Métro : itérateurs, I/O, try, HashSet, Objet, @Override

Vérifiez les conventions à suivre pour l'écriture du code (cf Moodle). Si vous masquez des méthodes utilisez impérativement la notation **@Override**.

L'objet de ce TD est de préparer le terrain pour les TD qui suivent. Vous allez analyser le contenu d'un fichier qui contient les lignes du métro parisien (simplifié) pour créer des objets.

Récupérez le fichier **Metro.txt** sur le Moodle. Ouvrez le dans un fichier texte et essayez d'analyser le contenu (essayez de comprendre la structure du fichier). Les consignes qui suivent vous proposent une organisation en méthodes et objets compatibles avec la suite du TD (TD3). Suivre ces consignes vous simplifie la suite des TD. S'il vous manque des informations, faites ce qui vous semble logique d'un point de vue code et organisation (donc, imaginez ce que vous pouvez en faire).

1. Commencez par créer une fonction **lireMetro(String nomDeFichier_) throws IOException** capable d'analyser ce fichier et d'en extraire le numéro de la ligne et chaque couple de noms de station. Vous utiliserez un ***fileReader***, un ***BufferedReader***, la construction ***try with resources*** de Java 7 et un ***StringTokenizer***.

2. Créez une classe **Ligne**. Cette classe permet de créer une ligne de métro (avec le numéro de la ligne).

3. Créez une classe **Station** : chaque station possède un nom et la liste des lignes auxquelles elle appartient. La liste des lignes d'une station est un ensemble (**Set**). Vous utiliserez un **HashSet** de **Ligne**s.

4. Créez une méthode d'instance de **Station** qui permet de rajouter une ligne à la station :

   ```
   public void ajouteLigne(Ligne ligne_) {
       this.lignes.add(ligne_);
   }
   ```

   Comme this.**lignes** est de type **Set** (ici **HashSet**), le rajout d'une nouvelle ligne se fait sans répétition (un **Set** contient des éléments sans répétition). Vérifiez.

5. En fait, non ça ne marche pas (et c'est normal). Pourquoi ? Comment fonctionne **HashSet** ? Modifiez la Classe **Ligne** pour que cela fonctionne. Vérifiez.

6. Modifiez **Station** pour qu'on puisse itérer sur ses lignes de la manière suivante :

   ```
   Station station = …… ;
   for(Ligne l : station) {
       ….
   }
   ```

7. Créez une classe **Reseau** : un réseau contient les stations et un graphe qui représente les connexions entre stations. À partir du code de **lireMetro**, créez la méthode de classe :

   public static Reseau CreeReseauAPartirDuFichier(String nomDeFichier_) throws IOException

8. Complétez la classe **Reseau** pour rajouter les méthodes nécessaires pour avoir accès aux informations sur le réseau. On considère que vous ne pourrez pas modifier le réseau. Sans savoir ce qu'on va faire de ces informations, trouvez (et implémentez) une couverture suffisante de méthodes pour l'accès aux informations sur les stations, lignes et graphe.

9. Fabriquez une méthode qui renvoie le nom des stations voisines d'une station donnée :

   public String[] stationsVoisinesDe(String nomDeStation_)