TP n° 2
Licence Informatique (L2)
« Programmation objet avancée »

F. BERTRAND

Année universitaire 2020–2021

1 Hiérarchie d'héritage

À partir des classes `BiblioMM`, `CD`, et `DVD` (présentes dans le code source disponible sur Moodle, répertoire `Biblio`) :

1. Construire une hiérarchie d'héritage (comme illustré par la figure 1) en introduisant une classe `EltMM` super-classe de `CD` et `DVD` permettant de factoriser les éléments communs aux deux classes.

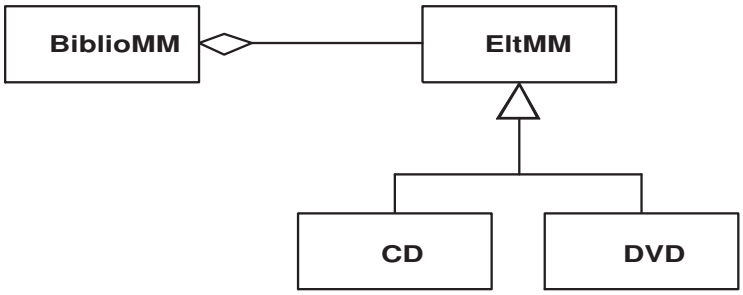

FIGURE 1 – La hiérarchie d'héritage à construire

2. Ajouter dans la classe `EltMM` une méthode `toString()` permettant d'afficher les attributs propres à `EltMM`. Puis utiliser cette méthode dans les versions de `toString()` définies dans `CD` et `DVD`.
3. Modifier la représentation interne de `BiblioMM` de manière à n'avoir plus qu'une seule liste ;
4. Ajouter une méthode `rechercherTitre()` prenant en paramètre une chaîne de caractères, correspondant au titre d'un élément multimédia et retournant une liste d'éléments multimédia dont le titre correspond au titre recherché.
5. Ajouter une méthode `emprunterTitre(String titre)` à la classe `BiblioMM` permettant d'emprunter un élément multimédia. Cette méthode fera appel à la méthode `rechercherTitre()`. L'emprunt ne pourra se faire que si le résultat de la recherche est constitué d'un seul élément et si celui-ci est disponible.
6. Construire une classe de test possédant une méthode `main` qui, après avoir créer une bibliothèque, insérera un `CD` et un `DVD` dans cette bibliothèque, puis testera la méthode `rechercherTitre()` et affichera l'objet retourné. Puis tester deux emprunts successifs d'un même `CD` (sans qu'il soit rendu) en s'assurant que cela n'est pas possible.

7. Définir une méthode donneType ayant la signature suivante :

```
1 public String donneType()
```

Cette méthode retournera une chaîne de caractères indiquant le type de l'objet. Par exemple, pour la classe EltMM, elle retournera la chaîne de caractères "EltMM".

Tester cette méthode en créant un CD et un DVD avec le même titre puis appeler rechercherTitre() avec ce titre en paramètre :
— si un élément du résultat est un CD, alors on appellera la méthode donneArtiste;
— si un élément du résultat est un DVD, alors on appellera donneRealisateur.
Exemple :

```
1 ArrayList<EltMM> elts = bibliotheque.rechercherTitre("Moby Dick");
2 for(EltMM elt : elts) {
3   System.out.println(elt.donneType()); // ex. affichage "DVD"
4 }
```

2 Transport de marchandises

On souhaite représenter (de manière simpliste) le coût de transport de colis. On dispose d'une classe Colis disponible dans le code source fourni.

On souhaite définir une classe Conteneur (dont la documentation est fournie, fichier Conteneur.html) qui permettra de transporter des colis qui iront tous au même endroit (même distance). La condition de chargement d'un colis dans le conteneur sera de s'assurer que le volume du colis, ajouté aux colis déjà présents, ne dépasse pas le volume maximal (ici le poids n'est pas pris en compte, acheminement par voie maritime). Le coût de transport sera égal à la distance multipliée par le poids.

Puis on souhaite définir une classe ConteneurUrgent qui aura pour conditions de chargement d'un colis :

1. la condition de la classe Conteneur;
2. plus celle de ne pas dépasser un poids maximal (les conteneurs urgents seront acheminés par voie aérienne).

Le coût de transport d'un colis pour ce type de conteneur sera égal à deux fois à celui d'un conteneur normal.

Puis définir une classe de test permettant de vérifier que les deux classes Conteneur et ConteneurUrgent) ont été correctement implémentées (pour chaque méthode, on comparera le résultat de l'appel avec la valeur attendue).

Dans toutes les classes développées, les attributs seront qualifiés private et si nécessaire des accesseurs seront définis.