**Les flux et les fichiers**

**1. Entrées-Sorties**

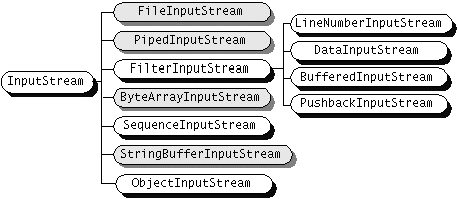
**1.1. Introduction**

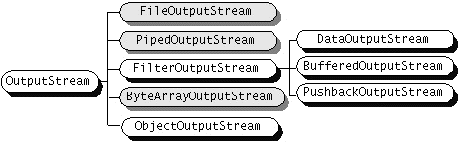
Les entrées/sorties sont basés sur les *flots* (*streams* ). Un *flot de données* est un canal de communication constitué d'une suite ordonnée de données ayant deux extrémités : une *entrée* et une *sortie*. *Java* fournit un moyen de manipuler des flots de données de manière identique quelque soit le système d'exploitation. L'ensemble des classes permettant d'utiliser les entrées/sorties est regroupé dans le package *java.io*.

Les librairies Java distinguent deux genres de flux :

• Les flux binaires *(Byte Stream)* qui peuvent représenter des données quelconques (nombres, données structurées, programmes, sons, images, …)

Le package ***java.io*** contient les classes ***InputStream*** et ***OutputStream*** qui sont des classes abstraites qui définissent les méthodes **d'entrée/sortie** de base. Elles manipulent des suites d'octets.



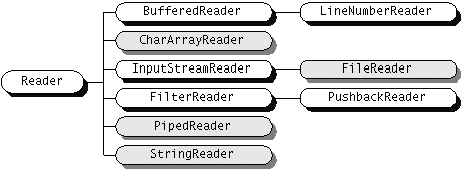


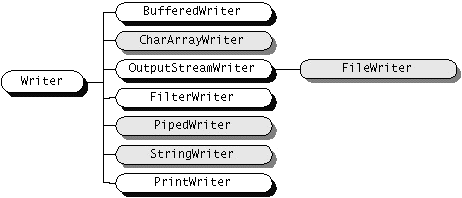
• Les flux de caractères *(Character Stream)* qui représentent des textes (chaînes de caractères)*.*

Les classes ***Reader*** et ***Writer***, tout comme les classes ***InputStream*** et ***OutputStream***, sont les classes de base à partir desquelles toutes les autres classes sont dérivées. Ces classes sont également des classes abstraites et manipulent comme unité de base le caractère (et non pas l'octet).

**La classe Reader :** C'est une classe abstraite qui est la classe mère de toutes les classes qui gèrent des flux de caractères en lecture.

**La classe Writer :** C'est une classe abstraite qui est la classe mère de toutes les classes qui gèrent des flux de caractères en écriture.





La communication avec un flux comprend trois phases :

* L'ouverture du flux, en entrée (pour la lecture) ou en sortie (pour l'écriture)
* La répétition de la lecture ou de l'écriture des données (généralement des octets ou des caractères)
* La fermeture du flux

Les classes les plus importantes pour utiliser les flux sont définies ci-dessous :

|  |  |  |
| --- | --- | --- |
|  | **Flux d'entrée** | **Flux de sortie** |
| **Flux binaires** | *DataInputStream* | *DataOutputStream* |
| **Flux de caractères** | *BufferedReader* | *PrintWriter* |

**1.2. Entrées/Sorties standard (console)**

Les flux suivants sont prédéfinis dans la classe System. Ces flux sont toujours ouverts et on ne les ferme pas.

• System.in Entrée standard (lecture du clavier) [ InputStream ]

• System.out Sortie standard (affichage à l'écran) [ PrintStream ]

• System.err Sortie des messages d'erreur [ PrintStream ]

**1.2.1. Lecture d’information au clavier**

**Exemple :** Il s’agit d’une classe clavier offrant les services de base que sont la lecture d’un entier, d’un flottant ou d’un caractère.

// classe fournissant des fonctions de lecture au clavier -

**import** java.io.\* ;

**public** **class** clavier

{ **public** **static** String lireString () // lecture d'une chaine

{ String ligne\_lue = **null** ;

**try**

{

InputStreamReader lecteur = **new** InputStreamReader (System.*in*) ;

BufferedReader entree = **new** BufferedReader (lecteur) ;

ligne\_lue = entree.readLine() ;

}

**catch** (IOException err)

{ System.*exit*(0) ;

}

**return** ligne\_lue ;

}

**public** **static** **float** lireFloat () // lecture d'un float

{ **float** x=0 ; // valeur a lire

**try**

{ String ligne\_lue = *lireString*() ;

x = Float.*parseFloat*(ligne\_lue) ;

}

**catch** (NumberFormatException err)

{

System.*out*.println ("\*\*\* Erreur de données \*\*\*") ;

System.*exit*(0) ;

}

**return** x ;

}

**public** **static** **double** lireDouble () // lecture d'un double

{ **double** x=0 ; // valeur a lire

**try**

{ String ligne\_lue = *lireString*() ;

x = Double.*parseDouble*(ligne\_lue) ;

}

**catch** (NumberFormatException err)

{ System.*out*.println ("\*\*\* Erreur de données \*\*\*") ;

System.*exit*(0) ;

}

**return** x ;

}

**public** **static** **int** lireInt () // lecture d'un int

{ **int** n=0 ; // valeur a lire

**try**

{ String ligne\_lue = *lireString*() ;

n = Integer.*parseInt*(ligne\_lue) ;

}

**catch** (NumberFormatException err)

{

System.*out*.println ("\*\*\* Erreur de données \*\*\*") ;

System.*exit*(0) ;

}

**return** n ;

}

// programme de test de la classe Clavier

**public** **static** **void** main (String[] args)

{ System.*out*.println ("donnez un flottant") ;

**float** x ;

x = clavier.*lireFloat*() ;

System.*out*.println ("merci pour " + x) ;

System.*out*.println ("donnez un entier") ;

**int** n ;

n = clavier.*lireInt*() ;

System.*out*.println ("merci pour " + n) ;

}

}

**1.2.2. Classe *Scanner***

Une autre manière de faire (souvent plus pratique) est d'utiliser la classe *Scanner* (java.util) qui permet également de saisir mais, en plus, de transformer et d'interpréter ce qui a été saisi au clavier.

**Exemple:**

**import** java.util.Scanner;

**public** **class** TestScanner {

**public** **static** **void** main(String[] args) {

Scanner scan = **new** Scanner(System.*in*);

System.*out*.print("Entrer un entier (ou 'x' pour quitter) : ");

**while** (scan.hasNextInt()) {

// hasNextFloat()

**int** i = scan.nextInt();

//float i = scan.nextFloat();

System.*out*.println("Valeur = " + i);

System.*out*.print("Entrer un entier (ou 'x' pour quitter) : ");

}

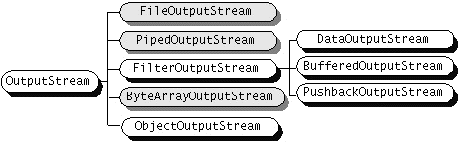
}

}

**2. Création séquentielle d’un fichier binaire**

La classe abstraite ***OutputStream*** sert de base à toutes les classes relatives à des flux binaires de sortie. La classe ***FileOutputStream***, dérivée de ***OutputStream*,** permet de manipuler un flux binaire associé à un fichier en écriture et possède plusieurs constructeurs :

|  |  |
| --- | --- |
| Constructeur | Rôle |
| FileOutputStream(String) | Si le fichier précisé n'existe pas, il sera créé. S'il existe et qu'il contient des données celles-ci seront écrasées. |
| FileOutputStream(String, boolean) | Le booléen permet de préciser si les données seront ajoutées au fichier (valeur true) ou écraseront les données existantes (valeur false) |



***FileOutputStream* f =new *FileOutputStream* (**"test.dat"**)**: associer l’objet f à un fichier de nom test.dat.

La classe ***DataOutputStream*** possède des méthodes permettant d’envoyer sur un flux (donc ici écrire dans un fichier) des valeurs. Exemple de méthodes : *writeInt*, *writeFloat*, etc. Cette classe dispose d’un constructeur recevant en argument un objet de type ***FileOutputStream* :**

***DataOutputStream*** sortie = **new** ***DataOutputStream*** (f);

Pour simplifier les deux instructions de créations peuvent être condensées en ***DataOutputStream*** sortie = **new** ***DataOutputStream***

( **new** ***FileOutputStream*** (nomfich)) ;

**Exemple** : Il s’agit d’un programme java qui lit des nombres entiers au clavier et qui les recopie dans un fichier binaire en utilisant les trois phases de la communication avec un flux qui sont l'ouverture du flux en sortie (pour l'écriture), la répétition de l'écriture des données (si différent de 0) et la fermeture du flux. Pour simplifier, on utilise la classe clavier.

////Programme de création de fichier

**public** **static** **void** main (String args[]) **throws** IOException

{

String nomfich ;

**int** n ;

System.*out*.print ("donnez le nom du fichier à créer : ") ;

nomfich = clavier.*lireString*();

DataOutputStream sortie = **new** DataOutputStream

( **new** FileOutputStream (nomfich)) ;

**do** { System.*out*.print ("donnez un entier : ") ;

n = clavier.*lireInt*() ;

**if** (n != 0)

{ sortie. writeInt (n) ;

}

}

**while** (n != 0) ;

sortie.close () ;

System.*out*.println ("\*\*\* fin creation fichier \*\*\*");

}

}

**3. Liste séquentielle d’un fichier binaire**

Le principe est le même pour la lecture avec ***FileInputStream*(String)** et ***DataInputStream*** qui peuvent être condensées en

***DataInputStream*** entree = *new* ***DataInputStream*** ( *new* ***FileInputStream*** (nomfich))

**Exemple:** Liste séquentielle du fichier binaire crée par le programme java de l’exemple précédent

**import** java.io.\* ;

**public** **class** Lecsfic1

{ **public** **static** **void** main (String args[]) **throws** IOException

{ String nomfich ;

**int** n = 0 ;

System.*out*.print ("donnez le nom du fichier à lister : ") ;

nomfich = clavier.*lireString*() ;

DataInputStream entree = **new** DataInputStream

( **new** FileInputStream (nomfich)) ;

System.*out*.println ("valeurs lues dans le fichier " + nomfich + " :") ;

**boolean** eof = **false** ; // sera mis à true par exception EOFile

**while** (!eof)

{ **try**

{ n = entree.readInt () ;

}

**catch** (EOFException e)

{ eof = **true** ;

}

**if** (!eof) System.*out*.println (n) ;

}

entree.close () ;

System.*out*.println ("\*\*\* fin liste fichier \*\*\*");

}

}

**4. Accès direct a un fichier**

Avec un fichier à accès direct, il est possible d'accéder à un élément en fournissant sa position, sautant ainsi par dessus d'autres éléments. La position d'un élément est donnée par un nombre entier positif ou nul.

Un fichier à accès direct possède un index, qui donne la position de l'élément courant du fichier. Lorsque le fichier est ouvert l'index est mis à la valeur 0, il peut être positionné par *seek(long pos*) que le fichier soit vide ou pas. L'index progresse au gré des opérations de lecture et d'écriture (il est incrémenté de 1 après l'opération). La valeur de l'index peut être connue par la fonction : *getFilePointer().* Il faut savoir que cet index avance par byte, ainsi à l'écriture d'un *int* il avance de 4.

La classe ***RamdonAccessFile*** encapsule les opérations de lecture/écriture d'un tel fichier. Elle implémente les interfaces ***DataInput*** et ***DataOutput***. Elle possède deux constructeurs qui attendent en paramètres le fichier à utiliser (sous la forme d'un nom de fichier ou d'un objet de type File qui encapsule le fichier) et le mode d'accès. Le mode est une chaîne de caractères qui doit être égale à «r» ou «rw» selon que le mode est lecture seule ou lecture/écriture.

**Exemple :**

**import** java.io.\* ;

**public** **class** Accdir

{ **public** **static** **void** main (String args[]) **throws** IOException

{

String nomfich ;

**int** n, num ;

RandomAccessFile entree ;

System.*out*.print ("donnez le nom du fichier à consulter : ") ;

nomfich = clavier.*lireString*() ;

entree = **new** RandomAccessFile (nomfich, "r") ;

**do**

{ System.*out*.print ("Numéro de l'entier recherche : ") ;

num = clavier.*lireInt*() ;

**if** (num == 0) **break** ;

entree.seek (4\*(num-1)) ;

n = entree.readInt() ;

System.*out*.println (" valeur = " + n) ;

}

**while** (num != 0) ;

entree.close () ;

System.*out*.println ("\*\*\* fin consultation fichier \*\*\*");

}

}

**Exercice 1:**  En cas de positionnement hors fichier, *seek* lance l’exception *IOException*. Pour éviter le lancement de cette exception, quelles sont les instructions à ajouter au programme précédent ?