Entrée/Sorties - Sérialisation

Mathieu Bourgais
mathieu.bourgais@insa-rouen.fr

2023

Préambule

Vous pouvez utiliser l'éditeur de texte de votre choix. Vous pouvez aussi utiliser un IDE comme Eclipse, Visual Studio Code ou encore Intellij, qui présente de nombreux avantages (auto-complétion, par exemple). Dans Eclipse, vous devez toujours travailler dans un projet (à créer au début de chaque TP/mini-projet) dans lequel vous définirez vos classes.

Pour rappel, pour compiler et exécuter un code source Java, vous pouvez utiliser votre éditeur favori, ou ouvrir un terminal puis naviguer jusqu'au répertoire désiré, et taper :

— javac monFichier.java
— java monFichier

1 Exercice 1 - Entrées/Sorties

Les entrées/sorties d'un programme se manipulent à l'aide de **flux**. On parle de flux d'entrée si l'information y est lue, et de flux de sortie si l'information y est écrite. En Java, l'API de gestion des flux se situe dans le package **java.io**.

Deux types de flux distincts sont à différencier. D'une part, les flux orientés **octets** permettent de manipuler des informations binaires sur 8 bits, et sont utilisés pour lire/écrire des images, des sons, etc. D'autre part, les flux orientés **caractères** permettent de manipuler des informations de type caractère sur 16 bits, et sont utilisés pour lire/écrire du texte.

Pour manipuler des flux d'octets, on utilisera les classes abstraites **InputStream** (pour les entrées) et **OutputStream** (pour les sorties), ainsi que leurs sous-classes concrètes respectives, comme par exemple **FileInputStream** et **FileOutputStream** pour des lectures/écritures dans un fichier.

Pour manipuler des flux de caractères, on utilisera les classes abstraites **Reader** et **Writer**, ainsi que leurs sous-classes concrètes respectives, comme par

exemple **FileReader** et **FileWriter** pour des lectures/écritures dans un fichier.

De plus, pour les lectures / écritures dans des fichiers, on préférera utiliser des flux bufferisés, par exemple à l'aide des classes **BufferedReader** et **BufferedWriter**, pour la manipulation de flux de caractères.

Les classes sus-mentionnées définissent un ensemble de méthodes **read()**, **read(byte buff[])**, **write(int c)**, **write(byte buff[])**, **write(String s)**, **newLine()** (permettant un retour à la ligne), etc, afin de réaliser facilement les différentes opérations de lecture / écriture. Pour directement lire une ligne sous forme de chaîne de caractères, à l'aide d'un **BufferedReader**, on pourra également utiliser la méthode **String readLine()**. Vous pouvez vous référer à la documentation de ces différentes classes pour plus d'informations : *https ://docs.oracle.com/javase/7/docs/api/index.html* (package java.io).

Copiez les différentes classes et interfaces développées lors des TD précédents au sein du répertoire dédié à l'exercice 1. Faites attention à ne pas accidentellement écraser l'interface **BanqueInt** nouvellement fournie pour cet exercice.

L'interface **BanqueInt** fournie définit deux nouvelles méthodes par rapport à votre classe Banque : **enregistrer(String file)**, permettant d'enregistrer l'état actuel de la banque au sein d'un fichier, et **charger(String file)**, permettant d'initialiser la banque à partir de données contenues dans un fichier. Implémentez ces deux nouvelles méthodes au sein de votre classe Banque.

Compilez et exécutez la classe **TestEnregistrerCharger** fournie. Vérifiez le bon fonctionnement de vos méthodes enregistrer et charger. En particulier, vérifier que le contenu des fichiers **premiereSauvegarde.txt** et **deuxiemeSauvegarde.txt** soit cohérent.

2 Exercice 2 - Sérialisation

La **sérialisation** permet l'envoie d'objets à travers les flux, notamment à travers des flux de fichiers, permettant ainsi une gestion aisée de la persistance. En Java, tout objet implémentant l'interface **Serializable** est sérialisable en une suite d'octets, qui pourra ensuite être relue afin de reconstruire l'objet. De plus, Java prend en charge la sérialisation des réseaux d'objets (ensemble d'objets, listes d'objets, tableaux d'objets, ...). Ainsi, pour faire en sorte qu'un objet soit sérialisable, il suffit que la classe de l'objet implémente l'interface Serializable, ne contenant aucune méthode, et ne demandant donc aucun code supplémentaire.

Afin de manipuler la lecture/écriture d'objets dans des flux, on utilisera alors les classes **ObjectInputStream** et **ObjectOutputStream**, qui définissent notamment les méthodes **writeObject(Object o)** et **readObject()**.

Copiez les différentes classes et interfaces développées lors des TD précédents

au sein du répertoire dédié à l'exercice 2. Faites attention à ne pas accidentellement écraser l'interface **BanqueInt** nouvellement fournie pour cet exercice.

Modifiez l'ensemble de vos classes afin de les rendre sérialisables.

Modifiez la classe **TestSerialisation** fournie, afin d'ajouter le code correspondant à l'enregistrement de l'état de la banque dans un fichier, et le code correspondant au chargement de la banque à partir d'un fichier.

Compilez et exécutez la classe **TestSerialisation**, et vérifiez la cohérence de la deuxième banque par rapport à la première après enregistrement/chargement.

Essayez d'ouvrir le fichier **troisiemeSauvegarde**. Que se passe-t-il ? Pourquoi ?