Cours MOOC EPFL d'introduction à la programmation orientée objet, illustré en Java

Second assignment (graded) : Constructors

J. Sam & J.-C. Chappelier

This assignment consists of two exercises to submit.

1 Exercise 1 — Green mice

The goal of this exercise is to create « mice » in different ways and to make them « evolve » over time.

1.1 Description

Download the source code available at the course webpage[1] and 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 `Labo.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;

[1] `https://d396qusza40orc.cloudfront.net/intropoojava/assignments-data/Labo.java`

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 `Labo.java`) in "OUTPUT submission" (not in "Additional"!).

The provided code which is reproduced below does the following :

- « constructs » mice ;
- makes them evolve using a method `evolue` (means "evolve" in French) ;
- displays the mice before and after making them evolve.

The body of the class `Souris` (means "Mouse" in French) is missing and this is what we ask you to write.

A mouse is characterized by its weight in grams (an `int`), its color (a `String`), its age in months (an `int`), its life expectancy (an `int`) and one indication about whether it is cloned or not (a boolean).

These attributes will be named respectively : `poids`, `age`, `couleur`, `esperanceVie` and `clonee`.

The public methods of the class `Souris` are :

- constructors that conform to the provided `main` method, with the following order for the parameters : the weight, the color, the age and the life expectancy. These last two parameters have as default value zero and 36 respectively. The value 36 is stored in a provided constant that you can use by writing `ESPERANCE_VIE_DEFAUT`. The constructors display the message `Une nouvelle souris !` ; (means `A new mouse !`)

- a copy constructor which should display the message `Clonage d'une souris !`; a cloned mouse has the same characteristics as the original mouse, *except* its life expectancy which is less : the 4 fifths of the original mouse's life expectancy ;

- a method `toString()` producing a representation of a `Souris` respecting ***strictly*** the following format :
  `Une souris <couleur>[, clonee,] de <age> mois et pesant <poids> grammes` (in only one line)
  where `<age>` has to be replaced by the age of the mouse and `<poids>` by its weight. The part of the sentence « `, clonee,` » won't be displayed unless the mouse has been cloned ;

- a method `vieillir` which will increase the age of the mouse by one unit. If the mouse is cloned, it should become green if it reaches an age superior to the half of its life expectancy ; even if it is not called explicitly in the method `main()`, this method must be public ; it will be tested ;

- and a method `evolue` aging a mouse from its current age up to its life expectancy.

Use the terminal for all the required displays and terminate them with an End Of Line. An execution example is provided below.

1.2 Execution example

```
Une nouvelle souris !
Une nouvelle souris !
Clonage d'une souris !
Une souris blanche de 2 mois et pesant 50 grammes
Une souris grise de 0 mois et pesant 45 grammes
Une souris grise, clonee, de 0 mois et pesant 45 grammes
Une souris blanche de 36 mois et pesant 50 grammes
Une souris grise de 36 mois et pesant 45 grammes
Une souris verte, clonee, de 28 mois et pesant 45 grammes
```

2 Exercise 2 — Library

The purpose of this exercise is to simulate in a very basic manner the management of a library. The library contains *exemplaires* (copies) of *œuvres* (books) written

by *auteurs* (authors). It consists of modeling each one of these elements in your program.

2.1 Description

Download the source code available at the course webpage[2] and 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 `Biblio.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 `Biblio.java`) in "OUTPUT submission" (not in "Additional"!).

The provided code which is reproduced below creates authors, books of these authors, stores in the library copies of these books, then :

[2] `https://d396qusza40orc.cloudfront.net/intropoojava/assignments-data/Biblio.java`

- lists all the copies of this library ;
- lists all the copies written in english ;
- displays the names of all the awarded authors having written a book of which the library has a copy ;
- and displays the number of copies of a given book ;

A possible execution example is provided further below.

The definitions of the classes `Auteur`, `Oeuvre`, `Exemplaire` and `Bibliotheque`, described here above, are missing and you are asked to provide them.

The `Auteur` class An `Auteur` (means "Author" in French) is characterized by his name (a `String`) as well as an indication allowing to know whether he has been awarded.

The methods that are specific to this class and are part of its user interface are :

- constructors that conform to the provided `main` method, with the following order for the parameters : the name and the indication allowing to know whether the author has been awarded ;
- a method `getNom` returning the name of the author ;
- a method `getPrix` returning `true` if the author has been awarded.

The `Oeuvre` class An `Oeuvre` (means "literary work"/"book" in French) is characterized by its title (of type `String`), (a constant reference to the) author who wrote it and the language in which it was writen (of type `String`).

The methods that are specific to this class and are part of its user interface are :

- constructors that conform to the provided `main` method, with the following order for the parameters : the title, the author and the language. If the language is not provided it should take the default value `"francais"` ;
- a method `getTitre` returning the title of the book ;
- a method `getAuteur` returning the author (it is tolerated here to return directly the reference to the author) ;

- a method `getLangue` returning the language of the book ;
- and a method `afficher` displaying the characteristics of the book respecting ***strictly*** the following format :
  `<titre>, <nom de l'auteur>, en <langue>`
  where `<titre>` is to be replaced by the tile of the book, `<nom de l'auteur>`, by the name of the author and `<langue>` by its language ;

We consider here that an `Oeuvre` cannot be copied.

See the execution example provided furhter below for displaying examples.

The `Exemplaire` class The class `Exemplaire` models a copy of a book. An instance of this class is characterized by a (reference to a) book of which it is a copy.

The methods that are specific to the class `Exemplaire` and that should be part of its user interface are :

- a constructor taking as argument a reference to an `Oeuvre` and displaying a message respecting ***strictly*** the following format :
  `Nouvel exemplaire ->  <titre>, <nom de l'auteur>, en <langue>`
  followed by an End Of Line ;
- a copy constructor displaying a message respecting ***strictly*** the following format :
  `Copie d'un exemplaire de -> <titre>, <nom de l'auteur>, en <langue>`
  followed by an End Of Line ;
- a method `getOeuvre` returning the book (`Oeuvre`) ;
- and a method `afficher` displaying a description of the copy respecting ***strictly*** the following format :
  `Un exemplaire de  <titre>, <nom de l'auteur>, en <langue>`
  without an End Of Line. The display method of the class `Oeuvre` will be used in order to produce this display.

The `Bibliotheque` class A `Bibliotheque` (means "library" in French) is characterized by its name and contains a list of copies (`Exemplaire`).

The list of copies will be modeled using a dynamic table (`ArrayList`). **This attribute should be obligatory called `exemplaires`**. The methods that are specific to the class `Bibliotheque` and are part of its user interface are :

- a constructor that conforms to the provided `main` method and displays the message :
  `La bibliothèque <nom> est ouverte !` followed by an End Of Line, where `<nom>` is to be replaced by the name of the library ;

- a method `getNom` returning the name of the library ;

- a method `getNbExemplaires` returning the number of copies stored in the library;

- a method `stocker` allowing to store one or multiple copies (`Exemplaire`) of a book in the library ; it should conform to the provided `main`, with the following order for the parameters : the reference to a book and the number n of copies to add ; this method will add to the library's list of `Exemplaire`, n `Exemplaire` of the provided `Oeuvre`, which is to be constructed. If the number of copies is not provided, this means that it is 1 by default. **Attention, the copies should imperatively be added in the end of the dynamic table** (method `add` of the `ArrayList`) ;

- a method `listerExemplaires` returning in an `ArrayList` all the copies of a book written in a given language ; if no language is given (empty string), all the copies of the library will be returned; A utility method is provided which allows then to display the content of the dynamic table returned by `listerExemplaires` (see the execution example provided further below) ;

- a method `compterExemplaires` returning the number of copies of a given book passed as a parameter ;

- a method `afficherAuteur` with a parameter of type boolean or without a parameter, that displays the names of the authors whose copy is stored in the library. If the boolean is provided and is equal to `true`, only the names of authors with a prize (awarded) will be displayed ; if it is equal to `false` only the names without a prize will be displayed. If the boolean is not provided, the method will display only authors with prizes. The name of the author will be displayed as many times as there are copies written by the author.

  Use an End Of Line after the display of each name ;

A possible execution example is provided further below.

2.2 Execution example

```
La bibliotheque municipale est ouverte !
Nouvel exemplaire -> Les Miserables, Victor Hugo, en francais
Nouvel exemplaire -> Les Miserables, Victor Hugo, en francais
Nouvel exemplaire -> L'Homme qui rit, Victor Hugo, en francais
Nouvel exemplaire -> Le Comte de Monte-Cristo, Alexandre Dumas, en francais
Nouvel exemplaire -> Le Comte de Monte-Cristo, Alexandre Dumas, en francais
Nouvel exemplaire -> Le Comte de Monte-Cristo, Alexandre Dumas, en francais
Nouvel exemplaire -> Zazie dans le metro, Raymond Queneau, en francais
Nouvel exemplaire -> The count of Monte-Cristo, Alexandre Dumas, en anglais
La bibliotheque municipale offre
        Un exemplaire de Les Miserables, Victor Hugo, en francais
        Un exemplaire de Les Miserables, Victor Hugo, en francais
        Un exemplaire de L'Homme qui rit, Victor Hugo, en francais
        Un exemplaire de Le Comte de Monte-Cristo, Alexandre Dumas, en francais
        Un exemplaire de Le Comte de Monte-Cristo, Alexandre Dumas, en francais
        Un exemplaire de Le Comte de Monte-Cristo, Alexandre Dumas, en francais
        Un exemplaire de Zazie dans le metro, Raymond Queneau, en francais
        Un exemplaire de The count of Monte-Cristo, Alexandre Dumas, en anglais
Les exemplaires en anglais sont
        Un exemplaire de The count of Monte-Cristo, Alexandre Dumas, en anglais
Les auteurs a succes sont
Raymond Queneau
Il y a 3 exemplaires de  Le Comte de Monte-Cristo
```