Plan

Programmation Java

Y. ALJ

Références

Les références pour ce cours sont :

- Head First Java, 2nd Edition : Kathy Sierra, Bert Bates (pour les exercices).
- Le cours de Gauthier Picard
  `https://www.emse.fr/~picard/cours/1A/java/`

Introduction à Java

- Java est un langage de programmation de haut niveau.
- Conçu au début des années 1990 par Sun Microsystemes et actuellement maintenu par Oracle.
- Java est un langage indépendant de la plateforme utilisée :
- On écrit le programme une fois.
- Ce programme peut être exécuté sur différentes plateformes
- L'esprit de Java : "Write Once Run Anywhere"

Premier programme

- On commence comme à chaque quand on apprend un nouveau langage de programmation par afficher ”Hello World”.

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

Premier programme : discussion

- En Java chaque ligne de code qui peut être exécutée doit être à l'intérieur d'une classe.
- Dans note exemple, on a appelé cette classe MaPremiereClasse.

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

Premier programme : discussion

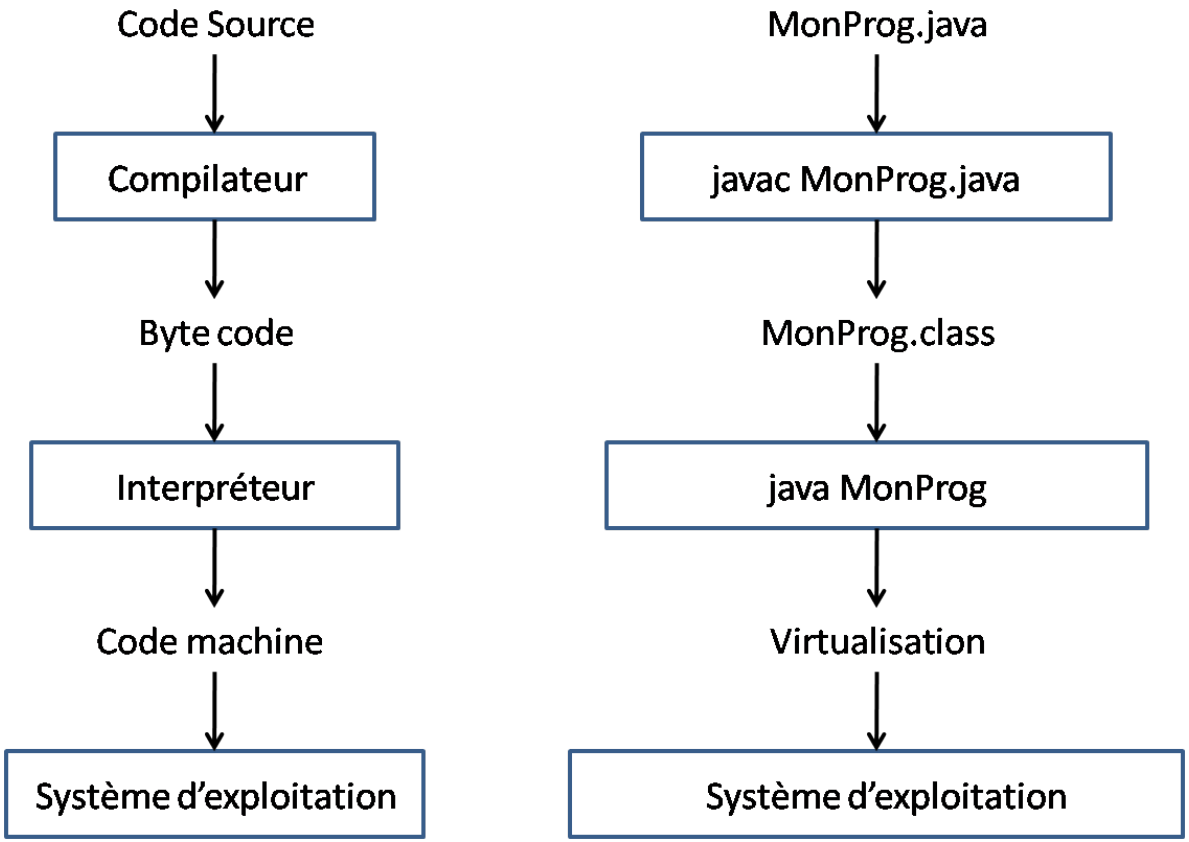
- En Java chaque application doit avoir un point d'entrée qui est une fonction (ou méthode) appelée main.

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

Premier programme : discussion

- Pour exécuter nos programmes notre fonction main doit obligatoirement avoir cette signature :
  `public static void main(String [] args)`
- public : tout le monde peut accéder à cette méthode.
- static : la méthode en question peut être appelée sans instancier la classe contenant la fonction main.
- void : méthode ne renvoie aucune valeur.
- main : nom de la méthode. Exemple : `void test()` ne renvoie rien et n'a pas de paramètres.

Processus de développement d'une application Java



Explication

1. Un programmeur Java écrit son code source : c'est un fichier `".java"` qui contient une classe.
2. Ce code source est alors compilé par le compilateur `javac` et traduit en un autre langage appelé bytecode. Le résultat est un fichier `".class"`. Ce bytecode n'est pas directement utilisable.
3. Le bytecode est interprété par la machine virtuelle java qui transforme ce bytecode en code machine compréhensible par le système d'exploitation.

Installation de Java

- On a besoin :
  - Pour compiler : `javac`.
  - Pour exécuter l'application : `java`.
- Deux produits java existent :
  - le JDK (Java Development Kit) : un environnement qui contient les outils pour compiler et pour exécuter les applications java.
  - le JRE (Java Runtime Environment) : un environnement d'exécution de classes java.

Familles de langages de programmation

- Chaque langage de programmation appartient à une famille de langage qui définit une méthodologie pour programmer (appelée aussi paradigme de programmation).
- Par exemple : le langage C est un langage de programmation procédurale. Un programmeur C commence par identifier l'ensemble des traitements qu'il souhaite effectuer, puis écrit des fonctions qui réalisent ces traitements.
- La programmation orientée objet POO (en anglais Object Oriented Programming) propose une méthodologie orientée sur les données.
- En POO, le programmeur identifie les objets sous forme d'un ensemble de données dans un premier temps.
- Dans un second temps, le programmeur écrit les traitements.
- Un objet en POO est une entité regroupant des données (appelées attributs) et des traitements (appelés méthodes).

Comparaison entre C et Java

Langage C	Langage Java
programmation fonctionnelle	programmation orienté objet
langage de bas niveau	langage de haut niveau
gestion de la mémoire par l'utilisateur	gestion par la JVM
+ applications rapides (temps d'exécution)	- applications moins rapides
- éventuelle mauvaise gestion de la mémoire	+ moins de problèmes de gestion de la mémoire

La multiplication en Java

```
class Multiplication{

    public static void main(String [] args){
        int a = 2;
        int b = 4;
        System.out.println("Le produit de a=" + a +
            " par b=" + b +" est " + multiply(a,b)
            );
    }
    public static int multiply(int a, int b){
        return a*b;
    }
}
```

La multiplication en C

```c
#include <stdio.h>

int multiply(int a, int b)
{
    return a*b;
}
void main()
{
    int a = 2;
    int b = 4;
    printf("La multiplcation de a=%i par b=%i est %
        i", a, b, multiply(a,b));
}
```

Les deux approches de programmation

Approche fonctionnelle

Que doit faire mon programme ?

Approche orientée objet

De quoi être composé mon programme ?

Syntaxe d'une classe

Le code source d'un programme Java est contenu dans un ou plusieurs fichiers d'extension ".java".

- Une seule classe publique par fichier.
- Le nom du fichier doit être le même que celui de la classe.
- Par convention, le nom d'une classe commence toujours par une majuscule.

Syntaxe d'une classe

```
class <nom de la classe>{
    <contenu de la classe>
}
```

Contenu d'une classe

Contenu d'une classe

- des attributs : variables typée.
- Dans l'exemple ci-contre les attributs sont : `solde` et `proprietaire`.
- des méthodes (ou opération) : ensemble d'instructions de traitement.
- Dans l'exemple ci-contre les méthodes sont : `double getSolde()` et `void credite(double val)`.

Exemple

```
class CompteBancaire{
    String proprietaire;
    double solde;
    double getSolde(){
        return solde;
    }

    void credite(double val)
        {
        solde = solde + val;
    }
}
```

Types de données primitifs

Les mêmes types qu'en C

- **short** : les entiers signés sur 16 bits.
- **int** : les entiers signés sur 32 bits.
- **long** : entiers signés sur 64 bits.
- **float** : réels sur 32 bits.
- **double** : réels sur 64 bits.
- **char** : charactères sur 16 bits.
- **void** : utilisé pour le type de retour de fonctions.

Plus deux nouveaux types

- **boolean** : prend les valeurs **true** ou **false**.
- **byte** : : les entiers signés sur 8 bits.

Plus une classe

- `String`

Exemple

```
boolean result = true;
char capitalC = 'C';
byte b = 100;
short s = 10000;
int i = 100000;
double d1 = 123.4;
// meme valeur que d1, mais avec la notation scientifique
double d2 = 1.234e2;
float f1  = 123.4f;
```

Portée d'une variable

```
class Portee{
    int a;
    public void test(int b){
        if(a>b){
            int c = b;
            b = a;
            a = c;
        }
        System.out.println(b);
    }
}
```

L'entier *a* est visible dans toute la classe Portee.

Portée d'une variable

```
class Portee{
    int a;
    public void test(int b){
        if(a>b){
            int c = b;
            b = a;
            a = c;
        }
        System.out.println(b);
    }
}
```

L'entier *b* est visible à l'intérieur de la méthode test.

Portée d'une variable

```
class Portee{
    int a;
    public void test(int b){
        if(a>b){
            int c = b;
            b = a;
            a = c;
        }
        System.out.println(b);
    }
}
```

L'entier *c* est visible à l'intérieur du bloc `if`.

Instanciation

Example

L'instanciation est l'opération qui consiste à créer un objet à partir d'une classe. Ceci se fait avec le mot clé `new`.

Création de la classe

```
class Point {
    public int x = 0;
    public int y = 0;
    // un constructeur
    public Point(int a, int b) {
        x = a;
        y = b;
    }
}
```

Instanciation

```
class DemoPoint {
    public static void main(String [] args){
        Point mon_origine = new Point(2, 3);
        System.out.println("Coordonnee X= " + mon_origine.x);
        System.out.println("Coordonnee Y= " + mon_origine.y);
    }
}
```

Constructeurs

- Pour instancier une classe, c'est à dire créer un objet à partir d'une classe il faut donc utiliser l'opérateur `new`.
- `new` appelle une méthode spéciale de la classe : **le constructeur**.

Exemple

```
class Personne{
int age;
char sexe;
float taille;
Personne(int age, char sexe, float
     taille){
    this.age = age;
    this.sexe = sexe;
    this.taille = taille;
  }
int getAge(){
return age;
}
}
```

- Un constructeur **porte le même nom de la classe** dans laquelle est définie.
- Un constructeur **n'a pas de valeur de retour** .
- Un constructeur peut avoir des arguments.
- **Plusieurs constructeurs** peuvent exister dans la même classe avec des arguments différents.
- Il faut **au moins un constructeur** dans une classe pour pouvoir instancier des objets.

Méthodes

Appel des méthodes

- En java, les méthodes ne peuvent pas être appelées seules (comme en C pour les fonctions).
- Elles sont toujours appelées sur un objet.
- On met un "." entre l'objet et la méthode appelée.

Exemple : définition de la classe

```
class Personne{
int age;
char sexe;
float taille;
Personne(int age, char sexe, float
     taille){
    this.age = age;
    this.sexe = sexe;
    this.taille = taille;
  }
int getAge(){
    return age;
  }
}
```

Exemple : instanciation + appel de méthode

```
Personne p = new Personne(20, 'H',
     1.80);
p.getAge();
```

Visibilité des membres

- Lors d'un grand projet, on écrit plusieurs classes Java.
- Ces classes interagissent entre elles.
- La visibilité d'une classe et de ses membres définit quelles autres classes y ont accès.
- Il y a quatre niveaux de visibilité en Java :
  - **`public`** on peut y avoir accès par n'importe quelle classe.
  - **`private`** on peut y accéder seulement à l'intérieur de la classe où c'est défini.
  - **`protected`** sera expliqué plus ultérieurement.
  - `par defaut` sera expliqué plus ultérieurement.

Mot clé static

```
public class MyClass {
  static void myMethod() {
    // code a executer
  }
}
```

- `myMethod()` est le nom de la méthode.
- `static` veut dire que la méthode appartient à la classe et non à l'objet instancié.
- On peut donc appeler une méthode déclarée en `static` sans instancier un objet de cette classe.

Exemple : différence entre static et public

```
public class MyClass {
  // method static
  static void myStaticMethod() {
    System.out.println("methode static peut etre appele sans creer d
        objet");
  }
  // methode publique
  public void myPublicMethod() {
    System.out.println("methode publique doit etre appelee apres la
        creation dun objet");
  }
  //methode main
  public static void main(String[] args) {
    myStaticMethod(); // appel de la methode static
    // myPublicMethod(); //cette ligne genere une erreur
    MyClass myObj = new MyClass(); // cree un objet de la class MyClass
    myObj.myPublicMethod(); // On appelle la methode publique
  }
}
```