Java et OOP

Cours no. 1

Qu'est-ce que Java ?

- Programmation orientée objet.
- Programmation structurée.
- Programmation distribuée.
- Programmation Web.
- Portabilité.
- Programmes compilés **et** interpretés.
- Simple ! (gestion de la mémoire)

Java basics

- Syntaxe dérivée du C :
  - affectations : `x=y=z;`
  - declarations : `int x; char z[];` mais aussi `char[] z;`
  - structures de contrôle : `while` `do-while` `if-else` `case-switch`
  - opérateurs arithmétiques, logiques, relationnels
  - `main`, arguments en ligne de commande
  - Autres (identifiez-les dans ce cours !)

Différences avec C

- Pas de directive de compilation.
- Pas de possibilité de travailler directement avec les adresses en mémoire.
- Pas nécéssaire de désallouer l'espace mémoire.
- Pas de `goto` (étiquettes de boucles)
- `struct` est remplacé et étendu par la notion de classe.
- Pas d'édition des liens.

Quelques conventions de nommage

- Les variables commencent par minuscules.
- Les noms de classes commencent par majuscules.
- Les constantes sont en majuscules, séparées par _

Types primitifs

- `boolean` – type non entier !
  - `true, false`, valeur implicite = `false`.
- `char` sur 2 octets – Unicode, 65536 caractères:

  Caractères entre 32 et 128 – les mêmes que dans ASCII.
- types entiers : `byte, short, int, long`, sur 1/2/4/8 octets.

  Conversions : implicite en ordre croissant de la taille,

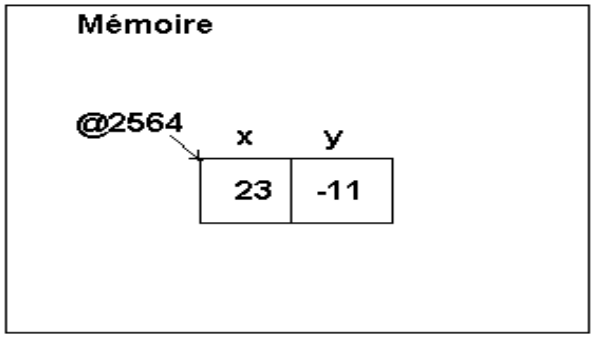
  explicite en ordre decroissant (perte d'information):

```
int x; long y = 10000000; x = (int)y;
```

- types réels : `float, double` sur 4, resp. 8 octets.
- même "polymorphisme" de la division :
  - si les 2 paramètres entiers, alors division entière
  - sinon, division réele.

Valeurs & références

- Une variable est caractérisée par
  - sa valeur et
  - sa place en mémoire.



- Manipulation : affectation, opérations, passage en paramètre.
  - Manipulation par valeur : copie.
  - Manipulation par référence : partage de la place en mémoire.

Tableaux

- Suite de composants (objets) du même type.
- Manipulé par référence.
- indexés à partir de 0.
- Déclaration (sans dimension) : `int[] tableau;`
- Définition avec dimension : `tableau = new int[2];`
- Déclaration et définition simultanées :

  `char[] s = new char[2];`
- Autre type de déclaration et définition simultanées :

  `boolean[] b = {false, true, false};`
- Longueur : `b.length == 3`.
- Exception d'indices hors des limites :

```
try { tableau[tableau.length] = 1; }
catch (ArrayIndexOutOfBoundsException e)
{System.out.println(e + "Bien interceptée !");}
```

Tableaux (2)

- Tableaux à deux dimensions :

```
int[][] matrice = new int[2][3];
int[][] liste_double;
liste_double = new int[2][];
liste_double[0] = new int[3];
liste_double[1] = new int[6];
```

- Référence non-initialisée = `null`.
- Pas besoin de désallouer la mémoire si le tableau n'est plus utile – c'est la tâche du processus de ramasse-miettes.
- Le "ramasse-miettes" cherche les zones de mémoire qui ne sont plus référencées.

Chaînes de caractères

- Chaînes constantes : "bonjour"
- L'affectation par "=" est erronée !
- == ne fait pas de comparaison caractère par caractère !
- Opérateur de concaténation : +
- Conversions :

```
boolean b;
System.out.println("bonjour" + b);
```

Qqs classes spéciales avec leurs attributs

- `Integer` : attributs `MAX_VALUE` et `MIN_VALUE`, méthode `intValue()`.

```
Integer tableau = new Integer[2];
tableau[0] = new Integer(5);
tableau[1] = new Integer(Integer.MIN_VALUE);
int resultat = tableau[0].intValue() + tableau[1].intValue();
```

- Similaires : tous les types primitifs numériques et `boolean`.
- `String` : méthode `valueOf()` pour conversion de variables de divers types en string.

Structure d'un programme Java

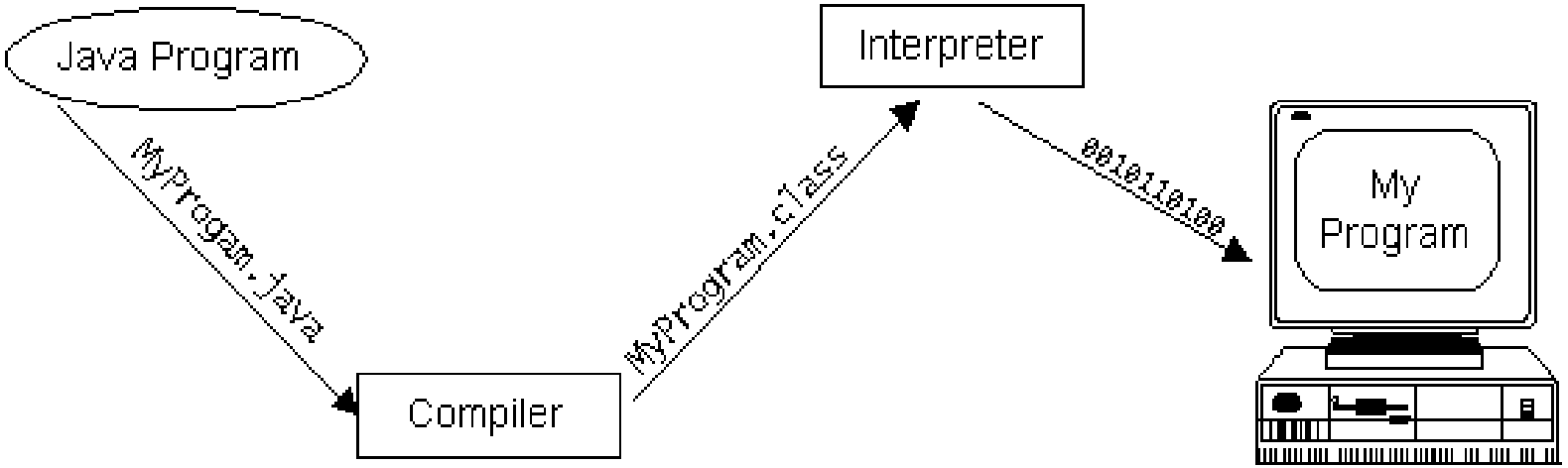
- Programme = ensemble de classes + interfaces ; chaque classe contient les attributs et les méthodes qu'elle définit.
- Un programme exécutable doit avoir une méthode `main` :

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

- Tout fichier source doit avoir l'extension `java`.
- Compilation : `javac nom_fichier.java` – crée un fichier bytecode du nom de **chaque classe**.
- Exécution :

  `javac nom_de_la_classe_ou_se_trouve_le_main`

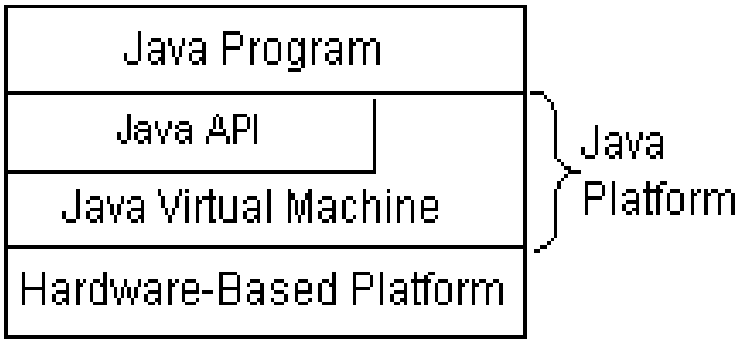
Compilation et interprétation



- Programme Java **compilé** en **bytecode**
- Bytecode **interprété** sur une Machine Virtuelle Java (JVM)
- JVM – spécification indépendente de machine.

Plate-forme Java

- JVM
- API (Application Programming Interface) = bibliothèque de classes et paquetages de base.



Paquetages API : `java.lang`, `java.util`, `java.awt`, `java.applet`, `java.io`, `java.net`, `java.swing`.