TP8: interfaces génériques

Introduction

Le but de ce TP va être d'approfondir un peu plus la notion de programmation générique en Java. Si vous avez un doute par rapport à ces notions, qui sont loin d'être triviales, nous vous recommendons de relire/revisionner le cours 7, deuxième partie. En particulier la mnémonique *PECS: Producer Extends Consumer Super* pourra nous être utile pour ce TP.

Pour ce TP8, un squelette vous est fourni téléchargeable sur Moodle. Il contient un script build.gradle et est déjà structuré selon les conventions gradle.

Toujours dans le cadre de notre RPG, nous allons cette fois-ci implémenter un marchant de potion magiques (healing ou mana) un peu plus sophistiqué que celui du TP précédent. En effet, maintenant le marchand peut vendre des potions mais aussi acheter des objet que le joueur désire lui vendre (ou revendre). Pour simplifier, nous allons toujours faire des transactions impliquant uniquement des potions magiques, néanmoins, comme nous faisons de la programmation générique, il serait déjà possible d'intégrer des transactions avec d'autre types d'`Items` (tels que des armes ou armures etc.). Lorsque nous utilisons les verbes "buy" et "sell" dans le code comme nom de méthodes, nous nous plaçons du **point de vue du marchand**, i.e., *le marchand achète au joueur* ou *le marchand vend au joueur* pour éviter les confusions[1].

Veuillez noter que ce TP comporte une grande partie de compréhension de code, car un squelette conséquent vous ai fourni. Prenez donc le temps nécessaire pour bien comprendre le code existant pour être en mesure de compléter les "trous".

[1] En effet, dans le TP précédent c'était l'inverse: le marchand avait une méthode sell(...) et le client une méthode buy(...).

Exercice 1

Nous vous fournissons les interfaces suivantes (présente dans le squelette):

```
1  public interface Item {
2      public int getWeight();
3      public String getName();
4      public int getPrice();
5  }
6  public interface Buyer<T>  {
7      int buy( Inventory<? extends Item> goods );
8  }
9  public interface Seller<T> {
10     boolean sell( Inventory<? extends Item> goods, int
       offeredAmount  );
11     Inventory<? extends Item> showInventory();
12 }
13 public interface Merchant<T extends Item> extends Buyer<T>,
       Seller<T> {}
```

Notez que cette fois, nous avons une interface `Potion`:

```
1  public interface Potion extends Item {
2      public int getPotionValue();
3  }
```

Celle ci joue le même rôle qu'aurait pu jouer une super-classe, mais comme dit en cours, il est souvent préférable de remplacer l'héritage par l'utilisation d'interfaces. Les classes ManaPotion ou HealingPotion implémentent donc cette interface comme le ferait toute autre potion, e.g., potion de stamina, d'intellect, etc. Mais comme dit en introduction, nous nous limiterons aux potions de mana ou vie pour simplifier, car ce sera suffisant pour mettre en pratique la généricité.

Nous avons maintenant une classe `Inventory` qui sert a représenter l'inventaire du marchand ou le sac du joueur. Les `Items` tels que des potions sont maintenant "encapsulés" dans des `InventoryItem` qui représentent en quelquesorte un conteneur virtuel et qui permet d'avoir des stacks (empillement) d'items du même type. Cela permet aussi de "chacher" des détails d'implémentations que nous voudrions transparents pour le moteur du jeu.

Le but de cet exercie est pour vous de compléter les signatures génériques de la classe `Inventory` ainsi que `PotionMerchant`. Les trous à compléter sont indiqués par des "...".

Exercice 2

Verifiez que votre code compile et s'éxécute avec la commande:

```
gradle clean build run
```

.

Complétez la méthode `removeItemInventoryFromInv(...)` de telle sorte que lorsque un InventoryItem a un ammount de 0, i.e., qu'il y a 0 exemplaire de cet objet, que cet InventoryItem soit enlevé de l'inventaire. (En effet, vous remarquerez qu'avec le code de test (scénario) fourni dans le main, le sac du joueur se retrouve avec une ItemInventory de potion de mana listée comme ayant 0 exemplaires).

Rendu

Ce TP n'est pas noté. Néanmoins, nous vous demandons de le déposer sur Moodle via le widget prévu à cet effet. Archivez votre projet tout entier dans **un seul** fichier .zip avec la convention de nommage suivante: **prenomNomTP8.zip**.