

Concept. et prog. objet


\1 TP n°4 : les flux

Dépublié

Détails

Écrit par stéphane Domas

Catégorie : M3105 - Concept. et prog, objet avancée (/index.php/menu-cours-s3/menu-mmi3-test)

 Publication : 23 octobre 2014

 Affichages : 1344

1°/ Flux de caractères

- L'objectif est de sauvegarder/lire l'ensemble de la population sous la forme d'un fichier texte. Pour cela, chaque humain doit être sauvegardé/lu en précisant son type et la valeur de ses attributs, séparés par des virgules. Pour le type, on peut utiliser un code simple avec 1 = Homme et 2 = Femme, Par exemple : 1,toto,21,60,65,34000 est le résultat de la sauvegarde d'une instance d'Homme s'appelant toto, de 21 ans, pesant 60 kg, avec une espérance de vie de 65 ans et un salaire de 34000.
- Comme on va lire/écrire des lignes de texte, les flux à utiliser sont `BufferedReader` pour la lecture et `PrintStream` pour écrire.
- Pour définir les méthodes de lecture/écriture, il existe plusieurs solutions. Ce TP est l'occasion d'en voir deux.

1.1°/ solution basique

1.1.1°/ Sauvegarde

- Comme il y a un certain nombre d'attributs communs aux Homme et Femme, les principes POO nous conduisent à définir une méthode de sauvegarde dans `Humain` qui écrive ces attributs dans le fichier. Ensuite, dans `Homme` et `Femme`, on redéfinit cette méthode tout en l'appelant grâce au mot-clé `super`.
- Pour cela, ajoutez et complétez dans la classe `Humain` le code suivant :

```
1 public void saveTxt(PrintStream ps) {
2     String toWrite;
3     /* affectation de toWrite avec la concaténation de la valeur
4        des attributs nom, age, poids, esperance de vie
5     */
6
7     // écriture de toWrite sur le flux
8 }
```

- Ajoutez et complétez dans la classe `Homme` le code suivant :

```
1 public void saveTxt(PrintStream ps) {
2     // écriture de la chaîne "1," sur le flux
3     // appel de la super methode
4     // écriture de la valeur de salaire sur le flux
5 }
```

- Ajoutez et complétez dans la classe `Femme` le code suivant :

```

1 public void saveTxt(PrintStream (http://java.sun.com/j2se/1.5.0/docs/api/java/io/PrintStream.html) ps)
2 {
3     // écriture de la chaîne "2," sur le flux
4     // appel de la super méthode
5     // écriture de la valeur de fertilité sur le flux
6 }

```

- Ajoutez et complétez dans la classe Population le code suivant :

```

1 public void saveTxt(String fileName) throws IOException {
2     PrintStream ps = null;
3     // instantiation de ps
4     // parcours de la population pour appeler saveTxt(ps) sur chaque individu
5 }

```

1.1.2°/ Lecture

- Le problème de la lecture s'apparente à celui de la poule et de l'oeuf.
- En effet, au moment de lire une ligne du fichier texte, on ne sait pas encore si elle commence par 1, ou 2,
- Si ce sont les classes Homme et Femme qui contiennent les méthodes de lecture, cela pose problème car il faut une instance de ces classes pour appeler leur méthode. Comme on ne sait pas déterminer l'instance à créer, le problème tourne en rond.
- Une solution simple consiste à déporter la lecture dans la classe Population et qui va interpréter la ligne lue pour créer une instance d'Homme ou de Femme en appelant le constructeur qui initialise tous les attributs.
- Pour cela, ajoutez et complétez dans la classe Population de code suivant :

```

1 public void loadTxt(String fileName) throws IOException {
2     BufferedReader br = null;
3     String line;
4     Humain h = null;
5     // instantiation de br
6     /*
7     tant que l'on peut lire une ligne line sur br :
8         séparer line pour obtenir un tableau tabs de String (cf. methode split)
9         si tabs[0] vaut "1"
10             récupérer dans tabs[1], tabs[2], ... le nom, l'âge, ... de l'homme
11             h = instance d'Homme avec ces valeurs
12         sinon tabs[0] vaut "2"
13             récupérer dans tabs[1], tabs[2], ... le nom, l'âge, ... de la femme
14             h = instance de Femme avec ces valeurs
15         finsi
16         si h != null ajouter h à la population
17     fintantque
18     */
19 }

```

1.2°/ Solution POO

- La solution basique n'est pas satisfaisante du point de vue de la POO car elle n'est pas symétrique : un humain sait sauvegarder ses attributs mais pas les lire.
- Il faudrait donc lire uniquement le type d'instance, la créer et ensuite appeler sa méthode loadTxt().
- En fait, c'est faisable car la classe `BufferedReader` encapsule un `FileReader` basique et peut donc lire un nombre donné de caractères.

- Or, que ce soit un homme ou une femme, une ligne commence toujours par 2 caractères : soit 1, soit 2,
- Pour mettre en place cette solution, ajoutez et complétez dans la classe `Population` le code suivant :

```

1 | public void loadTxtBetter(String fileName) throws IOException {
2 |     BufferedReader br = null;
3 |     char[] deb;
4 |     int nbLu;
5 |     Humain h = null;
6 |     // instanciation de br et de deb
7 |     /*
8 |     nbLu = lecture dans br de 2 caractères mis dans deb.
9 |         si deb[0] == '1'
10 |             h = instanciation d'Homme
11 |         sinon si deb[0] == '2'
12 |             h = instanciation de Femme
13 |         finsi
14 |         si h != null
15 |             h.loadTxt(br)
16 |             ajouter h à la population
17 |         finsi
18 |         nbLu = lecture dans br de 2 caractères mis dans deb.
19 |     fintantque
20 |     */
21 | }

```

- Ajoutez et complétez dans la classe `Humain` le code suivant :

```

1 | public abstract void loadTxt(BufferedReader br);

```

- Ajoutez et complétez dans la classe `Homme` le code suivant :

```

1 | public void loadTxt(BufferedReader br) {
2 |     String line = null;
3 |     // lire dans br une ligne line
4 |     // découper line pour obtenir un tableau tabs de String
5 |     // récupérer dans tabs[0], tabs[1], ... le nom, l'âge, ... de l'homme
6 |     // mettre à jour les attributs avec ces valeurs.
7 | }

```

- Ajoutez et complétez dans la classe `Femme` le code suivant :

```

1 | public void loadTxt(BufferedReader br) {
2 |     String line = null;
3 |     // lire dans br une ligne line
4 |     // découper line pour obtenir un tableau tabs de String
5 |     // récupérer dans tabs[0], tabs[1], ... le nom, l'âge, ... de la femme
6 |     // mettre à jour les attributs avec ces valeurs.
7 | }

```

- Petite question : pourquoi ne peut-on pas définir une partie du traitement dans la méthode de `Humain`, par exemple celle qui concerne les attributs communs ?

1.3°/ Utilisation

- Copiez TP3.java dans TP4.java et modifiez le nom de classe.
- Modifiez TP4 pour que :
 - après la génération initiale de la population, elle soit sauvegardée au format texte dans un fichier dont le nom est donné en paramètre au programme.
 - après les tours de jeu, la population est relue dans ce même fichier puis affichée. Si tout va bien, le résultat devrait être le même qu'au début.

2°/ Flux objet

- On remarque que la solution utilisant du texte est un peu fastidieuse à mettre en place.
- Or, Java propose de lire/écrire directement des objets grâce aux flux `ObjectInputStream` et `ObjectOutputStream`.
- Dans ce TP, on veut lire/écrire des Homme et Femme. Modifiez ces classes (ou directement Humain) pour qu'elles soient sérialisables.
- De plus, l'objectif est de lire/écrire une population entière. De nouveau, il y a deux solutions :
 - écrire les instructions de lecture/écriture en dehors de `Population`,
 - `Population` sait elle-même lire/écrire sa collection d'`Humain`.

2.1°/ lecture/écriture "externe"

- Dans ce cas, la création des flux se fait dans la méthode `main()` de TP4. Par exemple :

```

1  class TP4 {
2      public static void main(String[] args) {
3          Population pop = null;
4          ObjectInputStream ois = null;
5          ObjectOutputStream oos = null;
6          ...
7          // instantiation de pop
8          // création de population initiale
9          // instantiation de ois et oos
10         // sauvegarde de pop grâce à oos
11         // tours de jeu
12         // relecture de pop grâce à ois
13         // affichage de pop
14     }
15 }
```

- Ensuite, on utilise ces flux pour lire/écrire directement l'objet `pop`, à condition que la classe `Population` soit sérialisable.
- Modifiez `Population` en ce sens, ainsi que TP4 pour que la sauvegarde/relecture décrite en 1.3 se fasse en utilisant les flux objet ainsi créés.

2.2°/ lecture/écriture "interne"

- Dans ce cas, les méthodes de lecture/écriture font partie de la classe `Population`.
- Pour cela ajoutez et complétez dans la classe `Population` le code suivant :

```

1  public void saveBin(String fileName) throws IOException, ClassNotFoundException {
2      ObjectOutputStream oos = null;
3      // instantiation de oos pour écrire dans fileName
4      // sauvegarde de pop grâce à oos.
5  }
6
7  public void loadBin(String fileName) throws IOException, ClassNotFoundException {
8      ObjectInputStream ois = null;
9      // instantiation de ois pour lire dans fileName
10     // initialisation de pop grâce à ois.
11 }
```

- Modifiez TP4 pour que la sauvegarde et la relecture de la section 1.3 se fasse cette fois avec `saveBin()` et `loadBin()`.