Institut Universitaire de Technologie
Aix*Marseille Université

Module M3105

Conception et programmation objet avancées

Sébastien Thon

sebastien.thon@univ-amu.fr

Chapitre 1

Introduction

Nous allons utiliser le langage de programmation orienté objet Java

- Crée par SUN Microsystems en 1995
- SUN racheté par Oracle Corporation en 2009

Pourquoi Java ?

Java est très utilisé dans l'industrie du fait de ses qualités :

- ***Portabilité***
- ***Robustesse***
- ***Langage orienté objet***

- ***Portabilité***

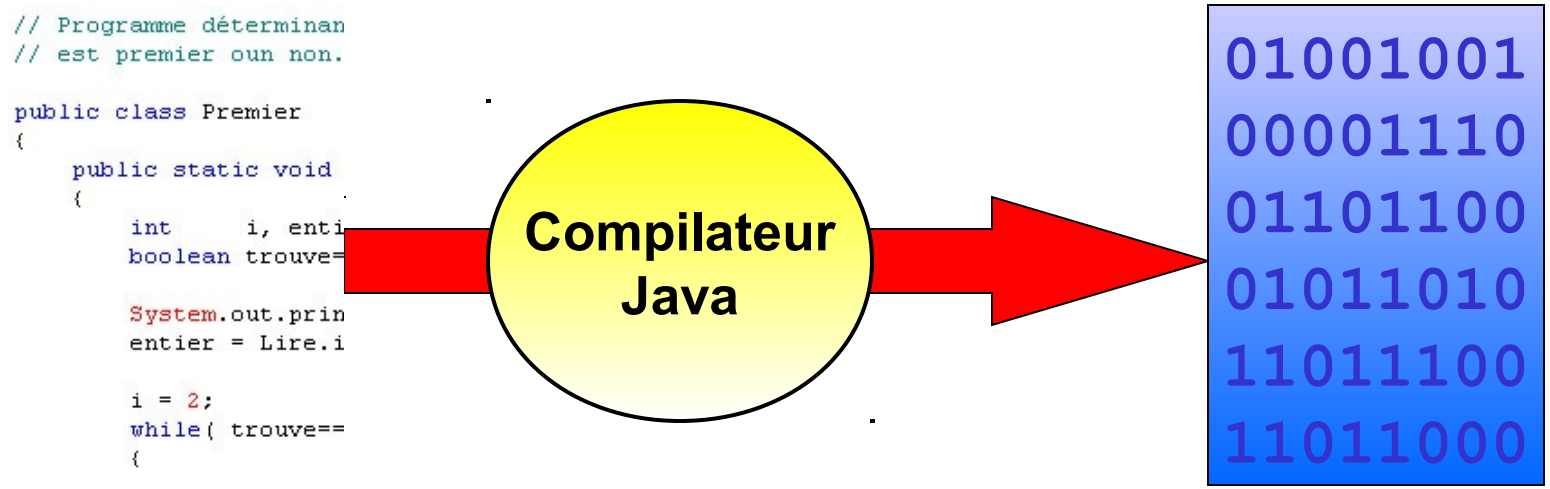
Avantage principal de Java : sa **portabilité**.

Un programme réalisé en Java peut fonctionner sur n'importe quelle machine disposant d'un interpréteur Java, que cette machine tourne sous Windows, Linux, MacOSX, etc.

Important dans le cadre d'applications exécutées via Internet, des machines très diverses étant connectées.

Cette portabilité est due au fait que le code source Java est compilé pour une machine dite **virtuelle** (appelée **JVM** : ***J**ava **V**irtual **M**achine*).

Le code machine résultant est nommé **ByteCode**.



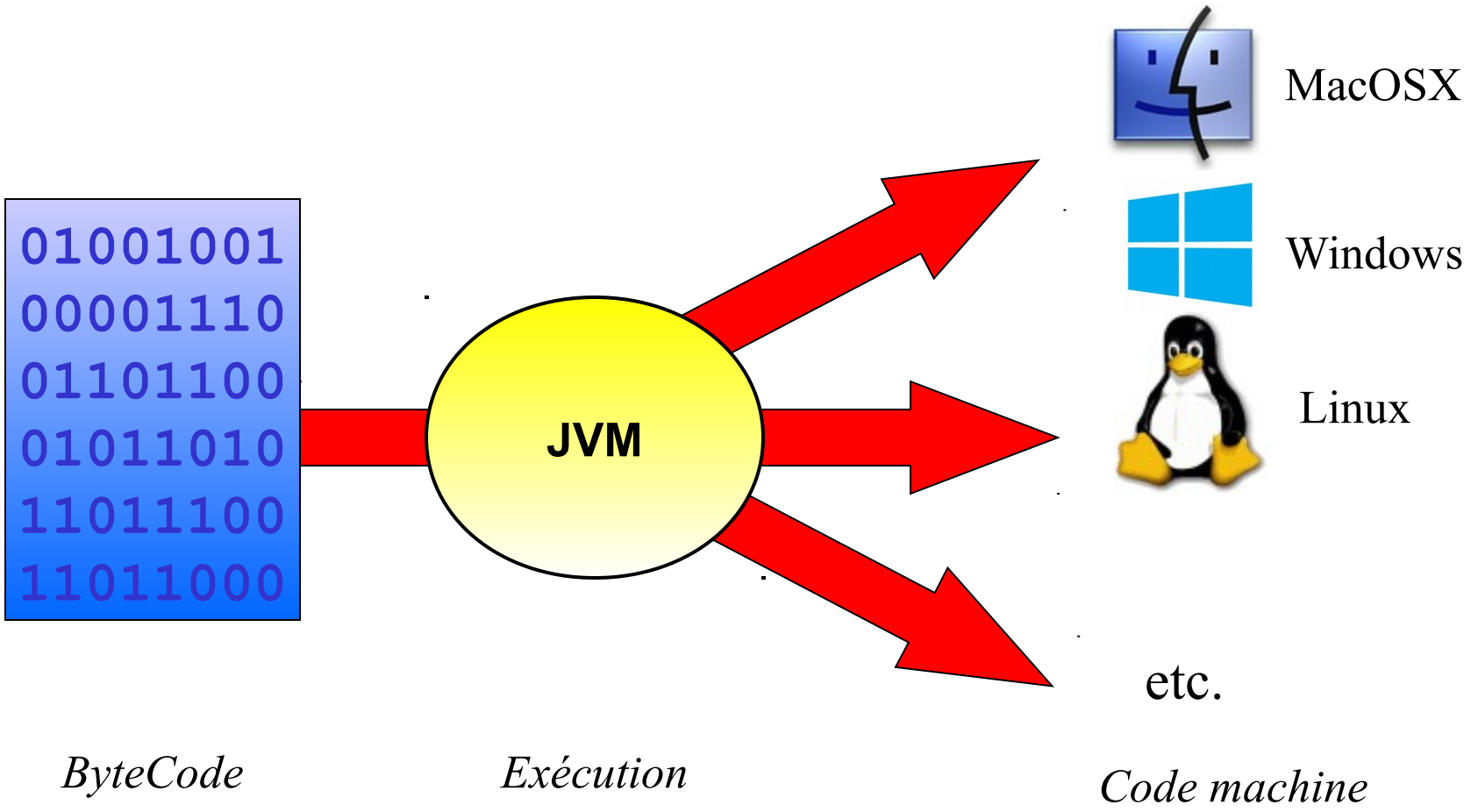
Code source en Java

Compilation

ByteCode

Lors de son **exécution** par la JVM, ce code sera **interprété**

→ transformation en un code machine compréhensible par le microprocesseur sur lequel tourne l'application.



Example devices powered by Oracle





Small

- RFID Readers
- Parking Meters
- Intelligent Power Module
- Smart Meters

Medium

- Routers & Switches
- Storage Appliances
- Network Management Systems
- Factory Automation Systems
- Security Systems

Large

- Multi Function Printers
- ATMs
- POS Systems
- In-Flight Entertainment Systems
- Electronic Voting Systems
- Medical Imaging Systems

→**Java intéresse beaucoup les industriels :**

Une application est écrite une fois pour toute en Java et fonctionnera sur n'importe quelle machine dotée d'une JVM.

→Réduction très importante des coûts de développement : pas besoin de réécrire le code source de l'application ou de le recompiler pour plusieurs systèmes différents.

- ***Robustesse***

Java = langage « **robuste** » :

- Langage fortement typé → permet moins de "bidouillages" que le langage C, par exemple.
- Détection de bugs classiques de programmes C et C++ comme le dépassement de tableau.
- Contrairement à C et C++, ne permet pas l'usage de pointeurs pour directement adresser des portions de mémoire.
- Gestion « propre » de la mémoire (désallocation).
- …

- Langage orienté objet

Programmation objet : réutilisabilité, sécurité, héritage, …

Langage C++ : compromis entre le langage C et un langage purement orienté objet comme Smalltalk.

Même s'il s'inspire de la syntaxe du C++, Java a été réécrit à partir de zéro, de manière à être totalement orienté objet.

Un exemple de programme

```
import java.util.*;

public class Bonjour
{
  public static void main (String [] arg)
  {
    String nom;
    int    age;
    Scanner clavier = new Scanner(System.in);

    System.out.println("Entrez votre nom : ");
    nom = clavier.nextLine();

    System.out.println("Entrez votre age : ");
    age = clavier.nextInt();

    System.out.println("Vous vous appelez " + nom
                    + " et vous avez " + age + " ans");
  }
}
```

Remarques

```
public class Bonjour
```

- Un programme peut être répartit en plusieurs **classes**, stockées dans des fichiers différents (voir programmation objet).

- Le programme doit être tapé dans un fichier d'extension **`.java`** portant le **même** nom que la classe. Par exemple, ici : **`Bonjour.java`**

- Tout le reste du programme est compris entre une accolade ouvrante '**{**' et une accolade fermante '**}**'.

Remarques (suite)

```
public static void main (String [] arg)
```

- Un programme peut être décomposé en plusieurs **fonctions** (sous programmes).

- Dans la classe principale du programme, il faut impérativement une fonction **`main()`**. Cette fonction sera la première à être exécutée (*main* = « principal »).

- La fonction **`main()`** reçoit en paramètre un tableau de chaînes de caractères correspondant aux **arguments** passés au programme sur la ligne de commande lors de l'exécution.

Compilation

Il faut compiler le code source du fichier **`Bonjour.java`** pour obtenir le fichier de bytecode (code machine pour une machine virtuelle).

On utilise la commande **javac** :

```
javac Bonjour.java
```

→ On obtient un fichier **`Bonjour.class`** contenant le bytecode.

Exécution

Pour exécuter le fichier de bytecode au moyen de la JVM, on utilise la commande **java** :

```
java Bonjour
```

Remarques :

- On ne spécifie pas l'extension **`.class`**
- Attention aux majuscules.

Cas d'un projet composé de plusieurs fichiers

Pour compiler un projet composé de **principal.java**, **compte.java** et de **banque.java** :

```
javac principal.java compte.java banque.java
```

→ crée les fichiers **principal.class**, **compte.class** et **banque.class**

Si principal.java contient la fonction **main**, on exécute le programme résultant avec la commande :

```
java principal
```

Autre possibilité

Ecrire un fichier texte (par exemple : projet) dans lequel vous écrivez la liste des noms des fichiers de votre projet :

projet

```
principal.java
compte.java
banque.java
```

On compile ensuite avec javac en précisant le nom de ce fichier texte précédé de @ :

```
javac @projet
```

Plateforme Java = JVM + diverses API :

- *Java Platform, Standard Edition* (**Java SE**) contient les API de base et est destiné aux ordinateurs de bureau ;

- *Java Platform, Enterprise Edition* (**Java EE**) contient, en plus du précédent, des API orientées entreprise et est destiné aux serveurs ;

- *Java Platform, Micro Edition* (**Java ME**) est destiné aux applications embarquées (lecteurs blu-ray, téléphonie, imprimantes, automobile, domotique, électroménager...)

(API : Application Programming Interface) = ensemble normalisé de classes, de méthodes et de fonctions)