INF1010
Programmation orientée-objet

Travail pratique 1
Objets, allocation dynamique, pointeurs intelligents, composition et agrégation

Département de génie informatique et logiciel

Polytechnique Montréal
15 janvier 2020

POLYTECHNIQUE MONTRÉAL

Objectifs

Permettre à l'étudiant de se familiariser avec les notions de base de la programmation orientée objet, l'allocation dynamique, les pointeurs intelligents ainsi que les principes de relation de composition et d'agrégation.

Remise du travail

- Date :**28 janvier 2020 à 23h55**
- **Une note de 0 sera attribuée aux équipes qui remettent leur travail en retard.**
- Remettre uniquement **tous** les fichiers .cpp et .h sous une archive **.zip**
- Nom du fichier zip : **matricule1_matricule2_groupe.zip** où matricule1 < matricule2. Exemple : 1234566_1234567_1.zip

Références

- Notes de cours sur Moodle
- Livre Big C++ deuxième édition
- https://en.cppreference.com/w/

Directives

- Les travaux s'effectuent obligatoirement en équipe de deux personnes faisant partie du même groupe.
- Les entêtes de fichiers sont obligatoires.
- Les fonctions doivent être documentées.
- **Le guide de codage sur Moodle doit être suivi.**

Conseils

- Lisez tout le document avant de commencer!
- Ayez lu vos notes de cours!

Faites attention si vous partagez votre travail, car le plagiat sera pénalisé par **une note de 0**.

Venez poser vos questions sur Slack! Vous devez utiliser votre adresse @polymtl.ca.
Lien pour faire son compte : https://join.slack.com/t/inf1010-h20/signup
Lien du workspace : https://inf1010-h20.slack.com

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 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 »

- `#include <nom_fichier>`
- `#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 prédéfinis 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.

Mise en contexte

L'entreprise CinéPoly est une nouvelle compagnie qui tente de faire sa place dans le monde des médias numériques, faisant compétition à Netflix, Amazon Prime, Crave et Disney+. Malheureusement, les programmeurs travaillant sur le projet n'ont pas les connaissances requises afin de mener le projet à bien. C'est pourquoi CinéPoly fait recours à vous et vos talents exceptionnels de programmeur.

Travail à réaliser

On vous demande de compléter les fichiers qui vous sont fournis pour pouvoir implémenter le système décrit ci-dessous.

Classe Auteur

Cette classe représente un auteur.

Cette classe contient les attributs suivants :

- `nom_ (string)`
- `anneeDeNaissance_ (unsigned int)`
- `nbFilms (unsigned int)`

Les méthodes suivantes doivent être implémentées :

- Le constructeur par paramètre.
- Les méthodes d'accès.
- `setNbFilms(unsigned int)`

Classe GestionnaireAuteurs

Cette classe gère les auteurs.

Cette classe contient les attributs suivants :

- `NB_AUTEURS_MAX (static constexpr size_t)` : Le nombre maximal d'auteurs dans la liste.
- `auteurs_ (Auteur[])` : Tableau de taille fixe contenant les auteurs. Ceci est une relation de composition avec la classe
- `nbAuteurs_ (std::size_t)` : Le nombre d'auteurs dans la liste.

Les méthodes suivantes doivent être implémentées :

- Le constructeur par défaut
  - Faire attention de bien initialiser toutes la variables membres.
- `ajouterAuteur(const Auteur&)`
  - Cette méthode doit ajouter un auteur à la liste de taille fixe des auteurs.
  - Cette fonction retourne un booléen représentant le succès de l'opération
- chercherAuteur(const string&)
  - Cette méthode cherche dans la liste des auteurs si un auteur du nom envoyé en paramètre existe.
  - Retourner un pointeur vers un auteur du même nom.
  - Si aucun auteur ne porte le nom envoyé par paramètre, retourner un pointeur nul.
- `chargerDepuisFichier(const std ::string&)`
  - Cette méthode doit ouvrir le fichier, et envoyer chaque ligne du fichier à la fonction `lireLigneAuteur`. Si une erreur se produit dans `lireLigneAuteur`, interrompre la fonction et retourner `false`. Retourner `true` aucune erreur ne s'est produite.
- `lireLigneAuteur(const std::string&)`
  - Cette méthode privée doit ajouter un auteur à partir d'un string.
  - Le format du string est
    `"NOM DE L'AUTEUR" ANNÉE_DE_NAISSANCE`.
    Utiliser `std::quoted` pour extraire le nom de l'auteur.
  - Cette fonction retourne un booléen représentant si l'opération est un succès.
- Les méthodes d'accès.

Classe Film

Cette classe représente un film. Un film peut être inaccessible dans certains pays et peut aussi être restreint à un âge minimum. Il est important de remarquer que cette classe n'a pas besoin d'un destructeur malgré le fait qu'il y ait de l'allocation dynamique ainsi que de comprendre pourquoi.

Cette classe contient les attributs suivants :

- `nom_ (string)`
- `anneeDeSortie_ (unsigned int)`
- `genre_ (Genre)` : Genre est un `enum` dans la classe `Film`.
- `pays_ (Pays)` : Pays est un `enum` qui se retrouve dans le fichier `Pays.h`.
- `estRestreintParAge_ (bool)` : représente si un utilisateur doit avoir au moins 16 ans pour regarder le film.

- `auteur_ (Auteur*)` : Pointeur vers l'auteur du film. Ceci est une relation d'agrégation.
- `paysRestreints_ (std::unique_ptr<Pays[]>)` : Liste de taille dynamique de tous les pays. Il est important de comprendre pourquoi pays utilise un `std::unique_ptr` et l'impact sur la gestion de la mémoire.
- `nbPaysRestreints_ (std::size_t)` : Le nombre de pays qui sont contenus dans la liste des pays restreints.
- `capacitePaysRestreints_ (std::size_t)` : La capacité (mémoire allouée) de la liste des pays restreints.

Les méthodes suivantes doivent être implémentées :

- Le constructeur par paramètre
  - Dans le constructeur, il faut allouer dynamiquement de l'espace pour le tableau `paysRestreints_` de taille `CAPACITE_PAYS_INITIALE` (constante statique qui représente la capacité initiale du tableau) en utilisant `std::make_unique` (interdiction d'utiliser new).
  - Faire attention de bien initialiser toutes la variables membres.
- `ajouterPaysRestreint(Pays)`
  - Cette méthode doit s'assurer d'avoir assez de place pour ajouter un pays à la liste des pays restreints.
  - S'il n'y a plus de place, un nouvel espace mémoire du double de la taille actuelle doit être alloué (utiliser `AUGMENTATION_CAPACITE_PAYS` pour éviter les nombres magiques).
  - Utiliser les fonctions `std::make_unique` ainsi que `std::move`.
  - On doit toujours pouvoir ajouter un pays.
- `supprimerPaysRestreints()`
  - Fait simplement mettre le nombre de pays à 0.
- `estRestreintDansPays(Pays)`
  - Cette méthode effectue une recherche dans le tableau des pays restreints et retourne `true` si le pays en paramètre s'y retrouve.
- Les méthodes d'accès.

Classe Librairie

Cette classe gère les films offerts par CinéPoly. Il est important de comprendre pourquoi cette classe a besoin d'un destructeur, contrairement à la classe Film.

Cette classe contient les attributs suivants :

- `films_ (Film**)` : Un tableau dynamique de films alloués dynamiquement.

- `nbFilms_ (std::size_t)`
- `capaciteFilms_ (std::size_t)`

Les méthodes suivantes doivent être implémentées :

- Le constructeur par défaut
  - Faire attention de bien initialiser toutes la variables membres.
- Le destructeur
  - Faire appel à fonction `supprimerFilms()`.
  - Ne pas oublier d'utiliser `delete[]`.
- `ajouterFilm(Film*)`
  - Cette méthode devient propriétaire du film passé en paramètre (composition).
  - Cette méthode doit s'assurer d'avoir assez de place pour ajouter un pays à la liste des pays restreints. S'il n'y a plus de place, un nouvel espace mémoire du double de la taille actuelle doit être alloué.
    - Utiliser `AUGMENTATION_CAPACITE_FILMS` pour éviter les nombres magiques.
  - Utilisez les opérateurs `new[]` ainsi que `delete[]`. Remarquez la différence entre cette méthode et la méthode `ajouterPays(Pays)` de la classe `Film`.
- `trouverIndexFilm(const std::string&)`
  - Fonction membre privée qui cherche un film comportant le nom envoyé en paramètre.
  - Retourner l'index du premier film comportant le nom.
  - Retourner -1 (`FILM_INEXSISTANT`) si le nom du film ne s'y trouve pas.
- `retirerFilm(const Film&)`
  - Cette méthode supprime la première occurrence du film envoyé en paramètre.
- Utiliser la fonction `trouverIndexFilm(const std::string&)` pour supprimer le film approprié. Ne rien faire si le film n'existe pas.
- Il ne doit pas y avoir de trou au milieu du tableau après avoir retiré un film.
  - Il n'est pas nécessaire de garder l'ordre du tableau.
- `chercherFilm(const std::string&)`
  - Cette méthode retourne un pointeur vers le film qui comporte le même nom que celui envoyé en paramètre.
  - Retourner un pointeur nul si le film n'existe pas.
- `chargerFilmDepuisFichier(const string&, GestionnaireAuteurs&)`
  - Cette méthode doit récupérer chaque ligne du fichier, puis les envoyer une à une à la fonction `lireLigneFilm`. Interrompre la fonction et retourner `false` si `lireLigneFilm` retourne `false`. Cette méthode doit s'assurer de supprimer les vieux films avant de charger les films du fichier.
- `chargerRestrictionsDepuisFichier(const string&)`

  - Cette méthode doit récupérer chaque ligne du fichier, puis les envoyer une à une à la fonction `lireLigneRestrictions`. Interrompre la fonction et retourner `false` si `lireLigneRestrictions` retourne `false`. Cette méthode doit s'assurer de supprimer les vielles restrictions de tous les films avant de commencer à les charger du fichier.
- `lireLigneFilm(const string&, const GestionnaireAuteurs&)`
  - Le string représente un film et est composé ainsi :
    `"NOM_DU_FILM" ANNEE_DE_SORTIE GENRE PAYS BOOL_RESTREINT_PAR_AGE "NOM AUTEUR"`
    Où chaque attribut est séparé d'un espace. Le nom du film ne contient pas d'espaces. Utiliser `std::quoted` pour extraire le nom du film et le nom de l'auteur.
  - Il est important de noter que `GENRE` ainsi que `PAYS` sont des valeurs numériques qui représentent leur valeur dans leur `enum` respectif.
    - Vous pouvez initialiser le pays ainsi : `to_enum<Pays>(int)`.
    - Vous pouvez initialiser le genre ainsi : `to_enum<Film::Genre>(int)`.
  - Si l'auteur n'existe pas, simplement retourner `false` pour indiquer qu'il y a eu une erreur. La méthode doit retourner `true` seulement si aucune erreur n'est survenue.
- `lireLigneRestrictions(const std::string&)`
  - Le string représente les restrictions d'un pays et est composé ainsi :
    `"NOM DU FILM" PAYS1 PAY2 PAYS3 ...`
    Le pays est la valeur numérique de son `enum`. Utiliser `std::quoted` pour extraire le nom du film.
    - Vous pouvez initialiser le pays ainsi : `to_enum<Pays>(int)`.
  - **Le nombre de pays restreints par film n'est pas connu à l'avance!**
  - Il faut s'assurer que le film existe, puis ajouter les restrictions en utilisant la méthode `ajouterPaysRestreint(Pays)` qui se trouve dans l'objet `Film`.
  - Si le film n'existe pas, simplement retourner `false`. Retourner `true` si aucune erreur n'est survenue.
  - Cette méthode est appelée par `chargerRestrictionsDepuisFichiers(const std::string&)`
- `supprimerFilms()`
  - Cette méthode privée fait simplement supprimer les films un par un et assigne la valeur `0` à la variable `nbFilms_`.
  - Ne pas changer la capacité du tableau (donc ne pas utiliser l'opérateur `delete[]`)
  - Cette fonction est utilisée par le destructeur de la classe ainsi que par la fonction de chargement des films depuis un fichier.

Classe Utilisateur

Cette classe représente un utilisateur de CinéPoly.

Cette classe contient les attributs suivants :

- `nom_ (std::string)`
- `age_ (unsigned int)`
- `nbFilmsVus_ (unsigned int)`
- `estPremium_ (std::size_t)` : Cette variable membre représente si l'utilisateur est un utilisateur payant. Un utilisateur payant n'a pas de restrictions sur le nombre de films qu'il peut regarder.
- `pays_ (Pays)` : Le pays de l'utilisateur. Utile pour savoir si un utilisateur peut regarder un film en fonctions des restrictions des films.

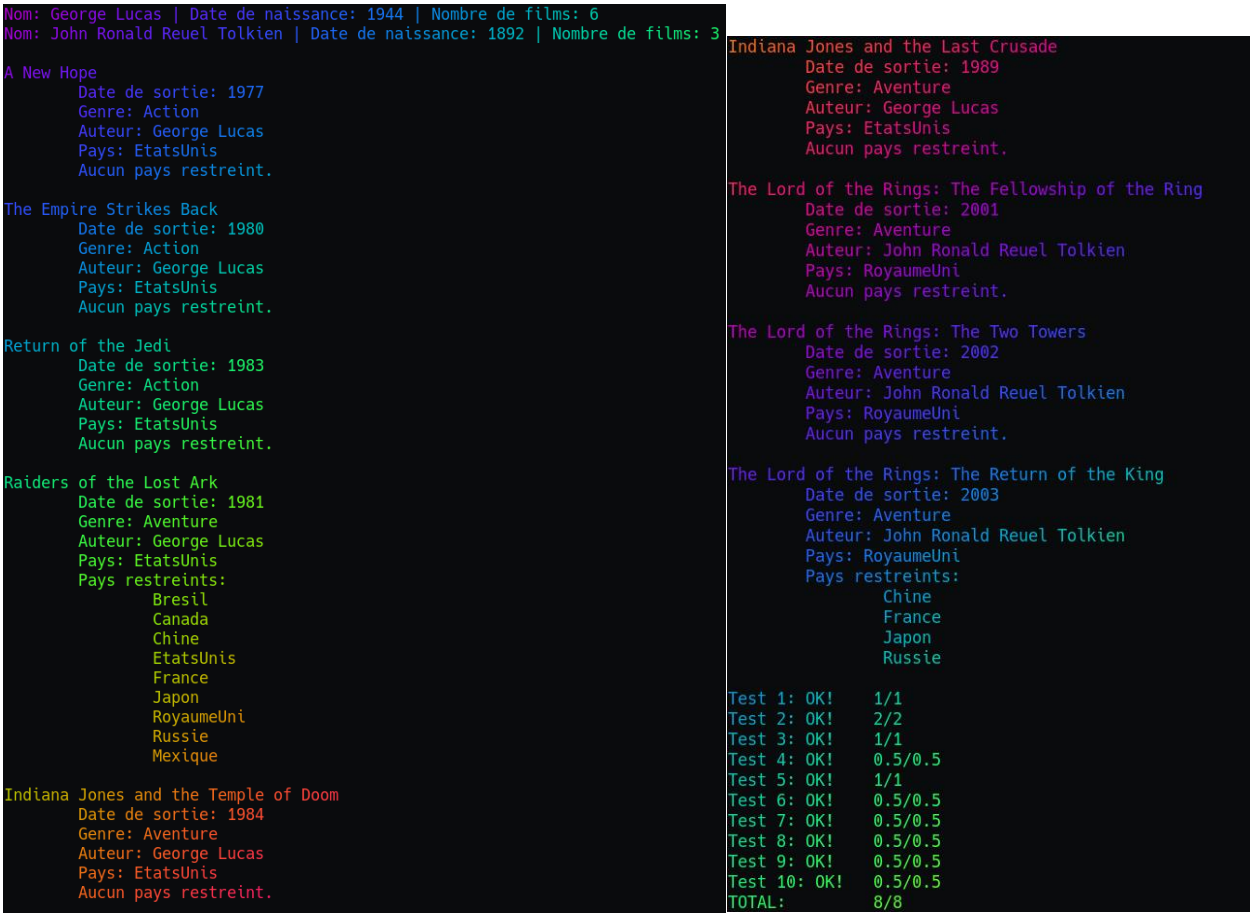
Les méthodes suivantes doivent être implémentées :

- Le constructeur par paramètre
  - Faire attention de bien initialiser toutes la variables membres.
- `filmEstDisponnible(const Film&)`
  - Cette méthode vérifie un film est disponible à un utilisateur. Deux conditions doivent être remplies pour qu'un film soit disponible à un utilisateur :
    - Le film n'est pas restreint dans le pays de l'utilisateur.
    - Si le film a une restriction d'âge, l'utilisateur doit avoir plus de 16 ans (`AGE_MINIMUM_FILMS_RESTREINTS`).
- `nbLimiteFilmsAtteint()`
  - Cette méthode retourne si l'utilisateur a atteint le nombre maximum de films qu'il peut regarder. Si l'utilisateur n'est pas premium et qu'il a regardé plus de 3 films (`NB_FILMS_GRATUITS`), la fonction doit retourner `true`.
- `regarderFilm(const Film&)`
  - Cette méthode doit vérifier que le film est disponible à l'utilisateur ET que l'utilisateur pas atteint le nombre de films maximum qu'il peut regarder.
    - Si les deux conditions sont remplies, incrémenter le nombre de film vus par l'utilisateur puis retourner `true`.
    - Sinon, retourner `false`.
- Les méthodes d'accès.

main.cpp

Le fichier main.cpp vous est fourni et ne devrait pas avoir à être modifié pour la remise. Une série de tests sont fournis dans ce fichier pour vous aider à vérifier le comportement de votre programme. Vous pouvez désactiver des blocs de tests en changeant les `#ifdef true` pour des `#ifdef false` afin de compiler le programme avant d'avoir tout terminé. Si un test échoue, allez voir le but du test dans la fonction main.

Sortie attendue



Fichiers de lecture

Trois fichiers de lecture vous sont fournis. `auteurs.txt`, `films.txt` et `restrictionsPays.txt`. Ces fichiers sont utilisés pour tester votre programme et vos méthodes de lecture des fichiers. Assurez-vous de mettre les fichiers à un endroit où votre programme peut les lire.

Compilation (requiert C++17)

Sous Windows : Utilisez la solution VS qui vous est fournie. Assurez-vous que vous compilez avec /W4.

Sous Linux : Un Makefile vous est fourni. Avec votre console, allez sous le même répertoire où se retrouve le fichier Makefile, puis entrez la commande `make` pour compiler, puis la commande `make run` pour rouler le programme. Assurez-vous de copier les fichiers txt à côté du programme compilé sous `bin/linux/debug`.

Fuites de mémoire

Sous Windows :

Le fichier `debogageMemoire.h` est présent pour vous aider à vérifier s'il y a des fuites de mémoire dans votre code. Exécutez la solution en débogage pour qu'il fonctionne. Si une fuite de mémoire est détectée, un message sera affiché dans la fenêtre de sortie (Output) de Visual Studio.

Sous Linux :

Utiliser Valgrind : Lancer la commande ci-dessous :

```
valgrind --tool=memcheck --leak-check=yes ./PATH_VERS_PROGRAMME
```

Pour installer valgrind : `sudo apt install valgrind`, `sudo pacman -S valgrind`, `sudo dnf valgrind`, etc. Si aucune fuite n'est détectée, vous devriez voir la ligne :

```
All heap blocks were freed -- no leaks are possible.
```

Spécifications générales

- Utilisez la liste d'initialisation pour l'implémentation de vos constructeurs.
- Ne modifiez pas les signatures des méthodes.
- Documentez le code source.
- Ne remettez que les fichiers .h et .cpp lors de votre remise
- Les entêtes de fichiers sont obligatoires
- Les fonctions doivent être documentées. Suivez les exemples des fonctions déjà documentées.
- Les fonctions doivent être implémentées dans le même ordre que la définition de la classe.
- **Le guide de codage sur Moodle doit être suivi.**
- **Bien lire le barème de correction ci-dessous.**

Correction

La correction du TP1 se fait sur 20 points :

- [4 points] Compilation du programme (le programme ne doit pas avoir d'avertissements).
- [2 points] Exécution du programme.s
- [8 points] Comportement exact des méthodes du programme.
- [4 points] Documentation du code et bonne norme de codage.
- [2 points] Gestion de la mémoire (pas de fuites).