Chapitre 6

Les flux

Plan

- Flux binaires (flux d'octets)
- Flux textes (flux de caractères)
- Conversions de flux

Flux

- Un flux représente une suite d'informations (qui s'écoule telle une rivière).
- D'un point de vue donné, un flux est :
  - soit entrant (input stream) telle une source
  - soit sortant (output stream) tel un puits
- Il est alors possible :
  - de lire (read) sur un flux entrant
  - d'écrire (write) sur un flux sortant
- Un flux peut être attaché à un composant tel qu'un clavier, un écran, un fichier, une connexion réseau.

Types de Flux

- Il existe deux catégories de flux :
  - les flux d'octets (ou flux binaires)
  - les flux de caractères (ou flux textes)
- Les flux d'octets permettent de lire et écrire une suite d'octets (sans aucune conversion)
- Les flux de caractères permettent de lire et écrire une suite de caractères (avec conversions éventuelles)

Flux binaires (illustration sur les fichiers)

Flux binaires

- Les flux binaires correspondent à des entrées/sorties binaires. Celles-ci sont :
  - rapides et efficaces :-)
  - peu compréhensible par l'utilisateur :-(
- Par exemple, l'entier (int) 1234 correspond à la séquence de 4 octets : 00 00 04 D2 tandis que l'entier (short) 1234 correspond à la séquence de 2 octets : 04 D2

Flux binaires

- La classe InputStream décrit tout flux binaire entrant. On trouve dans cette classe la méthode suivante :

  public abstract int read() throws IOException
- La classe OutputStream décrit tout flux binaire sortant. On trouve dans cette classe la méthode suivante :

  public abstract void write(int b) throws IOException
- Il est à noter que seul l'octet de poids faible du type int est considéré.

TestByteStream

Flux binaire entrant sur un fichier

- La classe FileInputStream (qui hérite de InputStream) décrit tout flux binaire entrant attaché à un fichier.
- Pour ouvrir un fichier en lecture (en fait, pour ouvrir un flux binaire entrant attaché à un fichier), on peut utiliser le constructeur suivant :

  public FileInputStream(String name)
  throws FileNotFoundException

Flux binaire sortant sur un fichier

- La classe FileOutputStream (qui hérite de OutputStream) décrit tout flux binaire sortant attaché à un fichier.
- Pour ouvrir un fichier en écriture (en fait, pour ouvrir un flux binaire sortant attaché à un fichier), on peut utiliser le constructeur suivant :

```
public FileOutputStream(String name)
   throws FileNotFoundException
```


Bufférisation du flux

- Afin de garantir une plus grande efficacité, il est possible d'utiliser une mémoire tampon dans l'accès au flux.
- Pour obtenir cette nouvelle fonctionnalité, on utilise les constructeurs suivants :

```
public BufferedInputStream(InputStream in)
public BufferedOutputStream(OuputStream out)
```


Lire des valeurs primitives

- La classe DataInputStream (qui hérite de InputStream) permet d'ajouter des fonctionnalités de lecture de données primitives à un flux binaire entrant.
- Pour obtenir ces nouvelles fonctionnalités, on utilise le constructeur suivant :

```
public DataInputStream(InputStream in)
```


Interface DataInput

- En fait, la classe DataInputStream implémente l'interface DataInput, c'est à dire :

```
public boolean readBoolean() throws IOException
public char readChar() throws IOException
public byte readByte() throws IOException
public short readShort() throws IOException
public int readInt() throws IOException
public long readLong() throws IOException
public float readFloat() throws IOException
public double readDouble() throws IOException
…
```


Ecrire des valeurs primitives

- La classe DataOutputStream (qui hérite de OutputStream) permet d'ajouter des fonctionnalités d'écriture de données primitives à un flux binaire sortant.
- Pour obtenir ces nouvelles fonctionnalités, on utilise le constructeur suivant :

```
public DataOutputStream(OutputStream out)
```


Interface DataOutput

- En fait, la classe DataOutputStream implémente l'interface DataOutput, c'est à dire :

```
public void writeBoolean(boolean b) throws IOException
public void writeChar(int c) throws IOException
public void writeByte(int v) throws IOException
public void writeShort(int v) throws IOException
public void writeInt(int i) throws IOException
public void writeLong(long l) throws IOException
public void writeFloat(float f) throws IOException
public void writeDouble(double d) throws IOException
…
```


Ouvrir un flux binaire entrant sur un fichier

```
DataInputStream in=null;
try {
   in = new DataInputStream(new
   BufferedInputStream(new
   FileInputStream(fileName)));
}
catch (FileNotFoundException e) {
   System.out.println(e); System.exit(1);
}
```

- On peut alors utiliser in.readInt(), in.readChar(), in.readDouble(), …

Ouvrir un flux binaire sortant sur un fichier

```
DataOutputStream out=null;
try {
   out = new DataOutputStream(new
   BufferedOutputStream(new
   FileOutputStream(fileName)));
}
catch (FileNotFoundException e) {
   System.out.println(e); System.exit(1);
}
```

- On peut alors utiliser out.writeInt(…), out.writeChar(…), out.writeDouble(…), …

Flux textes
(illustration sur les fichiers)

Flux textes

- Les flux textes correspondent à des entrées/sorties texte. Celles-ci sont :
  - peu efficaces :-(
  - compréhensibles par l'utilisateur :-)
- Par exemple, l'entier (int) 1234 correspond en mode texte à la chaîne "1234". Il en est de même pour l'entier (short) 1234.

Flux textes

- La classe Reader décrit tout flux texte entrant. On trouve dans cette classe la méthode suivante :
  public abstract int read() throws IOException
- La classe Writer décrit tout flux texte sortant. On trouve dans cette classe la méthode suivante :
  public abstract void write(int b) throws IOException
- Il est à noter que seuls les 2 octets de poids faible (codage Unicode) du type int sont considérés.

TestCharacterStream

Flux texte entrant sur un fichier

- La classe FileReader (qui hérite de Reader) décrit tout flux texte entrant attaché à un fichier.
- Pour ouvrir un fichier en lecture (en fait, pour ouvrir un flux texte entrant attaché à un fichier), on peut utiliser le constructeur suivant :
  public FileReader(String fileName)
  throws FileNotFoundException

Flux texte sortant sur un fichier

- La classe FileWriter (qui hérite de Writer) décrit tout flux texte sortant attaché à un fichier.
- Pour ouvrir un fichier en écriture (en fait, pour ouvrir un flux texte sortant attaché à un fichier), on peut utiliser le constructeur suivant :
  public FileWriter(String name)
  throws IOException

Bufférisation du flux

- Afin de garantir une plus grande efficacité, il est possible d'utiliser une mémoire tampon dans l'accès au flux.
- Pour obtenir cette nouvelle fonctionnalité, on utilise les constructeurs suivants :
  public BufferedReader(Reader in)
  public BufferedWriter(Writer out)

Lire des chaînes de caractères

- La classe BufferedReader offre la méthode suivante qui permet de récupérer une chaîne de caractères :
  public String readLine() throws IOException
- Il est important de noter que ce sont des caractères Unicode qui sont lus, représentés par 2 octets consécutifs.
- A line is considered to be terminated by any one of a line feed ('\n'), a carriage return ('\r'), or a carriage return followed immediately by a linefeed.

Ecrire des chaînes de caractères

- La classe Writer offre la méthode suivante :
  public void write(String s) throws IOException
- Il est important de noter que ce sont des caractères Unicode qui sont écrits, représentés par 2 octets consécutifs.
- Il est à noter que DataOutputStream offre la méthode équivalent suivante :
  public final void writeChars(String s) throws IOException

Utiliser print et println

- La classe PrintWriter offre des méthodes surchargées print et println.
- Il est ainsi possible d'envoyer la représentation texte de tout type primitif et de tout objet.
  - pour une valeur primitive, une conversion de celle-ci en une chaîne de caractères la représentant est effectuée ;
  - pour un objet, la méthode toString() est appelée.
- Pour obtenir ces fonctionnalités, on peut utiliser le constructeur :

  public PrintWriter(Writer out)

Ouvrir un flux texte entrant sur un fichier

```
BufferedReader in=null;
try {
   in= new BufferedReader(new
   FileReader(fileName));
}
catch (IOException e)  {
   System.out.println(e); System.exit(1);
}
```

- On peut alors utiliser in.readLine()

Ouvrir un flux texte sortant sur un fichier

```
PrintWriter out=null;
try {
   out = new PrintWriter(new BufferedWriter(new
   FileWriter(fileName)));
}
catch (IOException e) {
   System.out.println(e); System.exit(1);
}
```

- On peut alors utiliser out.print(…), out.println(…)

Conversions

Conversions

- Il est parfois nécessaire de convertir un flux binaire en un flux texte.
- Par exemple, il n'est pas possible de récupérer (ou créer) directement un objet Reader et un objet Writer à partir d'un socket.
- La solution consiste alors à convertir en flux textes les flux binaires obtenus.

InputStreamReader

- Cette classe qui hérite de Reader offre un constructeur permettant de convertir un flux entrant binaire passé en paramètres en un flux entrant texte.
- Ce constructeur est le suivant :

  public InputStreamReader(InputStream in)

OutputStreamWriter

- Cette classe qui hérite de Writer offre un constructeur permettant de convertir un flux sortant binaire passé en paramètres en un flux sortant texte.
- Ce constructeur est le suivant :

  public OutputStreamWriter(OutputStream out)

Illustration sur les sockets

- La classe Socket fournit les méthodes suivantes pour obtenir respectivement un flux entrant binaire et un flux sortant binaire :

  public InputStream getInputStream()
  throws IOException
  public OutputStream getOutputStream()
  throws IOException
- Il est alors possible de convertir ceux-ci en flux textes à l'aide des classes InputStreamReader et OutputStreamWriter.

Ouvrir un flux binaire entrant sur un socket

```
DataInputStream in=null;
try {
   in = new DataInputStream(new
   BufferedInputStream(socket.getInputStream()));
}
catch (IOException e) { System.out.println(e);
   System.exit(1);
}
```

- On peut alors utiliser in.readInt(), in.readChar(), in.readDouble(), …

Ouvrir un flux binaire sortant sur un socket

```
DataOutputStream out=null;
try {
   out = new DataOutputStream(new
   BufferedOutputStream(socket.getOutputStream())
   );
}
catch (IOException e) { System.out.println(e);
   System.exit(1);
}
```

- On peut alors utiliser out.writeInt(…), out.writeChar(…), out.writeDouble(…), …

Ouvrir un flux texte entrant sur un socket

```
BufferedReader in=null;
try {
  in= new BufferedReader(new
   InputStreamReader(socket.getInputStream()));
}
catch (IOException e)  { System.out.println(e);
   System.exit(1);
}
```

- On peut alors utiliser in.readLine()

Ouvrir un flux texte sortant sur un socket

```
PrintWriter out=null;
try {
  out = new PrintWriter(new BufferedWriter(
  new OutputStreamWriter(
   socket.getOutputStream()));
}
catch (IOException e) {
  System.out.println(e); System.exit(1);
}
```

- On peut alors utiliser out.print(…), out.println(…)

Illustration sur l'entrée standard

- L'entrée standard System.in est un objet InputStream qui représente un flux entrant binaire.
- Ce flux est élémentaire. Aussi pour faciliter les saisies semble-t-il possible de procéder selon l'une des 2 approches suivantes :
  - « envelopper » ce flux binaire dans un DataInputStream
  - convertir ce flux binaire en un flux texte

Première approche

```
DataInputStream in=null;
try {
   in = new DataInputStream(new
   BufferdInputStream(System.in));
}
catch (Exception e) {
   System.out.println(e); System.exit(1);
}
```

Quel est le problème ?

- On peut alors utiliser in.readInt(), in.readChar(), in.readDouble(), …

Seconde approche

```
BufferedReader in=null;
try {
   in = new BufferedReader(new
   InputStreamReader(System.in));
}
catch (Exception e) {
   System.out.println(e); System.exit(1);
}
```

- On peut alors utiliser in.readLine()

Classe Mediator

- La première approche n'est pas praticable.
- Aussi, si on souhaite lire une valeur primitive au clavier, faut-il utiliser la seconde approche en procédant comme suit :
  - lire une chaîne de caractères s
  - utiliser une opération de conversion telle que :
    - int i =Integer.parseInt(s)
    - double d = Double.parseDouble(s)
    - …
- La classe Mediator (qui n'existe pas dans le SDK) effectue toutes ces opérations.

Combiner flux entrant binaire et texte sur un fichier

```
DataInputStream inB=null;
BufferedReader inT=null;

try {
    InputStream in = new FileInputStream(fileName);
    inB=new DataInputStream(new BufferedInputStream(in));
    inT=new BufferedReader(new InputStreamReader(in));
}
catch (IOException e) {
    System.out.println(e); System.exit(1);
}
```

- On peut alors utiliser inB.readInt(), inB.readChar(), inB.readDouble() et inT.readLine()

Combiner flux sortant binaire et texte sur un socket

```
DataInputStream outB=null;
PrintWriter outT=null;

try {
   OutputStream out = socket.getOutputStream();
    outB = new DataOutputStream(
      new BufferedInputStream(out));
    outT = new PrintWriter(new BufferedWriter(
      new OutputStreamWriter(out)));
}
catch (IOException e) {
    System.out.println(e); System.exit(1);
}
```

- On peut alors utiliser outB.writeInt(…), outB.writeChar(…), outB.writeDouble(…), outT.print(...)