Modélisation Orientée OBjet

Premiers pas avec le langage Java

Valentin Honoré

valentin.honore@ensiie.fr

FISA 1A

La programmation orientée objet : un paradigme parmi d'autres

1. Programmation impérative
   - le plus répandu
   - **Principe :** opérations en séquence qui modifient l'état du programme
   - **Programmation procédurale** (ex : C) : concevoir en terme de structures de données et de traitements
   - **Programmation objet :** conception centrée sur des objets qui intéragissent entre eux
     - On ne s'intéresse plus à la mise en œuvre d'un objet, mais aux fonctionnalités qu'il fournit
     - Objet = entité du programme fournissant des fonctionnalités
     - Encapsule une structure de données et des méthodes qui manipulent cette structure de données
     - Expose des fonctionnalités

2. Programmation déclarative
   - **fonctionnelle :** applications comme un ensemble de fonctions mathématiques (ex : Lisp)
   - **logique :** les composants d'une application sont des relations logiques (ex : Prolog)

3. Programmation concurrente, synchrone etc

Plan du cours

Le langage de programmation JAVA

- Java est un langage de programmation
  - Première version sortie en 1995 (J.Gosling et P. Naughton, SUN)
  - Racheté par Oracle en 2010
  - Actuellement en JAVA SE 21 (Septembre 2023)
  - L'un des langages les plus utilisés depuis 30 ans
- Java permet de programmer des types d'application très variés
  - Des jeux, des serveurs, des applications mobiles, *back-end* etc
- Java possède un ensemble de bibliothèques très complet
  - `https://docs.oracle.com/en/java/javase/21/docs/api/index.html`
  - $\sim$ 2.000.000 de lignes de code
  - Pour tous les domaines applicatifs

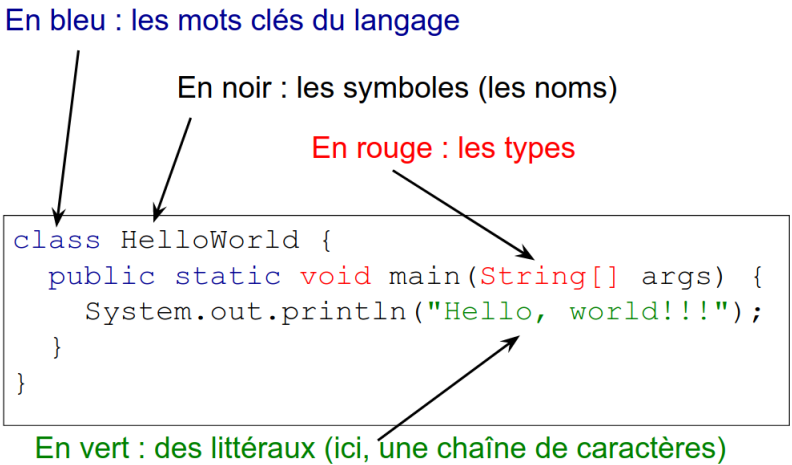
Dans un programma Java, on trouve :

- Des mots clés : des mots qui ont un sens réservé dans le langage :
  class, if, while, ||, &&, do...
- Des symboles : des noms servant à identifier des éléments
  - Constitué de caractères, chiffres (sauf au début), _ ou $
  - La casse est significative (majuscule vs minuscule)
- Des types : spécifie la nature de certains symboles
  Entier (`int`), chaîne de caractères (`String`), flottant (`float`)...
- Des valeurs littérales : des valeurs ayant un type donné
  42 (entier), "Bonjour, monde !" (chaîne de caractère), 3.14 (flottant)

Mon premier programme JAVA (1/3)

- Syntaxe très proche du C
- Langage typé : **ATTENTION NÉANMOINS**
  - les types sont déclarés par l'utilisateur
  - chaque variable, chaque méthode a un certain type
  - **JAVA n'est pas entièrement typé à la compilation**
  - Contrôlé à l'exécution : la conversion explicite et la manipulation de tableaux (possibles erreurs de typage)
- On va y revenir tout au long du cours. **NE PAS NEGLIGER CES ASPECTS DE TYPAGE**
- Langage fortement orienté objet (on va le voir tout au long du cours)
- Pas de pointeurs ! Mais... ne vous réjouissez pas trop vite : notion de *références*

Mon premier programme JAVA (2/3)



Mon premier programme JAVA (3/3)

```
/*************************************************
 *  Compilation:  javac HelloWorld.java
 *  Execution:    java HelloWorld
 *
 *  Prints "Hello, World".
 *
 *
 ************************************************/

public class HelloWorld {

    public static void main(String[] args) {

        // Prints "Hello, World" in the terminal window.
        System.out.println("Hello, World");
    }

}
```

ATTENTION A LA NOTATION DE VOS FICHIERS/CLASSES

- En JAVA, le nom du fichier et le symbole qui suit class doivent coïncider ! ! !

- Convention de nommage :
  - Majuscule pour le premier symbole
  - Minuscule pour tous les autres symboles

```
$ emacs HelloWorld.java
```

```
class HelloWorld{
...
}
```

Plan du cours

Principe

- *Compile once, execute anywhere*
  - Un programme Java est compilé en **bytecode**

```
$ javac MonSuperProgramme.java
```

  - le *bytecode MonSuperProgamme.class* est interprétable par une **machine virtuelle**

```
$ javac MonSuperProgramme.java
```

- La machine virtuelle Java est installable sur toutes les plateformes
- Portabilité à défaut de performance

Distributions Java

- Java SE : Standard Edition
  - "Let you develop and deploy Java applications on desktops and servers. Java offers rich user interface, performance, versatility, portability and security that today's application require."
- Java EE : Enterprise Edition
  - "the standard in community-driven enterprise software (...). Each release integrates new features that align with industry needs, improves application portability, and increases developer productivity."
- Java ME, Card, TV
  - systèmes embarqués
- et bien d'autres...

- **Modulaire**
- **Portable (JDK)**
- **Prédictible**
- Performance faible (virtualisation)
- Installation :

```
sudo apt install default-jdk
```

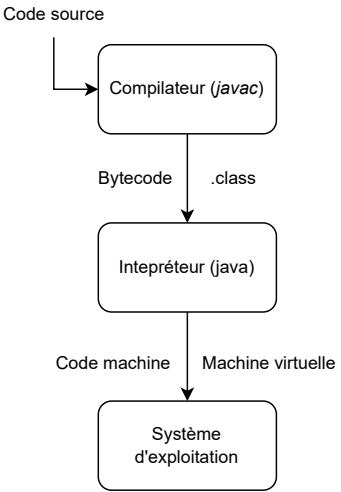


Figure – Processus d'interprétation du langage

- Compilation :

```
$ javac HelloWorld.java
```

- Virtualisation :

```
$ java HelloWorld
```

- Résultat :

```
$ Hello, World
```

Compilation, quelques détails de plus

- le mot clé `-classpath`
  - code compilé référence des classes d'autres répertoires
  - liste des répertoires, séparés par ' :' (' ;' sous Windows)
  - Exemple :

```
javac -classpath /code/exemple:/code/demo HelloWorld.java
```

- Création de "HelloWorld.class" dans un répertoire de compilation
  - option '-d' pour destination
  - Exemple :

```
javac -d code/bytecode -classpath /code/exemple:/code/demo HelloWorld.java
```

Et pour l'interprétation ?

- le mot clé `-classpath` doit aussi être utilisé
  - même syntaxe que pour la compilation
  - Exemple :

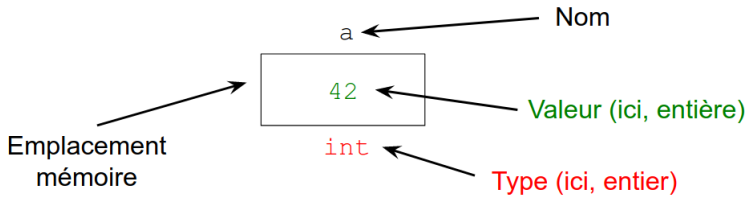
```
java -classpath /code/exemple:/code/demo HelloWorld
```

La suite : syntaxe Java, avec types, opérateurs, conversion de type etc

Plan du cours

Les variables en Java (1/2)

- Une variable est un emplacement mémoire
  - Qui possède un nom (le nom de la variable)
  - Un type (la nature de ce que contient la variable)
  - Et une valeur (le contenu de la variable)



Les variables en Java (2/2)

En Java, les types de base sont les :

- Booléens : `boolean` (true ou false)
- Entiers (signés) : `byte` (1 octet), `short` (2 octets), `int` (4 octets), long (8 octets)
- Réels : `float` (4 octets), `double` (8 octets)
- Caractères : `char` (un caractère unicode)
- Chaînes de caractères : `String` (plusieurs caractères)

Exemple de déclaration de variables

Définition avec :

- type symbole ; /* ? valeur indéfinie ? */
- ou type symbole = litteral ;

```
class HelloWorld {
    public static void main(String[] args) {
        String msg = "Coucou";/* Coucou */
        char c = '!'; /* ! */
        int x = 2; /* 2 */
        int y = x + 5; /* 7 (addition entiere) */
        float z; /* ? valeur indefinie ? */
        boolean b = true; /* true */
    }
}
```

Les littéraux en Java

- Un entier : une valeur entière sans symbole particulier
  - Si suivi de l, valeur de type `long` (2l)
  - Sinon de type `int` (2)
- Un réel : une valeur avec un point sans symbole particulier
  (OU AVEC EXPOSANT $3.14e7 == 3.14 \times 10^7$)
  - Si suivi de f, valeur de type `float` (3.14f)
  - Sinon de type `double` (3.14)
- Un caractère : un caractère entouré d'apostrophes ('a')
- Une chaîne de caractères : une suite de caractères entourée de guillemets ("Ceci est une chaîne")

Types de données primitifs en Java

Type	Classe éq.	Valeurs	Portée	Défaut
boolean	Boolean	`true` ou `false`	N/A	false
byte	Byte	entier signé	{-128 ... 128}	o
char	Character	caractère	{\uoooo ... \uFFFF}	\uoooo
short	Short	entier signé	{-32768 ... 32767}	0
int	Integer	entier signé	{-2147483648 ... 2147483647}	0
long	Long	entier signé	$\{-2^{31} \dots 2^{31} - 1\}$	0
float	Float	réel signé	$\{-3.4028234^{38} \dots 3.4028234^{38}\}$	0.0
double	Double	réel signé	$\{-1.797693134^{308} \dots 1.797693134^{308}\}$	0.0

Table – Types primitifs de données en Java

Plan du cours

Les opérateur en Java (1/2)

- Opérations arithmétiques sur les nombres (entiers ou flottants) +, -, *, /, % (modulo), ++ (incrémente), -- (décrémente)
- Opérations bit à bit (sur les entiers) & (et), | (ou), ^ (xor), ~ (complément), << (décalage à gauche), >> (décalage à droite)
- Opérations sur les booléens && (et), || (ou), ! (non)
- Opérations sur les chaînes de caractères + (concaténation)

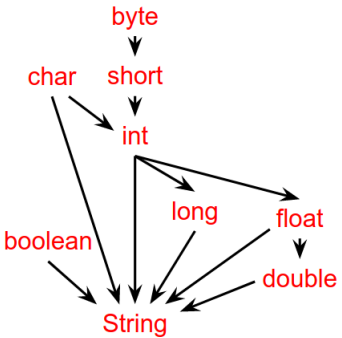
Les opérateur en Java (1/2)

- Opérations de comparaison $==$, $<=$, $<$, $>=$, $>$, $!=$ (différent)
- Opération d'affectation (tous types) $=$
- Combinaison d'affectation avec d'autres opérations possible $+=$, $/=$, $»=$ etc.
  (Exemple : `a += 42`)

Plan du cours

Conversion automatique de type

- Avant d'effectuer une opération, un type peut être converti vers un type plus fort

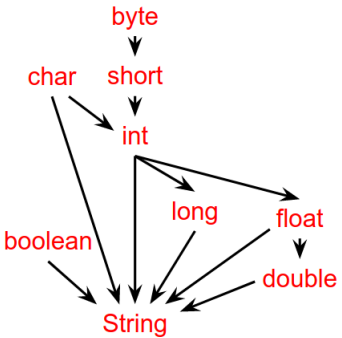


Type le plus faible

Type le plus fort

Conversion automatique de type

- Avant d'effectuer une opération, un type peut être converti vers un type plus fort



Type le plus faible

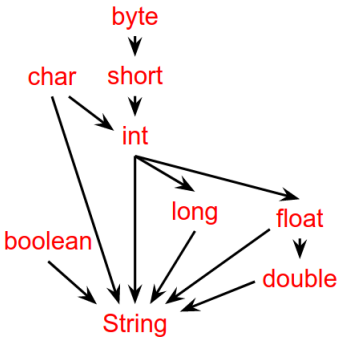
Type le plus fort

Remarque

Lors de la conversion d'un char vers un int, c'est le numéro de caractère qui est renvoyé (par exemple 'a' a la valeur 97)

Conversion automatique de type

- Avant d'effectuer une opération, un type peut être converti vers un type plus fort



Type le plus faible

Type le plus fort

```
int x = 3;
Double y = x + 2.2; // 5.2
String s1 = "Coucou" + x;
/* Coucou3 */
char c = 'a';
String s2 = "Coucou" + c;
/* Coucoua */
int d = c + 1;
/* 98 car 'a' = 97 */
```

Conversion explicite de type

Quand il n'existe pas de conversion implicite, on utilise la conversion explicite avec (`type`)

- Dans ce cas, Java tronque le nombre

```
double x = 97.7;
int y = (int)x; /* 97 */
char c = (char)y; /* 'a' car c vaut 97 */
byte b = (byte)(y * 3); /* 291 modulo 256 = 35 */
```

Plan du cours

Pour les blocs, c'est comme le C !

La structure d'un programme Java est donnée par des blocs

- Commence par un { et termine par un }
- Regroupe un ensemble de déclarations

```
class Test {
    public static void main(String[] args) {
        if(0 == 1) {
            System.out.println("Cette machine est bizarre");
            System.out.println("Elle pense que 0 == 1 !");
        }
    }
}
```

- Accolades optionnelles si seulement une déclaration dans le bloc

Structures conditionnelles (1/3)

Schéma conditionnel simple

- si alors ... sinon (si alors ... sinon ...)
- Parties else if et else optionnelles

```
class Exemple {
    static void main(String[] args) {
        int a = 21 * 2;
        if(a == 42) {
            System.out.println("Super !");
            System.out.println("Java, c'est facile !");
        }
        else if (a<42){
            System.out.println("Gros soucis!!!");
        }
        else
            System.out.println("Perturbation dans la Force");
    }
}
```

Structures conditionnelles (2/3)

Schéma conditionnel plus complexe

- Si val vaut v1, exécute body1
- Sinon, si val vaut v2, exécute body2
- Sinon, si ...
- Sinon, exécute bodyn

```
class Exemple {
    static void main(String[] args) {
        int c = a;
        switch(c) {
            case 'a': System.out.println("c est un a"); break;
            case 'b': System.out.println("Bizarre..."); break;
            default: System.out.println("Echec?"); break;
        }
    }
}
```

Structures conditionnelles (3/3)

Attention avec le switch

- Si pas de break, continue l'exécution
- Attention au code "spaghetti" !

```
class Exemple {
    static void main(String[] args) {
        char c = a;
        switch(c) {
            case 'a': System.out.println("uniquement a");
            case 'b': System.out.println("a ou b"); break;
            case 'c': System.out.println("uniquement c");
            default: System.out.println("ni a, ni b");
        }
    }
}
```

Structures de contrôle

- Boucle while
  Tant que cond faire body

- Boucle do ... while
  Faire body tant que cond

- Boucle for
  Exécute `init`
  Tant que cond faire
  `body` puis `iter`

```
while(cond) {
  body
}
```

```
do {
  body
} while(cond)
```

```
for(init; cond; iter) {
  body
}
```

Quelques exemples (rappels du C)

```
int x = 0;
while(x < 10) {
  System.out.println(x);
  x++;
}
```

```
for(int x=0; x<10; x++) {
 System.out.println(x);
}
```

```
int x=10;
do {
  x--;
} while(x >= 3);
```

En résumé !

- En bash :
  - Toutes les variables sont de type chaîne de caractères (`PATH=/usr/bin`)
  - Les blocs d'instructions sont délimités par des mots clés (do ... done, if ... ; then ... fi etc...)

- En Java :
  - Le type d'une variable est explicite (`int` x=100 → entier, `String` name="Arya" → chaîne de caractères)
  - Les blocs sont explicitement délimités par des accolades { ... }

Quelques étrangetés en conversion de type

- Avant d'effectuer une opération arithmétique, Java convertit un `byte`, un `short` ou un `char` en `int`

```
byte x = 1;
byte y = 2;
byte z = x + y;
```

- Interdit car :

  Java convertit x et y en `int` avant d'effectuer l'opération
  → le résultat est un `int`
  et il est impossible d'affecter un `int` dans un `byte`

Encore plus de bizarreries !

- Si les deux membres de l'opération sont des littéraux entiers (`byte`, `short`, `int`) ou caractères (`char`)

  $\rightarrow$ alors Java effectue normalement le calcul en `int`

- Mais après le calcul, s'il faut affecter le résultat à un des types entiers ou au type `char`, Java convertit le résultat si le nombre n'est pas trop grand

Encore plus de bizarreries !

```
char c = 'a';
char d = c + 1;
```

Interdit car c n'est pas un littéral
(il est impossible de convertir de `int` vers `char` lors de l'affectation)

```
char c = 'a' + 1;
```

Autorisé car 'a' est un littéral
('a' devient le nombre 97, la somme vaut 98, qui représente le caractère 'b')