TD 6 - Les associations

Programmation Orientée Objet

Objectif

– Comprendre le principe de l'implémentation d'une association et d'une composition en C++

1 Première partie

Ce sujet est tiré d'un exercice de Benoit Charroux.

Soit le diagramme des classes de la figure 1 modélisant une médiathèque où des adhérents peuvent faire 3 emprunts (ils empruntent un exemplaire d'une œuvre donnée). L'ensemble des adhérents est stocké dans une classe `Adherents`, et l'ensemble des œuvres de la médiathèque est conservé dans une classe `Œuvres`. Il y a deux types d'œuvres : des œuvres interprétées (limitées ici à des CD), et des œuvres non interprétées (uniquement des livres dans notre cas).

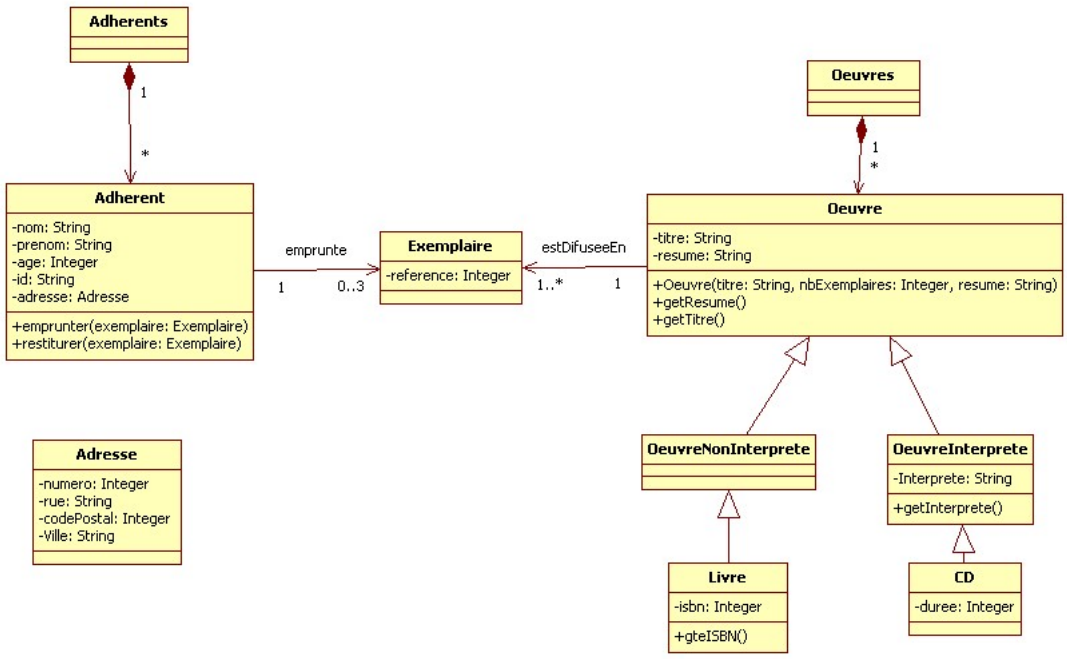

Fig. 1 – Diagramme de classe de l'application

Ecrivez la classe `Adherent`. Ajoutez-y un constructeur pour initialiser les données membres et éventuellement un destructeur.

Ecrivez la classe `Exemplaire` avec un constructeur pour initialiser la donnée membre reference.

Pour modéliser l'association unidirectionnelle `emprunte` entre les classes `Adherent` et `Exemplaire`, vous pouvez utiliser un tableau de 3 pointeurs. Pour mettre à jour cette association, ajoutez dans la classe `Adherent` une fonction membre appelée `addExemplaire`, qui ajoute un exemplaire à un adhérent, simulant ainsi un

emprunt. Ajoutez aussi une fonction appelée `removeExemplaire` qui modélise la restitution d'un exemplaire dans la médiathèque par un adhérent.

Complétez le programme suivant, puis testez vos classes avec :

```
void main(){
    Adherent belloir( ... ) ;
    Exemplaire exp1( ... ) ;
    belloir.addExemplaire( exp1 ) ;
    belloir.removeExemplaire( exp1 ) ;
}
```

On s'intéresse à présent à la relation de composition entre les classes `Adherents` et `Adherent`. Bien que ce soit une relation de type composé / composant, il est possible de la modéliser avec des pointeurs sur des adhérents. Il faudra cependant veiller à écrire correctement le destructeur de la classe `Adherents` pour qu'il détruise l'ensemble des adhérents.

Ecrivez la classe `Adherents`. Par simplicité, vous pouvez modéliser la relation de composition avec la classe `Adherent` par un tableau de pointeurs. Ajoutez à la classe `Adherents` des fonctions appelées `addAdherent` et `removeAdherent` permettant d'ajouter et de supprimer respectivement un adhérent de la médiathèque.

Ecrivez à présent la classe `Œuvre` ainsi que ses classes dérivées, puis écrivez un programme de test.

2 Deuxième partie

Comment modéliser l'association de 1 à n entre les classes `Œuvre` et `Exemplaire` ? Cette association doit être mise à jour dès la création d'une oeuvre comme dans le programme suivant où une nouvelle œuvre portant le titre de Don Juan et diffusée à 2 exemplaires est créée. C'est le constructeur de la classe `Œuvre` qui se charge de créer les exemplaires :

```
void main(){
    OEuvre donJuan(``Don Juan'', 2, ``C'est l'histoire...'' ) ;
}
```

On remarque que l'association de 1 à n entre les classes `Œuvre` et `Exemplaire` doit être mise à jour quand un adhérent emprunte un exemplaire : il faut alors commencer par mettre à jour la relation entre un adhérent et l'exemplaire qu'il désire, puis mettre à jour la relation entre l'exemplaire et l'oeuvre correspondante (modélisant ainsi qu'il y a un exemplaire de moins dans la médiathèque, et un exemplaire de plus pour un adhérent) ! Ecrivez alors les fonctions membres `emprunter` et `restituer` de la classe `Adherent`.