Programmation Orientée Objet – Java

Cours 1 : Intro et bases du langage

Viviane Pons

Master BIBS Université Paris-Saclay

Introduction : la programmation objet

Quoi, comment, pourquoi

Que signifie “Programmation Orientée Objet” ?

Introduction : la programmation objet

Quoi, comment, pourquoi

Que signifie “Programmation Orientée Objet” ?

C’est un **paradigme de programmation** c’est à dire une façon d’approcher la programmation informatique et la résolution de problèmes

Introduction : la programmation objet

Quoi, comment, pourquoi

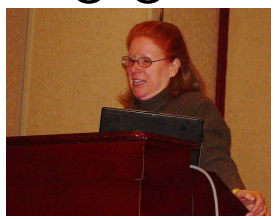
Que signifie “Programmation Orientée Objet” ?

C’est un **paradigme de programmation** c’est à dire une façon d’approcher la programmation informatique et la résolution de problèmes
Exemples d’autres paradigme : programmation impérative (C, Pascal, Basic, . . .) , programmation fonctionnelle (Lisp, Haskell, OCaml, . . .)

Quand ?

Vers la fin des années 60, sur les travaux d'Alan Kay et avec l'apparition du langage Smalltalk rendu public en 1980





3 des créateurs-trices du langage Smalltalk : Alan Kay, Dan Ingalls et Adele Goldberg

Alan Kay receiving the Kyoto Prize by Ryan Johnson under CC-BY-SA-4.0 – Dan Ingalls under CC-BY-SA-3.0 – Adele Goldberg at PyCon 2007 by Terry Hancock under CC-BY-SA-2.5

Pourquoi ?

- Organisation de l'information
- Une conception proche de l'humain (plus haut niveau)
- **encapsulation** : séparée la structure interne de la donnée et sa manipulation externe

Exemple

Problème : manipuler des nombres rationnels

```
?? add_rat(int n1, int d1, int n2, int d2) {
    int n = n1 * d2 + n2 * d1
    int d = d1 * d2

    return ??
}
```

Quel type de valeur renvoyer ? Comment s'assurer de la cohérence du programme ? Où faire la simplification ?

Exemple

Première idée : utilisée un type structuré.

```
rat add_rat(rat r1, rat r2) {
    rat r;
    r.n = r1.n * r2.d + r2.n * r1.d
    r.d = r1.d * r2.d;

    return r;
}
```

Mais ça ne résout pas tous les problèmes !

La solution objet

le fichier code ici

```
1
2  public class Rational {
3      public final int n;
4      public final int d;
5
6      Rational(int n, int d) {
7          if(d == 0) {
8              throw new IllegalArgumentException();
9          }
10         int div = Main.gcd(n, d);
11         n = n / div;
12         d = d / div;
13         this.n = n;
14         this.d = d;
15     }
16
17     public Rational add(Rational r2) {
18         return new Rational(n*r2.d + r2.n * d, d * r2.d);
19     }
20
21     public String toString() {
22         return n + "/" + d;
23     }
24 }
```

Quels Langages ?

Smalltalk (1980), Common List (1984), C++ (1985), Object Pascal (1986), Python (1991), PHP (1994), **Java (1995)**, JavaScript (1996), C# (2001), ...

Java

Petit historique

- Développé en 1995 par Sun pour répondre à des questions de **portabilité** et **sécurité** puis racheté par Oracle (2009)
- Langage de programmation libre depuis 2006
- Syntaxe inspirée du C
- Un des langages les plus utilisés d'après le classement RedMonk

Comment ça marche ?

Rappel : un processeur ne sait exécuter que des **fichiers binaires** qui correspondent à des instructions machines

Comment passer d'un programme à des instructions machines ?

Comment ça marche ?

Rappel : un processeur ne sait exécuter que des **fichiers binaires** qui correspondent à des instructions machines

Comment passer d'un programme à des instructions machines ?

- En compilant ! (Cobol, Fortran, C, C++, Pascal, OCaml)
- En interprétant ! (Langages de scripts tels que : bash, pearl, python, PHP)

Comment ça marche ?

Rappel : un processeur ne sait exécuter que des **fichiers binaires** qui correspondent à des instructions machines

Comment passer d'un programme à des instructions machines ?

- En compilant ! (Cobol, Fortran, C, C++, Pascal, OCaml)
- En interprétant ! (Langages de scripts tels que : bash, pearl, python, PHP)

Et Java alors ?

Java est un langage intermédiaire :

- Étape 1 : code écrit dans un fichier `mycode.java`

Java est un langage intermédiaire :

- Étape 1 : code écrit dans un fichier `mycode.java`
- Étape 2 : code **compilé** par `javac`

  `javac mycode.java`

  Le compilateur produit un fichier `mycode.class` en **Byte code** (spécifique à Java). Ce n'est pas un exécutable

Java est un langage intermédiaire :

- Étape 1 : code écrit dans un fichier `mycode.java`
- Étape 2 : code **compilé** par `javac`

  `javac mycode.java`

  Le compilateur produit un fichier `mycode.class` en **Byte code** (spécifique à Java). Ce n'est pas un exécutable
- Étape 3 : Byte code **interprété** par la **machine Java**

  `java mycode`

Avantages / inconvénients

Avantages : portabilité

- **langage interprété** : Un code unique quelque soit la plateforme : allocations mémoires, sécurité, spécificités des plateformes gérées par la machine java
- **langage interprété** : Des fichiers "compilés" `.class` similaires d'une plateforme à l'autre
- **langage compilé** : contrairement à un langage purement interprété, on ne partage cependant pas le code
- **langage compilé** : vérification statique de la

Inconvénients :

- **langage interprété** : moins de contrôle bas niveau
- **langage interprété** : moins rapide qu'un langage directement compilé
- **langage interprété** : nécessite une installation sur la machine cliente
- **langage compilé** : pas la souplesse d'un langage interprété

Démo : mon premier code Java

```java
public class HelloWorld {

    public static void main(String[] args) {

        System.out.println("Hello World !");
    }
}
```

Environnement d'exécution / de développement

Environnement d'exécution / de développement

Pour exécuter du Java

Il faut **JRE** = "Java Runtime Environnement": contient la machine Java (JVM) et l'API Java de base que nous verrons plus en détail

Environnement d'exécution / de développement

Pour exécuter du Java

Il faut **JRE** = "Java Runtime Environnement": contient la machine Java (JVM) et l'API Java de base que nous verrons plus en détail

Pour compiler du java (développer en Java)

Il faut **JDK** = Java Development Kit : contient compilateur `javac`, javadoc, JRE, ...

Quel IDE ?

Integrated Development Environment

On utilisera **Eclipse**

- tout en un : édition, compilation, exécution
- analyse syntaxique à la volée
- libre
- très courant (existe depuis 2001)

Refaisons “Hello World” avec Eclipse

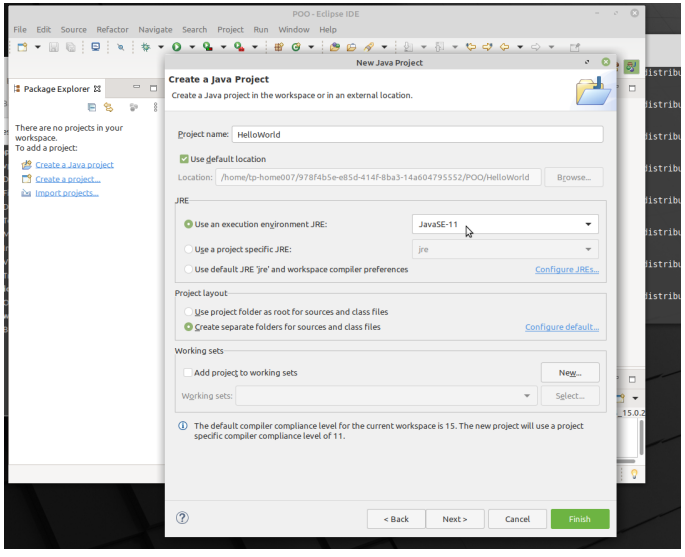


Figure 1: Ouvrir eclispe et créer un projet Java

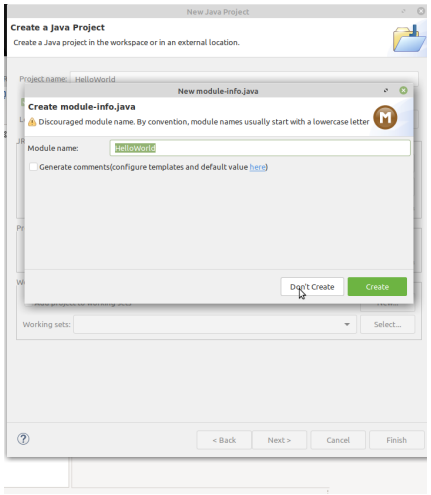


Figure 2: On choisit de ne PAS créer le fichier module

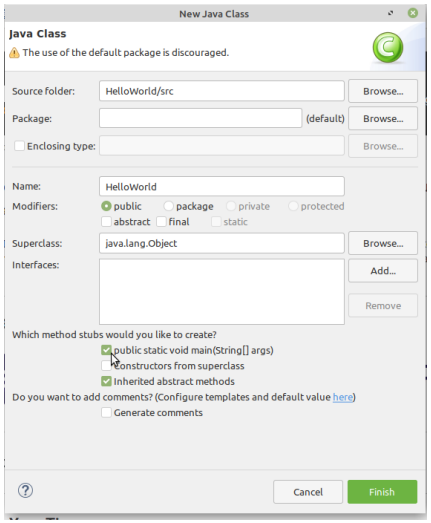


Figure 3: Créer une nouvelle classe HelloWorld

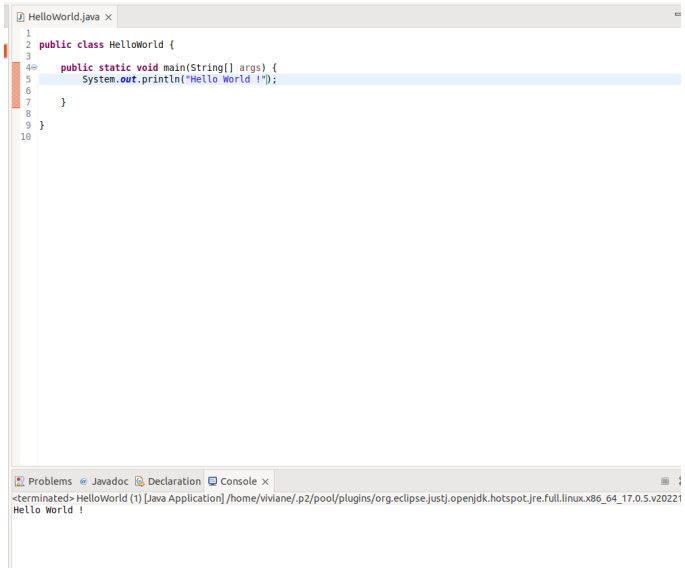


Figure 4: On écrit le code restant de la classe HelloWorld et on exécute avec le bouton vert

Codons en Java !

Quelle Syntaxe ? . . . Comme le C

Principes de base :

- on doit déclarer les variables
- les blocs de code sont entre { }
- il y a des ; à la fin des lignes

Pour le reste. . . Apprenons par l'exemple, faisons le TP !