Deuxième devoir (noté)
Constructeurs

Ce devoir comprend deux exercices à rendre.

1 Exercice 1 — Souris vertes

Le but de cet exercice est de créer des « souris » par différents biais et de les faire « évoluer » au cours du temps.

1.1 Description

Télécharger le programme fourni avec le devoir et le compléter

ATTENTION : vous ne devez modifier ni le début ni la fin du programme, juste ajouter vos propres lignes à l'endroit indiqué. Il est donc primordial de respecter la procédure suivante (les points 1 et 3 concernent spécifiquement les utilisateurs d'Eclipse) :

1. désactiver le formatage automatique dans Eclipse :
   `Window > Preferences > Java > Editor > Save Actions`
   (et décocher l'option de reformatage si elle est cochée)
2. sauvegarder le fichier téléchargé sous le nom `Labo.java` (avec une majuscule, notamment). Si vous travaillez avec Eclipse vous ferez cette sauvegarde à l'emplacement `[dossierDuProjetPourCetExercice]/src/` ;
3. rafraîchir le projet Eclipse où est stocké le fichier (clic droit sur le projet > refresh) pour qu'il le prenne en compte ;

4. écrire le code à fournir entre ces deux commentaires :

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

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

5. sauvegarder et tester son programme pour être sûr(e) qu'il fonctionne correctement, par exemple avec les valeurs données plus bas ;
6. rendre le fichier modifié (toujours `Labo.java`) dans « OUTPUT submission » (et non pas dans « Additional ! »).

Le code fourni et reproduit plus bas :

— « construit » des souris ;
— les fait évoluer au moyen d'une méthode `evolue` ;
— affiche les souris avant et après les avoir fait évoluer.

Le corps de la classe `Souris` manque et c'est ce qu'il vous est demandé d'écrire.

Une souris est caractérisée par son poids en grammes (un `int`), sa couleur (une `String`), son âge en mois (un `int`), son espérance de vie (un `int`) et une indication sur le fait qu'elle soit clonée ou pas (un booléen).

Ces attributs seront nommés respectivement : `poids`, `age`, `couleur`, `esperanceVie` et `clonee`.

Par ailleurs, les méthodes publiques de la classe `Souris` sont :

— des constructeurs conformes à la méthode `main` fournie, avec l'ordre suivant pour les paramètres : le poids, la couleur, l'âge et l'espérance de vie. Ces deux derniers paramètres ont pour valeur par défaut zéro et 36 respectivement. La valeur 36 est stockée dans une constante fournie que vous utiliserez en écrivant `ESPERANCE_VIE_DEFAUT`. Les constructeurs afficheront le message `Une nouvelle souris !;`

— un constructeur de copie qui doit afficher le message `Clonage d'une souris !;` une souris clonée a les mêmes caractéristiques que la souris d'origine, *sauf* son espérance de vie qui est moindre : les 4 cinquième de celle de la souris d'origine ;

— une méthode `toString()` produisant une représentation d'une `Souris` respectant ***strictement*** le format suivant :
`Une souris <couleur>[, clonee,] de <age> mois et pesant <poids> grammes` (sur une seule ligne)

où `<age>` est à remplacer par l'âge de la souris et `<poids>` par son poids. Le bout de phrase « `, clonee,` » ne sera affiché que si la souris a été clonée ;

— une méthode `vieillir` qui augmentera d'une unité l'âge de la souris. Si la souris est clonée, elle doit devenir verte si elle atteint un âge supérieur à la moitié de son espérance de vie ; même si elle n'est pas appelée explicitement dans la méthode `main()`, cette méthode doit être publique ; elle sera testée ;

— et une méthode `evolue` faisant vieillir la souris depuis son âge courant jusqu'à son espérance de vie.

Tous les affichages demandés se feront sur le terminal et seront terminés par un saut de ligne. Un exemple de déroulement est fourni plus bas.

1.2 Exemple de déroulement

```
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 Exercice 2 — Bibliothèque

Le but de cet exercice est de simuler de façon très basique la gestion d'une bibliothèque. La bibliothèque contient des *exemplaires* d'*œuvres* écrites par des *auteurs*. Il s'agira de modéliser chacun de ces éléments dans votre programme.

2.1 Description

Télécharger le programme fourni avec le devoir et le compléter

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

ATTENTION : vous ne devez modifier ni le début ni la fin du programme, juste ajouter vos propres lignes à l'endroit indiqué. Il est donc primordial de respecter la procédure suivante (les points 1 et 3 concernent spécifiquement les utilisateurs d'Eclipse) :

1. désactiver le formatage automatique dans Eclipse :
   `Window > Preferences > Java > Editor > Save Actions`
   (et décocher l'option de reformatage si elle est cochée)
2. sauvegarder le fichier téléchargé sous le nom `Biblio.java` (avec une majuscule, notamment). Si vous travaillez avec Eclipse vous ferez cette sauvegarde à l'emplacement `[dossierDuProjetPourCetExercice]/src/` ;
3. rafraîchir le projet Eclipse où est stocké le fichier (clic droit sur le projet > refresh) pour qu'il le prenne en compte ;
4. écrire le code à fournir entre ces deux commentaires :

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

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

5. sauvegarder et tester son programme pour être sûr(e) qu'il fonctionne correctement, par exemple avec les valeurs données plus bas ;
6. rendre le fichier modifié (toujours `Biblio.java`) dans « OUTPUT submission » (et non pas dans « Additional ! »).

Le code fourni et reproduit plus bas crée des auteurs, des œuvres de ces auteurs, stocke dans la bibliothèque des exemplaires de ces œuvres, puis :

— liste tous les exemplaires de la bibliothèque ;
— liste tous les exemplaires écrits en anglais ;
— affiche le nom de tous les auteurs à succès ayant écrit une œuvre dont la bibliothèque stocke un exemplaire ;
— et affiche le nombre d'exemplaires d'une œuvre donnée ;

Un exemple de déroulement possible est fourni plus bas.

Les définitions des classes `Auteur`, `Oeuvre`, `Exemplaire` et `Bibliotheque`, décrites ci-dessous, manquent et il vous est demandé de les fournir.

La classe `Auteur` Un auteur est caractérisé par son nom (une `String`) ainsi qu'une indication permettant de savoir s'il a été primé.

Les méthodes qui sont spécifiques à cette classe et font partie de son interface d'utilisation sont :

- des constructeurs conformes à la méthode `main` fournie, avec l'ordre suivant pour les paramètres : le nom et l'indication permettant de savoir si l'auteur a été primé ;
- une méthode `getNom` retournant le nom de l'auteur ;
- une méthode `getPrix` retournant `true` si l'auteur a été primé.

La classe `Oeuvre` Une `Oeuvre` est caractérisée par son titre (de type `String`), (une référence constante à) l'auteur qui l'a rédigée et la langue dans laquelle elle a été rédigée (de type `String`).

Les méthodes qui sont spécifiques à cette classe et font partie de son interface d'utilisation sont :

- des constructeurs conformes à la méthode `main` fournie, avec l'ordre suivant pour les paramètres : le titre, l'auteur et la langue. Si la langue n'est pas fournie elle vaudra par défaut `"francais"` ;
- une méthode `getTitre` retournant le titre de l'œuvre ;
- une méthode `getAuteur` retournant l'auteur (il est toléré ici de retourner directement la référence à l'auteur) ;
- une méthode `getLangue` retournant la langue de l'œuvre ;
- et une méthode `afficher` affichant les caractéristiques de l'œuvre en respectant ***strictement*** le format suivant :
  `<titre>, <nom de l'auteur>, en <langue>`
  où `<titre>` est à remplacer par le titre de l'œuvre, `<nom de l'auteur>`, par le nom de son auteur et `<langue>` par sa langue ;

On considère qu'une oeuvre n'est pas copiable.

Voir l'exemple de déroulement fourni plus bas pour des exemples d'affichage.

La classe `Exemplaire` La classe `Exemplaire` modélise un exemplaire d'une œuvre. Une instance de cette classe est caractérisée par (une référence à) l'œuvre dont elle constitue un exemplaire.

Les méthodes spécifiques à la classe `Exemplaire` et qui doivent faire partie de son interface d'utilisation sont :

- un constructeur prenant en argument une référence à une œuvre et affichant un message respectant ***strictement*** le format suivant :
  `Nouvel exemplaire -> <titre>, <nom de l'auteur>, en <langue>`
  suivi d'un saut de ligne ;

— un constructeur de copie affichant un message respectant ***strictement*** le format suivant :
`Copie d'un exemplaire de -> <titre>, <nom de l'auteur>, en <langue>`
suivi d'un saut de ligne ;
— une méthode `getOeuvre` retournant l'œuvre ;
— et une méthode `afficher` affichant une description de l'exemplaire respectant ***strictement*** le format suivant :
`Un exemplaire de  <titre>, <nom de l'auteur>, en <langue>`
sans saut de ligne. La méthode d'affichage de la classe `Oeuvre` sera utilisée pour réaliser cet affichage.

La classe `Bibliotheque` Une bibliothèque est caractérisée par un nom et contient une liste d'exemplaires.

La liste des exemplaires sera modélisée au moyen d'un tableau dynamique (`ArrayList`). **Cet attribut devra obligatoirement s'appeler `exemplaires`**. Les méthodes spécifiques à la classe `Bibliotheque` et qui font partie de son interface d'utilisation sont :

— un constructeur conforme à la méthode `main` fournie et affichant le message :
`La bibliothèque <nom> est ouverte !` suivi d'un saut de ligne,
où `<nom>` est à remplacer par le nom de la bibliothèque ;
— une méthode `getNom` retournant le nom de la bibliothèque ;
— une méthode `getNbExemplaires` retournant le nombre d'exemplaires contenus dans la liste ;
— une méthode `stocker` permettant d'ajouter un ou plusieurs exemplaires d'une œuvre dans la bibliothèque ; elle doit être conforme au `main` fourni, avec l'ordre suivant des paramètres : la référence à une œuvre et le nombre n d'exemplaires à ajouter ; cette méthode va ajouter à la liste d'exemplaires de la bibliothèque n exemplaires de l'œuvre fournie, qu'il s'agit de construire. Si le nombre d'exemplaires n'est pas fourni, cela signifie que sa valeur par défaut est 1. **Attention, les exemplaires devront impérativement être ajoutés à la fin du tableau dynamique** (méthode `add` des `ArrayList`) ;
— une méthode `listerExemplaires` retournant dans un `ArrayList` tous les exemplaires d'une œuvre écrite dans une langue donnée ; si aucune langue n'est donnée (chaîne vide), tous les exemplaires de la bibliothèque seront retournés ; Une méthode utilitaire est fournie qui permet ensuite d'afficher le contenu du tableau dynamique retourné par `listerExemplaires` (voir l'exemple de déroulement fourni plus bas) ;
— une méthode `compterExemplaires` retournant le nombre d'exem-

plaires d'une œuvre donnée passée en paramètre ;

— une méthode `afficherAuteur` avec un paramètre de type booléen ou sans paramètre, qui affiche les noms des auteurs dont un exemplaire est stocké dans la bibliothèque. Si le booléen est fourni et qu'il vaut `true`, seuls s'afficheront les noms des auteurs avec un prix ; s'il vaut `false` seuls les auteurs sans prix s'afficheront. Si le booléen n'est pas fourni, la méthode n'affichera que les noms des auteurs à prix. Les noms appararaîtront autant de fois qu'il y a d'exemplaires d'oeuvres écrites par l'auteur. Un saut de ligne sera fait après l'affichage de chaque nom ;

Un exemple de déroulement possible est fourni plus bas.

2.2 Exemple de déroulement

```
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
```