TD 11 - TP magasin CD - Tris de CDs

Objectif pédagogique

A l'issue de ce TP, vous devrez

- avoir réfléchi à la manière d'effectuer des tris
- avoir élaboré des diagrammes de séquence sur des cas simples

Ce TP est le début d'une suite de TPs centrée sur les diagrammes de séquences. Ces TPs vont se focaliser sur la notion de comportement d'une application.

Le TP fil rouge consistera à gérer un magasin de CD dont on vous donne les fichiers. Le TP sera à rendre ainsi que les différents diagrammes de séquences et diagrammes de classe demandés dans les TPs.

Attention :
Vous penserez à chaque fois à faire les diagrammes demandés, écrire la javadoc et respecter les conventions de nommage.

Rendu

L'ensemble de ces Tps sera contenu dans un dépôt git. Ce dépôt devra être nommé: "`2016_coo_cd_`*`login1`*`_`*`login2`*".

1 Mise en place du sous-projet git

Question 1

Créer le dépôt git "`2016_coo_cd_`*`login1`*`_`*`login2`*" destiné à contenir vos fichiers et partagez ce dépôt avec votre enseignant.

Question 2

Ajouter dans votre dépôt un fichier `README.txt` avec vos noms et prénoms.

2 Classes fournies

2.1 Classes de Données `Magasin`, `CD` et `InfoPiste`

2.1.1 Descriptif des classes

Ce sujet vous propose des classes déjà écrites permettant de charger et de modéliser un magasin. Ces classes **ne sont donc pas à écrire**. Il s'agit des classes suivantes:

- la classe `InfoPiste` qui stocke les données associées à une piste de CD ;
- la classe `CD` qui modélise un CD avec un titre, un nom d'artiste et un ensemble de pistes ;
- la classe `Magasin` qui modélise un magasin possédant plusieurs CDs.

2.1.2 Diagramme de classe

Le diagramme de la figure 1 décrit le contenu et les relations (simples) entre ces classes. Ce diagramme de classe est une première étape pour vous permettre de comprendre le contenu des classes transmises, soyez sur de bien le comprendre avant de passer à la suite.

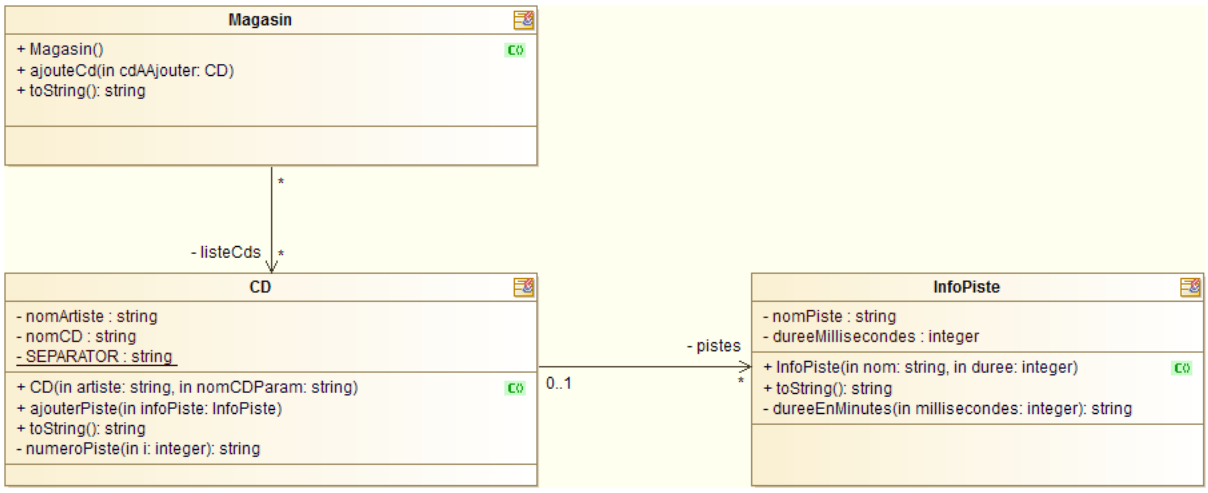

Figure 1: Diagramme de classe décrivant l'organisation des classes `InfoPiste`, `CD` et `Magasin` fournies sur arche.

2.2 Chargement des fichiers

2.2.1 Formats de fichier

Les fichiers proposés dans ce TP sont des fichiers XML provenant de la base de données `http://musicbrainz.org/`. Il vous est donc possible d'ajouter vos propres fichiers à ce TP à partir des fichiers XML proposés [1].

[1] Il sera même possible une fois que vous aurez eu les modules de "programmation réseau" de programmer une application capable de se connecter directement sur le site et de récupérer ces informations suite à une

Les fichiers utilisés sont les fichiers décrivant le contenu d'un album. Ce type de fichier s'obtient à partir `musicbrainz` en sélectionnant les onglet `release` puis en cliquant sur détail [2].

Question 3
Télécharger la base de test **fournie sur arche** et ajouter la au dépôt (un seul membre du binôme se charge de faire l'ajout pour éviter les conflits).

Attention :
Une base de test vous est déjà fournie sur arche. Ne perdez pas de temps à constituer votre base de test, vous pourrez faire cela en dehors des séance si l'idée vous intéresse.

2.2.2 Classes de chargement

Pour charger les CDs en mémoire, deux classes de chargement sont **fournies sur arche**:

- la classe `ChargeurCD` permettant de construire un CD à partir d'un nom de fichier;
- la classe `ChargeurMagasin` permettant de construire un magasin à partir d'un nom du répertoire contenant les fichiers XML descriptifs des CDs.

1.3

Ces deux classes de chargement fonctionnent de manière analogue. Chacune d'entre elles possède un constructeur qui prend en paramètre une chaine de caractère décrivant la localisation des fichiers XML et possède une méthode qui charge les données, crée l'objet choisi (un CD ou un Magasin en fonction du chargeur) et le retourne:

- pour la classe `ChargeurCD`
  - ☐ le constructeur prend en paramètre un nom de fichier;
  - ☐ la méthode `chargerCD` retourne un CD construit à partir du nom passé au constructeur.
- pour la classe `ChargeurMagasin`
  - ☐ le constructeur prend en paramètre un nom de répertoire;
  - ☐ la méthode `chargerMagasin` retourne un magasin construit à partir de tous les fichiers contenu dans le répertoire passé à la construction[3].

2.3 Compréhension du diagramme de séquence (15 min)

Le diagramme de séquence de la figure 2 décrit de manière plus précise le comportement de chargement pour un `ChargeurMagasin`.

requête

[2] Par exemple, la page `http://musicbrainz.org/release/6bfba6d5-71fc-454b-b3a0-63632a1459fa` décrit un album de Justin Bieber - My Worlds.

[3] `ChargeurMagasin` utilise `ChargeurCD` sur chacun des fichiers XML du répertoire.

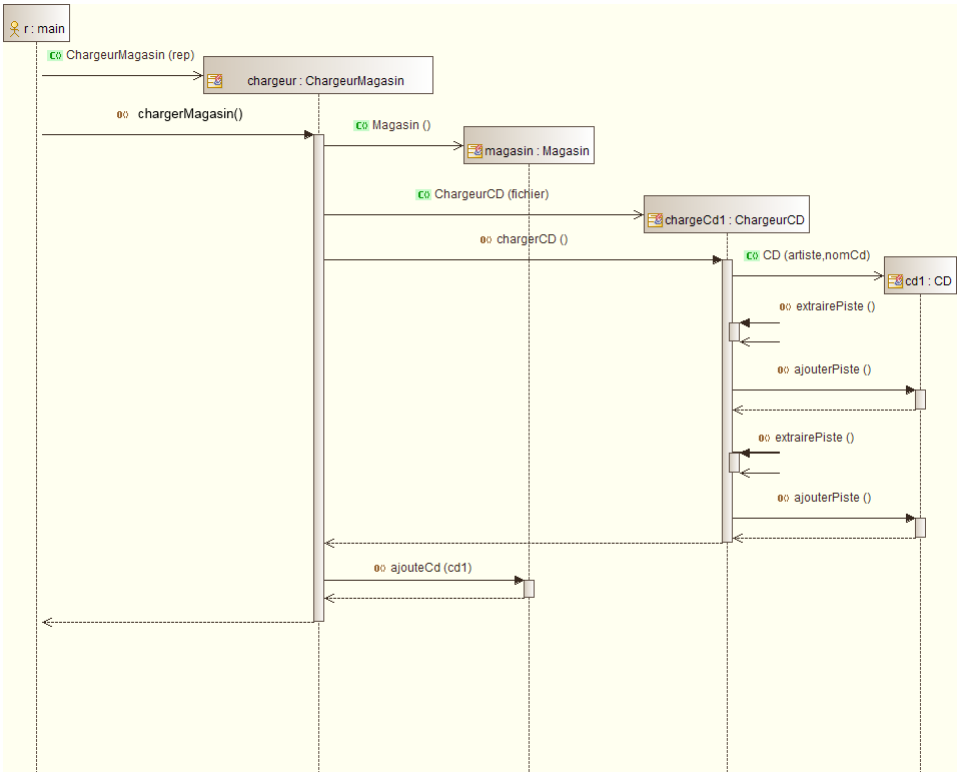

Figure 2: Diagramme de séquence décrivant le comportement de chargement des fichiers d'un magasin (code déjà écrit dans les classes fournies).

Question 4

Dans un fichier texte `chargement.txt`, écrivez votre compréhension du diagramme de séquence ci-dessus et ajoutez ce fichier texte à votre dépôt (il fait partie des résultats attendus dans ce TP).

2.4 Écriture de test (15 min)

On souhaite écrire une classe JUnit permettant de vérifier que la méthode `chargerMagasin` de la classe `ChargeurMagasin` fonctionne correctement.

- lorsque qu'un répertoire valide est donné (le répertoire d'exemple), la méthode `chargerMagasin` ne lève pas d'exception et retourne bien un magasin avec la bonne taille.
- lorsque qu'un répertoire non valide est donné (répertoire inexistant), la méthode `chargerMagasin` lève bien une `FileNotFoundException`.

Vous trouverez l'information pour vérifier la levée d'exception dans le polycopié du module (section 2.5.3).

Question 5
Ecrire la classe de test correspondante.

3 Tri de CDs

3.1 Conception des tris (10 min)

Une classe est censée être responsable de ses données. On souhaite ainsi que la comparaison entre cds se fasse dans la classe `CD`.

Un diagramme de séquence est un moyen de réfléchir aux méthodes utiles à créer dans les classes existantes pour ajouter une fonctionnalité de tri. Il **ne s'agit pas de faire apparaître** les listes et l'ensemble de l'algorithme dans le diagramme de séquence mais juste les méthodes utiles et l'ordre général dans lequel elles seront appelées (un diagramme de séquence n'est pas un algorithme).

Question 6
Réfléchir à la manière de concevoir un tri par titre de CDs et dessiner sur papier le diagramme de séquence correspondant. Appeler ensuite votre enseignant pour valider ce schéma.

Question 7
Modifier les classes existantes en ajoutant les signatures des méthodes présentées dans votre diagramme de séquence et utiles à l'écriture du code de tri. Vous pouvez compléter ces méthodes par des `throw new Error()`

3.2 Tri par Album (30 min)

Il existe différents moyens d'effectuer un tri. Nous vous proposons de faire un tri par `selection`. Pour rappel[4], un tri par sélection fonctionne de la manière suivante

- créez une liste vide destinée à contenir les éléments triés;
- trouvez dans la liste à trier l'indice du plus petit élément (recherche du minimum);
- insérez cet élément en fin de la nouvelle liste et supprimez le de la liste initiale;
- recommencez l'opération jusqu'à avoir tout trié.

Question 8
Écrire les algorithmes et le code JAVA des méthodes ajoutées.

[4] Vous êtes censés connaitre (ou savoir retrouver) au moins un algorithme de tri, cela fait partie des compétences de base en informatique.

3.3 Test du Tri par Album (15 min)

Question 9

Écrire une classe de test JUnit vérifiant que le tri fonctionne correctement (on se limitera à valider la position du premier et du dernier élément après le tri).

3.4 Tri par Artiste (10 min)

Question 10

Écrire une méthode de tri basée sur les noms des artistes et testez là.

3.5 Réflexion (10 min)

Question 11

Réfléchir à la manière dont le code des méthodes de tri a été conçu et mettez en évidence les problèmes que cela peut poser (par exemple si on souhaite trier par nombre de pistes).

Rendu

A l'issue de ce TP, vous devriez

- avoir lu et compris un diagramme de séquence simple ;
- savoir encapsuler les opérations de comparaisons dans la classe CD (une classe est responsable de ses propres données).