BO - HUB

B-INN-000

Introduction au Java

Programmation Orienté Objet

Introduction au Java

language: java
compilation: javac, gradle, automated compilation with IDE
build tool: gradle

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

Découverte du Java

Introduction

- Qu'est-ce que le Java ?
  - Java est un langage de programmation orienté objet, typé, "compilé" qui fonctionne grâce à une **J**ava **V**irtual **M**achine (**JVM**)
- Qu'est-ce que la **JVM** ?
  - La JVM est une machine virtuelle qui permet l'interprétation du code java compilé (**.class**), et aussi des archives java (**.jar**)
  - C'est l'un des avantages du Java ! Grâce à la **JVM** le code est très portable, il peut être exécuté partout tant qu'une **JVM** est présente !
  - D'autres langages peuvent aussi fonctionner sur la **JVM** c'est le cas du **Scala** et du **Kotlin**

Pré-requis

Si vous n'avez pas déjà installé les outils nécessaires à ce workshop suivez les instructions situées ici :

`https://github.com/alwyn974/workshop-java/blob/master/REQUIREMENTS.md`

Le Java ! La théorie

Les modifieurs de visibilité et d'accès

En java, on peut changer la visibilité d'une variable, d'une classe, d'une fonction, pour cela il faut utiliser des **keyword** spécifique

- Les modifieurs de visibilité
  - `private` personne ne peut modifier/accéder à cette variable à part la classe actuelle
  - `protected` uniquement les classes enfants ou appartenant au même package peuvent modifier/accéder à cette variable
  - `public` tout le monde peut modifier/accéder à cette variable

Par défaut une variable, classe ou fonction déclarée sans modifieur de visibilité est en privé

- Les modifieurs d'accessibilité
  - `static` une variable ou une fonction statique peut être accédée sans instancier la classe, elle est aussi unique à cette classe
  - `final` pour faire simple, c'est l'équivalent du `const` en C/C++
  - `super` bon techniquement, ce n'est pas un modifieur d'accessibilité, au contraire il permet d'accéder à des méthodes/variables de la classe parente

Les variables statiques sont toutes les mêmes peu importe l'instance de la classe

Le keyword **super** est pratique lorsque l'on veut utiliser la méthode de la classe parente et y ajouter quelque chose, ou bien pour modifier une variable de la classe parente à la place d'une variable déclaré dans la classe enfant

LES TYPES

Vu que le Java est basé sur le C++ et donc le C, les types restent à peu près les mêmes, pour certains ils sont améliorés ou ajoutés

Variable

- `boolean` permet de stocker les états `true` et `false` (comme bool en C/C++)
- `byte` permet de manipuler des entiers codés sur 1 octet (comme char)
- `double`, `int`, `long`, `float`, `short` restent les mêmes
- Vos objets à vous

Classes

- `enum` les enums en java sont beaucoup plus complet que ceux en C/C++, vu que ce sont des objets, on peut y assigner plus qu'un simple Integer comme valeur
- `interface` permet de créer une interface
- `abstract class` permet de créer une classe abstraite
- `class` permet de créer une classe normale

Le polymorphisme - L'héritage

L'héritage en java fonctionne comme en C++ à quelques différences près.
On ne peut hériter que d'une seule classe, néanmoins on peut hériter de plusieurs interfaces
Exemple:

```
public class Animal {
    public void eat() {
        System.out.prinln("Animal can't eat");
    }
}

public interface ILiveable {
  void live();
}

public interface IMovable {
    void move(int x, int y);
}

public class Cat extends Animal implements ILiveable, IMovable {

    private final String name;

    public Cat(String name) {
        this.name = name;
    }

    /**
     * Overriden function from interface IMovable
     * @param x the x coordinate
     * @param y the y coordinate
     */
    @Override
    public void move(int x, int y) {
        System.out.println("X: " + x + " Y:" + y);
    }

    /**
     * Overriden function from interface IMovable
     */
    @Override
    public void live() {
        System.err.println("I'm dead bruh");
    }

    /**
     * Overriden method from Animal class
     */
    @Override
    public void eat() {
        System.out.println("I'm hungry !!!");
    }

}
```

Le polymorphisme - La surcharge

Comme en C++ on peut surcharger une méthode avec différents types de paramètre tout en ayant le même nom
Exemple:

```
public class Main {

    public static void main(String[] args) {
        System.out.println("coucou"); //type of parameter is String
        System.out.println(1); //type of parameter is Integer
        System.out.println(0.0); //type of parameter is Double
        System.out.println(0.0f); //type of parameter is Float
        System.out.println(5 < 0); //type of parameter is Boolean
        System.out.println(new Object()); //type of parameter is Object, print an
            Object call the method toString() of the object
    }

}
```

Le polymorphisme - La redéfinition

Comme en C++, si l'on hérite d'une classe, d'une interface ou d'une classe abstraite, on peut surcharger les méthodes présentes dans la classe parente sauf si les méthodes ont le modifieur `private` ou `final`

Contrairement au C++ on ne peut pas hériter de toutes les variables/méthodes d'une classe, uniquement des méthodes et variables `public` et `protected`

Instancier un objet

C'est presque comme en C++, sauf qu'il faut utiliser new partout.

En java, on ne s'occupe pas de la gestion mémoire, c'est le Garbage Collector (**GC**) de la JVM qui s'en occupe automatiquement

Il faut quand même penser à faire un code un minimum optimisé

Exemple:

```
public class Main {

    public static void main(String[] args) {
        Object object = new Object(); //create object variable with Object type
        String str = new String("Hey"); //create str variable with String type
        String string = "toto"; //the good way to create a String
        int[] array = new int[42]; //create array with int[] type
    }

}
```

Le Java ! La pratique :3

Structure d'un projet java

Création d'un objet

Pour créer une classe