Entrées-Sorties

Fichiers et package java.io

Lecture et écriture de fichier texte

Lecture au clavier

Sérialisation

Objets et fichiers de type texte et binaire

Entrées-sorties

- Ce sont des opérations qui permettent à un programme d'échanger des données avec son environnement :
  - Lire au clavier
  - Ecrire à l'écran
  - Lire dans un fichier
  - Ecrire dans un fichier
  - …

Fichiers

- L'harmonisation des entrées-sorties permet de voir les opérations correspondantes comme des lectures ou écritures dans des fichiers. On distingue :
  - Les fichiers (de type) texte
  - Les fichiers binaires (et les fichiers d'objets)

Package java.io

- Fournit (des classes qui proposent) des services :
  - d'entrées-sorties au moyen de flux de données (data streams)
  - de sérialisation d'objets
  - d'accès au système de fichiers

Quelques classes

- FileInputStream et FileOutputStream
  pour lire et écrire des octets
- FileReader et FileWriter
  pour lire et écrire des caractères dans un fichier
- BufferedReader et BufferedWriter
  pour lire et écrire des chaînes de caractères
- DataInputStream et DataOutputStream
  pour lire et écrire des données de base (int, float, char, boolean …)

Lecture d'un fichier texte

```
FileReader f;
int lu; char c;
f = new FileReader("monfichier.txt");
while ((lu = f.read()) != -1) {
   c = (char)lu;
   ...
}
f.close();
```

crée un objet FileReader et ouvre le fichier monfichier.txt en lecture
À gérer : **`FileNotFoundException`**

lit un caractère et renvoie son code ou –1 en fin de fichier
À gérer : **`IOException`**

ferme le fichier
À gérer : **`IOException`**

Lecture d'un fichier texte

```
FileReader f;
int lu; char c;
try {
  f = new FileReader("monfichier.txt");
  while ((lu = f.read()) != -1) {
      c = (char)lu;
      ...
  }
  f.close();
} catch (FileNotFoundException e) {
  System.err.println("impossible d'ouvrir "+monfichier.txt+"!");
  System.err.println(e);
} catch (IOException e) {
  System.err.println("erreur de lecture dans "+monfichier.txt+"!");
  System.err.println(e);
}
```

Lecture au clavier

- **`System.in`** est un InputStream
  (on ne peut y lire que des octets) (**`read`**)
- **`new InputStreamReader(System.in)`**
  crée un objet qui peut lire un ou des caractères
  (**`read`**)
- **`new BufferedReader(`**
  `new InputStreamReader(System.in)`
  `)`
  crée un objet qui peut lire une ligne (**`readLine`**)

Ecriture d'un fichier texte

```
FileWriter f;
f = new FileWriter("monfichier.txt");
f.write('a');
f.write("bcde");
f.close();
```

crée un objet FileWriter et ouvre le fichier monfichier.txt en écriture (l'écrase s'il existe)
À gérer : **IOException**

écrit un caractère dans le fichier
À gérer : **IOException**

écrit une chaîne de caractères dans le fichier
À gérer : **IOException**

ferme le fichier
À gérer : **IOException**

Ecriture d'un fichier texte

```
FileWriter f;
try {
  f = new FileWriter("monfichier.txt");
  f.write('a');
  f.write("bcde");
  f.close();
} catch(IOException e) {
  System.err.println(e);
}
```

Objets et fichiers

- Les objets peuvent être lus et écrits dans des fichiers (de type texte ou binaire).
- Il est possible d'utiliser des fichiers d'objets (binaires) en déclarant sérialisable la classe des objets à enregistrer/charger.
  `class Chose implements Serializable {…}`
  Ces fichiers contiennent des informations sur la classe des objets et leurs variables d'instance.

Enregistrement d'un objet

La classe des objets enregistrés doit être sérialisable.

```
Chose uneChose;
ObjectOutputStream f;
try {

   f = new ObjectOutputStream(new FileOutputStream("uneChose.obj"));
   f.writeObject(uneChose);
   f.flush();
   f.close();

} catch(InvalidClassException e) {
   System.err.println(e);
} catch(NotSerializableException e) {
   System.err.println(e);
} catch(IOException e) {
   System.err.println(e);
}
```

Chargement d'un objet

La classe des objets enregistrés doit être sérialisable.

```
Chose uneChose;
ObjectInputStream f;
try {

   f = new ObjectInputStream(new FileInputStream("uneChose.obj"));
   uneChose = (Chose)f.readObject();
   f.close();

} catch(ClassNotFoundException e)    { System.err.println(e);
} catch(InvalidClassException e)     { System.err.println(e);
} catch(StreamCorruptedException e) { System.err.println(e);
} catch(OptionalDataException e)     { System.err.println(e);
} catch(IOException e)               { System.err.println(e);
}
```

lit un objet dans le fichier

Objet dans un fichier texte

```
Chose uneChose;
PrintWriter f;
try {
   f = new PrintWriter(new FileWriter("uneChose.txt"));
   f.println(uneChose);
   f.flush();
   f.close();
} catch(IOException e) {
   System.err.println(e);
}
```

écrit un objet dans le fichier
en utilisant la méthode toString()

Objet depuis un fichier texte

```
Chose uneChose;
BufferedReader f;
try {
  f = new BufferedReader(new FileReader(new File("uneChose.txt")));
  uneChose = new Chose(f.readLine());
  System.out.println(uneChose);
  f.close();
} catch (Exception e) {
  System.err.println(e);
}
```

lit une ligne de texte dans le fichier

crée un objet à partir de la ligne de texte lue dans le fichier en utilisant le constructeur qui extrait au moyen d'un StringTokenizer les valeurs des variables d'instances dans une chaîne de caractères délimitée

Objet dans fichier binaire

- Les valeurs des variables d'instances d'objets peuvent être lues et écrites dans des fichiers binaires :

- DataInputStream
  - readInt, readByte, readShort, readLong
  - readUTF
  - readBoolean
  - readFloat, readDouble
- DataOutputStream
  - writeInt, writeByte, writeShort, writeLong
  - writeUTF
  - writeBoolean
  - writeFloat, writeDouble

Conversions type de base ↔ texte

```
class Integer {
  private int value;

  public Integer(int value) {
    this.value = value;
  }
  public int intValue() {
    return value;
  }
  public toString() {
    String texte = "";
    texte += value;
    return texte;
  }
  public Integer(String texte) {
    value = Integer.parseInt( texte );
  }
}
```

rien que de l'encapsulation …

… mais qui permet de convertir en texte

… ou à partir d'un texte

procède au balayage du texte caractère par caractère et calcule la valeur entière notée dans le texte par additions et multiplications par 10

Autres conversions

- quelques méthodes définies dans **`Integer`**
  - `byte byteValue();`
  - `short shortValue();`
  - `long longValue();`
  - `float floatValue();`
  - `double doubleValue();`
- Classes **`Byte, Short, Integer, Long`**
- Classes **`Float, Double`**

StringTokenizer

```
String texteADecouper;
StringTokenizer decoupeur;

texteADecouper = "15:32:55";
decoupeur = new StringTokenizer( texteADecouper, ":" );

while ( decoupeur.hasMoreTokens() ) {
  System.out.println( decoupeur.nextToken() );
}
```

produit sur la sortie standard :

```
15
32
55
```

Conversions objet ↔ texte

```
class Heure {
  private int heures, minutes, secondes;

  public Heure(int h, int m, int s) {
    heures = h;
    minutes = m;
    secondes = s;
  }
  public Heure(String texte) {
    StringTokenizer st = new StringTokenizer(texte, ":");
    heures = Integer.parseInt( st.nextToken() );
    minutes = Integer.parseInt( st.nextToken() );
    secondes = Integer.parseInt( st.nextToken() );
  }
  public toString() {
    String texte = "";
    texte += heures + ":" + minutes + ":" + secondes;
    return texte;
 }
}
```