CARLOTTA CASTELLUCCIO

Mon projet contient la solution aux étapes 1-11 du TP C++ et de tous les étapes du TP Java Swing. Le makefile dans le dossier cpp sert à la compilation et à l'exécution du server; le fichier main contient le main avec lequel j'ai vérifié le correct fonctionnement des étapes 1-10.

Q1-Comme pour la fonction d'affichage, la fonction pour jouer l'objet ne modifie pas l'objet et elle doit être déclarée dans les classes **Photo** et **Video** et dans la **classe de base** afin de permettre un appel polymorphique sur la hiérarchie de classes. Cependant, contrairement à la fonction d'affichage, elle ne peut pas avoir d'implémentation au niveau de la classe de base (car a priori chaque type d'objet nécessite un utilitaire différent pour être joué). Comment appelle-t'on ce type de méthode et comment faut-il les déclarer ?

Si vous avez fait correctement ce qui précède, il ne sera plus possible d'instancier des objets de la classe de base. Pourquoi ?

Ce type de méthodes sont des méthodes abstraites, cela veut dire qui sont une spécification d' un concept qui doit être implémentée dans les sous-classes. Il faut les déclarer en utilisant le mot clé 'virtual' et en ajoutant '=0' après la définition.

Une classe qui a une méthode abstraite est une classe abstraite et donc elle ne peut plus être instanciée.

Q2- Cette boucle n'a pas besoin de connaître le type des éléments : elle doit pouvoir traiter de la même manière tous les objets dérivant de la classe de base.

Quelle est la propriété caractéristique de l'orienté objet qui permet de faire cela ? Qu'est-il spécifiquement nécessaire de faire dans le cas du C++ ? Quel est le type des éléments du tableau : est-ce que ce tableau contient les objets ou des pointeurs vers ces objets? Pourquoi ? Comparer à Java.

La propriété caractéristique de l'orienté objet qui permet un traitement uniforme des objets différents s'appelle polymorphisme de type. On doit créer un tableau des pointeurs de la classe de base, en manière que chaque élément peut pointer à une instance différente des sous-classes (photo ou vidéo dans ce cas spécifique). Sur chaque élément du tableau on peut appeler la méthode abstraite de la classe de base et sera toujours la bonne version de l' implémentation de cette méthode qui sera appelée.

En Java c'est pareil, parce que on utilise les références aux objets (qui sont une espèce de pointeurs), donc on doit créer un tableau de références aux objets de la superclasse et (grâce à la liaison tardive, i.e. la choix des méthodes est faites à l'exécution) toujours la correcte redéfinition de la méthode abstraite est appelée.

Q3- Que faut-il faire pour que l'objet Film ait plein contrôle sur ses données et que son tableau de durées des chapitres ne puisse pas être modifié (ou pire, détruit) à son insu ? (c'est l'objet qui doit pouvoir modifier ce qui lui appartient, pas les autres !)

Attention, le même problème se pose si un accesseur retourne directement ce tableau sans prendre les précautions nécessaires : la encore le contenu du tableau n'est pas recopié et l'appelant peut le modifier à sa guise. Quelle est la solution très simple que propose C/C++ pour éviter ce problème ?

Il faut implémenter une copie profonde dans le setter et dans le constructeur pour initialiser le tableau des durées des chapitres avec le tableau d' entiers passé en argument, en manière telle que on recopie la valeur pointée par chaque élément du deuxième tableau dans le correspondant élément du premier. Au contraire si on recopie les pointeurs de chaque élément on a pour chaque entier deux pointeurs qui pointe sur celle valeur, donc il pourrait être modifié à l' insu de l' objet, qui ne peut pas donner la garantie de la validité des ces données.

Le même problème se pose dans le getter : en effet on ne peut pas renvoyer directement le pointeur au tableau des durées des chapitres ; on doit le recopier dans un autre tableau et puis renvoyer le pointeur de cette copie.

Q4- Parmi les classes précédemment écrites quelles sont celles qu'il faut modifier afin qu'il n'y ait pas de fuite mémoire quand on détruit leurs instances ?

De même, la copie d'objets peut poser problème dans certains cas. Pourquoi et que faudrait-il faire ?

L'unique classe que on doit modifier est film, car c'est la seule à avoir un pointeur (en particulier un tableau) comme variable d'instance, donc il faut implémenter un destructeur qui détruit le pointé (dans le cas d' un tableau on utilise delete[] parce que on doit détruire tous les éléments du tableau).

Pour la même raison il faut faire attention dans le cas de la copie d' un objet qui contient un pointeur : il faut redéfinir le 'copy constructor' pour initialiser un objet avec un autre objet et l' operateur= pour l'affectation, en manière telle que on assure la copie profonde et pas la copie superficielle de tous le champs de l' objet.

Q5- Le groupe ne doit pas détruire les objets quand il est détruit car un objet peut appartenir à plusieurs groupes (on verra ce point à la question suivante). On rappelle aussi que la liste d'objets doit en fait être une liste de pointeurs d'objets. Pourquoi ? Comparer à Java.

En utilisant une liste de pointeurs on peut détruire le pointeur sans détruire le pointé et cela permet d' enlever un objet dans un groupe sans le détruire, en évitant de créer des pointeurs pendants.

En Java on n'a pas la possibilité de détruire un objet car cela c'est un devoir de la ramasse miettes : on peut seulement 'détruire' la référence à l' objet (en le déclarant null), donc l' objet continue à exister pour les autres groupes, jusqu'à tous les références à l'objet sont détruites (on obtient ce truc en C++ grâce aux smart pointers).

Q6- Les méthodes précédentes permettent d'assurer la cohérence de la base de données car quand on crée un objet on l'ajoute à la table adéquate. Par contre, ce ne sera pas le cas si on crée un objet directement avec new (il n'appartiendra à aucune table). Comment peut-on l'interdire, afin que seule la classe servant à manipuler les objets puisse en créer de nouveaux ?

On peut le faire en déclarant private les constructeurs (ou protected en cas de vidéo qui est superclasse de film) et en définissant la classe manager friend de toutes les autres. Dans cette manière les constructeurs peuvent être appelés seulement par la classe manager et pas à l'extérieur (comme par exemple dans le main).

Q7- Votre méthode processRequest() devra pouvoir accéder aux données de la classe créée à la question précédente. Sachant que cette méthode peut appartenir à n'importe quelle classe, quelle est la solution la plus simple ?

La solution plus simple c'est que la classe à la quelle processRequest() appartient aie un pointeur à un objet de la classe manager, afin qu' on puisse appeler sur cet instance tous les méthodes nécessaires.