Java DUT 1 Feuille TP8
Université Paris-Est Marne-la-Vallée

Exercice 1.—

On veut écrire un programme permettant de manipuler les joueurs d'un jeu de rôles en ligne où le nombre de joueurs est important.

Pour chaque joueur, on dispose de données diverses dont au moins

- un pseudo de type `String` qui est un identifiant unique dans le jeu;
- un nom de type `String`;
- un niveau du personnage qui sera un entier;
- la liste des pseudos de ses amis dans le jeu.

1. Écrire un classe `Player` contenant une map de nom `data` dont le type est `Map<String,Object>` et qui associe aux chaînes "pseudo", "name", "level", "friends" les données correspondantes pour le joueur. On écrira un constructeur, un getter et une méthode `toString` pour la classe `Player`.
2. Écrire une classe `PlayerTest` qui permet de lire une ligne de données d'un joueur dans un fichier `data.txt` et construit un `Player` ayant ces données. Une ligne de données sera sous la forme suivante: entre crochets, le pseudo, le nom, le niveau, et les amis. Chaque champ entre les crochets est séparé par une virgule. Par exemple, une ligne de `data.txt` est

   ```
   <mpb, beal, 5, dp, mac, zip, pa>
   ```

   et l'affichage du `Player` sera

   ```
   {friends=[zip, pa, mac, dp], level=5, pseudo=mpb, name=beal}
   ```

Exercice 2.— Écrire une classe `Game` qui permet de stocker l'ensemble des joueurs. La classe `Game` doit permettre de trouver rapidement les données d'un joueur ayant un certain pseudo. Pour ceci, on demande que la classe `Game` contienne une map `players` de type `Map<String, Player>` qui associe à chaque pseudo le joueur ayant ce pseudo.

1. Écrire la classe `Game`. On écrira un constructeur, un getter et une méthode `toString` pour la classe `Game`.
2. Écrire une classe `FileToGame` qui contient une méthode statique `Game readGame` permet de lire un fichier `data.txt` contenant sur chaque ligne les données d'un `Player` selon le format décrit ci-dessus et construit un `Game`.

   Par exemple sur le fichier

   ```
   <mpb, beal, 5, dp, mac, zip, pa>
   <mac, crochemore, 10, dp, beal, pa>
   <zip, zipstein, 10, beal, mac, dr>
   ```

et l'affichage du `Game` sera

```
{zip={friends=[mac, beal, dr], level=10, pseudo=zip, name=zipstein},
mac={friends=[pa, beal, dp], level=10, pseudo=mac, name=crochemore},
mpb={friends=[zip, pa, mac, dp], level=5,pseudo=mpb, name=beal}}
```

3. Écrire une classe `GameTest` qui contient une méthode `main` comme ci-dessous et testez le programme.

```
public class GameTest {
   public static void main(String[] args) throws IOException {
       Game game = FileToGame.makeGame();
       System.out.println(game);
   }
}
```

Exercice 3.— Dans la classe `Game`,

1. écrivez une méthode `highestLevel()` qui permet de calculer le plus haut niveau présent parmi les joueurs;
2. écrivez une méthode `highestLevelSet` construisant l'ensemble des pseudos des joueurs qui ont les personnages de plus haut niveau;
3. écrivez une méthode `add` permettant d'ajouter un nouveau joueur;
4. écrivez une méthode `commonFriends` renvoyant l'ensemble des pseudos des amis communs à deux joueurs.