Programmation Objet en Java - Feuille de TP 10

Exercice 1.

On veut écrire un programme permettant de manipuler les joueurs d'un jeu de rôle en ligne où le nombre de joueurs est important.
Pour chaque joueur, on dispose de données diverses dont au moins :

- le *login* de type `String` qui est un identifiant unique dans le jeu ;
- le nom de type `String` ;
- le niveau du personnage qui sera un entier ;
- l'ensemble des *logins* de ses amis dans le jeu.

1. Écrire un classe `Player` contenant le nom, le niveau et les amis du joueur (mais pas le *login*, c'est l'exercice 2 qui s'en occupe), avec un constructeur qui prend en paramètre tous les champs et les initialise.

2. Votre constructeur est-il visible de l'extérieur de la classe ? Si oui, êtes-vous sûr que votre code est correct ? Par exemple, que se passe-t-il avec le code suivant ?

```java
public static void main(String[] args) {
  HashSet<String> friends = new HashSet<>();
  friends.add("Alice");
  Player player = new Player("Bob", 10, friends);
  friends.add(null);
}
```

   Si ce n'est pas déjà fait, modifier la visibilité du constructeur pour le rendre correct.

3. Ajouter une méthode d'affichage (voir exemple ci-dessous).

4. Les données de l'ensemble des joueurs du jeu sont stockées dans un fichier. Chaque ligne représente un joueur sous la forme suivante : entre crochets, le *login*, le nom, le niveau, et les amis. Chaque champ entre les crochets est séparé par une virgule. Par exemple, une ligne du fichier peut être :

   `<Yampai804, lennox, 93, Eudora969, Pillow644, FairHill851>`

   et l'affichage du `Player` correspondant sera :

   `name = lennox, level = 93, friends = [Eudora969, Pillow644, FairHill851]`

   Écrire une méthode `makeLoginPlayerFromLine` qui prend en paramètre une **seule** ligne de données (c'est à dire une chaîne de caractères) formattée comme indiqué ci-dessus et renvoie un nouveau couple formé du *login* et d'un `Player` fabriqué à partir de ces données. Attention, cette méthode n'est pas un constructeur mais fait appel au constructeur de `Player` (on appelle cela une méthode *factory*).

Exercice 2. On va créer une classe `Game` pour stocker l'ensemble des joueurs. La classe `Game` doit permettre de trouver rapidement les données d'un joueur à partin d'un *login*. Pour ceci, on demande que la classe `Game` contienne une `Map` nommée `players` qui associe à chaque *login* le joueur correspondant.

1. Écrire la classe `Game` avec un constructeur qui prend en paramètre la `Map` et n'est pas visible à l'extérieur de la classe. Ajouter une méthode qui permet d'obtenir un joueur à partir de son *login* et une méthode d'affichage (un joueur par ligne).

2. Écrire une méthode `makeGameFromFile` qui prend en paramètre un fichier (son chemin) contenant les données d'un ensemble de joueurs selon le format décrit ci-dessus et fabrique un `Game` à partir de ces données. Attention, cette méthode est aussi une *factory* qui construit un `Game`.

   Note : pour simplifier, on considère que le fichier en paramètre est forcément bien formé.

   Par exemple, pour un `Game` crée à partir du fichier :

   ```
   <Mesa682, marylou, 46, Alex419, MaunaLoa295>
   <Malta493, corbin, 43, MaunaLoa295>
   <Alex419, xenia, 76, Mesa682, MaunaLoa295>
   <MaunaLoa295, cyndi, 28, Mesa682>
   ```

   l'affichage sera :

   ```
   login = Mesa682, name = marylou, level = 46, friends = [Alex419, MaunaLoa295]
   login = Malta493, name = corbin, level = 43, friends = [MaunaLoa295]
   login = Alex419, name = xenia, level = 76, friends = [Mesa682, MaunaLoa295]
   login = MaunaLoa295, name = cyndi, level = 28, friends = [Mesa682]
   ```

3. Écrire une classe `GameTest` pour tester la création et l'affichage d'un jeu à partir des fichiers de données fournis.

Exercice 3.

1. Écrire une méthode `anyOfHighestLevel` qui permet de renvoyer un joueur qui a le plus haut niveau parmi les joueurs d'un jeu (il peut y en avoir plusieurs, on veut n'importe lequel de ceux là). Pour cela, vous utiliserez sur l'ensemble des joueurs la méthode de `Collections` qui permet de calculer le maximum. Attention, il ne s'agit pas de calculer la liste des niveaux puis de calculer son maximum. En particulier, vous ne devez pas ajouter d'accesseur pour le niveau du joueur pour faire cet exercice.

2. Écrire une méthode `highestLevelPlayers` construisant l'ensemble des *logins* des joueurs qui ont les personnages de plus haut niveau. Pour le fichier `players_data_small.txt`, on obtient par exemple :

   ```
   [Elkader520, Brookings203, Oacoma699, NorthManitou715]
   ```

3. Écrire une méthode `commonFriends` renvoyant l'ensemble des *logins* des amis communs à deux joueurs identifiés par leurs *logins*. Pour cela, vous utilisez la méthode permettant de faire l'intersection de deux ensembles vue en cours. Avec le fichier `players_data_medium.txt`, pour "Warwick460" et "Interior294", on obtient par exemple :

   ```
   [SaintOlaf212]
   ```

4. ★ Écrire une méthode `network` qui prend en paramètre le *login* d'un joueur et renvoie l'ensemble des *logins* de ses amis directs ou indirects (ses amis, les amis de ses amis, les amis des amis de ses amis, ...). Avec le fichier `players_data_very_small.txt`, pour "Amo105", on obtient par exemple :

   ```
   [Camak669, McDermott373, Barneston758, Midtown451]
   ```

5. Avez-vous ajouté un accesseur sur l'ensemble des amis d'un joueur pour faire l'une des deux dernières questions ? Si oui, êtes-vous sûr que c'est nécessaire ? Et êtes-vous sûr que votre code est correct ?