Bases de la POO / Java

Les flux d'entrée-sortie

API pour les flux

- Deux bibliothèques fournies en standard

- **java.io**
  - Gestion d'E/S au travers de flux
  - Sérialisation

- **java.nio** (new io) à partir de Java 1.4
  - Amélioration des performances
  - Environnement uniforme quel que soit l'OS (Channel)
  - Extension aux E/S asynchrones

E/S

- **Echange** de données entre le programme et une autre source
  - un fichier, une imprimante, le clavier, …
- L'échange se fait au travers d'un **flux** existant entre la source et la destination des données
- Toute opération d'E/S suit le schéma suivant :
  - Ouverture d'un flux
  - Lecture ou écriture des données
  - Fermeture du flux

Classes java.io

- Flux de données
  - Classes «**InputStream**» et «**OutputStream**»
  - Classes « **Reader** » et « **Writer** »
- Système de fichiers
  - Classe «**File**»
- Sérialisation
  - Interfaces «**Serializable**» et «**Externalizable**»

Flux (Stream)

- Représente un **canal de communication**
- Dans lequel circulent des données
  - octets (Byte) ou caractères(Character)
  - Codage des caractères UNICODE sur 2 octets
- Ce flux peut être en entrée (ou lecture), ou en sortie (ou écriture)
  - Peut utiliser un buffer pour le traitement de lots

Package java.io

- https://docs.oracle.com/javase/7/docs/api/java/io/package-tree.html

- Noms des classes
  - Origine ou destination du flot
    - tampon, fichier, tableau, tube
  - Sens (lecture ou écriture)
    - Input, Output, Reader, Writer
  - Octets ou Caractères
    - Stream ou Reader/Writer

- BufferedInputStream : tampon/Lecture/octet
- ByteArrayOutputStream : tableau/Ecriture/octet
- BufferedWriter : tampon/caractère

Flux d'octets

- Toutes les classes qui manipulent des flux d'octets héritent de **InputStream** ou **OutputStream**

- Flux d'E/S standard
  - **in, out et err**

- java.lang.System
  - *System.out.println("coucou") ;*

InputStream

- int read()
  - retourne l'octet suivant dans le flot
  - ou -1 si la fin du flot est atteinte
- int read(byte[] b)
  - lit dans le flot au plus b.length octets qui sont placés dans le tableau b
  - Résultat : nombre d'octets lu ou -1 (si fin flot atteinte)
- int read(byte[] b, int offset, int len)
  - Idem mais len octets lus à partir de offset

OutputStream

- void write(int b)
  - permet d'écrire l'octet de poids faible de b dans le flot
- void write(byte[] b)
  - écrit dans le flot les b.length octets stockés dans le tableau
- void write(byte[] b, int offset, int len)

Exemple1 : lire un octet


```
try{
  byte b;
  int val = System.in.read();
  if(val != -1) b = (byte)val;
  System.out.write(b);
}
catch(IOException e){}
```

Flux de caractères

- Les classes de flux de caractères dérivent des classes abstraites **Reader** et **Writer**
- Méthodes équivalentes à celles des classes InputStream et OuputStream
  - seul le type des données lues est différent
- L'unité n'est pas l'octet mais le caractère
  - Les méthodes read() assurent la lecture de deux octets insécables (codage par défaut)

Exemple2 : lire des caractères au clavier


```
public static String lireLigne()
{
   InputStreamReader flotCar = new InputStreamReader(System.in);
   BufferedReader fluxEntree = new BufferedReader(flotCar);
   String s = null;
   try
   {
        s = fluxEntree.readLine();
   }
   catch (IOException e){}
   return s;
}
```

Flux tampon

- améliorent les performances
- classe **BufferedReader**
  - **public String readLine() throws IOException**
  - lit une ligne de texte et la retourne comme un objet String

Nouvelles classes utiles

- Java.util
- **Scanner**
- **StringTokenizer**

Flux sur fichiers

- **FileInputStream, FileOutputStream**
  - pour créer des flux d'octets
- **FileReader, FileWriter**
  - pour convertir des flux d'octets en flux de caractères
- **RandomAccessFile**
  - permet écriture et lecture dans le même flux

Classe File

- représente le nom d'un fichier ou d'un répertoire
- import java.io.*;

- **boolean canRead()**
- **boolean canWrite()**
- **boolean isFile()**
- **boolean isDirectory()**
- **boolean exists()**
- . . .

Exemple3 : écrire dans un fichier


```
String s = "abc\ndef\nghi";
StringReader sr = new StringReader(s);
BufferedReader entree = new BufferedReader(sr);

FileWriter fw = new FileWriter("fic.data");
BufferedWriter bw = new BufferedWriter(fw);
PrintWriter sortie = new PrintWriter(bw);
String s1;
while ((s1=entree.readLine())!=null)
{
      sortie.println(s1);
      //System.out.println(s1);
}
sortie.close();
```

Exemple4 : lire dans un fichier


```
  File f = new File("fic.data");
  if (!f.exists()) return;
  if (!f.isFile()) return;

  String s2;
  BufferedReader entree =
new BufferedReader(new FileReader("fic.data"));

  while((s2=entree.readLine())!=null)
      System.out.println(s2);
  entree.close();
```

Flux d'objets

- Classes
  - ObjectInputStream et ObjectOutputStream
  - Permettent de lire et d'écrire des graphes d'objets

- **Sérialisation**