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

Second devoir
Constructeurs

J.-C. Chappelier & J. Sam

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 `labo.cc` fourni et le compléter suivant les instructions données ci-dessous.

ATTENTION : vous ne devez en aucun cas modifier ni le début ni la fin du programme fourni, juste ajouter vos propres lignes à l'endroit indiqué. Il est donc impératif de respecter la procédure suivante :

1. sauvegarder le fichier téléchargé sous le nom `labo.cc` ou `labo.cpp` ;
2. écrire le code à fournir (voir ci-dessous) entre ces deux commentaires :

```
/*****************************************
 * Complétez le programme à partir d'ici.
 *****************************************/

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

3. sauvegarder et tester son programme pour être sûr(e) qu'il fonctionne correctement, par exemple avec les valeurs utilisées dans l'exemple de déroulement donné plus bas ;

4. soumettre le fichier modifié (toujours `labo.cc` ou `labo.cpp`) dans « My submission » puis « Create submission ».

Le code fourni :

— « 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 `double` qui devra s'appeler `poids`), sa couleur (une `string` appelée `couleur`), son âge (un `unsigned int` appelé `age`), son espérance de vie (un `unsigned int` appelé `esperance_vie`) et une indication sur le fait qu'elle soit clonée ou pas (un booléen appelé `clonee`). Par ailleurs, les méthodes publiques de la classe `Souris` sont :

— des constructeurs conformes au `main` fourni, 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. Ces 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 cinquièmes de celle de la souris d'origine ;

— un destructeur qui affichera le message `Fin d'une souris...` ;

— une méthode `afficher()` permettant d'afficher sur le terminal les caractéristiques de la souris en 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 le `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
Fin d'une souris...
Fin d'une souris...
Fin d'une souris...
```

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 `biblio.cc` fourni et le compléter suivant les instructions données ci-dessous.

ATTENTION : vous ne devez en aucun cas modifier ni le début ni la fin du programme fourni, juste ajouter vos propres lignes à l'endroit indiqué. Il est donc impératif de respecter la procédure suivante :

1. sauvegarder le fichier téléchargé sous le nom `biblio.cc` ou `biblio.cpp` ;
2. écrire le code à fournir (voir ci-dessous) entre ces deux commentaires :

```
/*****************************************
 * Complétez le programme à partir d'ici.
 *****************************************/

/*****************************************
```

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

3. sauvegarder et tester son programme pour être sûr(e) qu'il fonctionne correctement, par exemple avec les valeurs utilisées dans l'exemple de déroulement donné plus bas ;
4. soumettre le fichier modifié (toujours `biblio.cc` ou `biblio.cpp`) dans « My submission » puis « Create submission ».

Le code fourni 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 au `main` fourni, avec l'ordre suivant pour les paramètres : le nom et l'indication permettant de savoir si l'auteur a été primé. Par défaut un auteur n'est pas primé ;
— une méthode `getNom` retournant le nom de l'auteur ;
— une méthode `getPrix` retournant `true` si l'auteur a été primé.

Par ailleurs, il ne devra pas être possible de copier un `Auteur`.

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 au `main` fourni, avec l'ordre suivant pour les paramètres : le titre, la référence à l'auteur et la langue ;
— une méthode `getTitre` retournant le titre de l'œuvre ;

— une méthode `getAuteur` retournant *une référence constante* sur l'auteur (veillez à bien respecter ce type) ;
— une méthode `getLangue` retournant la langue de l'œuvre ;
— et une méthode `affiche` 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 ;
— un destructeur affichant un message respectant ***strictement*** le format suivant :
`L'oeuvre "<titre>, <nom de l'auteur>, en <langue>" n'est plus disponible.`

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

Par ailleurs, il ne devra pas être possible de copier une `Oeuvre`.

La classe `Exemplaire` La classe `Exemplaire` modélise les exemplaires d'un 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 de : <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>`
sur une seule ligne terminée par un saut de ligne ;
— d'un destructeur affichant un message respectant ***strictement*** le format suivant :
`Un exemplaire de "<titre>, <nom de l'auteur>, en <langue>" a été jeté !`
sur une seule ligne terminée par un saut de ligne ;
— une méthode `getOeuvre` retournant une *référence constante* à l'œuvre ;
— et une méthode `affiche` affichant une description de l'exemplaire respectant ***strictement*** le format suivant :
`Exemplaire de : <titre>, <nom de l'auteur>, en <langue>`
sans saut de ligne.

La classe `Bibliotheque` Une bibliothèque est caractérisée par un nom et contient un ensemble de pointeurs sur des exemplaires. L'ensemble sera modélisé au moyen d'un `vector`.

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

- un constructeur conforme au `main` fourni 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 `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, dont la valeur par défaut est 1 ; cette méthode va ajouter à l'ensemble d'exemplaires de la bibliothèque n exemplaires de l'œuvre fournie, dynamiquement alloués ; les nouveaux exemplaires devront impérativement être ajoutés à la *fin* du tableau dynamique ;
- une méthode `lister_exemplaires` affichant 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 affichés ;
  les exemplaires devront être affichés au moyen de la méthode d'affichage qui leur est spécifique et il y aura un saut de ligne à la fin de l'affichage de chaque exemplaire (voir l'exemple de déroulement fourni plus bas) ;
- une méthode `compter_exemplaires` retournant le nombre d'exemplaires d'une œuvre donnée passée en paramètre ;
- une méthode `afficher_auteurs` prenant en paramètre un booléen (valant par défaut `false`) indiquant si l'on veut afficher uniquement les auteurs à prix ;
  cette méthode affichera les noms des auteurs dont un exemplaire est stocké dans la bibliothèque ; si le booléen vaut `true`, seuls s'afficheront les noms des auteurs avec un prix ; un saut de ligne sera fait après l'affichage de chaque nom ; le nom d'un auteur sera répété autant de fois qu'il y a d'exemplaires écrits par cet auteur.
- un destructeur qui affiche le message suivant :
  `La bibliothèque <nom> ferme ses portes,`
  `et détruit ses exemplaires :`
  où `<nom>` est le nom de la bibliothèque, puis libère les zones mémoires liées à ses exemplaires.

2.2 Exemple de déroulement

```
La bibliothèque municipale est ouverte !
Nouvel exemplaire de : Les Misérables, Victor Hugo, en français
Nouvel exemplaire de : Les Misérables, Victor Hugo, en français
Nouvel exemplaire de : L'Homme qui rit, Victor Hugo, en français
Nouvel exemplaire de : Le Comte de Monte-Cristo, Alexandre Dumas, en français
Nouvel exemplaire de : Le Comte de Monte-Cristo, Alexandre Dumas, en français
Nouvel exemplaire de : Le Comte de Monte-Cristo, Alexandre Dumas, en français
Nouvel exemplaire de : Zazie dans le métro, Raymond Queneau, en français
Nouvel exemplaire de : The Count of Monte-Cristo, Alexandre Dumas, en anglais
La bibliothèque municipale offre les exemplaires suivants :
Exemplaire de : Les Misérables, Victor Hugo, en français
Exemplaire de : Les Misérables, Victor Hugo, en français
Exemplaire de : L'Homme qui rit, Victor Hugo, en français
Exemplaire de : Le Comte de Monte-Cristo, Alexandre Dumas, en français
Exemplaire de : Le Comte de Monte-Cristo, Alexandre Dumas, en français
Exemplaire de : Le Comte de Monte-Cristo, Alexandre Dumas, en français
Exemplaire de : Zazie dans le métro, Raymond Queneau, en français
Exemplaire de : The Count of Monte-Cristo, Alexandre Dumas, en anglais
 Les exemplaires en anglais sont :
Exemplaire de : The Count of Monte-Cristo, Alexandre Dumas, en anglais
 Les auteurs à succès sont :
Raymond Queneau
 Il y a 3 exemplaires de Le Comte de Monte-Cristo
La bibliothèque municipale ferme ses portes,
et détruit ses exemplaires :
Un exemplaire de "Les Misérables, Victor Hugo, en français" a été jeté !
Un exemplaire de "Les Misérables, Victor Hugo, en français" a été jeté !
Un exemplaire de "L'Homme qui rit, Victor Hugo, en français" a été jeté !
Un exemplaire de "Le Comte de Monte-Cristo, Alexandre Dumas, en français" a été jeté !
Un exemplaire de "Le Comte de Monte-Cristo, Alexandre Dumas, en français" a été jeté !
Un exemplaire de "Le Comte de Monte-Cristo, Alexandre Dumas, en français" a été jeté !
Un exemplaire de "Zazie dans le métro, Raymond Queneau, en français" a été jeté !
Un exemplaire de "The Count of Monte-Cristo, Alexandre Dumas, en anglais" a été jeté !
L'oeuvre "The Count of Monte-Cristo, Alexandre Dumas, en anglais" n'est plus disponible.
L'oeuvre "Zazie dans le métro, Raymond Queneau, en français" n'est plus disponible.
L'oeuvre "Le Comte de Monte-Cristo, Alexandre Dumas, en français" n'est plus disponible.
L'oeuvre "L'Homme qui rit, Victor Hugo, en français" n'est plus disponible.
L'oeuvre "Les Misérables, Victor Hugo, en français" n'est plus disponible.
```