

Introduction à la programmation en Java

Cours 6

Stéphane Airiau

Université Paris-Dauphine

Entrée et sortie

Entrée/sortie: échange de données entre le programme et une source:

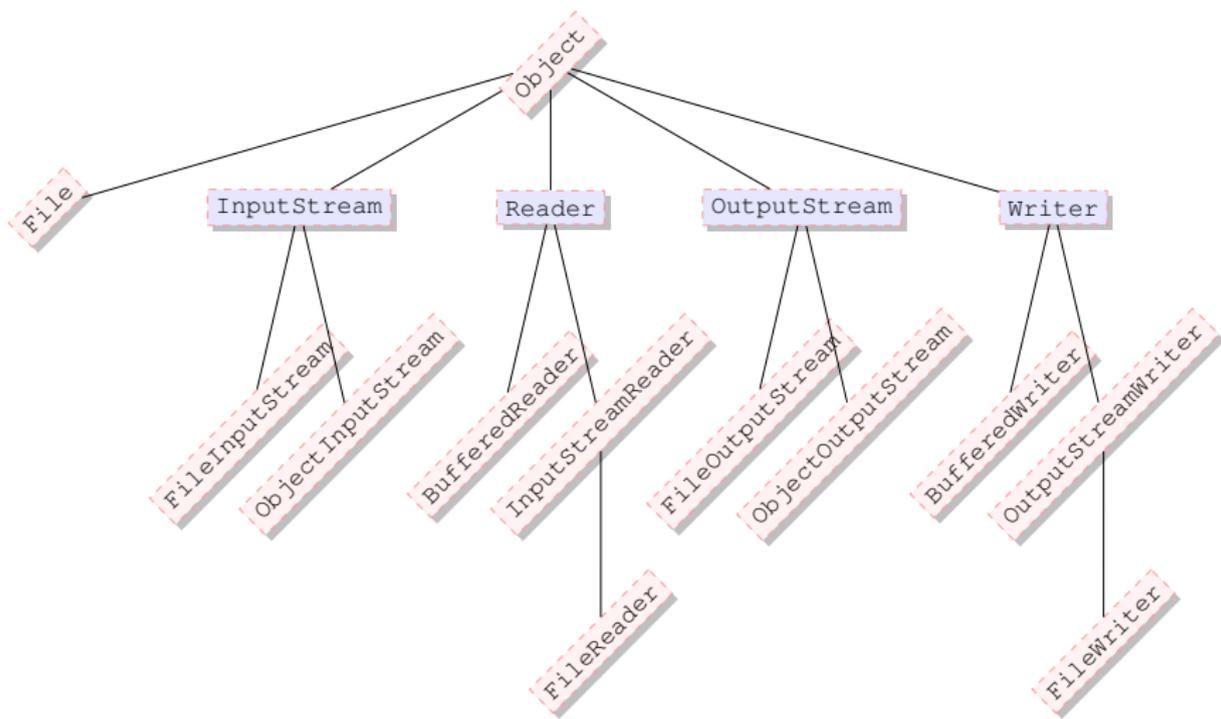
- entrée: au clavier, lecture d'un fichier, communication réseau
- sortie: sur la console, écriture d'un fichier, envoi sur le réseau

⇒ Java utilise des flux (stream en anglais) pour abstraire toutes ses opérations.

de manière générale, on observera trois phases:

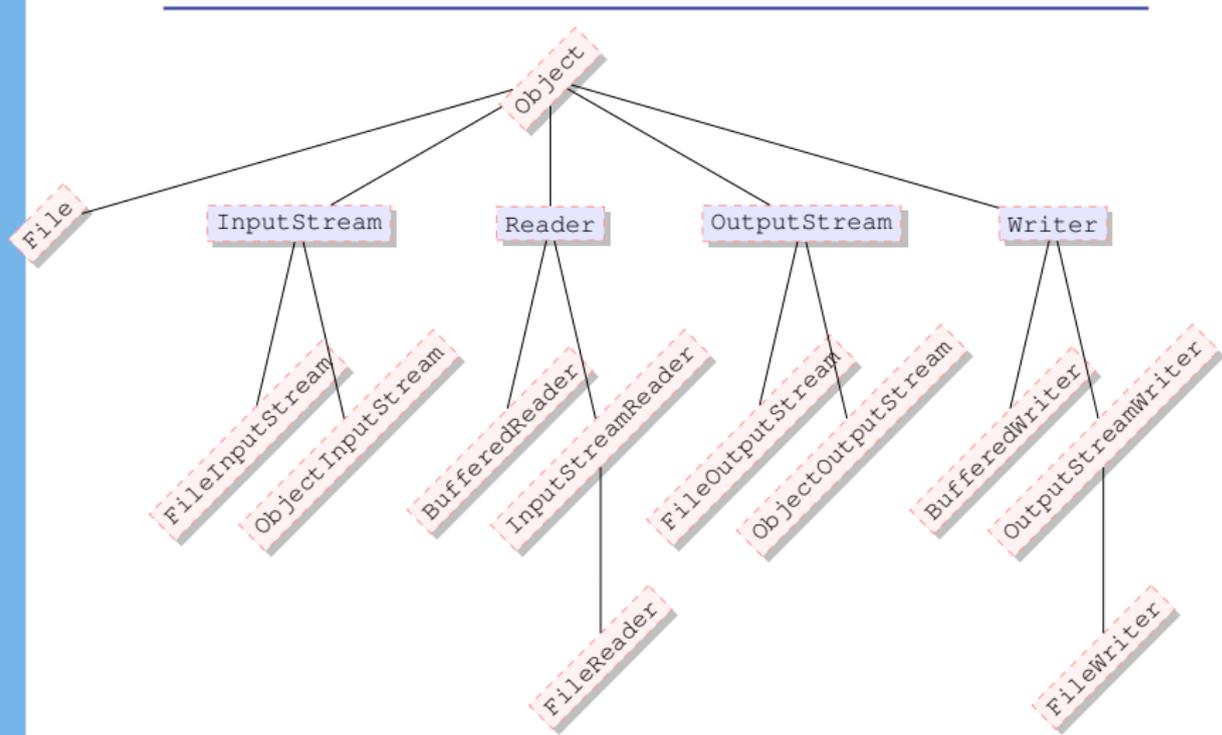
- 1- ouverture du flux
- 2- lecture/écriture du flux
- 3- fermeture du flux

le package java.io (fragment)



La classe `File` permet d'obtenir des informations sur les fichiers

- nom, chemin absolu, répertoire parent
- s'il existe un fichier d'un nom donné en paramètre
- droit: l'utilisateur a-t-il le droit de lire ou d'écrire dans le fichier
- la nature de l'objet (fichier, répertoire)
- la taille du fichier
- obtenir la liste des fichiers
- effacer un fichier
- créer un répertoire
- accéder au fichier pour le lire ou l'écrire



Flux

Les flux transportent des bytes ou des char.

Direction du Flux:

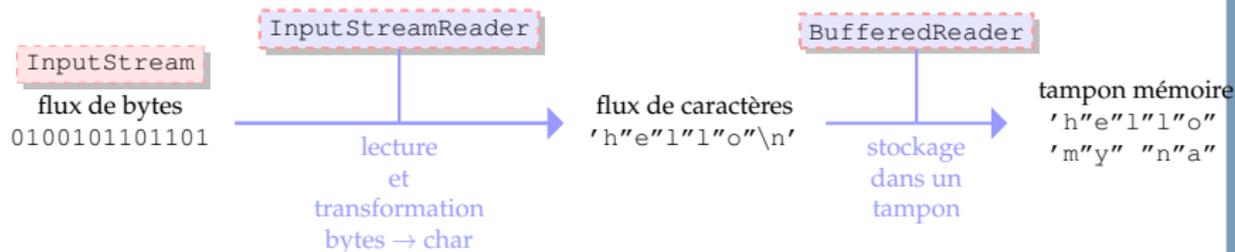
- objets qui gèrent des flux d'entrée: **in**
 - `InputStream`, `FileInputStream`, `FileOutputStream`
- objets qui gèrent des flux de sortie: **out**
 - `OutputStream`, `FileOutputStream`, `FileOutputStream`

Source du flux:

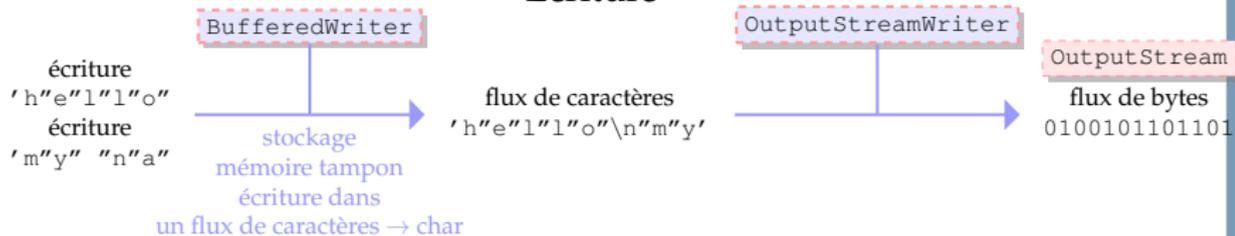
- **fichiers**: on pourra avoir des flux vers ou à partir de fichiers
 - `FileInputStream` et `FileOutputStream`
- **objets**: on pourra envoyer/recevoir un objet via un flux
 - `ObjectInputStream` et `ObjectOutputStream`

Processus de lecture et d'écriture

Lecture



Écriture



Selon le type de la source ou de la destination (fichier, objet), on utilisera

- `FileReader` à la place de `InputStreamReader`
- `FileOutputStream` ou `ObjectOutputStream` comme implémentation de la classe abstraite `OutputStream`

Exemple: Lecture d'un fichier

Lecture du premier octet d'un fichier

```
1 FileInputStream fis =
2     new FileInputStream(new File("ex.txt"));
3 byte[] huitLettres = new byte[8];
4 int nbLettresLues = fis.read(huitLettres);
5 for(int i=0;i<8;i++)
6     System.out.println(Byte.toString(huitLettres[i]));
```

Affiche un fichier sur la console

```
1 BufferedReader reader =
2     new BufferedReader(new FileReader(new File("ex.txt")));
3 String line = reader.readLine();
4 while(line != null) {
5     System.out.println(line);
6     line = reader.readLine();
7 }
8 reader.close();
```

N.B. Les codes ne sont pas corrects (gestion des exceptions)

But: envoyer toute l'information d'un objet

↳ mécanisme de sérialisation

- la classe doit implémenter l'interface `Serializable`
- l'interface `Serializable` n'a pas de méthodes: c'est juste un marqueur.
- Java transforme l'objet automatiquement en un code pas lisible pour les humains

NB: Si un attribut de la classe est un objet d'une classe `MaClasse`

- `MaClasse` est sérialisable : ✓
- `MaClasse` n'est pas sérialisable : on peut utiliser le mot-clé `transient` pour indiquer de ne pas enregistrer cet attribut

Exemple

```
1 IrreducibleGaulois panoramix =
2     new IrreducibleGaulois ("Panoramix", 1.75);
3
4 ObjectOutputStream oos =
5     new ObjectOutputStream (
6         new FileOutputStream (
7             new File ("panoramix.txt")));
8
9 oos.writeObject (panoramix);
10 oos.close ();
11
12 ObjectInputStream ois =
13     new ObjectInputStream (
14         new FileInputStream (
15             new File ("panoramix.txt")));
16
17 IrreducibleGaulois copyPanoramix =
18     (IrreducibleGaulois) ois.readObject ();
19 System.out.println (copyPanoramix.nom);
20 ois.close ();
```

N.B. Le code n'est pas correct (gestion des exceptions)

Lire depuis la console, afficher sur la console

- `System.in`:
 - entrée standard
 - objet de type `InputStream`
- `System.out`:
 - sortie standard
 - objet de type `PrintStream` qui hérite de `OutputStream`

La classe `Scanner` permet de récupérer ce que vous tapez

```
1 Scanner scan = new Scanner(System.in);  
2 int n = scan.nextInt();  
3 double x = scan.nextDouble();  
4 String s = scan.nextLine();
```

Exceptions et entrée/sortie

```
1  try {
2      FileInputStream fis = new FileInputStream(new File("test.txt"));
3      byte[] buf = new byte[8];
4      int nbRead = fis.read(buf);
5      System.out.println("nb bytes read: " + nbRead);
6      for (int i=0;i<8;i++)
7          System.out.println(Byte.toString(buf[i]));
8      fis.close();
9
10     BufferedReader reader =
11         new BufferedReader(new FileReader(new File("test.txt")));
12     String line = reader.readLine();
13     while (line!= null) {
14         System.out.println(line);
15         line = reader.readLine();
16     }
17     reader.close();
18 } catch (FileNotFoundException e) {
19     e.printStackTrace();
20 }
21 catch (IOException e) {
22     e.printStackTrace();
23 }
```