

Sérialisation en Java

Emmanuel BRUNO

Université de Toulon, LIS UMR CNRS 7020

emmanuel.bruno@univ-tln.fr

2026-10-14

Table des matières

1 Définition de la Sérialisation	1
2 Qu'est-ce qui est sérialisé ?	1
3 Utilisation de la Sérialisation	1
4 Interface Serializable	1
5 Compression/Décompression	2

1 Définition de la Sérialisation

- **Qu'est-ce que la sérialisation ?**
 - Transformation de l'état d'un objet (valeurs des variables d'instance) en une suite d'octets.
- **Pourquoi sérialiser ?**
 - **Conservation de l'état** : Sauvegarder l'état d'un objet pour une utilisation future.
 - **Reconstruction** : Restaurer un objet avec le même état à partir des octets sérialisés.
- **Outils en Java** :
 - `ObjectOutputStream` : Pour écrire l'état de l'objet en octets.
 - `ObjectInputStream` : Pour lire les octets et reconstruire l'objet.

2 Qu'est-ce qui est sérialisé ?

- **Variables d'instance** : Valeurs des variables d'instance (pas des variables de classe).
- **Informations sur les classes** :
 - Nom de la classe

- Noms, types et modificateurs des variables à sauvegarder

- **Compatibilité** : Informations pour détecter les modifications de classe entre sérialisation et désérialisation.

3 Utilisation de la Sérialisation

- **Sauvegarde** : Conserver un objet dans un fichier ou une base de données pour une récupération ultérieure.
- **Configuration** : Sauvegarder la configuration d'un composant (ex. JavaBean) pour une réutilisation future.
- **Communication distante (RMI)** :
 - Sérialiser les paramètres non primitifs d'une méthode.
 - Transmettre les octets sur le réseau.
 - Reconstruire le paramètre sur la machine distante.

4 Interface Serializable

- **Condition de sérialisation** : Un objet doit être une instance d'une classe implémentant `Serializable`.
- **Interface marquante** : Ne contient aucune méthode, sert à marquer les classes sérialisables.
- **Classes sérialisables** : La plupart des classes du JDK le sont.
- **Exceptions** :

- Certaines classes ne peuvent pas être sérialisées (ex. `InputStream`).
- D'autres ne doivent pas l'être pour des raisons de sécurité.
- **Variables transientes :**
 - Utiliser `transient` pour exclure une variable d'instance de la sérialisation :
`private transient int val;`
 - À la désérialisation, la variable reçoit sa valeur par défaut ou doit être recalculée si nécessaire.

5 Compression/Décompression

- **Paquetage `java.util.zip` :** Classes filtres pour compresser des flots.
 - **GZIP** : `GZIPInputStream` et `GZIPOutputStream` pour le format GZIP.
 - **ZIP** : `ZipInputStream` et `ZipOutputStream` pour le format ZIP.
- **Lecture performante des fichiers ZIP :**
Utiliser `java.util.zip.ZipFile`.

```
// Définition d'un enregistrement
// immuable Dog qui implémente
// Serializable
record Dog(int id, String name)
implements Serializable {};
```

```
String filename="/
tmp/mesChiens.bin.gz";
```

```
// Utilisation d'un bloc try-with-
// resources pour assurer la fermeture
// des flux
```

```
try (ObjectOutputStream oos = new
ObjectOutputStream(
    new GZIPOutputStream (
        new FileOutputStream
(filename)))) {
```

```
// Sérialisation et compression des
// objets Dog
oos.writeObject(new Dog(1,"Médor"));
oos.writeObject(new Dog(2,"Rex")); }
```

```
try (ObjectInputStream ois =
new ObjectInputStream(
    new GZIPInputStream(
        new
FileInputStream(filename)))) {

    // Boucle infinie pour
    // lire tous les objets Dog du fichier
    while (true) {
        // Lecture et affichage
        // de chaque objet Dog
        System.out.println((Dog)
ois.readObject());
    }
} catch (EOFException e) {
    // Fin de fichier atteinte,
    // sortie de la boucle
} catch (Exception e) {
    // Gestion des autres
    // exceptions potentielles
    e.printStackTrace();
}
```