Département de génie informatique et logiciel

POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

INF1010 – Programmation orientée-objet
TP1

Aut 2023

Objectifs Permettre à l'étudiant de se familiariser avec les concepts de classes, l'allocation dynamique, de composition et d'agrégation et la classe vector.

Remise du travail Date : **21 septembre 2023, avant 23h 30**

- **Une note de 0 sera attribuée aux équipes qui remettent leur travail en retard.**
- Remettre **tous** les fichiers directement sous une archive **.zip** (sans dossier à l'intérieur et sans autres fichiers).
- ✂**Toute archive autre que zip ou remise ne respectant pas ces consignes sera refusée et se méritera une note de 0. Attention, une archive .7z ne compte pas comme une archive zip.** ✂
- Respectez le format de la remise! Nom de l'archive zip : **`matricule1_matricule2_groupe.zip`**
  Exemple : 1234567_1234568_1.zip

Références

- Notes de cours sur Moodle

Directives

- Les travaux s'effectuent obligatoirement en équipe de deux personnes faisant partie du même groupe.
- Les entêtes de fichiers sont obligatoires.
- Les fonctions que vous décidez d'ajouter au programme doivent être documentées.

Conseils

- Lisez les conseils, l'aperçu et les spécifications avant de commencer!
- Ayez lu vos notes de cours!
- Si vous avez des difficultés au cours du TP, rappelez-vous que de nombreux problèmes demandés sont assez couramment rencontrés en C++. Consulter la documentation sur cppreference, les notes de cours, et les forums sur Google peuvent vous donner de bonnes pistes de résolution!
- Veuillez utiliser la version v17 de C++.

- Si votre programme ne compile pas, veuillez mettre en commentaires les instructions qui ne compilent pas.

✂Tout cas de plagiat sera automatiquement reporté au Comité d'examen de fraude (CEF) ✂

Spécifications générales

✂ Le non-respect des spécifications générales entraînera des pénalités. ✂

- Tout warning à la compilation sera pénalisé. Si vous utilisez Visual Studio de Microsoft, vous devez activer l'option **/W4**. Sur Visual Studio, **assurez-vous de compiler avec x64**, certains warnings pourraient ne pas s'afficher sinon.
- Sur Visual Studio, vous devez activer C++17.
- **L'inclusion des fichiers d'en-tête (`.h`) doit être sensible aux majuscules et minuscules**. Malheureusement, sur Windows les fichiers ne sont pas sensibles aux majuscules et minuscule, donc des erreurs d'inclusions pourraient passer inaperçues. Relisez-vous bien.
- Toutes les classes, fonctions et entêtes de fichiers doivent être documentées. Suivez l'exemple de la documentation déjà présente dans le TP.
- Toutes les méthodes doivent être définies dans le fichier d'implémentation (`.cpp`) **dans le même ordre** que leur déclaration dans le fichier d'en-tête (`.h`). Le non-respect de cette règle entraînera une pénalité au niveau du style.
- Utilisez le plus possible la liste d'initialisation des constructeurs. L'utilisation du corps des constructeurs à la place de la liste d'initialisation entraînera une pénalité de style. L'ordre des variables (attributs) dans la liste d'initialisation doit être la même que celle dans la liste des attributs de la classe dans le fichier d'en-tête.
- Suivez le guide de codage sur Moodle.
- Modifications/ajouts au guide de codage à respecter :
    - Vous pouvez utiliser le style d'accolades que vous désirez, **tant que vous êtes uniformes**. Sachez cependant qu'il est généralement attendu d'un(e) développeur(se) suivre la convention établie par le projet, ou la convention établie par la langue (si elle existe) en créant un nouveau projet. Dans le cas du TP fourni, les accolades ouvrantes sont sur leur propre ligne (style Allman).
    - Mettez un espace avant la parenthèse ouvrante des énoncés de contrôle comme if, for, while et switch, mais pas avant les parenthèses d'un appel de fonction. Ne mettez jamais d'espace tout juste après une parenthèse ouvrante ou tout juste avant une parenthèse fermante.
    - Le deux-points (`:`) et le signe égal (`=`) devraient être entourés d'espaces, sauf dans le cas des étiquettes de `case` et les spécificateurs d'accès (`private`, `protected` et `public`) qui ne devraient pas avoir d'espace avant le deux-points. Un espace

devrait toujours suivre les éléments de ponctuation comme la virgule (`,`), le deux-points (`:`) et le point-virgule (`;`).
  - N'utilisez pas `NULL` ou `0` pour les pointeurs, mais bien `nullptr`. Dans le cas de test de `nullptr` avec un pointeur, soyez explicite : préférez `if (p != nullptr)` à `if (p)`.
  - N'utilisez pas de `else` après un `return`.

Mise en contexte

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

Aperçu des classes du projet

- ***Acteur***: Classe qui représente un acteur d'un film ;
- ***Critique*** : Classe qui représente une critique d'un film;
- ***Film*** : Classe qui représente un film;
- ***Polyflix*** : Classe qui représente l'ensemble des films;

Travail à réaliser

Implémenter les classes des fichiers .cpp fournis.

Classe Acteur

Acteur qui représente un acteur. Cette classe contient les attributs suivants :

- Nom.
- Année de Naissance.
- Biographie.

Les méthodes suivantes doivent être implémentées :

- Un constructeur par défaut et par paramètres. Par défaut les attributs ont les valeurs "" et 0.
- Les méthodes d'accès et de modification des attributs.
- Une méthode d'affichage des informations qui concernent un acteur, tel que présenté dans la sortie des fichiers.

Classe Critique

Critique qui représente une critique de film. Cette classe contient les attributs suivants :

- Auteur de la critique du film.
- Commentaire du film.
- Note du film.

Les méthodes suivantes doivent être implémentées :

- Un constructeur par défaut et par paramètres. Par défaut les attributs ont les valeurs "" et 0.
- Les méthodes d'accès des attributs.
- Les méthodes de modifications des attributs.
- Une méthode d'affichage des informations qui concernent la critique, tel que présenté dans la sortie des fichiers.

Classe Film

La classe *Film* sert à représenter un film avec l'ensemble des critiques et des acteurs. La classe Film est une composition de critiques et d'acteurs. Cette classe contient les attributs suivants :

- Vector de pointeur d'acteurs.
- Vector Pointeur de critiques.
- Titre, l'année de sortie, le réalisateur, la catégorie et la durée du film.

Les méthodes suivantes doivent être implémentées :

- Un constructeur par défaut et par paramètres. Par défaut les attributs ont les valeurs "", 0 et ACTION.
- Le destructeur.
- Les méthodes d'accès et de modification des attributs.
- La méthode trouverActeur() vérifie si un acteur est présent dans le film en vérifiant son nom. S'il est présent dans le film, on retourne son pointeur, sinon on retourne nullptr.
- La méthode trouverCritique() vérifie si une critique est présente dans le film en vérifiant l'auteur de la critique. S'il est présent dans le film, on retourne son pointeur, sinon on retourne nullptr.
- isActeurPresent() et isCritiquePresent() vérifient si le nom de l'acteur est présent dans le film et si l'auteur de la critique est présent dans le film.
- ajouterActeur() et ajouterCritique() ajoutent l'acteur et la critique dans les vectors s'ils n'existent pas.
- retirerActeur() et retirerCritique() retirent l'acteur et la critique dans les vectors s'ils existent.
- obtenirMoyenne() retourne la moyenne des notes des critiques.
- Une méthode d'affichage des informations qui concernent un film tel que présenté dans la sortie.

Classe Polyflix

La classe *Polyflix e*st celle qui fait le lien entre toutes les classes précédentes. Cette classe est une composition de Film. Cette classe contient les attributs suivants :

- Nom de l'utilisateur.
- Le mot de passe.
- Vector de pointeurs à des films.

Les méthodes suivantes doivent être implémentées :

- Un constructeur par défaut et paramètre initialise le nom et le mot de passe. Par défaut les attributs ont les valeurs "".
- Un destructeur supprime tous les films.
- Les méthodes get et set.
- La méthode connecter() permet de vérifier le nom et le mot de passe de l'utilisateur.
- La méthode ajouterFilm() qui permet d'ajouter le film dont les attributs sont reçus en paramètre s'il n'existe pas dans le vector. La vérification de l'existence du film se fait selon le titre du film. On oublie les acteurs et les critiques dans l'ajout du film.
- La méthode retirerFilm() permet de retirer le film s'il existe en utilisant son titre et son année de sortie reçus en paramètre. La vérification de l'existence du film se fait selon le titre du film. Faut faire la verification.
- La méthode chercherFilmParTitre() retourne le film selon le titre reçu par paramètre s'il existe.
- La méthode listerFilmsParCategorie() retourne un vecteur de films de la catégorie passée en paramètre.
- La méthode getNombreTotalFilms() retourne le nombre de films.
- La méthode getNombreFilmsParCategorie() retourne le nombre de films de la catégorie.
- La méthode obtenirFilmAvecLaPlusHauteNote() retourne le film ayant obtenu la plus grande note.
- La méthode obtenirFilmAvecLaPlusBasseNote() retourne le film ayant obtenu la plus petite note.
- La méthode obtenirFilmPlusRecent() retourne le film le plus récent. On suppose qu'il n'existe pas deux films ayant la même date de sortie.
- La méthode afficherListeFilms() affiche toutes les informations concernant les films. Pensez à faire appel aux méthodes de la classe Film.
- La méthode afficherFilmsParCategorie() affiche les films correspondant à la catégorie reçu en paramètre. Pensez à faire appel aux méthodes de la classe Film.

main.cpp

Le programme principal roule une série de tests.

La correction du TP se fait sur 20 points :

- [3 pt] Compilation du programme sans avertissements.
- [4 pt] Exécution du programme.
- [10 pt] Comportement exact de l'application
- [2 pt] Qualité et documentation du code.
- [1 pt] Absence de fuites de mémoire.

Relisez bien les spécifications générales et les consignes de remise au début du TP avant de remettre votre travail!