TP 1 : Classes et Objets

©2022 Ghiles Ziat
ghiles.ziat@epita.fr

Dans les exercices qui suivent, vous êtes libres de répondre aux questions en utilisant le langage de votre choix parmi Java et C++, sauf lorsqu'il est explicitement demandé d'utiliser un des deux langages ou les deux.

Exercice I : Premier programme (**Java et C++**)

Q1 – Créez deux programmes *Player.java* et *player.cc* qui affichent sur la sortie standard la chaîne de caractère "Hello world"

Q2 – Compilez et exécutez les deux programmes.

Remarque :

- Pour compiler un (ou plusieurs) fichier(s) source(s) java, il faut lancer dans un terminal la commande `javac File.java`. On peut ensuite lancer l'exécutable produit en faisant `java File`
- Pour compiler un (ou plusieurs) fichier(s) source(s) c++, il faut lancer dans un terminal la commande `g++ file.cc`. On peut ensuite lancer l'exécutable produit en faisant `./a.out`

Q3 – Ajoutez à votre classe des attributs modélisant le nom et l'age d'un joueur.

Q4 – Définissez un constructeur avec paramètres pour la classe `Player`.

Q5 – Ajoutez un champ entier `count` qui compte le nombre d'instances créées.

Q6 – Ajoutez à la classe un champ `team` de type `boolean`. On veut garantir la parité de telle sorte à ce que le nombre d'instances ayant un champ `team` à `true` soit le même (à 1 près) que le nombre d'instances ayant le champ `team` à `false`. Comment faire ?

Q7 – Définissez une méthode `print(): void` qui affiche sur la sortie standard les informations relatives à une instance de la classe `Player`.

Q8 – Mettez à jour votre `main` pour créer quelques instances de la classe `Player` et afficher leurs informations.

Exercice II : Visibilité

Dans cet exercice, on veut restreindre l'instanciation d'une classe à une seule instance. Pour cela, nous allons passer par une méthode `getInstance` qui crée une instance uniquement s'il n'en existe pas encore. Sinon elle renverra une référence vers l'objet qui existe déjà.

Q1 – Créez une classe `Singleton` avec un champs `value` de type `String`.

Q2 – Créez un constructeur pour la classe. **Attention à la visibilité :** on veut **forcer** les autres classes à passer par la méthode `getInstance` pour avoir accès à une instance.

Q3 – Créez la méthode `getInstance`. Cette méthode doit être la seule façon de créer une instance de `Singleton`. Doit-il s'agir d'une méthode de classe ou d'instance ?

Q4 – Créez un main (en dehors de la classe `Singleton`) qui initialise deux objets `Singleton`. Affichez le résultat de l'égalité physique entre les deux objets.

Exercice III : Passage des paramètres

Le *passage de paramètres* (*parameter passing* en anglais) définit le type de valeur qui est passé à la fonction pour chaque paramètre. Java et C++ ont deux stratégies d'évaluation différentes et que l'on va découvrir ici.

Q1 – (C++) Définissez une fonction `void swap` qui prend deux variables en paramètre et échange leurs valeurs. Définissez dans une fonction `main` deux variables locales et échangez leurs valeurs en appelant `swap`.

Q2 – (Java) Qu'affiche le code suivant ?

```java
class Passing {
    static void f(int x) {
        x = 42;
        System.out.println("1. x = " + x);
    }

    static void g(int[] array) {
        array[0] = 37;
        System.out.println("4. array[0] = " + array[0]);
    }

    public static void main(String[] args) {
        int x = 11;
        f(x);
        System.out.println("2. x = " + x);
        int[] array = {5};
        System.out.println("3. array[0] = " + array[0]);
        g(array);
        System.out.println("5. array[0] = " + array[0]);
    }
```

```
21 }
```

Que peut-on en conclure sur le passage de paramètre en Java ?

Exercice IV : Portée de l'information

Q1 – Définissez une classe `Car` avec des champ entiers `registration`, `speed` et `distance`.

Q2 – Définissez un constructeur pour la classe `Car` qui crée une instance avec une vitesse et une distance parcourue initialement nulles. On veut de plus que chaque voiture ait un matricule (champ `registration`) unique. Quelle mécanisme mettre en oeuvre pour que ce soit le cas ?

Q3 – Définissez une méthode `print(): void` qui affiche sur la sortie standard les informations relatives à une instance de la classe `Car`.

Q4 – Définissez une méthode `void accelerate(int increment)` qui additionne l'incrément à la vitesse courante.

Q5 – Définissez une méthode `move()` qui incrémente la distance parcourue de l'instance appelante d'une quantité égale à sa vitesse courante.

Q6 – Définissez une méthode `boolean race(int arrival)` qui tire aléatoirement et uniformément [1] un nombre en 1 et 10, appelle la méthode `accelerate` avec cette valeur et appelle ensuite la méthode `move`. La méthode retournera `true` si la distance parcourue est supérieure au paramètre `arrival`.

Q7 – On veut organiser une course qui durera 500 unités de distances entre deux voitures : définissez un `main` qui crée deux voitures et les fait concourir tant qu'une des deux n'a pas franchi la ligne d'arrivée.

1. En java, on pourra créer une instance de la classe `java.util.Random` et utiliser sa méthode `nextInt()`. En C++, on pourra utiliser la fonction `int rand (void);`