Projet encadré Java : sérialisation et C^ie

Carl Esswein

V2

Dernière mise à jour : mars 2017.

Table des matières

Ce document est un *mini guide de prise en main rapide* de la sérialisation en Java.

La **sérialisation** est un procédé qui consiste à transcrire l'état d'un objet présent en mémoire dans un format de type flux d'octets permettant, soit de l'enregistrer dans un fichier, soit de le transmettre sur un réseau. Le procédé symétrique, la **désérialisation**, permet – à partir du flux (fichier ou réseau) – de reconstituer en mémoire l'objet sérialisé dans l'état dans lequel il se trouvait au moment de la sérialisation.

Dans le cadre du projet encadré java, nous allons utiliser la sérialisation pour sauvegarder dans un fichier les données de nos applications de façon persistente. Il serait beaucoup plus efficace, et structurant, d'utiliser une base de données, mais nous allons utiliser la sérialisation qui est plus simple à mettre en œuvre. Gardez à l'esprit que c'est uniquement pour une question de simplification, dans le cadre de ce projet.

Prise en main

Ecrire **dans un fichier** nécessite de créer un objet de type `java.io.FileOutputStream`[1]. Un des constructeurs prend en paramètre le nom du fichier à créer :

```
FileOutputStream fos = new FileOutputStream("myData.ser");
```

Pour **sérialiser** un(des) objet(s) java, il convient d'associer un flux sortant d'objets à ce fichier en créant une instance de `ObjectOutputStream` à partir de l'instance précédente :

```
ObjectOutputStream oos = new ObjectOutputStream(fos);
```

Dès lors, écrire un objet dans le fichier se fait simplement en passant une référence de l'objet à la méthode `write` aqéquate de l'`ObjectOutputStream` créé :

```
oos.writeObject("Hello Polytech!");
oos.writeInt(12345);
oos.writeObject(new java.util.Date());
oos.writeObject(myObject);
```

Naturellement, vous n'oublierez pas de fermer le flux : `oos.close();`
… de préférence dans un bloc `finally`.

La **désérialisation** est le procédé inverse permettant de lire des objets à partir d'un flux. Dans notre cas (lecture d'un fichier), il va falloir un `ObjectInputStream` associé à un `FileInputStream`, et vous utiliserez des méthodes `read` (`readObject()` *etc.*).


Cf. http://docs.oracle.com/javase/tutorial/essential/io/objectstreams.html pour plus d'infos sur la gestion des flux d'objets.


Les manipulations ci-dessus se font en binaire : ce sont des flux d'octets qui sont manipulés. Pour gérer des flux de caractères, utiliser `FileReader` et `FileWriter` qui sont plus adaptés.


Pour pouvoir être sérialisé, un objet **doit implémenter l'interface `Serializable`**. Les classes n'implémentent pas cette interface ne pourront pas être sérialisées : `writeObject(…)` lèvera une exception de type `NotSerializableException`.
`Serializable` est une *interface de marquage*, *i.e.* une interface ne contenant ni constante ni signature de méthode. Elle sert uniquement à indiquer sémantiquement qu'une classe est sérialisable.
Par défaut, tous les attributs d'un objet qu'on sérialise, disons `obj1`, sont sérialisés. Pour les attributs de types non primitifs, leur classe doit implémenter `Serializable`, sans quoi la sériélisation d'`obj1` lèvera une `NotSerializableException`.
La plupart des classes de l'API standard sont sérialisables.


NB : pour indiquer dans une classe qu'un attribut ne doit pas être sérialisé, il suffit de faire précéder sa déclaration du mot clé `transient` qui indique qu'il s'agit d'un attribut temporaire :

```
private Date dateOfBirth;
private transient int age;
```

[1] Toutes les classes et interfaces citées dans ce document appartiennent, sauf mention contraire, au package `java.io`.

Exercices

1. Exercice 1

1. Tester la **sérialisation** en enregistrant la date et l'heure en cours (via `java.time.LocalTime` par exemple) dans un fichier « theTime.dat ».

2. Tester, dans un 2e programme, la **désérialisation** de la date et de l'heure depuis le fichier ci-dessus puis leur affichage sur la sortie standard.

2. Exercice 2 – *back to the zoo*

A partir des classes de l'exercice 1.2 (« Zoo ») :

1. Faites les modifications nécessaires dans vos classes pour rendre tous vos animaux sérialisables.

2. Créer une classe de test qui crée une liste d'animaux (par exemple une `ArrayList<Animal>`). Ecrire et tester une méthode sérialisant cette liste et une méthode la désérialisant.

3. Rajouter un attribut non sérialisable à la classe `Vache`, et :
   a. tester la sérialisation d'une liste contenant au moins une vache,
   b. refaites la même chose après avoir déclaré l'attribut `transient`.

Quelques références web

3. Gestion des entrées/sorties et des flux

- http://docs.oracle.com/javase/tutorial/essential/io/index.html

4. Sérialisation

- Of course : http://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html

- http://docs.oracle.com/javase/8/docs/technotes/guides/serialization/