TD 8
Collections : `List`, `Set`

1 Rappel de Cours

Jusqu'à présent en PO, vous avez créé des objets et modélisé différentes structures de données afin de comprendre les notions de conceptualisation, modélisation, et utilisation.

Comme dans beaucoup de langages, un certain nombre de concepts souvent utilisés en programmation sont déjà définis/modélisés et accessibles *via* l'utilisation de **bibliothèques**.

C'est le cas en JAVA pour les structures permettant d'enregistrer un ensemble de données, aussi appelées `Collections`.

Ce TD a pour objectif de vous faire découvrir ces bibliothèques qui reprennent les notions vues lorsque vous avez créé des listes, ou des arbres, mais aussi les notions d'héritage.

2 Exercices

Exercice 1.

Question 1. Ouvrez sur votre ordinateur ou sur votre smartphone la documentation javadoc de l'interface `List`. Retrouvez les principales implémentations de cette classe évoquées en CM. Quelles sont leurs différences ?

Question 2. Au vu de ces différences, dites quelle implémentation de `List` est la plus pertinente dans les cas ci-dessous :

1. Liste représentant la file d'impression d'une imprimante. Les jobs rentrent en fin de Liste, et sont traités dans leur ordre d'arrivée.
2. Liste représentant le top-N des chansons les plus écoutées sur Spotify. La chanson en ième position est la ième plus écoutée.
3. Liste des étudiants du module de PO. La Liste est construite une fois au début du semestre, puis est peu modifiée. Les traitements consistent à mettre à jour la présence et les notes de chaque étudiant (en appelant des méthodes de la classe Etudiant).
4. Liste des onglets d'un navigateur web. La Liste est très dynamique : on peut ajouter des onglets et en retirer n'importe où dans la Liste.
5. Liste représentant une ligne dans une image, chaque élément de la ligne est un pixel.

Question 3. Écrivez la ligne d'initialisation d'une Liste, de la forme `List<...> x = new ...<...>()`, dans les cas ci-dessous :

1. Liste d'entiers, l'implémentation choisie doit permettre des accès aléatoires rapides ;
2. Liste de String, l'implémentation choisie doit permettre des suppressions rapides n'importe où dans la Liste ;
3. Liste de Pixel, l'implémentation choisie doit permettre des accès aléatoires rapides ;
4. Liste de ? ? ? (à trouver) ? ? ?, le tout doit représenter une image composée de pixels ;
5. Liste de ? ? ? (à trouver) ? ? ?, le tout doit représenter l'ensemble des lignes d'un texte, chaque ligne est composée de mots. Ce texte est composé sur un traitement de texte : chaque ligne est susceptible de changer et d'être supprimée à tout moment.

Exercice 2. Examen décembre 2014
Le but de cet exercice est d'écrire un objet `ControleParental` qui doit permettre d'empêcher l'accès à certains sites web. Les spécifications sont les suivantes :

- La liste des sites interdits a déjà été préparée et est fournie en paramètre au constructeur sous la forme d'une `List<String>` : vous avez juste à la stocker.
- En plus du constructeur, votre objet doit avoir une méthode :
  `public boolean siteOK(String url)`
  Cette méthode renvoie `true` si `url` n'est pas un site interdit, et `false` sinon.
  Attention ! On veut que la méthode `siteOK` réponde vite, même si la liste des sites interdits est très longue. Choisissez bien la structure de données pour stocker les sites interdits.

Question 1. Selon les spécifications données, Proposez une implémentation pour la classe `ControleParental`.

Exercice 3.
On veut réaliser une classe `Presence` qui permet de gérer la présence des étudiants en PO. Chaque étudiant est représenté par un entier entre 0 et $N-1$. Les méthodes de `Presence` sont :

- `public void ajoute(int numEtudiant, String seance)` : indique que l'étudiant
- `numEtudiant` a été présent à la séance `seance` (les séances sont de la forme : « cm4 », « tp6 »,...)
- `public List<String> getPresence(int numEtudiant)` : renvoie la liste des séances où l'étudiant `numEtudiant` a été présent.
- `public List<Integer> getPresenceSummary()` : renvoie une liste de N éléments où l'élément i indique le nombre de séances où l'étudiant i a été présent.

Question 1. Proposez une implémentation pour la classe `Presence`. Cette classe doit avoir un seul attribut, cet attribut doit être une collection composée (par exemple une `List` de `List`). Le constructeur de `Presence` prend comme paramètre le nombre N d'étudiants.