IFT1169 – TRAVAIL PRATIQUE #2 – 05 février 2008

La virtualité du langage C++!

Mohamed Lokbani

Équipes : le travail peut-être fait en binôme mais vous ne remettez qu'un travail par équipe.
Remise : une seule remise est à effectuer par voie électronique **le jeudi 13 mars 2007, 23h59 au plus tard, sans possibilités de prorogation**.
Conseils : n'attendez pas le dernier jour avant la remise pour engager votre travail. Vous n'aurez pas le temps nécessaire pour le réaliser.

But : vous allez concevoir dans ce travail pratique tout d'abord, sur papier une architecture orientée objet. Par la suite vous allez mettre cette architecture en pratique sous la forme d'un programme. Les connaissances développées dans ce TP sont essentiellement l'héritage, la surcharge des opérateurs, la ligature dynamique, la compilation séparée et finalement la gestion de projet contenant plusieurs fichiers.

Énoncé : on vous demande d'écrire un programme permettant d'effectuer certaines opérations reliées à la gestion des fonds documentaires d'une bibliothèque. Pour cela, nous allons définir les éléments suivants :

Un **catalogue** recense les fonds documentaires dont dispose la bibliothèque. L'interface de ce catalogue contient les éléments suivants :

- MaxItem (type int) est le nombre maximal d'items qu'une bibliothèque peut contenir.
- NbrItem (type int) est le nombre d'items déjà enregistrés dans la bibliothèque.
- Un tableau d'objets du type Item.

Un **item** (une entrée) dans une bibliothèque est identifié par les éléments suivants :

- Une cote (i.e un id) (type int)
- Un titre (type string)
- L'année de production/publication (type int)

Un **film**, en plus d'être un item, est identifié aussi par :

- Le nom de son réalisateur (type string)

Un **livre**, en plus d'être un item, est identifié aussi par :

- Le nom de l'auteur (type string)
- Nombre de pages (type int)

À signaler : même si le film (resp. le livre) a été réalisé (resp. écrit) par plusieurs réalisateurs (resp. auteurs), il y a un seul champ et il sera du type **string**.

Organisation des fichiers : votre travail consiste aussi à distribuer le code sous la forme de plusieurs fichiers. Chaque fichier va contenir une description d'un des éléments de votre fond documentaire. L'organisation des fichiers doit être comme suit :

	Interface	Implémentation
Item	Item.h	Item.cpp
Film	Film.h	Film.cpp
Livre	Livre.h	Livre.cpp
Catalogue	Catalogue.h	Catalogue.cpp
tp2h05.cpp		

Vous pouvez ajouter d'autres fichiers si vous le jugez nécessaire.

À noter qu'il vous est demandé ce qui suit:

-1- vous devez optimiser votre schéma d'héritage (design) de telle manière à réduire les redondances tout en respectant l'organisation précédemment décrite. Par la suite vous devez implémenter le design choisi (i.e. le coder).
-2- vous devez concevoir les opérateurs suivants:

a) l'opérateur d'entrée (<<) pour la saisie des données du clavier, et l'opérateur de sortie (>>) pour l'affichage des informations vers l'écran. Ces opérateurs doivent être définis pour toutes les classes mentionnées précédemment. Par exemple:

Soit « UnItem » une instance de « Item », alors :

```
cin >> UnItem;     // permet de lire les données nécessaires à une instance de Item.
cout << UnItem;    // permet d'afficher dans un format libre les données associées à une instance de Item.
```

Soit « UnCatalogue » une instance de « Catalogue », alors :

```
cout << UnCatalogue;    // permet d'afficher en sortie le catalogue complet de la bibliothèque
                        // (i.e. tous les Items).
```

b) les opérateurs de comparaison « == » et « != » permettant de comparer deux instances de « Items », de « Livre », « Film » ou « Catalogue ».

c) l'opérateur « += » uniquement pour la classe « Catalogue ». Il permet d'ajouter une nouvelle instance de « Item » dans la bibliothèque. Faire attention de ne pas dépasser le nombre maximal d'items qu'une bibliothèque peut contenir.

Il vous est permis d'ajouter d'autres opérateurs si vous le jugez nécessaire.

-3- vous devez prévoir les méthodes suivantes :

a) constructeurs et destructeurs appropriés pour les différentes classes.

b) la méthode « ChercheCatalogue » de la classe « Catalogue ». Cette méthode doit nous permettre de rechercher le fond documentaire de la bibliothèque. La recherche devra se faire sur une combinaison des paramètres suivants : cote et/ou le titre et/ou l'année.

Il vous est permis de surdéfinir plusieurs méthodes si vous le jugez nécessaire.

-4- la méthode main doit se trouver dans le fichier « tp2h05.cpp » et va nous permettre de tester votre tp2. Pour cela vous devez prévoir ce qui suit :

a) demander d'abord le nombre d'items que peut contenir votre bibliothèque.
b) enregistrer (i.e. ajouter) les items livres et/ou films dans le fond documentaire.
c) rechercher un item dans votre bibliothèque.
d) afficher les informations associées à un seul item de la liste des éléments obtenus suite à une recherche.
e) afficher les informations associées à tous les items obtenus suite à une recherche.
f) imprimer tout le catalogue.

-5- vous devez finaliser vos tests en utilisant les redirections, i.e.: tp2h05.exe < in.txt > out.txt
Le fichier « in.txt » va contenir toutes les instructions entrées « pseudo manuellement » (comme si vous les aviez tapées à la main l'une à la suite de l'autre). La redirection de la sortie écran vers un fichier, ne demande pas vraiment un traitement spécial, sauf pour la gestion des erreurs de sortie. Ainsi donc votre travail devra finalement être exécuté comme suit : « tp2h05.exe < in.txt > out.txt » et vous devez inclure dans votre remise les deux fichiers « int.txt » et « out.txt ».

-6- vous devez répondre dans votre rapport aux questions suivantes :

a) quels ont été les avantages d'avoir utilisé le polymorphisme dans votre design?

b) que signifie la ligature dynamique. Est-il nécessaire de l'inclure dans votre design, si oui ou? Et comment? Si vous avez répondu par l'affirmatif à cette question, vous devez ressortir cela à travers un exemple d'utilisation dans la fonction main.

c) tout en développant votre réponse, pouviez-vous éteindre encore plus (par héritage) la dernière classe dérivée de votre désign?

-7- Il vous est permis d'utiliser toutes les notions du langage C++ déjà apprises (ne pas utiliser les méthodes de la bibliothèque des modèles standard STL.

Remise : il est important de noter que votre TP sera compilé avec « gcc3.4.2 ». Si par choix vous décidez d'utiliser un autre compilateur, vérifiez que le code que vous avez produit (devant normalement fonctionner correctement chez vous) fonctionne aussi bien sur les ordinateurs de la DESI. Pour avoir la version du compilateur, utilisez la commande "**gcc –v**", qui devra donner le numéro de version « **3.4.2** ».
Par ailleurs, assurez-vous de la présence de l'option « pedantic » sur la ligne de compilation. (Cette option n'est pas activée par défaut dans l'utilitaire « devcpp ». Il faudra donc penser à l'activer).

Au total vous devez remettre **12 fichiers** (**9** fichiers h/cpp sauf si votre design nécessite plus que cela, les **2** fichiers in.txt et out.txt servant à simuler l'entrée et la sortie de votre programme et finalement le rapport). Pour éviter d'en oublier un lors de la remise, nous vous conseillons de regrouper ces 12 fichiers (ou plus) dans un seul fichier compressé **tp2h05.zip** et de remettre ce dernier comme suit :

1. commencez d'abord par vous connecter sur la machine « **remise** » comme il a été pratiqué dans la démo #01.
2. envoyez l'ensemble de vos fichiers par la procédure de remise électronique habituelle (Pour obtenir de l'aide sur cette commande, tapez dans un Xterm : man remise). Respectez les noms des fichiers.

   remise ift1169 tp2 tp2h05.zip

3. vérifiez que la remise s'est effectuée correctement.

   remise –v ift1169 tp2

Barème : ce TP1 est noté sur 13 points.

Compilation et respect des spécifications	**1**
Codage, commentaires etc.	**5**
Rapport	**3**
Tests (fournis et non fournis)	**4**

En plus du précédent barème, vous risquez de perdre des points dans les cas suivants ….

- La non remise électronique (volontaire ou par erreur) est sanctionnée par la note 0.
- Les programmes ne contenant pas d'en-tête, -1 point.
- Un programme qui ne compile pas : 0.
- Un programme qui compile mais ne réalise pas les choses prévues dans la spécification : 0.
- Les avertissements (warnings) non corrigés : cela dépend de la quantité! À partir de -0.25 et plus.
- Le non respect du nom du fichier entraîne une erreur de compilation donc un des points de la spécification n'a pas été respecté : 0.
- Aberration dans le codage : même si tous les chemins mènent à Rome, faites l'effort nécessaire pour éviter de prendre le plus long!

Des questions à propos de ce TP?

Une seule adresse : dift1169@iro.umontreal.ca

Pour faciliter le traitement de votre requête, inclure dans le sujet de votre email, au moins la chaîne: [IFT1169] et une référence au **travail02**.

Mise à jour

05-02-2008 diffusion