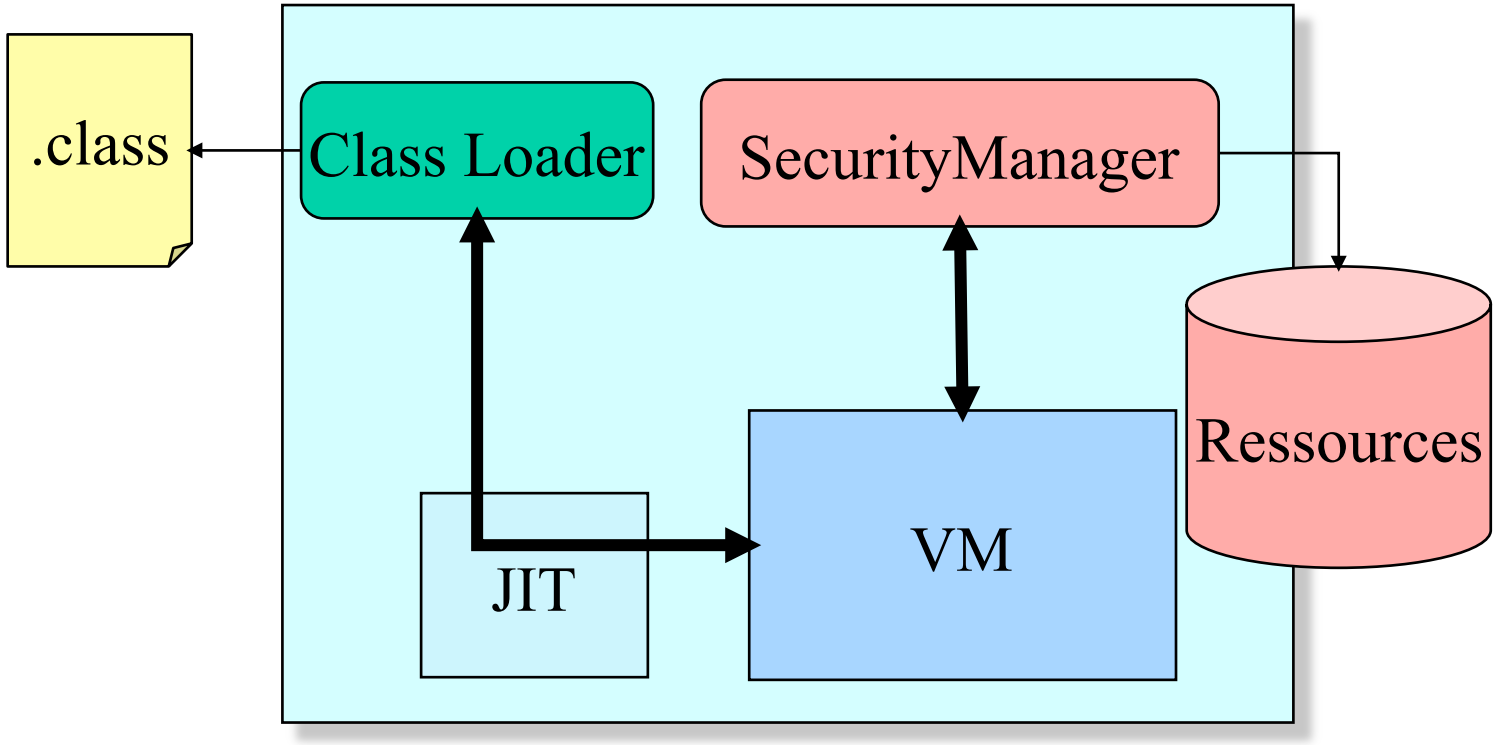
code source : `Test.java`

javac

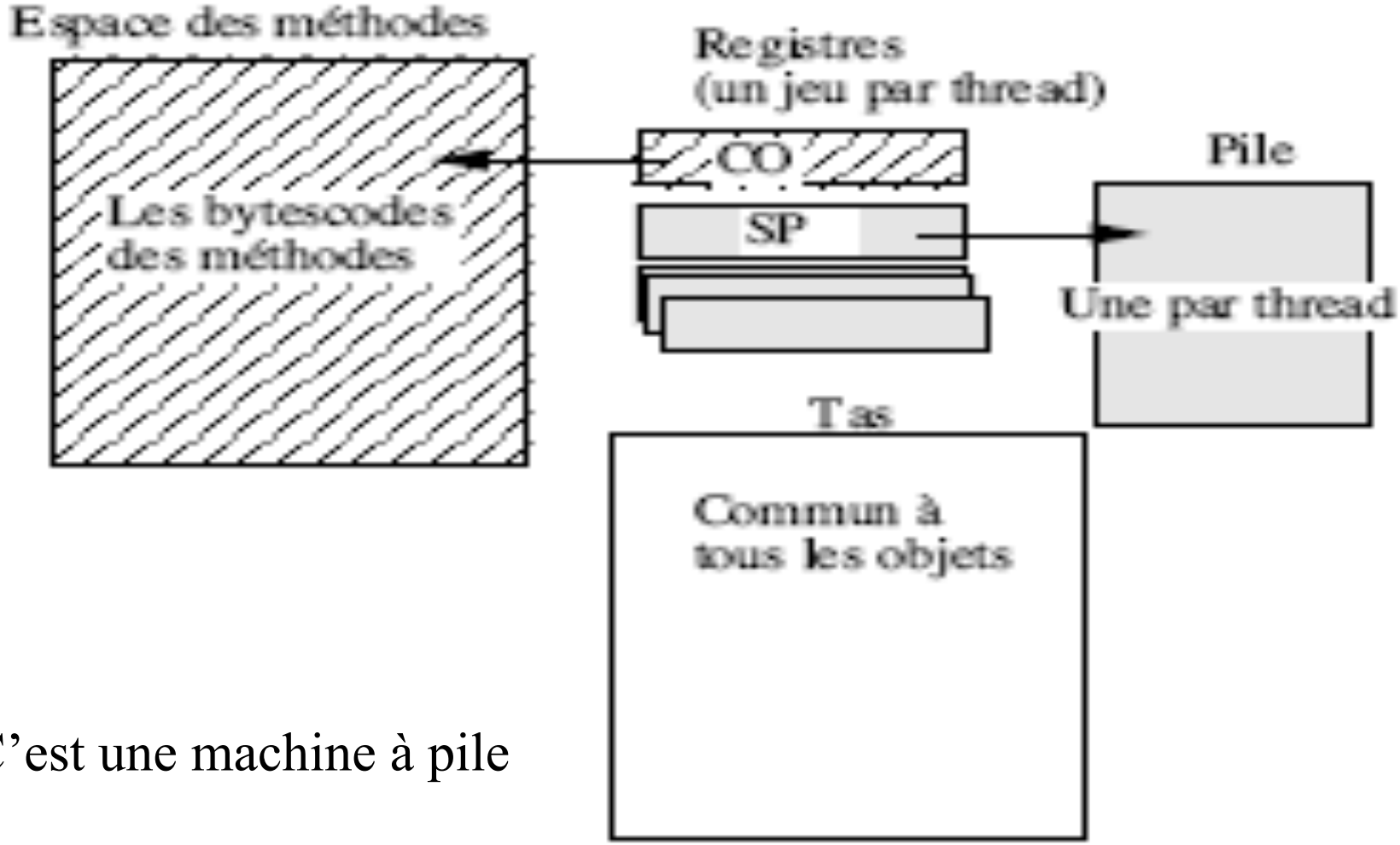
```
0 iconst_0
1 istore_1
2 goto 30
5 getstatic #10 <Field java.io.PrintStream out>
8 new #5 <Class java.lang.StringBuffer>
11 dup
12 ldc #1 <String "Hello ">
….
27 iinc 1 1
30 iload_1
31 bipush 10
33 if_icmplt 5
36 return
```

byte-code : `Test.class`

Structure de la JVM

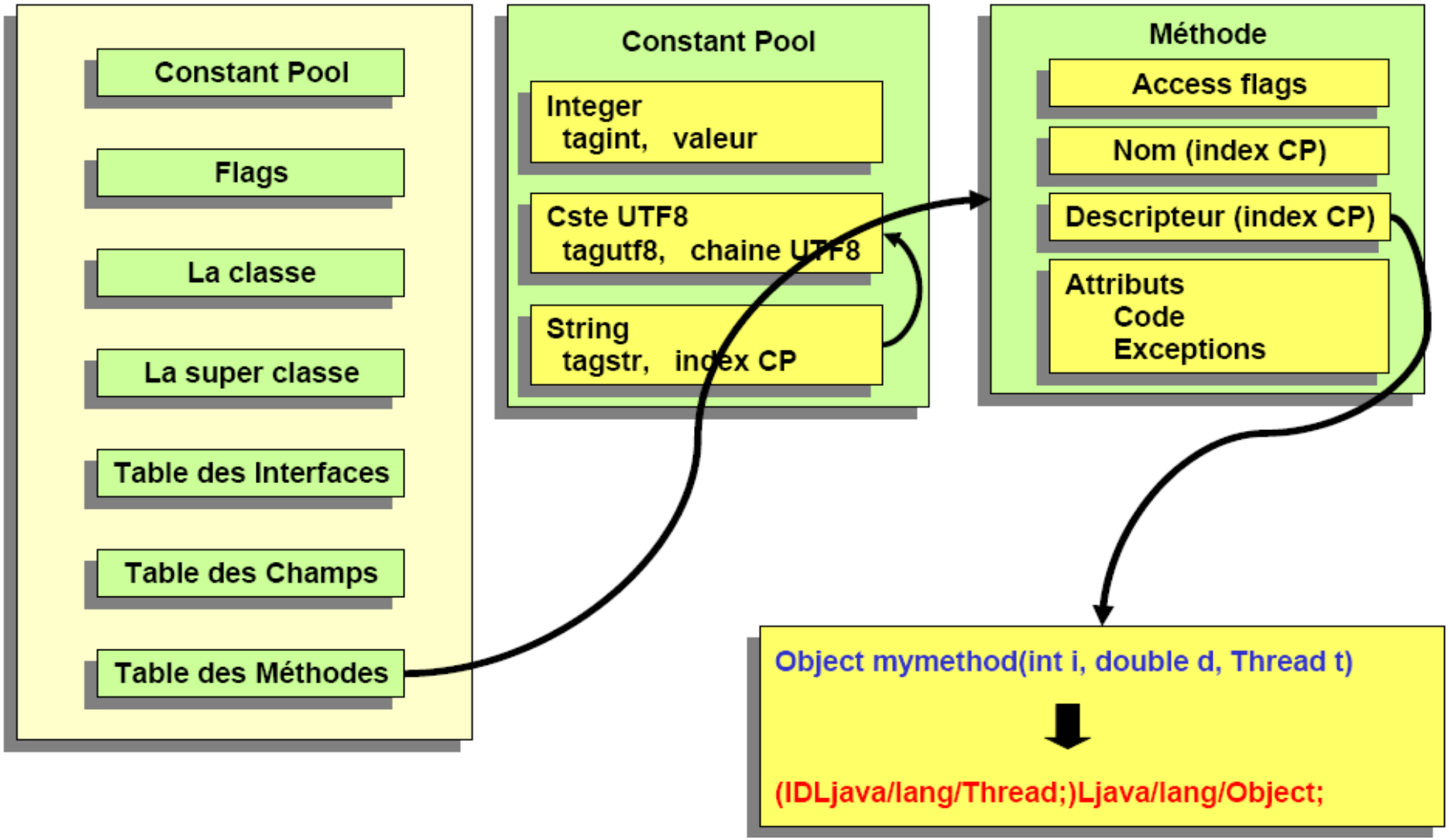


Organisation mémoire de la VM

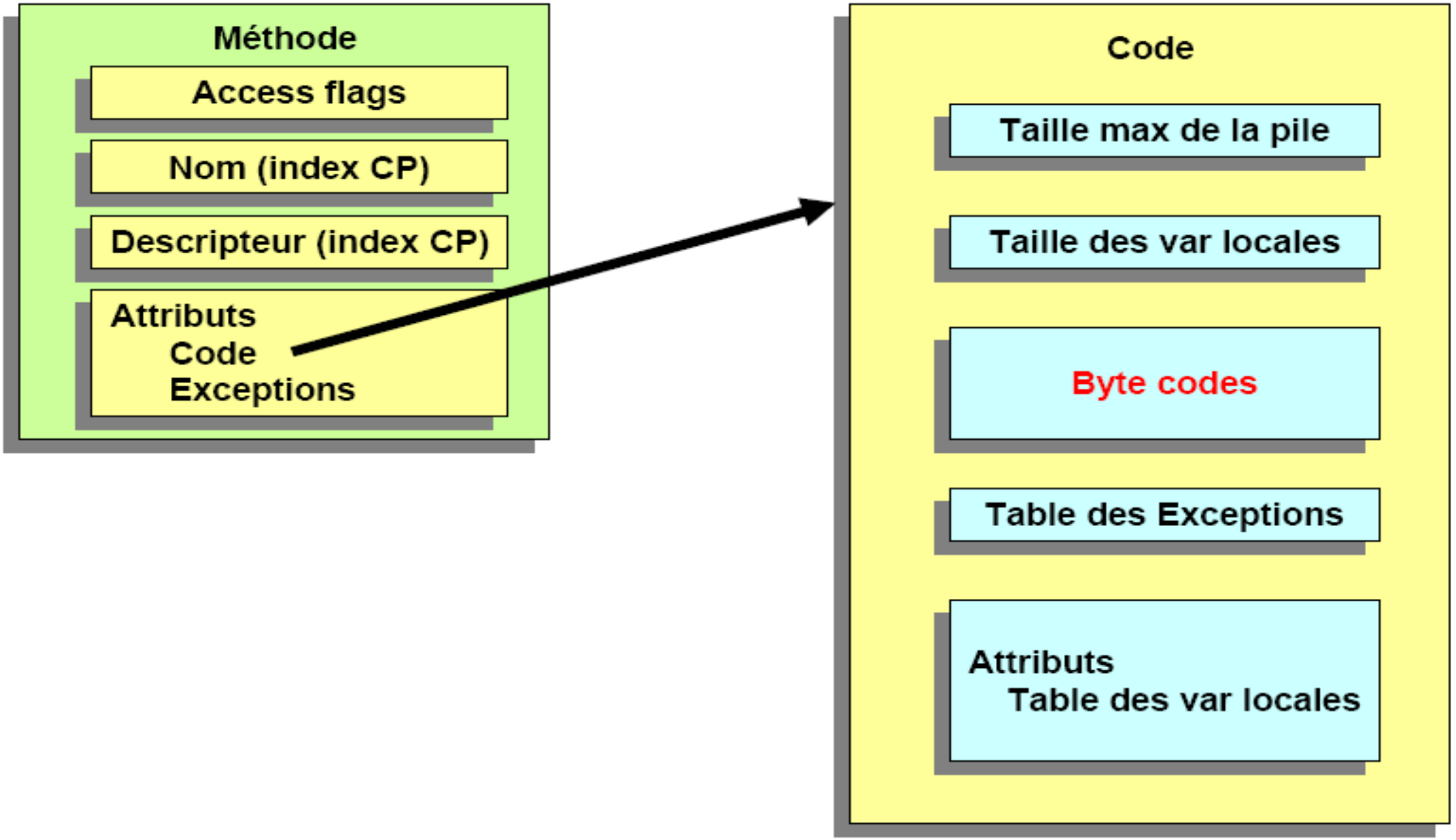


C'est une machine à pile

Structure du Bytecode



Strcuture du Bytecode

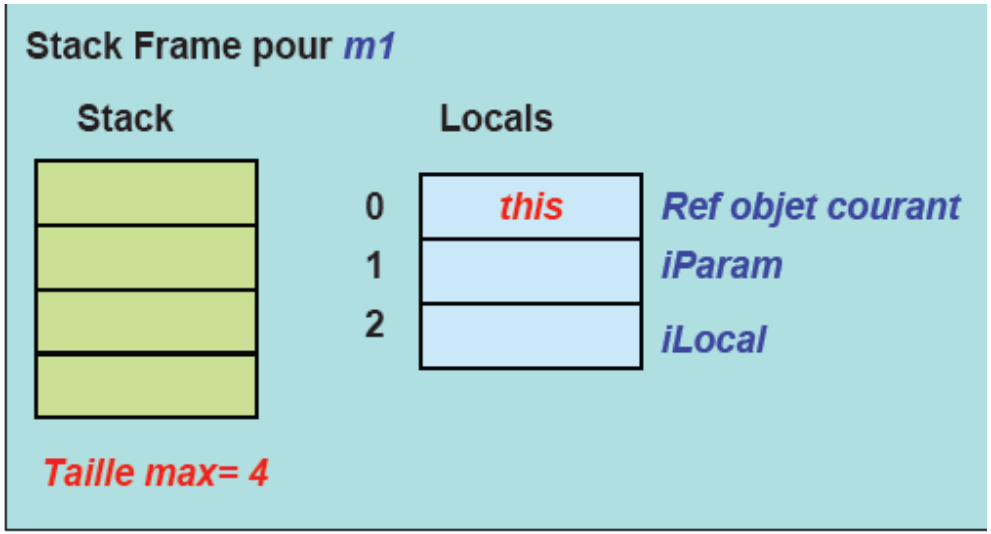


Fonctionnement de la JVM

```
void m1 ( int iParam )
{
    int iLocal = 30;

    iLocal  =  iLocal + (500000 + (4 * iParam));
       ……..
}
```

Toute méthode java s'éxécute au sein d'une mémoire privée: la StackFrame.
La taille de cette mémoire est connue dès la compilation



Bytecodes pour L'expression

```
iload_2
ldc #4          <Integer 500000>
iconst_4
iload_1
imul
iadd
iadd
istore_2
```

3ème variable locale
index 4 Constant pool

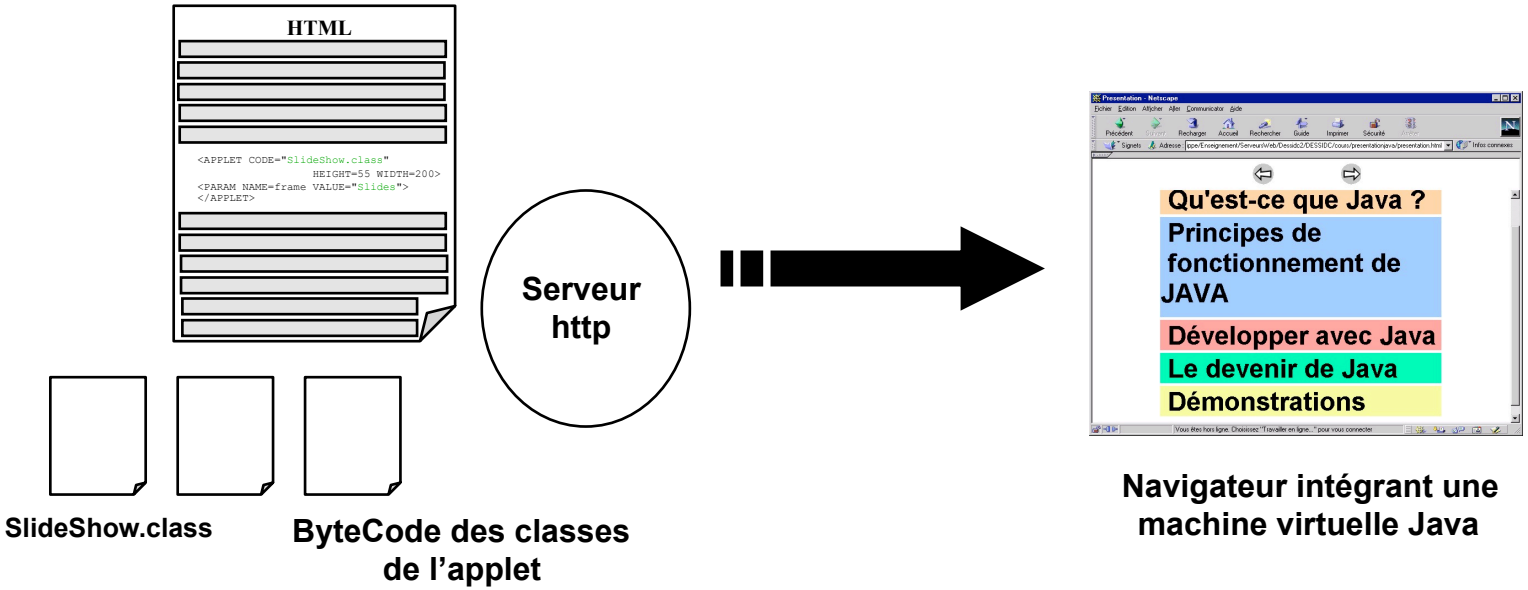
2ème variable locale (paramètre)

3ème variable locale

Principe de fonctionnement d'une applet

- Exécution d'une applet dans un navigateur Web
  - Référence au code de l'applet dans le document HTML :

```
<APPLET CODE="SlideShow.class" HEIGHT=55 WIDTH=200>
<PARAM NAME=frame VALUE="Slides">
</APPLET>
```



La technologie Java

- Un langage de programmation orienté objet
- Une machine virtuelle (JVM)
- Des bibliothèques de classes standards (JAVA API)
  - plus de 1500 classes dans java 1.3, 1.6 est sur le point de sortir
- Ensemble d'outils pour la compilation, l'exécution, le déboguage, la documentation, l'archivage…

Les bibliothèques Java

- Organisation en paquetages (JAVA API)

java.lang	classes essentielles	objets, types
java.util	données complexes	arbres, tables de hashage, tableaux, listes
java.io	entrées/sorties	standards, fichiers
java.net	réseau	sockets, web
java.rmi	distribution	appel de méthode à distance
java.sql	bases de données	
java.awt	graphiques	
java.java3D	graphiques	
…	JavaCard	
…	tél portables	

La technologie Java

- Un langage de programmation orienté objet
- Une machine virtuelle (JVM)
- Des bibliothèques de classes standards (JAVA API)
  - plus de 1500 classes dans java 1.3, 1.6 est sur le point de sortir
- Ensemble d'outils pour la compilation, l'exécution, le déboguage, la documentation, l'archivage…

Environnement de développement

- JDK Java Developer's Kit : environnement standard (Windows ou Solaris) diffusé gratuitement par Sun (http://java.sun.com)
  - des outils de développement
    - compilateur , debugger, « interpréteur » de Byte-Code
  - les "librairies" de classes standards
  - documentation
- différentes évolutions du JDK correspondant à différentes évolutions de la plate-forme Java

Les différentes éditions de Java

- 3 éditions de Java

Standard Edition J2SE

Fourni les compilateurs, outils, runtimes, et APIs pour écrire, déployer, et exécuter des applets et applications dans la langage de programmation Java

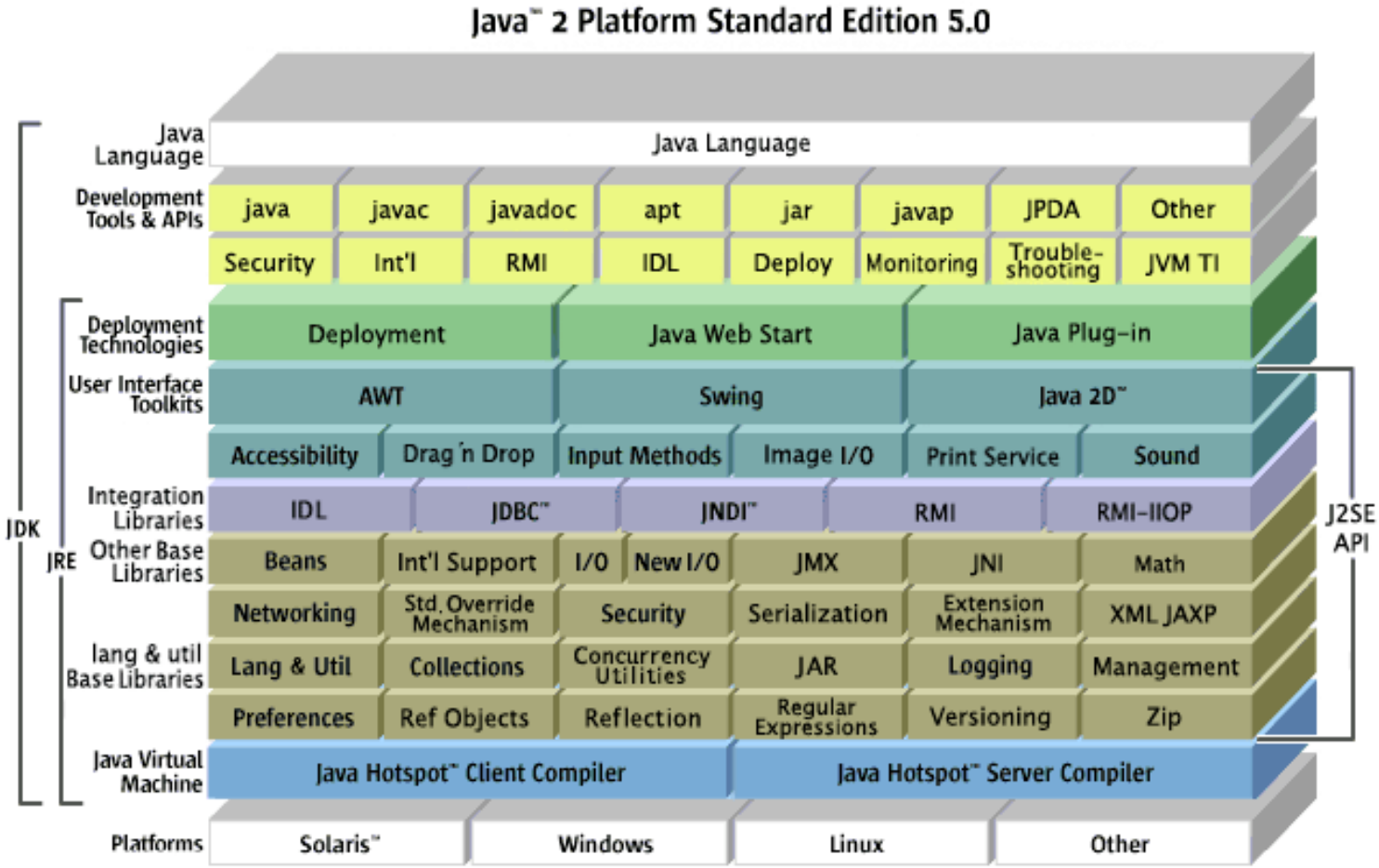
Entreprise Edition J2EE

Destinée au développement d'applications « d'entreprise » *(«business applications»)* robustes et interopérables. Simplifier le développement et le déploiement d'applications distribuées et articulées autours du web.

Mobile Edition J2ME

Environnement d'exécution optimisé pour les dispositifs « légers » :
- **Carte à puce (smart cards)**
- **Téléphones mobiles**
- **Assistants personnels (PDA)**

J2SE 5.0 (SDK 1.5)



Compiler et exécuter un programme Java (1/2)

- La classe principale : contient la méthode main, dans le fichier HelloWorld.java

```
public class HelloWorld {
   public static void main(String[] args) {
        System.out.println("Hello World!");
   }
}
```

- Compiler

```
vania@seneca:~/Code/test$ javac HelloWorld.java
vania@seneca:~/Code/test$ ll
total 16
-rw-r--r--   1 vania  vania  426 Sep 17 10:38 HelloWorld.class
-rw-r--r--   1 vania  vania  112 Sep 17 10:38 HelloWorld.java
vania@seneca:~/Code/test$
```

- Exécuter

```
java HelloWorld
```

Compiler et exécuter un programme Java (1/2)

- Compiler plusieurs classes

```
javac HelloWorld.java A.java B.java
```

- Compiler et mettre les classes dans un répertoire particulier
  - Par défaut, le répertoire destination est le répertoire courant

```
javac -d ./build HelloWorld.java A.java B.java
```

- Trouver les classes

```
vania@seneca:~/Code/test$ java HelloWorld
Exception in thread "main" java.lang.NoClassDefFoundError: HelloWorld
vania@seneca:~/Code/test$ java -classpath ./build HelloWorld
Hello World!
```

Pour conclure rapidement…

- Java est une technologie
  - langage, machine virtuelle, bibliothèques, outils
  - imposée sur le marché
  - on travaille toujours sur l'amélioration des performances
  - de plus en plus de nouvelles bibliothèques et classes
  - on n'est pas prêt à changer pour l'instant, à voir ce que donnera .NET
- Références
  - http://java.sun.com
  - « JAVA in a nutshell, 3nd Edition », David Flanagan - O'Reilly 1999
  - « Thinking in Java », Bruce Eckel - Prentice-Hall 1998 (www.BruceEckel.com)
  - …