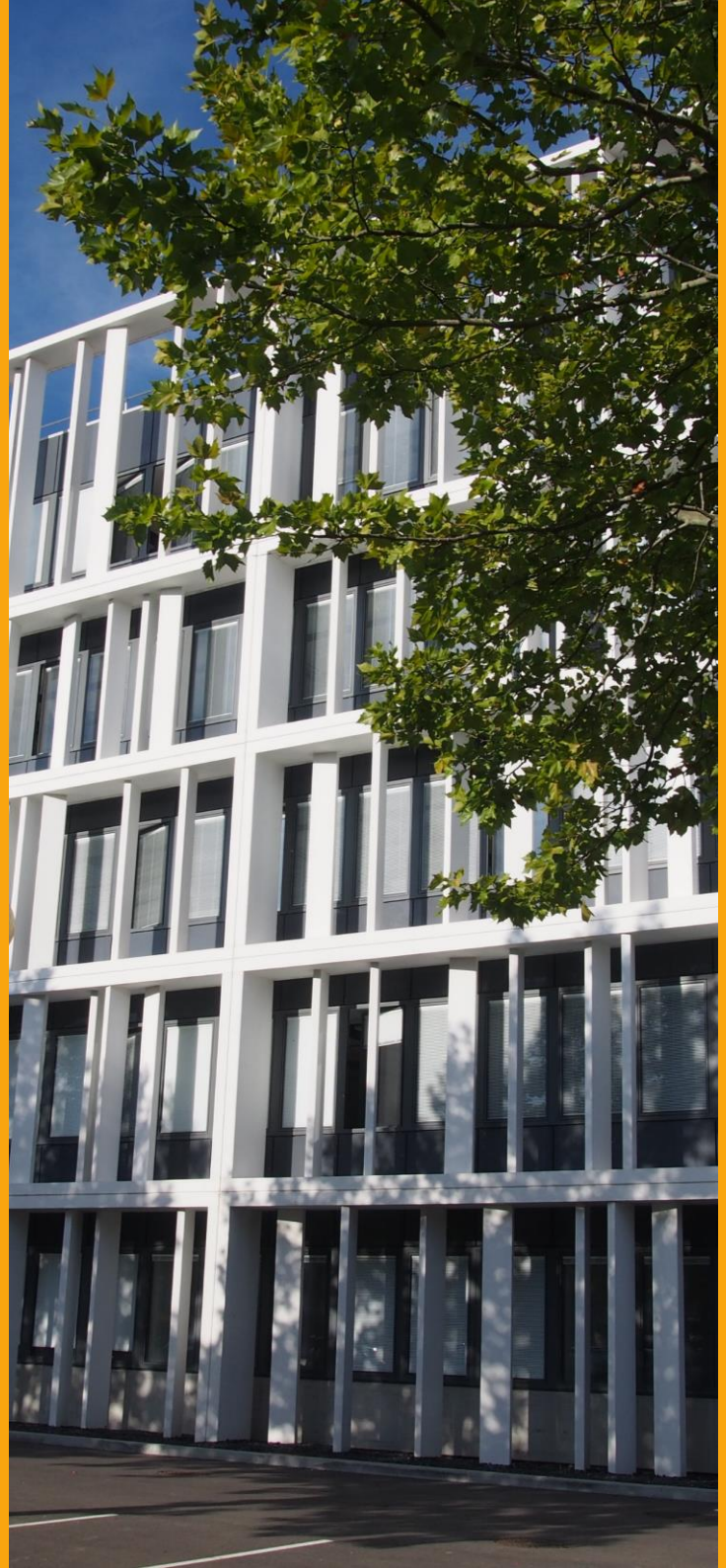
RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

UNIVERSITÉ TOULOUSE III PAUL SABATIER
FACULTÉ SCIENCES ET INGÉNIERIE

Introduction Java



Mathieu RAYNAL
mathieu.raynal@irit.fr
http://www.irit.fr/~Mathieu.Raynal

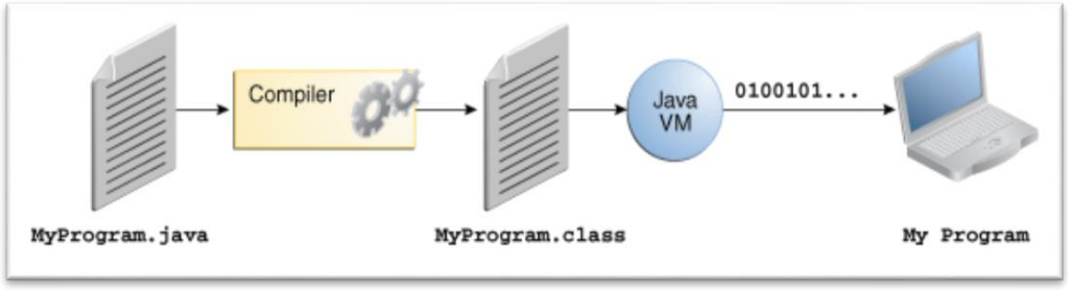
Présentation du langage JAVA

Java en bref …

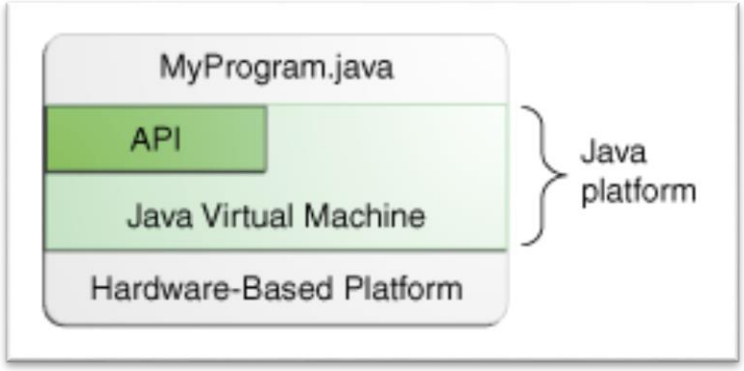
- Langage de programmation libre
  - Développé par Sun (1995)
  - Racheté par Oracle (2009)
- Portable « write once, run everywhere »
- Interprété
  - Bytecode
  - Java Virtual Machine (JVM)
- Programmation Orientée Objet
- Une bibliothèque de classes : API

Java Virtual Machine (JVM)

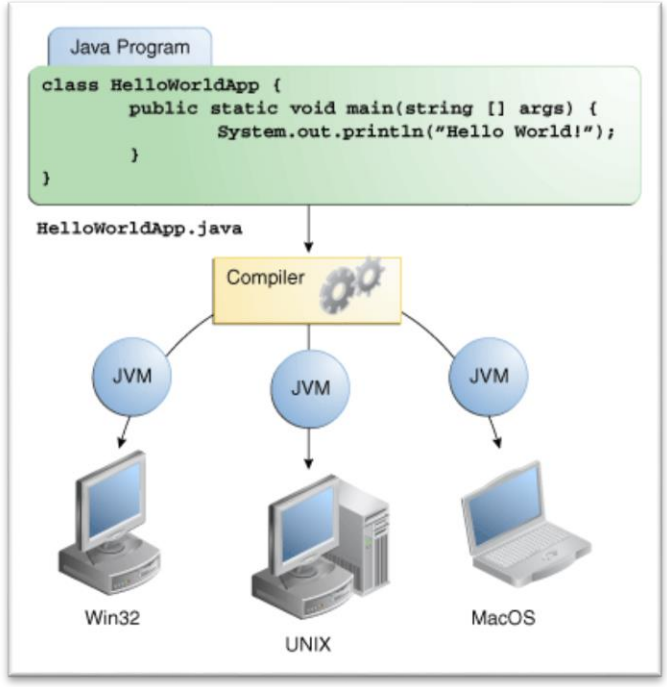
- Machine virtuelle
  - le compilateur Java génère du byte code et non de l'assembleur



  - le byte code est exécuté sur une machine virtuelle : la JVM



- Avantages
  - Plus de gestion manuelle de la mémoire
  - Indépendance de la plateforme cible



Quelques définitions

- JVM : Java Virtual Machine
  → Machine virtuelle permettant d'interpréter et d'exécuter le bytecode Java.

- JRE : Java Runtime Environement
  → Kit destiné au client pour pouvoir exécuter un programme Java
  → Il contient la JVM et les bibliothèques standards

- JDK : Java Development Kit
  → Kit destiné au programmeur
  → Contient JRE, exemples, compilateur …

- API : Application Programming Interface
  → Bibliothèque de classes standards

https://docs.oracle.com/javase/8/docs/api/

Environnement de développement

Compiler un fichier Java

- La compilation de chaque fichier .java génèrera un fichier d'extension .class
- Commande de compilation

  javac *[options]* fichiers_source
- Les options possibles

  -classpath path
  - path = chemin d'accès aux classes extérieures au projet et nécessaires à la compilation

  -d rep
  - spécifie le répertoire dans lequel les fichiers .class seront enregistrés

  -deprecation
  - Indique les méthodes *deprecated* et comment les remplacer

Exécuter une application Java

- Commande d'éxecution

  java *[options]* nomdelaclasse *[paramètres]*

- Les options possibles

  -**classpath** path

  - Path = chemin d'accès aux classes nécessaires pour exécuter le projet

Documentation

- Permet de faire des documentations identiques à celle de l'API
- Commande

```
javadoc –d rep fichiers_source
```

- Commentaires pris en compte dans la documentation
  /** */
- Mots clés
  @see
  @author
  @version
  @param
  @return
  @throws

Création de fichiers JAR

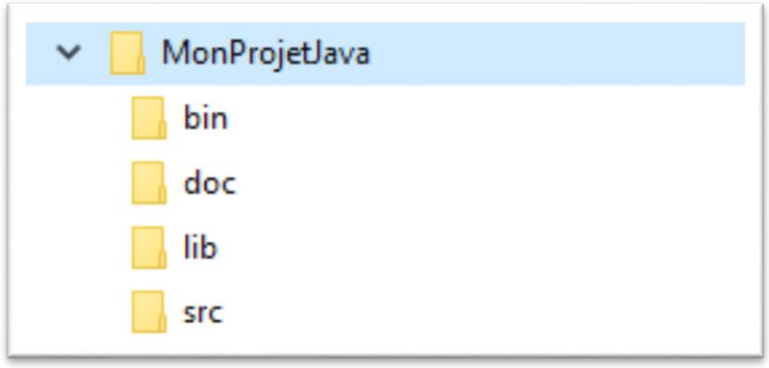
- JAR = Java ARchive
- Regroupe un ensemble de fichiers sous forme d'archive
  - Souvent sous forme d'arborescence
- Commande
- Options

  -c : création, avec ou sans contenu

  -u : Mise-à-jour

De bonnes pratiques

- Structurer son répertoire de travail
  - Un dossier pour les sources (.java)
  - Un dossier pour les fichiers bytecode (.class)
  - Un dossier pour les ressources extérieures
  - Un dossier pour la documentation



Syntaxe Java

A savoir …

- Ce n’est pas un script !
  - Toutes les instructions sont dans des « fonctions » des classes
  - *On ne parlera plus de fonction mais de **méthode***
- Toutes les instructions terminent par un point virgule
- On utilise les accolades pour délimiter un bloc d’instructions
- Le programme s’exécute à partir de la fonction main de la classe qui est appelée
  - Le programme exécute toutes les instructions de la méthode **main**

La méthode principale : main

```
public static void main(String[] args)
{
        …
}
```

- Il peut y en avoir une par classe
  - Toute classe qui a une méthode main peut être exécutée
- Pour les projets comportant plusieurs classes, il est préférable de faire une classe à part pour la méthode main
- Le tableau args contient les différents éléments, *sous forme de chaines de caractères,* passés en argument au moment du lancement du projet

Afficher du texte

- Affichage simple

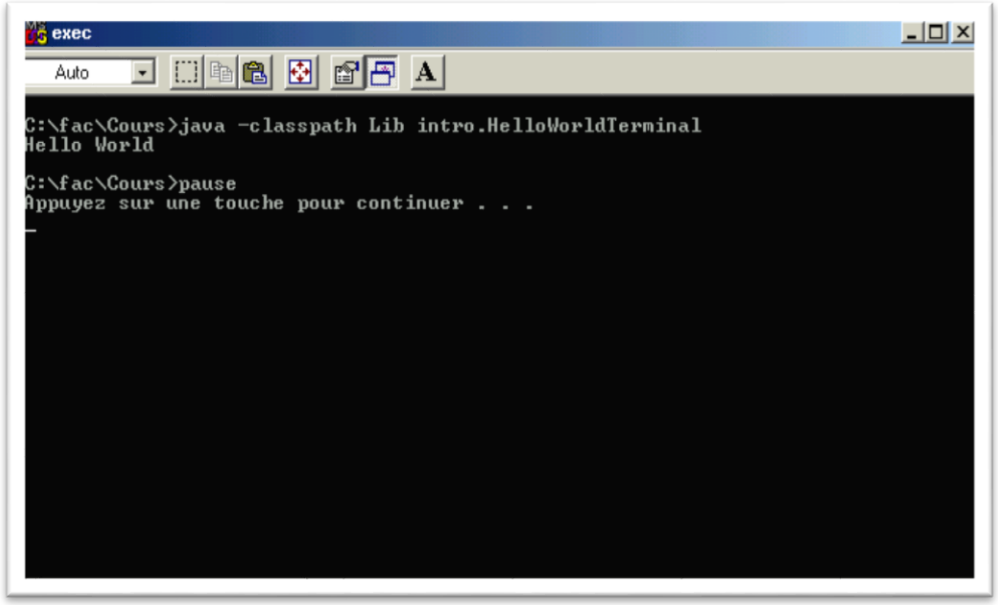
```
System.out.print(“Bonjour”);
```

- Affichage puis retour à la ligne

```
System.out.println(“Bonjour”);
```

- Exemple « Hello World »

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



Les variables

- Elles sont de trois types
  - Types primitifs,
  - Références à un tableau,
  - Références à un objet.

- Déclaration d'une variable
  - Type de la variable suivi du nom de cette variable

```
int somme;
String nom;
```

- Affectation d'une valeur à une variable

```
somme = 0;
Nom = "Roger";
```

- Elles doivent être initialisées **explicitement** avant de pouvoir s'en servir.

Les types primitifs

- Les différents types primitifs :
  - boolean (true,false)
  - byte (8 bits)
  - char (16 bits)
  - short (16 bits)
  - int (32 bits)
  - long (64 bits)
  - float (32 bits)
  - double (64 bits)
- Tous les types numériques sont signés
- Il n’existe pas de transtypage implicite

Les tableaux

- Déclaration
  - Type d'éléments présents dans le tableau suivi de [], puis du nom du tableau

```
int[] tab;
```

- Création
  - Il faut commencer par créer un tableau en indiquant le nombre d'éléments souhaités dans ce tableau

```
tab = new int[42];
```

  - Il faut initialiser chaque cellule du tableau avant de pouvoir l'utiliser

```
tab[0] = 0;
```

Les tableaux

- Taille du tableau connue grâce à length

```
int taille = tab.length
```

- Les éléments du tableau
  - Pour un tableau de taille N, les éléments sont stockés de l'indice 0 à l'indice N-1
  - Pour accéder à un élément du tableau

```
int premierElement = tab[0];
int dernierElement = tab[tab.length-1];
```

  - Les différents éléments du tableau sont manipulables comme toute autre variable

Structure conditionnelle : **if** … **else** …

- Les instructions du bloc A seront exécutées si la condition est vraie
- Le bloc else n'est pas obligatoire
  - Si il est présent, les instructions du bloc B seront exécutées si la condition du if est fausse
- Dans tous les cas, les instructions C sont exécutées

```
if(condition)
{
        instructions du bloc A;
}
else
{
        instructions du bloc B;
}
Instructions C;
```

Les conditions

- On teste
  - une égalité entre deux éléments avec **==**
  - une inégalité entre deux éléments avec
    - Différent : **!=**
    - Supérieur strict : **>**
    - Supérieur ou égal : **>=**
    - Inférieur strict : **<**
    - Inférieur ou égal : **<=**

- Plusieurs conditions peuvent être regroupées par
  - ET : **&&**
  - OU : **||**

Structure conditionnelle : **switch**

- Le programme exécute les instructions en fonction de la valeur de la variable
- Les différentes valeurs testées sont définies par **case** et se terminent par un break
- Pour les cas non traités, il y a le bloc **default**

```
switch(variable)
{
  case val1 :
      instructions;
  break;
  …
  case valn :
      instructions;
  break;
  default:
      instructions;
  break;
}
```

Les boucles

```
for(init compteur ; condition arrêt ; modif cpt)
{
        instructions;
}
```

```
while(condition d'arrêt)
{
        instructions;
}
```

```
do
{
        instructions;
} while(condition d'arrêt);
```

Exercices

1. Afficher les puissances de 2 de 2^1 à 2^{10}

2. Afficher les tables de multiplication de 1 à 9 sous la forme

3. Afficher les chaines de caractères passées en argument au lancement du programme