Cours MOOC EPFL d'initiation à la programmation illustré en Java

# Second assignment
# Conditionnal branching

J. Sam & J.-C. Chappelier

## 1 Exercise 1 — Bike rental

The aim of this exercise is to write a program that calculates bicycle rental amounts (for an online 24/24 service).

The rental rates are as follows:

- From 00:00 to 07:00 and from 17:00 to 24:00: 1 Franc/hour
- From 07:00 to 17:00: 2 Francs/hour

The program will ask the user to enter the start and end hours of the rental as integers (part of the code furnished) and will compute and display the rental cost.

You will adopt the following simplifications:

- the hours are integers (we don't mind about minutes; each started hour will be fully paid);
- the start time (i.e., hour) of the rental period must always be less than the end time (i.e., hour) of the rent. This means that one cannot hire a bicycle more than 24 hours and that each rental period must lie within the same day.

If the inputs are correct, then the program will calculate and show the rental amount on the screen strictly following the format presented in the examples below. If the times entered by the user are not between 0 and 24 (inclusive), then an error message strictly respecting the following format will be shown and the program will terminate:

```
Les heures doivent être comprises entre 0 et 24 !
```

If the start time of the rent is equal to the end time, then an error message strictly respecting the following format will be shown and the program will terminate:

```
Bizarre, vous n'avez pas loué votre vélo bien longtemps !
```

If the end time of the rent is less than the start time, then an error message strictly respecting the following format will be shown and the program will terminate:

```
Bizarre, le début de la location est après la fin ...
```

The provided file already contains some instructions, that must be used as they are, in order to produce the requested output. You must not use the instruction `return;` in this program.

## 1.1 Instructions

Download the source code `Velo.java` available at the course webpageand complete it.

**WARNING:** you should modify neither the beginning nor the end of the program, only add your own lines as indicated. It is therefore very important to respect the following procedure (the points 1 and 3 concern only Eclipse users):

1. remove automated formatting of the code in Eclipse:

   `Window > Preferences > Java > Editor > Save Actions` (and untick the formatting option if it's on);

2. save the downloaded file as `Velo.java` (respect the upper case). If you work with Eclipse, save it to `[projectFolderUsedForThatExercise]/src/`;

3. refresh the Eclipse project where the file is stored (right click on the project > "refresh") in order to take into account the new file;

4. write your code between these two provided comments:

```
/*******************************************
 * Completez le programme a partir d'ici.
 *******************************************/

/*******************************************
 * Ne rien modifier apres cette ligne.
 *******************************************/
```

5. save and test your program to be sure that it works properly, for example using the values given below;

6. upload the modified file (still named `Velo.java`) in "OUTPUT submission" (not in "Additional"!).

## 1.2 Execution examples

It is mandatory that your code respects the following formatting:

1) Example where the renting cost calculation is done using both rates:

```
Donnez l'heure de début de la location (un entier) : 10
Donnez l'heure de fin de la location (un entier) : 19
Vous avez loué votre vélo pendant
2 heure(s) au tarif horaire de 1.0 franc(s)
7 heure(s) au tarif horaire de 2.0 franc(s)
Le montant total à payer est de 16.0 franc(s).
```

2) Example where the renting cost calculation is done using a single rate:

```
Donnez l'heure de début de la location (un entier) : 18
Donnez l'heure de fin de la location (un entier) : 20
Vous avez loué votre vélo pendant
2 heure(s) au tarif horaire de 1.0 franc(s)
Le montant total à payer est de 2.0 franc(s).
```

## 2 Exercise 2 – Mushroom identification

### 2.1 Introduction

The purpose of this exercise is to write a Java program that asks questions to the user in order to guess which mushroom the user has in mind (among a list known in advance).

In order to guess the user's mushroom, the program can ask **at most 3** questions[1] that only have true/false answers (the user will reply to the questions of the program with `false` for no, and `true` for yes; see the execution example which is given further below).

The 6 possible mushrooms are (French names are mandatory):

- l'agaric jaunissant;
- l'amanite tue-mouches;
- le cèpe de Bordeaux;
- le coprin chevelu
- la girolle;
- et le pied bleu.

The "cèpe de Bordeaux" is the only one to have pores ("tubes" in French), the other mushrooms having gills ("lamelles" in French).

Both "coprin chevelu" and "agaric jaunissant" grow in meadows ("dans les prés" in French), while all the other mushrooms grow in the woods ("en forêt" in French).

The only mushrooms to have a convex cap ("chapeau convexe") are the "agaric jaunissant", the "amanite tue-mouches" and the "pied bleu".

And only mushrooms to have a annulus/ring ("anneau") are the "agaric jaunissant", the "amanite tue-mouches" and the "coprin chevelu".

**Comment:** `clavier.nextBoolean()` allows to read booleans.

[1]But that does not mean that these three questions are the same each time!

## 2.2 Instructions

For this assignment, we do not impose any starting Java code. We only ask you to use the provided questions and mushroom names as provided in the source code available at the course webpage.

**WARNING**: you should modify neither the mushroom names nor the questions provided in that file.

What you have to do is to write the whole Java code (using the provided `System.out.print` statements) so that it can find, using **at most 3** questions, the mushroom the user wants it to guess (among the above listed mushrooms).

Pay attention not to modify the text of the questions (you may move them around for changing the order if necessary); our automatic grader is based on the exact text of these questions in order to evaluate the program submitted.

You are otherwise free to write your program as you wish.

The main difficulty is to find which questions must be asked and according to which order. All the orders do not lead to a good program; namely, do not allow to find the answer within at most 3 questions.

**Note:**

1. We assume that the user follows the rules. If the user's answers are incoherent or incorrect, the output of the program is unspecified, which means that it could be whatever you want.

   We won't test such silly situations. Our grader will only provide correct and coherent answers to your program.

2. Since you cannot modify the provided pieces of code (messages), be aware that your program will "speak" French. Our grader is unable to deal with translations.

## 2.3 Execution examples

Notice that the examples below are only provided to illustrate how the interaction *could* be. We do not claim that the questions asked here are neither relevant nor asked in a proper order allowing to guess within at most 3 questions!

Furthermore, we don't claim that those two examples come from the same program. They only are possible examples of how the program could interact with the user.

**Example 1:**

```
Pensez à un champignon : amanite tue mouches, pied bleu, girolle,
cèpe de Bordeaux, coprin chevelu ou agaric jaunissant.

Est-ce que votre champignon a des lamelles (true : oui, false : non) ? true
Est-ce que votre champignon a un anneau (true : oui, false : non) ? false
Est-ce que votre champignon a un chapeau convexe (true : oui, false : non) ? false
==> Le champignon auquel vous pensez est la girolle.
```

**Example 2:**

```
Pensez à un champignon : amanite tue mouches, pied bleu, girolle,
cèpe de Bordeaux, coprin chevelu ou agaric jaunissant.

Est-ce que votre champignon vit en forêt (true : oui, false : non) ? true
Est-ce que votre champignon a des lamelles (true : oui, false : non) ? false
==> Le champignon auquel vous pensez est le cèpe de Bordeaux.
```