TP3 Entrées/sorties

Développement Logiciel (L2-S4)

Vendredi 7 février 2014

Le but de ce TP est de créer un petit programme qui permet de gérer la liste de personnages d'un jeu. Les fonctions requises concernent l'ajout et la suppression de personnages ainsi que la sauvegarde, le chargement et l'export d'une liste de personnages. Un personnage est décrit par un nom, un nombre de points de vie et une force de combat. On a deux types de personnages les héros et les monstres. Les héros ont une arme (épée, sabre ou bâton) et les monstres ont un type (Magicien ou Ogre).

Préambule Récupérez les sources à l'adresse suivante :
`http://www.lri.fr/~garcia/devlog/tp3.zip`

Très important : Ce TP est noté. Avant de partir, vous serez noté par votre chargé de TP sur les critères suivants : Questions réussies (questions qui fonctionnent et qui respectent le sujet du TP) ; Utilisation des concepts du cours (Visibilité des variables et méthodes, héritage, exceptions ...) ; Clarté du code (Lisibilité, commentaires).

Vous avez également la possibilité de terminer le TP chez vous si ce n'est pas le cas durant la séance. Dans ce cas, vous devez envoyer les sources dans une archive NOMPRENOMTP3.zip à votre chargé de TP. Le titre du mail doit être [DevLog] TP3 de Nom Prenom

Exercice 1 : Création des classes

- 1. Ecrire les classes Personnage, Hero et Monstre ainsi que leurs constructeurs et accesseurs et les méthodes *toString.*

Exercice 2 : Lecture de la base de données

Les personnages sont rangés dans le fichier personnages.txt. Ouvrez ce fichier. Dans cette base, les héros sont sous la forme :*Hero\t nom\t pointsdevie\t force\t arme*. Les monstres sont sous la forme : *Monstre\t nom\t pointsdevie\t force\t type*

- 1. Ecrire une méthode *void litfichier()* permettant d'afficher le contenu du fichier personnages.txt. Pour ceci vous pouvez utiliser les classes *BufferedReader* et *FileReader* qui permettent la lecture de fichier.
- 2. Créer une méthode *Personnage creePersonnage(String ligne)* qui transforme une des lignes lues précédemment en un personnage (un héro ou un monstre). Pour ceci vous pouvez utiliser la fonction *split* de la classe *String* qui permet de séparer une String en plusieurs éléments suivant un motif. Vous devez aussi convertir une String en entier, pour cela vous utiliserez la fonction *static parseInt(String s)* de la classe Integer. Faites attentions aux exceptions. Aide, pour transformer
- 3. Créer une méthode *ArrayList<Personnage>parcourebase()* qui lit les lignes de personnages.txt et qui pour chaque ligne crée le personnage associé et l'ajoute à l'ensemble donné en retour.
- 4. Affichez les résultats et vérifiez que c'est bien correct.

Exercice 3 : Création de personnages

Il est aussi demandé de pouvoir ajouter de nouveaux personnages. Pour ceci on utilise une interface texte dans laquelle le système pose des questions auxquels il faut répondre. Par exemple, le système demande "Quel est le nom ?" et l'utilisateur répond "John Snow".

- 1. Ecrire la fonction *static Personnage dialogue()* permettant un dialogue entre l'homme et la machine afin de créer un personnage. Pour ceci vous pouvez utiliser *System.in* qui est un *InputStream*.
- 2. Ecrire une fonction *ArrayList<Personnage>ajoutePersonnage(ArrayList<Personnage>les personnages)* créant un personnage de cette manière et permettant de l'ajouter à l'ensemble des personnages obtenus par la fonction *parcourebase()*. Dans le cas ou le personnage possède le même nom qu'un personnage déjà existant dans la base, on veillera à avertir l'utilisateur et à lui demander un autre nom.
- 3. Modifier votre fonction pour qu'elle puisse boucler si l'utilisateur veut rajouter d'autres personnages. Cela est fait en posant la question "Voulez vous rajouter d'autres personnages (OUI ou NON) ?".

Exercice 4 : Mise en mémoire

Après avoir ajouter des personnages dans la liste, l'étape suivante consiste à les sauvegarder. Pour ceci il faut mettre tous les personnages de l'ArrayList obtenue dans le fichier personnage.txt.

- 1. Créer la fonction *abstract String toBase()* permettant de transformer un personnage dans le format du fichier personnages.txt dans la classe Personnage. Surchargez là dans les classes Héro et Monstre afin de s'adapter.
- 2. Créer la fonction *void ajouteDansBase(ArrayList<Personnage>lespersonnages)* permettant de mettre les héros et les monstres dans le fichier en utilisant les fonctions de transformation précédentes (*toBase()*)et *BufferedWriter*.
- 3. Créer la fonction *void principaleajout()* qui appelle les fonctions des différents exercices afin d'ajouter des personnages dans la base. Elle commence par appeler *parcourebase* pour avoir les personnages contenus dans la base puis *ajoutepersonnage* pour créer les personnages et enfin *ajouteDansBase* pour tout mettre dans la base de connaissance.
- 4. Tester votre travail.

Exercice 5 (Bonus) : Recherche dans la base

Il s'agit de créer une méthode permettant de chercher certains personnages dans la base en utilisant un système de dialogue. Un tel système peut fonctionner de la manière suivante :

```
Quel type de personnage souhaitez vous (Hero ou Monstre)?
> Hero
Quel critère voulez vous utiliser (âge, nom arme)?
> nom
Entrez un nom (ou une partie du nom).
> John
Voici les personnages correspondant à votre recherche :
- John Snow age: 24 ptsdevie:40 force:80 arme:épée
- John Wayne age:55 ptsdevie:0 force:50 arme:pistolet
```

Vous êtes libre de la manière d'organiser ce dialogue mais à la fin le système doit afficher les personnages intéressants ou écrire "Je suis désolé aucun personnage ne correspond à votre demande".

– 1. Créer la méthode *ArrayList<Personnage>filtre(ArrayList<Personnage>lespersonnages, String critere, String valeur)* parcourant lespersonnages afin de n'avoir en sortie que ceux correspondant bien au critères.
– 2. Créer la fonction *void recherche()*. Elle commence par appeler *parcourebase()* afin d'avoir l'ensemble des personnages connus. Puis à partir de cette liste demande à l'utilisateur son critère et sa valeur et à partir de ceux ci filtre l'ensemble des personnages. Ces étapes sont faites jusqu'à ce que l'utilisateur soit satisfait.