Licence Mathématiques-Informatique 2ème année 11 Juin 2006

Université d'Artois
U.F.R. des Sciences Jean Perrin

Programmation Orientée Objet

Corrigé de l'examen

Exercice 1 : La mediathèque

Une médiathèque propose à ses adhérents la possibilité d'emprunter des livres et des disques. On appelle dans la suite de l'énoncé ouvrage, tout exemplaire d'un livre ou d'un disque de la médiathèque.

Elle souhaite organiser la mémorisation de son fonds comme indiqué ci-dessous :

- pour chaque livre, on mémorise le nom de chaque auteur, ainsi que le titre, le nom de l'editeur, l'année de publication, le nombre de pages et un code qui identifie l'ouvrage d'une manière unique ;
- pour chaque disque, on mémorise le nom de chaque auteur (cela peut-être le nom d'un groupe, ou d'un chanteur, ou d'un compositeur, ou de plusieurs compositeurs, chanteurs, etc ...), le titre de l'album mais aussi le titre de chaque morceau du disque, l'éditeur et l'année de publication, ainsi qu'un code qui identifie l'ouvrage d'une manière unique ;
- le code unique de chaque ouvrage est une chaîne de caractères dont les trois premier caractères sont les trois premières lettres[1] du premier auteur (ou moins si son nom a moins de trois caractères, ou " ?" si le nom du premier auteur est la chaîne vide), suivi d'un numéro. Ce numéro correspond à l'ordre de création de l'ouvrage dans le fond documentaire.

Q 1 . Donnez les classes nécessaires pour modéliser les ouvrages de la médiathèque. Vous donnerez la description des classes en termes d'attributs, et signatures de méthodes et de constructeurs. Vous implémenterez les constructeurs, et la méthode `toString` de chaque classe.

Réponse :

Dans un premier temps, on peut proposer la modélisation suivante :

```
public class Ouvrage {
    /** un compteur d'instances d'ouvrages */
    private static int cptOuvrages;
    /** les auteurs de l'ouvrage */
    private List<String> lesAuteurs;
    /** le titre de l'ouvrage */
    private String titre;
    /** le code de l'ouvrage */
    private String code;
    /** l'editeur de l'ouvrage */
    private String editeur;
    /** l'annee de publication */
    private anneePublication;

    Ouvrage(String titre, List<String> lesAuteurs,
            String editeur, int annee){
        this.titre = titre;
        this.lesAuteurs = lesAuteurs;
        cptOuvrages++;
        if (lesAuteurs == null
            || lesAuteurs.get(0) == null
```

[1] On rappelle que la méthode `substring(int indice_debut, int indice_fin)` de la classe `String` extrait une sous-chaîne de caractères.

```
            || lesAuteurs.get(0).length() == 0)
            identifiant = "?";
        else if (lesAuteurs.get(0).length() < 3)
            identifiant =
                 lesAuteurs.get(0).substring(0,lesAuteurs.get(0).length());
        else
            identifiant = lesAuteurs.get(0).substring(0,3);
        identifiant += cptOuvrages;
        this.editeur = editeur;
        anneePublication = annee;
    }
    // et accesseurs sur
    // le titre, le code, l'éditeur, l'année de publication
    // pour les auteurs : le nom du premier auteur ou du nieme
    public getAuteur(int indice){
      if (indice < lesAuteurs.size())
        return lesAuteurs.get(indice);
      else return null;
    }

    public String toString(){
        StringBuffer sb = new StringBuffer();
        sb.append("Code : "+code+" -- "+titre+"\n");
        sb.append(anneePublication+" -- "+editeur+"\n");
        sb.append("Auteurs : ");
        for(String auteur : lesAuteurs)
            sb.append(auteur+" --");
        return sb.toString();
    }

class Livre extends Ouvrage {

    private int nbrePages;

    Livre(String titre, List<String> lesAuteurs, String editeur,
          int annee, int nbrePages){
        super(titre, lesAuteurs, editeur, annee);
        this.nbrePages = nbrePages;
    }

    public String toString(){
        return super.toString()+"\n"+"Nbre de pages : "+nbrePages;
    }
}

class Disque extends Ouvrage {

    private List<String> lesMorceaux;

    Disque(String titre, List<String> lesAuteurs, String editeur,
           int annee, List<String> lesMorceaux){
        super(titre, lesAuteurs, editeur, annee);
        this.lesMorceaux = lesMorceaux;
    }
```

```
    public String toString(){
        StringBuffer sb = new StringBuffer();
        sb.append(super.toString());
        sb.append("\n");
        sb.append("Titres des plages : ");
        for(String morceau : lesMorceaux)
            sb.append(morceau+" --");
        return sb.toString();
    }
}
}
```

Quelques remarques :

- il y a d'autres solutions pour compter le nombre d'ouvrages dans la médiathèque (et donc gérer la génération du code pour chaque ouvrage) : on pouvait aussi décider de compter le nombre d'ouvrages à chaque fois qu'il y a un ajout dans la classe `Mediatheque`, il suffisait alors supprimer l'attribut `cptOuvrages` et de passer un compteur dans le constructeur d'ouvrage ;
- pour les accesseurs en lecture, il ne faut pas fournir un accesseur direct sur la liste des auteurs, car dans ce cas-là vous permettriez une modification de vos auteurs par n'importe quel objet externe. Il vaut mieux fournir le nom d'un auteur en fonction de son indice, par exemple. Si vous souhaitez vraiment fournir directement la liste des auteurs, alors il faut faire une copie de celle-ci, afin de préserver l'intégrité de votre `Ouvrage` :

```
public List<String> getLesAuteurs(){
  return new ArrayList<String>(lesAuteurs);
}
```

- en regardant la classe de la `Mediatheque`, notamment `estEmprunte(String code)`, on sait que la classe `Ouvrage` va devoir au moins avoir une methode `emprunte(Mediatheque m)`. Nous terminerons l'impolémentation des classes `Ouvrage`, `Livre` et `Disque` après la classe `Mediatheque`.

Une médiathèque souhaite organiser la mémorisation de son fonds comme indiqué ci-dessous :

- dans le fond documentaire, les livres et les disques sont enregistrés dans une même structure. On accède à un ouvrage par son code (les codes seront mémorisés au niveau de la médiathèque et au niveau de chaque ouvrage) ;
- les ouvrages empruntés sont par contre mémorisés dans deux structures différentes, une pour les disques et l'autre pour les livres. Un disque emprunté est donc mémorisé à la fois dans le fonds documentaire et dans la structure des disques empruntés ;

Q 2 . Complétez la classe `Mediatheque`. Pour la partie qui concerne l'action d'emprunter ou de rendre un ouvrage, c'est l'ouvrage qui indiquera à la médiathèque (passée en paramètre) quelle structure doit être mise-à-jour.

Réponse :

```
public class Mediatheque {
    /** Le fond documentaire : on accède à un ouvrage par son code */
    private Map<String,Ouvrage> lesOuvrages = new HashMap<String,Ouvrage>();

    /** Les disques et les livres empruntés */
    private List<Disque> lesDisquesEmpruntes = new ArrayList<Disque>();
    private List<Livre> lesLivresEmpruntes = new ArrayList<Livre>();

    public void ajouteLivre(String titre, List<String> lesAuteurs,
                            String editeur, int annee, int nbrePages){
        Livre unLivre = new Livre(titre,lesAuteurs,editeur,annee,nbrePages);
        lesOuvrages.put(unLivre.getCode(),unLivre);
```

```
}

public void ajouteDisque(String titre, List<String> lesAuteurs, String edite
                         int annee, List<String> lesMorceaux){
    Disque unDisque = new Disque(titre,lesAuteurs,editeur,annee,lesMorceaux)
    lesOuvrages.put(unDisque.getCode(),unDisque);
}


public void afficheTousLesOuvrages(){
    for(Ouvrage o : lesOuvrages.values())
      System.out.println(o);
}

public List<Ouvrage> ouvragesDeLAuteur(String auteur){
    // ... on ajoutera une méthode aParticipe() à la classe Ouvrage ensuite
    List<Ouvrage> ouvrages = new ArrayList<Ouvrage>();
    for(Ouvrage o : lesOuvrages.values())
      if (o.aParticipe(auteur))
         ouvrages.add(o);
    return ouvrages;
}

public boolean est_Il_Emprunte(Ouvrage o){
    // bcp de possibilités ...
    // par exemple :
    // je regarde s'il est dans les disques empruntés
    for(Disque d : lesDisquesEmpruntes)
      if (d.getCode().equals(o.getCode()))
         return true;
    // si je ne l'ai pas trouvé, je regarde s'il est dans les livres
    for(Livre l : lesLivresEmpuntes)
      if (l.getCode().equals(o.getCode()))
         return true;
    return false;
}

public void emprunte(String code){
  // on emprunte l'ouvrage dont le code est donné
  // en paramètre
  lesOuvrages.get(code).emprunte(this);
}
// sur la méthode précédente, rien  à faire, par contre, on voit :
// que lesOuvrages possede une méthode get(String code)
// qui permet de retourner un Ouvrage
// que la classe Ouvrage doit posséder une méthode emprunte(Mediatheque m)
// dont le comportement est précisé dans l'énoncé

public void rend(String code){
  // on rend l'ouvrage dont le code est donné en paramètre
  // on faut comme pour la méthode précédente
  lesOuvrages.get(code).rend(this);
}
```

```
    // et voilà les méthodes qui seront utilisées
    // dans les méthodes rend et emprunte
     public void ajouteUnLivreEmprunte(Livre l){
         lesLivresEmpruntes.add(l);
     }

     public void ajouteUnDisqueEmprunte(Disque d){
         lesDisquesEmpruntes.add(d);
     }

     public void supprimeUnLivreEmprunte(Livre l){
         lesLivresEmpruntes.remove(l);
     }

     public void supprimeUnDisqueEmprunte(Disque d){
         lesDisquesEmpruntes.remove(d);
     }

}
```

Quelques remarques en vrac :

1. `lesOuvrages` : pour accéder à un ouvrage par son code, c'était une `Map` qu'il fallait utiliser. On aurait pu aussi avoir une classe *sur mesure*, par exemple :

   ```
   class FondDocumentaire {
     private List<Ouvrage> ouvr = new ArrayList<Ouvrage>();
     public add(Ouvrage o){
       ouvr.add(o);
     }
     public Ouvrage get(String code){
       for (Ouvrage o : ouvr)
         if (o.getCode().equals(code))
           return o;
       return null;
     }
   }
   ```

2. Et pourquoi pas le code suivant :

   ```
       public void ajouteLivre(Livre l){ lesOuvrages.put(l.getCode(),l); }
       public void ajouteDisque(Disque d){ lesOuvrages.put(d.getCode(),d); }
   ```

   ?

   Parce qu'une seule méthode `ajouteOuvrage` aurait suffit pour ces deux cas :

   ```
       public void ajouteOuvrage(Ouvrage o){ lesOuvrages.put(o.getCode(),o); }
   ```

   et je n'aurais pas demandé deux méthodes...

3. Pour la méthode `estIlEmprunte(Ouvrage o)`, on ne peut pas se contenter d'un

   ```
   return lesDisquesEmpruntes.contains(o) || lesLivresEmpruntes.contains(o);
   ```

   car il y aura un probleme de typage (`lesDisquesEmpruntes` ne peut contenir que des `Disque`s, `lesLivresEmpruntes` que des `Livre`s, or `o` est soit l'un, soit l'autre) ;

4. on doit donc compléter les classes `Ouvrage`, `Disque` et `Livre` :

   ```
   public abstract class Ouvrage {
     public abstract void emprunte(Mediatheque m);
     public abstract void rend(Mediatheque m);
   ```

```
  public boolean aParticipe(String auteur){
    for(String aut : lesAuteurs)
      if (aut.equals(auteur))
        return true;
    return false;
  }
}
public class Livre {
  public abstract void emprunte(Mediatheque m){
    // je sais que je suis un Livre
    m.ajouteUnLivreEmprunte(this);
  }
  public abstract void rend(Mediatheque m){
    // je sais que je suis un Livre
    m.supprimeUnLivreEmprunte(this);
  }
}
public class Disque {
  public abstract void emprunte(Mediatheque m){
    // je sais que je suis un Disque
    m.ajouteUnDisqueEmprunte(this);
  }
  public abstract void rend(Mediatheque m){
    // je sais que je suis un Disque
    m.supprimeUnDisqueEmprunte(this);
  }
}
```
