TP 1 : Classes and Objects

©2022 Ghiles Ziat
ghiles.ziat@epita.fr

In the following exercises, you are free to answer the questions using the language of your choice among Java and C++, except when it is explicitly requested to use one of the two languages or both.

Exercice I : First program (**Java and C++**)

Q1 – Create two programs *Player.java* and *player.cc* which prints on the standard output the string "Hello world" character

Q2 – Compile et execute both programs.

Note :

- To compile one (or more) java source file(s), you have to run the command in a terminal `javac File.java`. You can then run the executable produced by doing `java File`
- To compile one (or more) c++ source file(s), you have to run the command in a terminal `g++.cc file`. You can then run the executable produced by doing `./a.out`

Q3 – Add attributes to your class that model the name and age of a player.

Q4 – Define a constructor with parameters for the class `Player`.

Q5 – Add an integer field `count` that counts the number instances created.

Q6 – Add to the class a field `team` of type `boolean`. We want to guarantee parity in such a way that the number of instances having a `team` field set to `true` is the same (within 1) as the number of instances having the `team` field set to `false`. How to do so ?

Q7 – Define a `print():void` method that prints on the standard output the information relating to an instance of the `Player` class.

Q8 – Update your `main` to create some instances of the class `Player` and display their information.

Exercice II : Visibility

In this exercise, we want to limit the instantiation of a class to a single instance. For this, we will go through a method `getInstance` which creates an instance only if none exists yet. Otherwise it will return a reference to the object that exists already.

Q1 – Create a `Singleton` class with a `value` field of type `String`.

Q2 – Create a constructor for the class. **Beware of the visibility :** we want to **force** the other classes to use the `getInstance` method to access an instance.

Q3 – Create the `getInstance` method. This method should be the only way to create an instance of `Singleton`. Should it be a class or instance method ?

Q4 – Create a main (outside the `Singleton` class) that initializes two `Singleton` objects. Display the result of physical equality between the two objects.

Exercice III : Parameter passing

Parameter passing defines the type of value that is passed to the function for each parameter. Java and C++ have two different evaluation strategies which we will discover here.

Q1 – (C++) Define a function `void swap` that takes two variables in parameter and exchange their values. Set in a function `main` two local variables and exchange their values by calling `swap`.

Q2 – (Java) What does the following code output ?

```
1  class Passing {
2      static void f(int x) {
3          x = 42;
4          System.out.println("1. x = " + x);
5      }
6
7      static void g(int[] array) {
8          array[0] = 37;
9          System.out.println("4. array[0] = " + array[0]);
10     }
11
12     public static void main(String[] args) {
13         int x = 11;
14         f(x);
15         System.out.println("2. x = " + x);
16         int[] array = {5};
17         System.out.println("3. array[0] = " + array[0]);
18         g(array);
19         System.out.println("5. array[0] = " + array[0]);
20     }
21 }
```

What can we conclude about parameter passing in Java ?

Exercice IV : Scope of information

Q1 – Define a class `Car` with integer fields `registration`, `speed` and `distance`.

Q2 – Define a constructor for the `Car` class that creates an instance with a speed and distance traveled initially set to 0. We also want each car to have unique registration numbers. What can we do to make this happen ?

Q3 – Define a `print():void` method that prints on the standard output the information relating to an instance of the `Car` class.

Q4 – Define a method `void accelerate(int increment)` which adds the increment to the current speed.

Q5 – Define a `move()` method that increments the distance traveled by the calling instance by an amount equal to its current speed.

Q6 – Define a method `boolean race(int arrival)` which draws randomly and uniformly [1] a number between 1 and 10, calls the `accelerate` method with this value and then calls the `move` method. The method will return `true` if the distance traveled is greater than the parameter `arrival`.

Q7 – We want to organize a race between two cars, that will last 500 units of distance. Define a `main` that creates two cars and makes them compete as long as one of the two has not crossed the finishing line.

1. In java, we can create an instance of the class `java.util.Random` and use its `nextInt()` method. In C++, we can use the function `int rand (void);`