Chapitre 1 : les entrées sorties

1.1 Entrées/sorties en Java

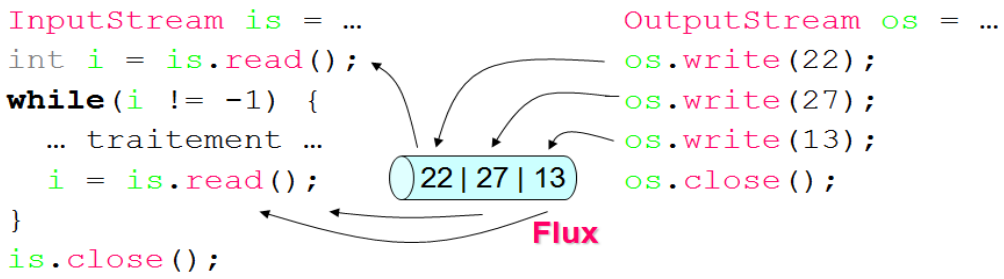
Les entrées/ sorties sont utiliser pour lire et/ou écrire des données à partir d'un fichier, de la mémoire, du réseau. Il existe deux API d'entrées/sorties, l'une bloquantes (`java.io.*`) et l'autre non-bloquantes (`java.nio.*`).

Il existe aussi deux façon de gérer les entrées/sorties, l'une par *paquet* et l'autre par *flux*. La gestion par *paquet* consiste à communiquer directement par paquets de taille fixe, et est utilisé avec le protocole UDP. La gestion par *flux* consiste à communiquer via un canal FIFO, la taille des messages reçus et émis n'est pas la même, et est utilisé avec le protocole TCP, l'accès au fichier, etc. e

Il existe deux modes d'entrée/sortie, l'un qui est *binaire* et l'autre est *caractère*. Le mode *binaire* consiste à écrire et lire des données brutes. Le mode *caractère* consiste à écrire et lire des caractères.

Gestion de flux bloquant et binaire :

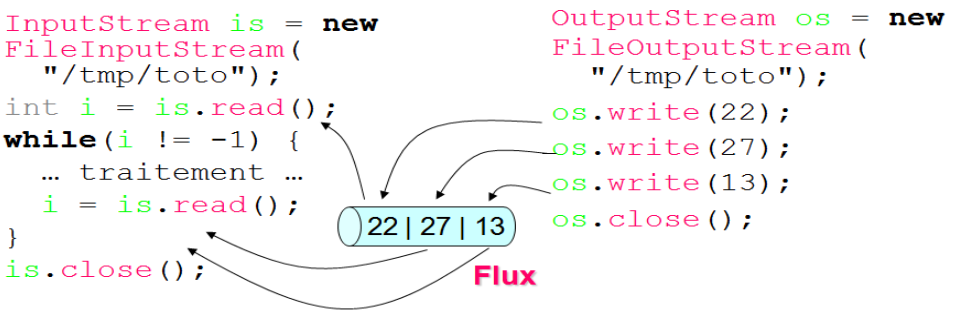
- Le flux de sortie utilise `OutputStream`. Ce dernier permet d'écrire des données. L'instruction `void write(int b)` écrit un octet dans le flux.
- Le flux d'entrée utilise `InputStream`. Ce dernier permet de lire des données. L'instruction `int read()` lit un octet du flux et retourne `-1` en cas de fin de flux.



Listing 1.1 : exemple d'utilisation d'InputStream/OutputStream

Gestion de flux de fichier bloquant et binaire :

- Le flux de fichier de sortie utilise `FileOutputStream`. Ce dernier permet d'écrire des données dans un fichier et hérite de `OutputStream`. L'instruction `void write(int b)` écrit un octet dans le fichier.
- Le flux de fichier de sortie utilise `FileInputStream`. Ce dernier permet de lire des données dans un fichier et hérite de `InputStream`. L'instruction `int read()` lit un octet du fichier.



Listing 1.2 : exemple d'utilisation d'FileInputStream/FileOutputStream

La **sérialisation** est une représentation d'un objet Java *sous forme binaire*, et est utilisée **pour échanger** des objets (envoi d'objets, persistance…). Les objets sérialisables doivent implémenter l'interface `serializable`.

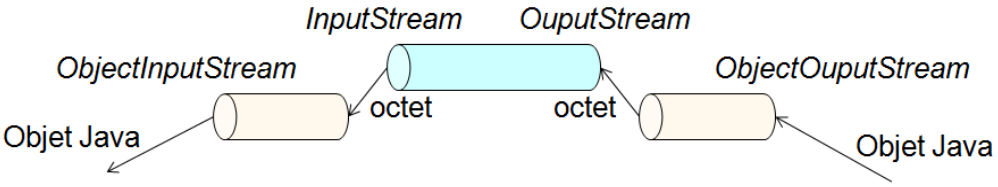


Figure 1.1 : sérialisation d'objets

Gestion de flux d'objet bloquant et binaire :

▪ Le flux d'Objet de sortie utilise `ObjectOutputStream`. Ce dernier permet d'écrire des objets sérialisables dans un flux et repose sur un `OutputStream`. L'instruction `void writeObject(Object b)` écrit un objet dans un flux.
▪ Le flux d'Objet d'entrée utilise `ObjectInputStream`. Ce dernier permet de lire des objets sérialisables dans un flux et repose sur un `InputStream`. L'instruction `Object readObject()lit` un objet d'un flux.

Repose sur un InputStream

```
InputStream is = new
ObjectInputStream(new
FileInputStream(
  "/tmp/toto"));

Object o = is.readObject();
```

```
OutputStream os = new
ObjectOutputStream(new
FileOutputStream(
  "/tmp/toto"));

os.writeObject(new Toto(…));
```

Listing 1.3 : exemple d'utilisation d'ObjectInputStream/ObjectOutputStream