Programmation & Algorithmique 2

TP - Séance 6

26 mars 2021

Matière visée : Héritage et exceptions.

1 Héritage

1.1 Gestion d'une bibliothèque

On veut modéliser la gestion d'une bibliothèque : on définira un certain nombre de classes : `Main, Ouvrage, BiblioTab, Bibliotheque, Periodique, CD, Livre`. Les livres auront comme caractéristiques : auteur, titre, éditeur ; les périodiques : nom, numéro, périodicité ; les CDs : titre, auteur. De plus tous les ouvrages auront une date d'emprunt (potentiellement nulle), une cote (le numéro par ordre de création). On implémentera également sur chaque objet une méthode `toString()` renvoyant toutes les informations sur l'ouvrage sous forme d'une chaîne de caractères.

La classe `BiblioTab` permettra de stocker dans une structure les Ouvrages (ajout et suppression, la suppression prenant en argument la cote de l'ouvrage). Elle aura également une méthode `toString()` affichant le nombre d'ouvrages, puis chaque ouvrage successivement. La classe `Bibliotheque` sera simplement une version abstraite déclarant les mêmes méthodes que `BiblioTab` mais sans les implémenter. `BiblioTab` héritera de `Bibliotheque`.

La classe `Main` ne contiendra que la méthode main et testera la bibliothèque en y insérant et supprimant quelques ouvrages, puis en affichant le contenu de la bibliothèque.

1. Représentez les différentes classes dans un graphe d'héritage. On mettra en évidence pour chaque classe les méthodes et les champs qu'elle définit, redéfinit ou hérite. On souhaite que tous les champs soient déclarés privés et que l'on ne puisse y accéder de l'extérieur que par des méthodes ; et
2. Implémentez les classes ci-dessus. Pour la classe `BiblioTab` on utilisera un tableau de longueur suffisante (on vérifiera quand même à chaque opération d'insertion que l'on ne dépasse pas les bornes du tableau). Quel sont les inconvénients de cette méthode ?

Dans ce qui suit, on veut implémenter une deuxième version de la bibliothèque, que l'on appellera `BiblioList` et qui héritera également de `Bibliotheque`. Cette nouvelle implémentation utilisera la classe ArrayList.

1. Modifiez le minimum de choses dans la classe Main pour permettre l'utilisation de `BiblioList` ; et
2. Implémentez, à l'aide des ArrayLists, la classe `BiblioList`.

2 Exceptions

2.1 ConsoleReader

Créez une classe `ConsoleReader` contenant la méthode `int readInt(String prompt)` qui ne pourra, en aucun cas, lancer d'exceptions.

Cette méthode doit demander à l'utilisateur d'entrer un entier et retourner ce dernier. La méthode doit répéter la demande aussi longtemps que l'utilisateur n'a pas entré d'entier. Donnez tout de même la possibilité à l'utilisateur de quitter le programme.

Pour ce faire, veuillez utiliser la classe `Scanner` ou `JOptionPane` pour intéragir avec l'utilisateur.

2.2 ArrayLists bornés

Créez une classe `BoundedArrayList`. Un objet `BoundedArrayList` doit se comporter de la même façon qu'un `ArrayList` standard de Java, hormis le fait que celui-ci doit lancer une exception lorsque sa taille dépasse une borne donnée en paramètre du constructeur. Pour cet exercice vous devez donc :

1. créer votre propre exception *non testée* `BoundOutreachedException` afin de représenter l'erreur ; et
2. implémenter uniquement la méthode `boolean add(Object o)` en s'assurant que le `BoundedArrayList` respecte sa propriété.

2.3 Attraper votre exception

Créez une classe test `BoundedArrayListTest` qui se comporte comme suit :
— elle utilise la classe `ConsoleReader` pour demander à l'utilisateur d'enter deux entiers k et l ;
— elle crée un `BoundedArrayList` b de borne k ;
— elle ajoute l fois un objet o (que vous êtes libre de choisir) dans b ;
— si la borne n'est pas dépassée après l'ajout de tous les éléments, la méthode imprime `"Remplissage termine avec succes"` ; et
— si la borne est dépassée, la méthode imprime `"Echec du remplissage: borne depassee"`.