INF1010

Programmation Orientée-Objet

Travail pratique #1

Allocation dynamique, composition et agrégation

Objectifs :	Permettre à l'étudiant de se familiariser avec les notions de base de la programmation orientée objet, l'allocation dynamique de la mémoire, le passage de paramètres, les méthodes constantes et les principes de relation de composition et d'agrégation
Remise du travail :	Dimanche 31 janvier 2016, 23h
Références :	Notes de cours sur Moodle & Chapitre 2-7 du livre Big C++ 2e éd.
Documents à remettre :	Les fichiers .cpp et .h complétés réunis sous la forme d'une archive au format .zip.
Directives :	Directives de remise des Travaux pratiques sur Moodle Les en-têtes (fichiers, fonctions) et les commentaires sont obligatoires. Les travaux dirigés s'effectuent obligatoirement en équipe de deux personnes faisant partie du même groupe. Veuillez suivre le guide de codage

Informations préalables :

La directive de précompilation « #ifndef »

La directive de précompilation « #ifndef » signifie « if not defined » (si non défini). Comme le type de directive le laisse deviner, cette directive est évaluée avant la phase de compilation du code source. Dans les travaux pratiques, vous l'utiliserez dans les fichiers d'en-têtes (.h), pour éviter la double inclusion. Un fichier peut inclure deux fichiers d'entête, par exemple prenons deux fichiers a.h et b.h. Il se peut que a.h soit inclus dans le fichier b.h. On se retrouve alors à inclure deux fois le fichier a.h, ce qui entrainerait une erreur de compilation, car on ne peut définir deux fois la même classe. La directive « #ifndef » nous évite donc cette double inclusion. Pour utiliser la directive « #ifndef », il faut respecter la syntaxe suivante :

```
#ifndef NOMCLASSE_H
#define NOMCLASSE_H
// Définir la classe NomClasse ici
#endif
```

La directive de précompilation « #include»

La directive de précompilation pour l'inclusion de fichiers « #include »

1. #include <nom_fichier>
2. #include "nom_fichier"

Ce qui différencie ces deux expressions est l'emplacement où le fichier spécifié est recherché. Pour la seconde forme, le précompilateur commence tout d'abord par rechercher dans le même répertoire que le fichier compilé. Par la suite, il procède de la même manière que la première forme, c'est-à-dire dans des répertoires prédéfinit par l'environnement de développement intégré.

En résumé, lorsqu'on inclut un fichier source qui se trouve dans le projet, on utilise la seconde forme. Et lorsque l'on inclut un fichier qui provient d'une bibliothèque externe au projet, on utilise la première forme.

Travail à réaliser

Le travail consiste à implémenter un programme pouvant servir de base pour la réalisation d'un système de gestion d'inventaire d'une bibliothèque qui sera développé tout au cours de la session.

Vous devez réaliser la définition et l'implémentation des classes *Abonne*, *Livre*, *Emprunt*, *Bibliotheque*.
Pour vous aider, le fichier d'en-tête (sans les commentaires) de la classe Abonne vous est fourni. Vous devrez implémenter les méthodes du fichier source (.cpp) correspondant ainsi que les classes complètes manquantes.

Pour chaque attribut, vous devrez déterminer vous-même s'il s'agit d'une agrégation ou d'une composition puis faire l'implémentation de la manière appropriée.

Classe Abonne

Créer une classe *Abonne* qui représente un abonné de la bibliothèque. Cette classe contient les attributs suivants :

- Matricule (string)
- Prénom (string)
- Nom (string)
- Age (entier)

Les méthodes suivantes doivent être implémentées :

- Un constructeur par défaut qui initialise les trois premiers attributs à la chaîne de caractères vide et l'âge à zéro.
- Un constructeur par paramètres qui initialise les attributs selon les paramètres. Quatre paramètres - un pour chaque attribut.
- Un destructeur.
- Les méthodes d'accès et de modification des attributs.
- Une méthode d'affichage des informations qui concernent un abonné, tel que présenté dans l'exemple à la fin du document.

Classe Livre

Créer une classe *Livre* qui représente un livre dans la bibliothèque. Cette classe contient les attributs suivants :

- Cote (string)
- Titre (string)
- Année (entier)
- Age minimal requis pour le lecteur (entier)
- Nombre d'exemplaires possédés (entier)
- Nombre d'exemplaires disponibles (entier)

Les méthodes suivantes doivent être implémentées :

- Un constructeur par défaut qui initialise les deux premiers attributs à la chaîne de caractères vide et l'année, l'âge le nombre d'exemplaires possédés et le nombre d'exemplaires disponibles à zéro.
- Un constructeur par paramètres qui initialise les cinq premiers attributs selon les paramètres (tous les exemplaires sont disponibles lors de la création d'une instance de l'objet).
- Un destructeur.
- Les méthodes d'accès et de modification des attributs.
- Une méthode d'affichage des informations qui concernent un livre, tel que présenté dans l'exemple à la fin du document.

Classe Emprunt

La classe *Emprunt* sert à représenter l'emprunt d'un livre par un abonné. Cette classe contient les attributs suivants :

- Pointeur d'abonné.
- Pointeur de livre.
- Date prévue de retour (entier).
  Le format est « AAMMJJ ». Par exemple, l'entier « 160213 » représente « 13 février 2016 ».

Les méthodes suivantes doivent être implémentées :

- Un constructeur par paramètres avec les paramètres abonne (Abonne*), livre (Livre*) et date (entier).
- Un destructeur.
- Les méthodes d'accès et de modification des variables membres.
- Une méthode d'affichage des informations qui concernent un emprunt, tel que présenté dans l'exemple à la fin du document.

Classe Bibliotheque

La classe *Bibliotheque* est celle qui fait le lien entre toutes les classes précédentes. Cette classe contient les attributs suivants :

- Tableau de pointeurs d'abonnés: maximum de 100 abonnés.
- Nombre d'abonnés (entier).
- Tableau de pointeurs de livres: maximum de 1000 livres.
- Nombre de livres (entier).
- Tableau de pointeurs d'emprunts : Maximum de 200 emprunts.
- Nombre d'emprunts (entier).

Note : Tous les tableaux de pointeurs doivent être alloués dynamiquement.

Les méthodes suivantes doivent être implémentées :

- Un constructeur par défaut qui initialise les tableaux avec les dimensions indiquées et initialise les pointeurs à null.
- Un destructeur.
- La méthode « ajouterAbonne() » qui permet d'ajouter l'abonné reçu en paramètre.
- La méthode « retirerAbonne() » qui permet de retirer l'abonné en utilisant le matricule reçu en paramètre.
- La méthode « ajouterLivre() » qui permet d'ajouter le livre reçu en paramètre.
- La méthode « retirerLivre() » qui permet de retirer le livre en utilisant la cote reçue en paramètre.
- La méthode « rechercherTitre() » qui prend en paramètre une chaine de caractères (string) et affiche les titres de tous les livres de la bibliothèque qui contiennent cette chaine dans leur titre.
  Si la recherche est infructueuse , un message clair s'affiche.
  Aide : http://www.cplusplus.com/reference/string/string/find/ .
- La méthode « rechercherCote() » qui prend en paramètre une cote de livre (string). La méthode affiche les informations du livre correspondant s'il existe (nom, année, nombre d'exemplaires possédés et disponibles).
  Si la recherche est infructueuse, un message clair s'affiche.
- La méthode « emprunter() » qui prend en paramètres le matricule d'un abonné, la cote d'un livre et la date de retour. Cette méthode vérifie s'il est possible pour l'abonné d'emprunter le livre en question, dans quel cas un nouvel emprunt est ajouté au tableau d'emprunts. Il est possible pour un abonné d'emprunter le livre si ce dernier est disponible, qu'il a l'âge minimal requis, qu'il n'a pas déjà emprunté ce livre et qu'il n'a pas atteint le nombre maximal d'emprunts (2 par abonné).
  Cette méthode doit retourner une valeur booléene indiquant si l'emprunt a été fait ou non.
  Note : N'oubliez pas de considérer l'attribut du nombre d'exemplaires disponible de la classe *Livre*.
- La méthode « retourner() » qui prend en paramètres le matricule d'un abonné et la cote d'un livre. Si l'abonné avait bien emprunté ce livre, l'emprunt en question est retiré du tableau d'emprunts. Sinon, aucune action n'est faite
  Cette méthode doit retourner une valeur booléene indiquant si le retour a été fait ou non.
- La méthode « infoAbonne() » prend en paramètre un matricule d'abonné et affiche les informations qui le concerne (nom, prénom, emprunts).

Main.cpp

Le programme principal contient des directives à suivre pour instancier différents objets et essayer les différentes méthodes implémentées.

Le résultat final devrait être similaire à ce qui suit :

```
CREATION ET AFFICHAGE DES ABONNES
John, Doe. 23 ans. #1839456
Nicolas, Gagnon. 8 ans. #1630236
Martin, Tremblay. 18 ans. #1269348

CREATION ET AFFICHAGE DES LIVRES
GA403. Big C++. 2009. 8 ans et plus.
QA203. Calcul a plusieurs variables partie 1. 2011. 3 ans et plus.
QA204. Calcul a plusieurs variables partie 2. 2011. 3 ans et plus.
AC409. Le chateau d'Ortrante. 1764. 16 ans et plus.

Ajout des livres et abonnes a la bibliotheque

RECHERCHE PAR TITRE...: Calcul
QA203. Calcul a plusieurs variables partie 1. 2011. 3 ans et plus.
QA204. Calcul a plusieurs variables partie 2. 2011. 3 ans et plus.

RECHERCHE PAR cote...: AC409
AC409. Le chateau d'Ortrante. 1764. 16 ans et plus.

TESTS D'EMPRUNTS
Echec emprunt
BD302 emprunte par 1630236
Echec emprunt
Echec emprunt
QA204 emprunte par 1630236
Echec emprunt

INFO ABONNE AVANT RETOUR
Informations sur l'abonne:
Nicolas, Gagnon. 8 ans. #1630236
Emprunt de l'usager #1630236. Livre BD302. Retour prevu le 160204.
Emprunt de l'usager #1630236. Livre QA204. Retour prevu le 160204.

TESTS RETOUR LIVRE
BD302 remis par 1630236
Echec remise

INFO ABONNE APRES RETOUR
Informations sur l'abonne:
Nicolas, Gagnon. 8 ans. #1630236
Emprunt de l'usager #1630236. Livre QA204. Retour prevu le 160204.
Press any key to continue . . .
```

Question

Quel est le type de relation (composition / agrégation) que vous avez choisi entre la classe *Bibliotheque* et chacune des classes *Abonne*, *Livre* et *Emprunt* ? Justifiez.
Remarque : L'utilisation de pointeurs ne veut pas toujours dire qu'il s'agit d'agrégation.

Correction

La correction du TP1 se fera sur 20 points. Voici les détails de la correction:

- (03 points) Compilation du programme;
- (03 points) Exécution du programme;
- (04 points) Comportement exact des méthodes du programme;
- (02 points) Documentation du code et bonne norme de codage;
- (02 points) Utilisation correcte du mot-clé *const* et dans les endroits appropriés;
- (02 points) Utilisation adéquate des directives de précompilation;
- (02 points) Allocation et désallocation appropriée de la mémoire;
- (02 points) Réponse à la question;