INF1010
Programmation orientée-objet

Travail pratique 3
Héritage, polymorphisme et héritage multiple

Département de génie informatique et logiciel

Polytechnique Montréal
12 février 2020

POLYTECHNIQUE MONTRÉAL

Objectifs

Permettre à l'étudiant de se familiariser avec les notions d'héritage, de polymorphisme, de conversion dynamique de type, d'héritage multiple et de fonctions virtuelles.

Remise du travail

- Date : **25 février 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

Le présent TP constitue une suite du programme `CinéPoly` réalisé lors des précédents TPs, il aura pour objectif d'intégrer les notions d'héritage, de polymorphisme, de conversion dynamique de type, d'héritage multiple et de fonctions virtuelles.

Afin de diversifier l'offre, `CinéPoly` ajoute des séries à sa librairie qui inclut un ensemble d'épisodes par saison, pour une ou plusieurs saisons par série. De cette façon, nous avons maintenant des films et des séries qui héritent une partie ou l'ensemble de leurs attributs et méthodes de classe mère Media.

L'héritage permet de créer une classe mère possédant ses propres caractéristiques et de l'utiliser pour créer de nouvelles classes filles. Pour réussir ce TP, il faudra vous familiariser avec la fonction `C++ dynamic_cast<T*>(Objet*)`. Cet opérateur permet de convertir à l'exécution un pointeur de la classe mère en pointeur de la classe fille.
Dans ce TP, il vous permettra par exemple de convertir un pointeur de type Media en pointeur de type Série.

Les fonctions virtuelles permettent à une classe dérivée de surcharger une méthode et que celle-ci soit appelée par le compilateur `C++` sur un pointeur de cette classe, même si le pointeur a été défini comme un pointeur de la classe mère. De plus, elles peuvent servir à définir des classes abstraites en terminant la définition d'une fonction virtuelle par un = 0. Une classe ayant au moins une méthode abstraite est une classe abstraite et on ne peut pas créer d'objet de cette classe. Une classe possédant seulement des fonctions virtuelles pures ne peut être instanciée et est appelée une interface.

Pour vous aider, la solution du TP précédent est fournie. Vous n'avez qu'à implémenter les nouvelles méthodes décrites plus bas et supprimer les attributs qui n'ont plus lieu d'être. Les méthodes à modifier et à retirer vous ont été indiquées.

ATTENTION :

- **Tout au long du TP, n'utiliser que les pointeurs intelligents.**
- **Tout au long du TP, assurez-vous d'utiliser les opérateurs sur les objets et non sur leurs pointeurs ! Vous devez donc déréférencer les pointeurs si nécessaire.**
- **Vous serez pénalisés pour les utilisations inutiles du mot-clé *this*. Utilisez-le seulement où nécessaire.**
- **Il est fortement conseillé d'utiliser les fichiers fournis, plutôt que de continuer avec vos fichiers du TP2.**

Remarque : Pour plus de précision sur le travail à faire et des changements à effectuer, veuillez-vous référer aux fichiers .h

Travail à réaliser

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

`Classe Episode`

Cette classe représente un épisode.

Cette classe contient les attributs suivants :

- `nom_ (string)`
- `duree_ (string)`
- `numEpisode_ (unsigned int)`

Les méthodes suivantes doivent être implémentées :

- Le constructeur avec/sans paramètres
  - Faire attention à bien initialiser tous les attributs.
- `opérateur==(unsigned int)`
  - L'opérateur compare un `unsigned int numEpisode` avec un `episode`
  - L'opérateur retourne `true` si les deux numéros sont égaux, `false` sinon.
- `opérateur<<(ostream&, const Episode&)`
  - L'opérateur permet d'afficher tous les attributs de l'épisode
  - L'opérateur doit pouvoir être appelé en cascade.
- `opérateur>>(istream&, const Episode&)`
  - L'opérateur permet d'initialiser tous les attributs de l'épisode
  - L'opérateur doit pouvoir être appelé en cascade.
- Les méthodes d'accès qui vous semblent utiles.

`Classe Saison`

Cette classe représente une Saison.

Cette classe contient les attributs suivants :

- `episodes_ (vector<unique_ptr<Episode>>)`
- `numSaison_ (unsigned int)`
- `nbEpisodesmax__ (unsigned int)`

Les méthodes suivantes doivent être implémentées :

- Le constructeur avec/sans paramètres
  - Faire attention à bien initialiser tous les attributs.
- `Saison(Saison &)` constructeur par copie
- Le destructeur
  - Vide le vecteur episodes_.
- `opérateur==( unsigned int)`
  - L'opérateur compare un `unsigned int numSaison` avec une `saison`
  - L'opérateur retourne `true` si les deux numéros sont égaux, `false` sinon.

- `opérateur+=(Episode*)`
  - L'opérateur ajoute un épisode au vecteur `episodes_`
  - Si `episode` reçu en paramètre existe dans le vecteur `episodes_`, il faut supprimer l'ancien `episode` avant d'ajouter la nouvelle.
  - L'opérateur doit pouvoir être appelé en cascade.
  - Utilisez cette ligne de code dans votre méthode pour trier les épisodes par ordre croissant des numéros d'épisode :
    `sort(episodes_.begin(),episodes_.end(), Episode::SortByNumEpisode())`
  - L'opérateur utilise la méthode `trouverIndexEpisode`
- `opérateur-=(unsigned int)`
  - L'opérateur retire un épisode du vecteur `episodes_`
  - L'opérateur doit pouvoir être appelé en cascade.
  - L'opérateur utilise la méthode `trouverIndexEpisode`
- `opérateur<<(ostream&, const Saison&)`
  - L'opérateur affiche tous les attributs de la `saison`
  - L'opérateur affiche tous les épisodes du vecteur `episodes_`
  - L'opérateur doit pouvoir être appelé en cascade.
- `opérateur>>(istream&, const Saison&)`
  - L'opérateur permet d'initialiser tous les attributs de la classe `saison`
  - L'opérateur doit pouvoir être appelé en cascade.
- `trouverIndexEpisode(unsigned int)`
  - Méthode privée qui cherche un épisode comportant le numéro envoyé en paramètre
  - La méthode retourne l'index du premier épisode comportant le numéro
  - La méthode retourne `-1` si le numéro de l'épisode ne s'y trouve pas.
- Les méthodes d'accès qui vous semblent utiles.

Classe GestionnaireSaison

Cette classe gère les saisons.

Cette classe contient les attributs suivants :

- `saisons_ (vector<unique_ptr<Saison>>)`

Les méthodes suivantes doivent être implémentées :

- Le destructeur
  - Vide le vecteur saisons_
- `opérateur+=(Saison*)`
  - L'opérateur ajoute une `saison` au vecteur `saisons_`
  - Si la `saison` reçue en paramètre existe dans le vecteur `saisons_`, il faut supprimer l'ancienne `saison` avant d'ajouter la nouvelle.
  - L'opérateur doit pouvoir être appelé en cascade.
  - Utiliser cette ligne de code dans votre méthode pour trier les saisons par ordre croissant des numéros de saison :

```
sort(saisons_.begin(), saisons _.end(), Saison::SortByNumSaison());
```

  - L'opérateur utilise la méthode `trouverIndexSaison`

- `opérateur-=(unsigned int)`
  - L'opérateur retire une saison du vecteur `saisons_`
  - L'opérateur doit pouvoir être appelé en cascade.
  - L'opérateur utilise la méthode `trouverIndexSaison`
- `ajouterEpisode(unsigned int, Episode*);`
  - La méthode ajoute un `episode` à la `saison` dont le `numéro` est reçu en paramètre
  - La méthode utilise les méthodes suivantes :
    - `trouverIndexSaison`
    - `operateur+=` de la classe `Saison`
- `retirerEpisode(unsigned int, unsigned int);`
  - La méthode retire un `episode` de la `saison` dont le `numéro` est reçu en paramètres
  - La méthode utilise les méthodes suivantes :
    - `trouverIndexSaison`
    - `operateur-=` de la classe `Saison`
- `Afficher(ostream&)`
  - Méthode abstraite
- `trouverIndexSaison(unsigned int)`
  - Méthode privée qui cherche une saison comportant le numéro envoyé en paramètre
  - La méthode retourne l'index de la première saison comportant le numéro
  - La méthode retourne -1 si le numéro de la saison ne s'y trouve pas.
- Les méthodes d'accès qui vous semblent utiles.

Classe Auteur

Cette classe représente un auteur.

Les méthodes suivantes doivent être implémentées :

- Constructeur sans paramètres.
- `opérateur>>(istream&, const Auteur&)`
  - L'opérateur remplacera la méthode `lireLigneAuteur` de la classe `GestionnaireAuteur`
  - L'opérateur permet d'initialiser tous les attributs de la classe `auteur`
  - L'opérateur doit pouvoir être appelé en cascade.

Classe GestionnaireAuteurs

Cette classe gère les auteurs.

Les méthodes suivantes doivent être supprimées :

- La méthode `lireLigneAuteur` doit être retirée, car elle sera remplacée par `l'opérateur >>` de la classe `Auteur`

Les méthodes suivantes doivent être modifiées :

- `chargerDepuisFichier` doit être adaptée pour faire appel à `l'opérateur>>` de la classe `Auteur`

`Classe Media`

Cette classe représente un média. Cette classe est recyclée de la classe Film du TP2.

Cette classe contient les attributs suivants :

- `nom_` `(string)`
- `anneeDeSortie_` `(unsigned int)`
- `genre_` `(Genre)`
- `estRestreintParAge_` `(Bool)`
- `pays_` `(Pays)`
- `auteur_` `(Auteur*)`
- `typeMedia_` `(Media::TypeMedia)`
- `paysRestreints_` `(vector<Pays>)`

Les méthodes suivantes doivent être recyclées de la classe Film du TP1/2 :

- `ajouterPaysRestreint(Pays);`
- `supprimerPaysRestreints();`
- `estRestreintDansPays(Pays);`
- `estRestreintParAge();`
- `afficher(ostream&);`
- `getGenre();`
- `getNom();`
- `getAuteur();`

Les méthodes suivantes doivent être modifiées :

- `operator<<(ostream&, const Media&)` doit être modifié pour faire appel à la méthode `affiche` de la classe `Media`

Les méthodes suivantes doivent être implémentées :

- Le constructeur avec paramètres
  - Faire attention à bien initialiser tous les attributs.
- `Media(Media &)` constructeur par copie
- Le destructeur virtuel
- `Media(Auteur*, Media::TypeMedia)`
  - Faire attention à bien initialiser tous les attributs.
- `lire(istream&);`
  - L'opérateur permet d'initialiser tous les attributs de la classe `Media`
  - L'opérateur doit pouvoir être appelé en cascade.
- `operator>>(istream&, Media&);`
  - L'opérateur utilise la méthode `lire` de la classe `Media`
- `clone()` méthode qui retourne un objet alloué dynamiquement qui est une copie de l'objet courant.
- Les méthodes d'accès qui vous semblent utiles.

Classe Film

Cette classe représente un film, elle hérite de la classe Media

Cette classe contient les attributs suivants :

- `duree_ (string)`

Les méthodes suivantes doivent être implémentées :

- Le constructeur avec paramètres
  - Faire attention à bien initialiser tous les attributs.
  - Ce constructeur utilise le constructeur de la classe `Media`
- `afficher(ostream& os)`
  - L'opérateur affiche tous les attributs de `Film`
  - L'opérateur doit pouvoir être appelé en cascade.
- `Lire(istream&);`
  - L'opérateur permet d'initialiser tous les attributs de la classe `Film`
  - L'opérateur doit pouvoir être appelé en cascade.
  - L'opérateur utilise la méthode `lire` de la classe `media`
- `clone()` méthode qui retourne un objet alloué dynamiquement qui est une copie de l'objet courant.

Classe Série

Cette classe représente une série, elle hérite des classes `Media` et `GestionnaireSaison`

Les méthodes suivantes doivent être implémentées :

- Le constructeur avec paramètres
  - Faire attention à bien initialiser tous les attributs.
  - Ce constructeur fait appel aux constructeurs des classes `Media` et `GestionnaireSaison`,
- `Serie(Serie &)` constructeur par copie
- `afficher(ostream& os)`
  - L'opérateur affiche tous les attributs de la classe `Serie`
  - L'opérateur doit pouvoir être appelé en cascade.
- `clone()` méthode qui retourne un objet alloué dynamiquement qui est une copie de l'objet courant.

Classe Librairie

Cette classe gère les films offerts par `CinéPoly`.

Les méthodes suivantes doivent être retiré :

- `chargerFilmsDepuisFichier` elle sera remplacée par la méthode `chargerMediasDepuisFichier`
- `trouverIndexFilm` elle sera remplacée par la méthode `trouverIndexMedia`
- `opérateur+=(Film*)` sera remplacée par `opérateur+=(Media*)`

Les méthodes suivantes doivent être modifiées :

- `chercherFilm(const string&);`
  - La méthode doit être modifiée pour utiliser la méthode `chercherMedia`
  - Penser à utiliser un `dynamic_cast` au besoin
- `Librairie(Librairie &)`
  - La méthode doit être modifiée pour prendre en considération le polymorphisme
  - La méthode utilise la méthode `clone()` de la classe `Media`
- `opérateur-=(const string&)`
  - La méthode doit être modifiée pour retirer un `média` à la place d'un `film`
- `opérateur=(Librairie &)`
  - La méthode doit être modifiée pour prendre en considération le polymorphisme
- `lireLigneRestrictions(const string&)`
  - La méthode doit être modifiée pour utiliser la méthode `chercherMedia`.
  - Le fichier `restrictionsPays.txt` a été modifié pour prendre en charge le type de média.

Les méthodes suivantes doivent être implémentées :

- `ajouterSaison(const string&, Saison*);`
  - La méthode ajoute une `saison` à la `série` dont le `nom` est reçu en paramètre
  - La méthode utilise les méthodes suivantes :
    - `trouverIndexMedia` de la classe `Librairie`
    - `getTypeMedia` de la classe `Media`
    - `operateur+=` de la classe `GestionnaireSaison`
- `retirerSaison(const string&, unsigned int);`
  - La méthode retire une `saison` de la `série` dont le `nom` est reçu en paramètre
  - La méthode utilise les méthodes suivantes :
    - `trouverIndexMedia` de la classe `Librairie`
    - `getTypeMedia` de la classe `Media`
    - `operateur-=` de la classe `GestionnaireSaison`
- `ajouterEpisode(const string&, unsigned int, Episode*);`
  - La méthode ajoute un `épisode` à la `saison` de la `série` dont le `nom` et le `numéro` sont reçus en paramètre
  - La méthode utilise les méthodes suivantes :
    - `trouverIndexMedia` de la classe `Librairie`
    - `getTypeMedia` de la classe `Media`
    - `ajouterEpisode` de la classe `GestionnaireSaison`
- `retirerEpisode(const string&, unsigned int, unsigned int);`
  - La méthode retire un `épisode` de la `saison` de la `série` dont le `nom` et le `numéro` sont reçus en paramètres
  - La méthode utilise les méthodes suivantes :
    - `trouverIndexMedia` de la classe `Librairie`
    - `getTypeMedia` de la classe `Media`
    - `retirerEpisode` de la classe `GestionnaireSaison`

- `chercherSerie(const string&);`
  - La méthode retourne la `série` dont le `nom` est reçu en paramètre
  - Si la `série` n'existe pas on return un `pointeur nul`
  - La méthode utilise la méthode `chercherMedia` de la classe `Librairie`
- `chercherMedia(const string& , Media::TypeMedia);`
  - Méthode privée qui cherche un `média`
  - La méthode retourne le `média` dont le `nom` est reçu en paramètre
  - La méthode utilise les méthodes suivantes :
    - `trouverIndexMedia` de la classe `Librairie`
    - `getTypeMedia` de la classe `Media`
- `lireLigneMedia(const string&, GestionnaireAuteurs&);`
  - La méthode est fournie il ne faut pas la modifier
- `lireLigneFilm(const string&, GestionnaireAuteurs&);`
  - Méthode privée qui lit une `ligne` à partir d'un `string` reçu comme paramètre
  - La méthode ajoute un `film` à la `librairie`
  - La méthode retourne `true` si la lecture c'est bien passer, `false` sinon
  - La méthode utilise les méthodes suivantes :
    - `chercherAuteur` de la classe `GestionnaireAuteurs`
    - `operateur>>, set` et `getNbMedias` de la classe `Auteur`
    - `operateur>>` de la classe `Media`
    - `operateur+=` de la classe `Librairie`
- `lireLigneSerie(const string&, GestionnaireAuteurs&);`
  - Méthode privée qui lit une `ligne` à partir d'un `string` reçu comme paramètre
  - La méthode ajoute une `série` à la `librairie`
  - La méthode retourne `true` si la lecture c'est bien passer, `false` sinon
  - La méthode utilise les méthodes suivantes :
    - `chercherAuteur` de la classe `GestionnaireAuteurs`
    - `operateur>>, set` et `getNbMedias` de la classe `Auteur`
    - `operateur>>` de la classe `Media`
    - `operateur+=` de la classe `Librairie`
- `lireLigneSaison(const string&, GestionnaireAuteurs&);`
  - Méthode privée qui lit une `ligne` à partir d'un `string` reçu comme paramètre
  - La méthode ajoute une `saison` d'une `série` à la `librairie`
  - La méthode retourne `true` si la lecture c'est bien passer, `false` sinon
  - La méthode utilise les méthodes suivantes :
    - `operateur>>` de la classe `Saison`
    - `ajouterSaison` de la classe `Librairie`
- `lireLigneEpisode(const string&, GestionnaireAuteurs&);`
  - Méthode privée qui lit une `ligne` à partir d'un `string` reçu comme paramètre
  - La méthode ajoute un `épisode` d'une `saison` d'une `série` à la `librairie`
  - La méthode retourne `true` si la lecture c'est bien passer, `false` sinon
  - La méthode utilise les méthodes suivantes :

  - `operateur>>` de la classe `Episode`
  - `ajouterEpisode` de la classe `Librairie`
- `trouverIndexMedia(const string&);`
  - Méthode privée qui cherche un `média` comportant le `nom` envoyé en paramètre
  - Retourne l'index du premier média comportant le nom
  - Retourne `-1` si le nom du média ne s'y trouve pas.
  - La méthode utilise la méthode `getNom` de la classe `Media`
- Les méthodes d'accès qui vous semblent utiles.

Classe Utilisateur

La classe représente un utilisateur de CinéPoly. Cette classe n'a pas de changement.

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.

Fichiers de lecture

Trois fichiers de lecture vous sont fournis. `auteurs.txt`, `Medias.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. Les fichiers Medias.txt et restrictionsPays.txt ont été modifiés comme suit :

`fichiers Medias.txt :`

Film							
Type Media	Auteur	Nom du film	Année de sortie	genre	pays	restriction	durée
Série							
Type Media	Auteur	Nom du film	Année de sortie	genre	pays	restriction	
Saison							
Type Media	Numéro	Nombre d'épisodes par saison	Nom de la série				
Episode							
Type Media	numéro	Nom	durée	Nom de la série	Numéro de la saison		

`fichiers restrictionsPays.txt :`

Type Media	Nom	Restriction

Sortie attendue

```
                Episode 01: Episode:01 | Duree: 00:00:00

        Saison 01: 3/20(En cour)
                Episode 01: Episode:01 | Duree: 00:00:00
                Episode 02: Episode:02 | Duree: 00:00:00
                Episode 03: Episode:03 | Duree: 00:00:00

        Saison 01: 2/20(En cour)
                Episode 01: Episode:01 | Duree: 00:00:00
                Episode 03: Episode:03 | Duree: 00:00:00

Nom: Auteur Test | Date de naissance: 2019 | Nombre de Film/Serie: 0
Film Test
        Date de sortie: 2019
        Genre: Action
        Auteur: Auteur Test
        Pays: EtatsUnis
        Aucun pays restreint.
        Duree: 00:00:00

Serie Test
        Date de sortie: 2019
        Genre: Action
        Auteur: Auteur Test
        Pays: EtatsUnis
        Aucun pays restreint.
        Saison 01: 2/20(En cour)
                Episode 01: Episode:01 | Duree: 00:00:00
                Episode 03: Episode:03 | Duree: 00:00:00
        Saison 02: 0/20(En cour)

Serie Test
        Date de sortie: 2019
        Genre: Action
        Auteur: Auteur Test
        Pays: EtatsUnis
        Aucun pays restreint.
        Saison 01: 2/20(En cour)
                Episode 01: Episode:01 | Duree: 00:00:00
                Episode 04: Episode:04 | Duree: 00:00:00
        Saison 02: 1/20(En cour)
                Episode 05: Episode:05 | Duree: 00:00:00
```

```
Serie Test
        Date de sortie: 2019
        Genre: Action
        Auteur: Auteur Test
        Pays: EtatsUnis
        Aucun pays restreint.
        Saison 01: 2/20(En cour)
                Episode 01: Episode:01 | Duree: 00:00:00
                Episode 04: Episode:04 | Duree: 00:00:00


Nom: George Lucas | Date de naissance: 1944 | Nombre de Film/Serie: 6
Nom: John Ronald Reuel Tolkien | Date de naissance: 1892 | Nombre de Film/Serie: 3
Nom: David Benioff | Date de naissance: 1970 | Nombre de Film/Serie: 1
Nom: Paul Scheuring | Date de naissance: 1968 | Nombre de Film/Serie: 1


A New Hope
        Date de sortie: 1977
        Genre: Action
        Auteur: George Lucas
        Pays: EtatsUnis
        Aucun pays restreint.
        Duree: 02:05:00
Raiders of the Lost Ark
        Date de sortie: 1981
        Genre: Aventure
        Auteur: George Lucas
        Pays: EtatsUnis
        Pays restreints:
                Bresil
                Canada
                Chine
                EtatsUnis
                France
                Japon
                RoyaumeUni
                Russie
                Mexique
        Duree: 01:55:00
Indiana Jones and the Temple of Doom
        Date de sortie: 1984
        Genre: Aventure
        Auteur: George Lucas
        Pays: EtatsUnis
        Aucun pays restreint.
        Duree: 01:58:00
```

```
Indiana Jones and the Last Crusade
        Date de sortie: 1989
        Genre: Aventure
        Auteur: George Lucas
        Pays: EtatsUnis
        Aucun pays restreint.
        Duree: 02:08:00
The Lord of the Rings: The Fellowship of the Ring
        Date de sortie: 2001
        Genre: Aventure
        Auteur: John Ronald Reuel Tolkien
        Pays: RoyaumeUni
        Aucun pays restreint.
        Duree: 03:48:00
The Empire Strikes Back
        Date de sortie: 1980
        Genre: Action
        Auteur: George Lucas
        Pays: EtatsUnis
        Aucun pays restreint.
        Duree: 02:07:00
Return of the Jedi
        Date de sortie: 1983
        Genre: Action
        Auteur: George Lucas
        Pays: EtatsUnis
        Aucun pays restreint.
        Duree: 02:16:00
The Lord of the Rings: The Two Towers
        Date de sortie: 2002
        Genre: Aventure
        Auteur: John Ronald Reuel Tolkien
        Pays: RoyaumeUni
        Aucun pays restreint.
        Duree: 03:55:00
The Lord of the Rings: The Return of the King
        Date de sortie: 2003
        Genre: Aventure
        Auteur: John Ronald Reuel Tolkien
        Pays: RoyaumeUni
        Pays restreints:
                Chine
                France
                Japon
                Russie
        Duree: 03:29:00
```

```
Game of Thrones
        Date de sortie: 2011
        Genre: Action
        Auteur: David Benioff
        Pays: EtatsUnis
        Aucun pays restreint.
        Saison 01: 10/10(Terminer)
                Episode 01: Winter Is Coming                    | Duree: 00:40:20
                Episode 02: The Kingsroad                       | Duree: 00:40:20
                Episode 03: Lord Snow                           | Duree: 00:40:20
                Episode 04: Cripples, Bastards and Broken Things | Duree: 00:40:20
                Episode 05: The Wolf and the Lion               | Duree: 00:40:20
                Episode 06: A Golden Crown                      | Duree: 00:40:20
                Episode 07: You Win or You Die                  | Duree: 00:40:20
                Episode 08: The Pointy End                      | Duree: 00:40:20
                Episode 09: Baelor                              | Duree: 00:40:20
                Episode 10: Fire and Blood                      | Duree: 00:40:20
        Saison 02: 10/10(Terminer)
                Episode 01: The North Remembers         | Duree: 00:40:20
                Episode 02: The Night Lands             | Duree: 00:40:20
                Episode 03: What Is Dead May Never Die  | Duree: 00:40:20
                Episode 04: Garden of Bones             | Duree: 00:40:20
                Episode 05: The Ghost of Harrenhal      | Duree: 00:40:20
                Episode 06: The Old Gods and the New    | Duree: 00:40:20
                Episode 07: A Man Without Honor         | Duree: 00:40:20
                Episode 08: The Prince of Winterfell    | Duree: 00:40:20
                Episode 09: Blackwater                  | Duree: 00:40:20
                Episode 10: Valar Morghulis             | Duree: 00:40:20
Prison Break
        Date de sortie: 2005
        Genre: Action
        Auteur: Paul Scheuring
        Pays: EtatsUnis
        Aucun pays restreint.
        Saison 01: 22/22(Terminer)
                Episode 01: Pilot                               | Duree: 00:40:20
                Episode 02: Allen                               | Duree: 00:40:20
                Episode 03: Cell Test                           | Duree: 00:40:20
                Episode 04: Cute Poison                         | Duree: 00:40:20
                Episode 05: English, Fitz or Percy              | Duree: 00:40:20
                Episode 06: Riots, Drills and the Devil (Part 1) | Duree: 00:40:20
                Episode 07: Riots, Drills and the Devil (Part 2) | Duree: 00:40:20
                Episode 08: The Old Head                        | Duree: 00:40:20
                Episode 09: Tweener                             | Duree: 00:40:20
                Episode 10: Sleight of Hand                     | Duree: 00:40:20
                Episode 11: And Then There Were 7               | Duree: 00:40:20
                Episode 12: Odd Man Out                         | Duree: 00:40:20
                Episode 13: End of the Tunnel                   | Duree: 00:40:20
                Episode 14: The Rat                             | Duree: 00:40:20
                Episode 15: By the Skin and the Teeth           | Duree: 00:40:20
```

```
                Episode 17: J-Cat                                       | Duree: 00:40:20
                Episode 18: Bluff                                       | Duree: 00:40:20
                Episode 19: The Key                                     | Duree: 00:40:20
                Episode 20: Tonight                                     | Duree: 00:40:20
                Episode 21: Go                                          | Duree: 00:40:20
                Episode 22: Flight                                      | Duree: 00:40:20
        Saison 02: 22/22(Terminer)
                Episode 01: Manhunt             | Duree: 00:40:20
                Episode 02: Otis                | Duree: 00:40:20
                Episode 03: Scan                | Duree: 00:40:20
                Episode 04: First Down          | Duree: 00:40:20
                Episode 05: Map 1213            | Duree: 00:40:20
                Episode 06: Subdivision         | Duree: 00:40:20
                Episode 07: Buried              | Duree: 00:40:20
                Episode 08: Dead Fall           | Duree: 00:40:20
                Episode 09: Unearthed           | Duree: 00:40:20
                Episode 10: Rendez-vous         | Duree: 00:40:20
                Episode 11: Bolshoi Booze       | Duree: 00:40:20
                Episode 12: Disconnect          | Duree: 00:40:20
                Episode 13: The Killing Box     | Duree: 00:40:20
                Episode 14: John Doe            | Duree: 00:40:20
                Episode 15: The Message         | Duree: 00:40:20
                Episode 16: Chicago             | Duree: 00:40:20
                Episode 17: Bad Blood           | Duree: 00:40:20
                Episode 18: Wash                | Duree: 00:40:20
                Episode 19: Sweet Caroline      | Duree: 00:40:20
                Episode 20: Panama              | Duree: 00:40:20
                Episode 21: Fin Del Camino      | Duree: 00:40:20
                Episode 22: Sona                | Duree: 00:40:20

Test 1: OK!
Test 2: OK!
Test 3: OK!
Test 4: OK!
Test 5: OK!
Test 6: OK!
Test 7: OK!
Test 8: OK!
Test 9: OK!
Test 10: OK!
Test 11: OK!
Test 12: OK!
Test 13: OK!
Test 14: OK!
Test 15: OK!
Test 16: OK!
Test 17: OK!
Test 18: OK!
Test 19: OK!
Test 20: OK!
```

Compilation

Sous Windows : Faites une solution Visual Studio.
Sous Linux et MacOS: Un Makefile vous est fourni. Mettez tous les fichiers .h sous TP3/include, mettez tous les fichiers .cpp sous TP3/src et mettez le Makefile directement dans TP3. make -C path/to/TP3 all compile le programme. make -C path/to/TP3 run roule le programme.

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.

Compilez avec -W4.
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, 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.
- Ajouter le mot-clé *const* chaque fois que cela est pertinent.
- Ajouter des références dans les paramètres lorsque cela est pertinent
- 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 TP3 se fait sur 20 points :

- [4 points] Compilation du programme (le programme ne doit pas avoir d'avertissements).
- [2 points] Exécution du programme.
- [2 points] Comportement exact des méthodes du programme.
- [2 points] Utilisation correcte des méthodes virtuelles et la conversion dynamique de type.
- [3 points] Utilisation correcte du polymorphisme.
- [1 point] Utilisation correcte du mot clé *const*.

- [1 point] Utilisation correcte du mot clé *this*
- [1 point] Passage de paramètres adéquat.
- [4 points] Documentation du code et bonne norme de codage.