NFP121, Cnam/Paris
Les entrées-sorties
ou l'usage du patron Décorateur
+ Le patron DAO

version du 16 Décembre 2016

Notes de cours

Bibliographie

- **Le cours de J-L Dewez, NFP121/2009**

Sommaire

- **La base**
  - **Le patron Décorateur : le cours 8**
- **Les Entrées-Sorties**
- **Le patron DAO**
  - **Data Access Object et sa fabrique (DAOFactory)**
  - **Une introduction à JDBC (Java DataBase Connectivity)**
    - **Derby** https://db.apache.org/derby/
    - **Hsqldb** http://hsqldb.org/
    - **MySQL** http://dev.mysql.com/downloads/connector/j/
- Sérialisation ? : un autre support

Le monde des Entrées/Sorties

Tour de Babel, selon Bruegel l'ancien, XVIème

Le monde des Entrées/Sorties

- OS différents
- Fichiers différents
- Entrées différentes
- Sorties différentes
- Codages différents
  - → Classes abstraites minimalistes
    - → Décorées…

Des OS différents

- **Windows NT**
- **Windows XP**
- **Windows 10 and so on…**
- **Macintosh OS X El Capitan…**
- **GNU Linux**
- **Red Hat Linux**
- **BSD**

Des fichiers différents

fichiers normaux

* texte : courrier, sources des programmes, scripts, configuration ...
* exécutables : programmes en code binaire

fichiers répertoires

ce sont des fichiers conteneurs qui contiennent des références à d'autres fichiers.

fichiers spéciaux (style unix)

situés dans ***/dev***, ce sont les points d'accès préparés par le système aux périphériques. Le montage va réaliser une correspondance de ces fichiers spéciaux vers leur répertoire "point de montage".
par exemple, le fichier /dev/hda permet l'accès et le chargement du 1er disque IDE

fichiers liens symboliques

Ce sont des fichiers qui ne contiennent qu'une référence (un pointeur) à un autre fichier.
Cela permet d'utiliser un même fichier sous plusieurs noms sans avoir à le dupliquer sur le disque.

Des entrées différentes

- Lecture d'un fichier depuis le disque.
- Réception d'une page web depuis un serveur distant.
- Réception d'un signal au travers d'un réseau
- Scanner, video camera, ...
- Souris, clavier, joystick

Des sorties différentes

- Ecriture d'un fichier sur disque local.
- Envoi d'une requête vers un serveur.
- Envoi d'une commande vers un robot.
- Impression d'un document vers un fax.
- Affichage graphique sur un écran.

Des codages différents

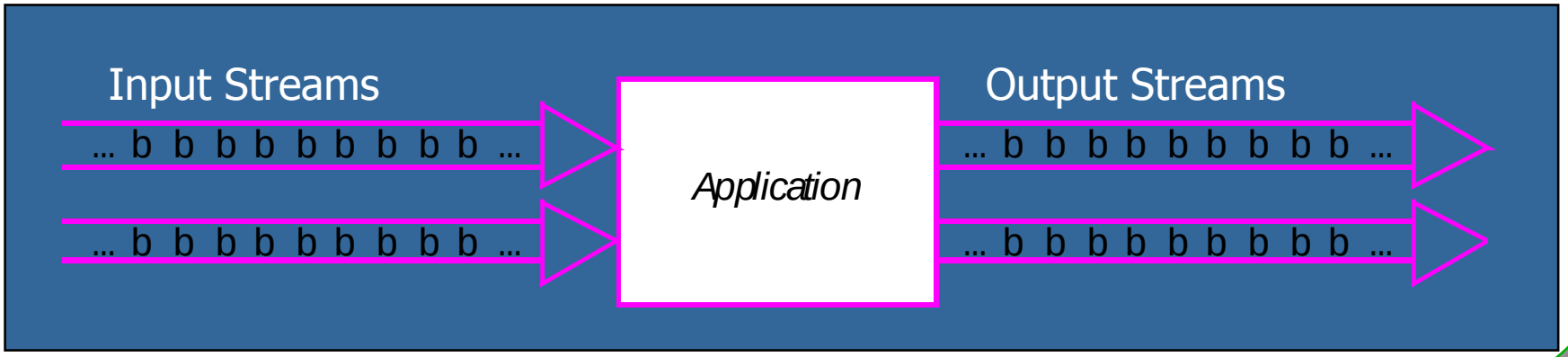
ISO 8859-1	Latin-1	Western European languages (French, German, Italian, etc.)
ISO 8859-2	Latin-2	Eastern European languages (Czech, Hungarian, Polish, etc.)
ISO 8859-3	Latin-3	Southern European languages (Maltese & Turkish)
ISO 8859-4	Latin-4	Northern European languages (Latvian, Lithuanian, etc.)
ISO 8859-5	Cyrillic	Russian, Bulgarian, Ukrainian, Belarusian, Serbian, Macedonian
ISO 8859-6	Arabic	Arabic
ISO 8859-7	Greek	Greek
ISO 8859-8	Hebrew	Hebrew
ISO 8859-9	Latin-5	Turkish (replaces Latin-3)
ISO 8859-10	Latin-6	Northern European (unifies Latin-1 and Latin-4)
ISO 8859-11	Thai	Thai
ISO 8859-13	Latin-7	Baltic languages (replaces Latin-4, supplements Latin-6)
ISO 8859-14	Latin-8	Celtic languages
ISO 8859-15	Latin-9	Western European languages (replaces Latin-1)
ISO 8859-16	Latin-10	Eastern European languages (replaces Latin-2)

Abstraction E/S

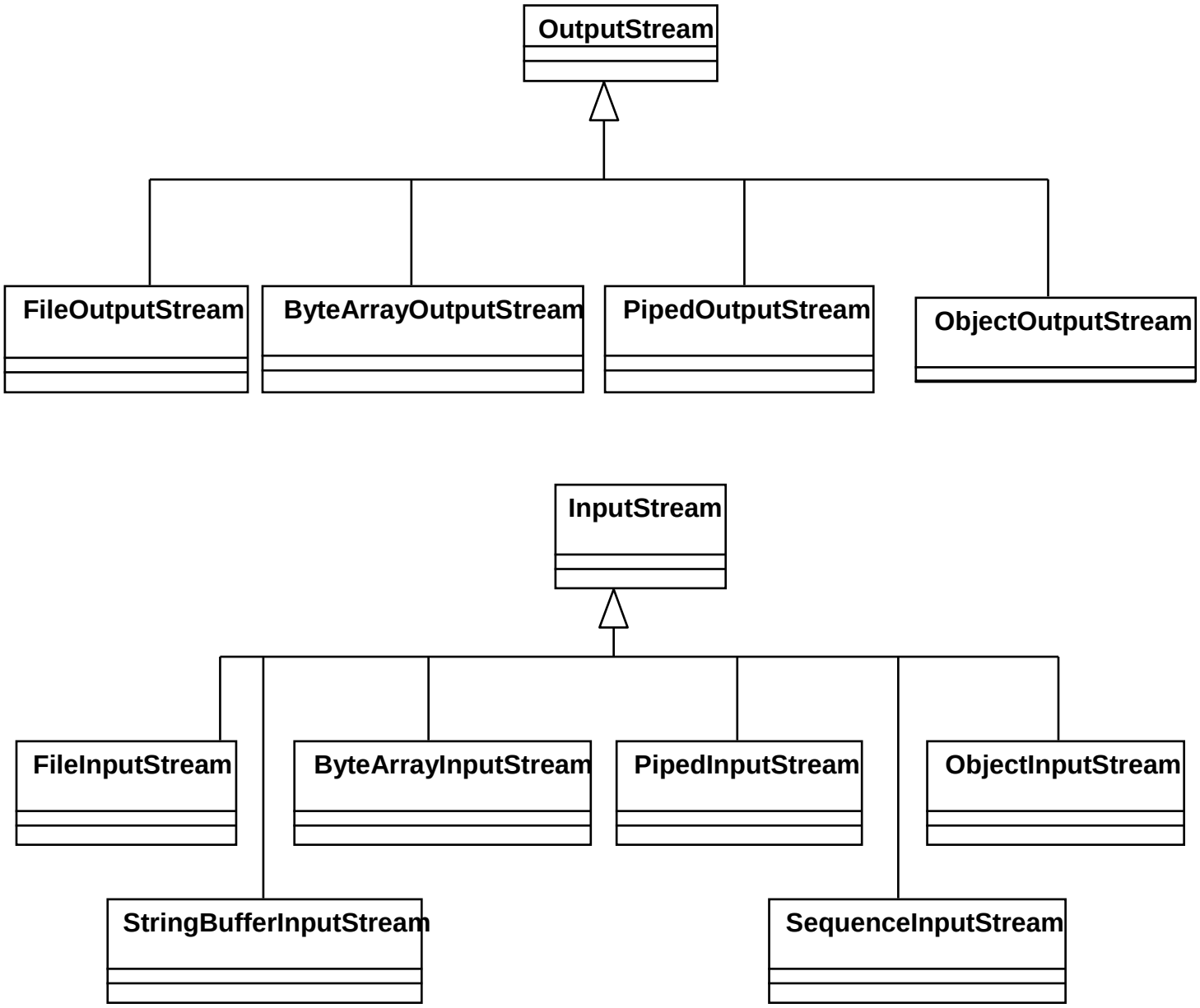
- **Réaliser une interface commune pour lire et écrire tous les types de données**
- **Cette interface sera implémentée par la notion de flots d'E/S**
  - **input et output *streams*.**
- **Toute E/S peut être représentée par une suite de bits.**
- **Cette suite est découpée en octet/Byte.**

Input/Output streams

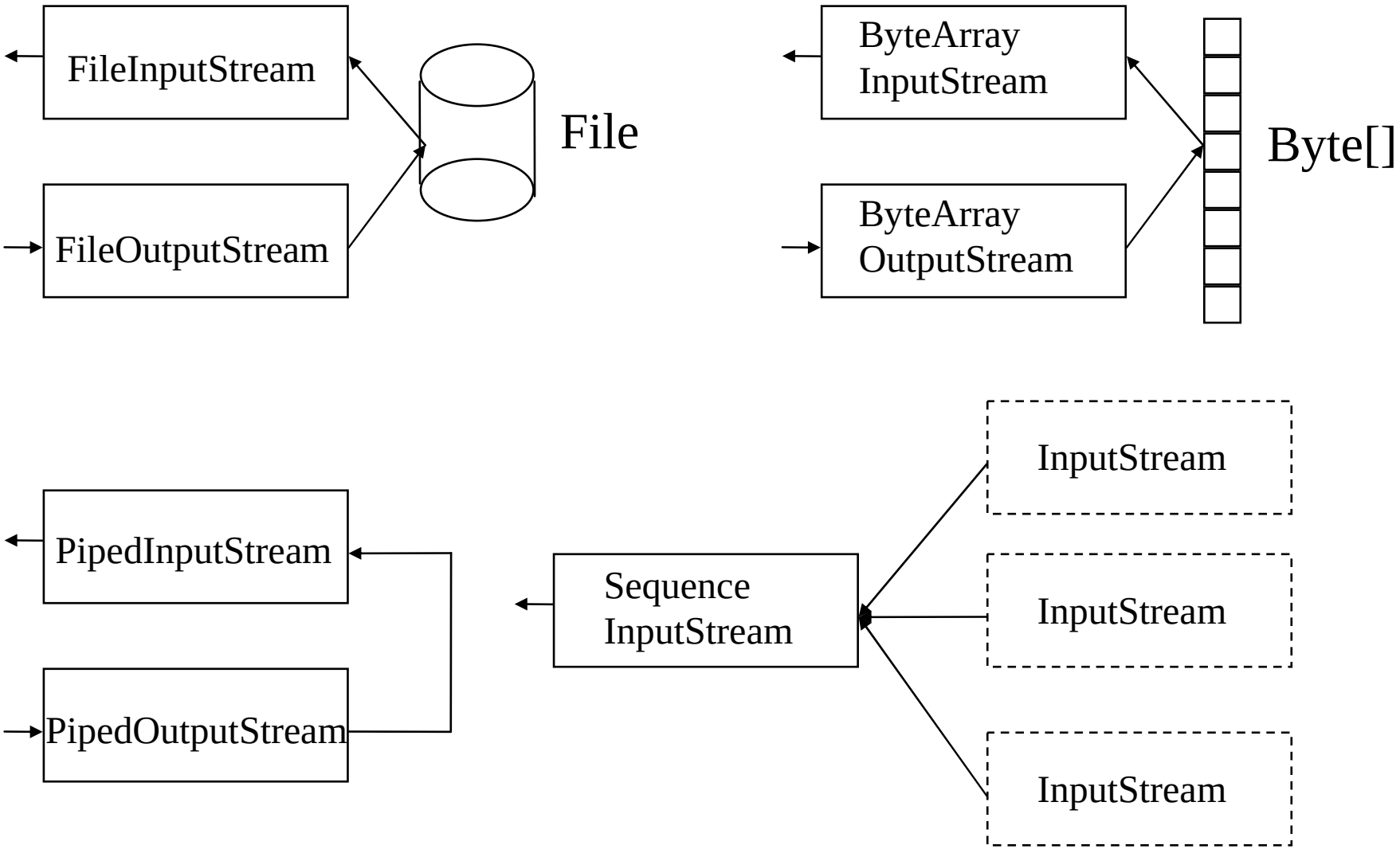
- **Un flot d'E/S est donc une suite d'octets attachée à une source en entrée ou en sortie.**
- **On peut lire ou écrire des données séquentiellement.**
- **Un octet ou plusieurs octets à la fois.**



Streams et supports



Input et Output



Autres sources de flots

Il existe aussi

StringBufferInputStream (Deprecated)

ObjectInputStream et ObjectOutputStream

et des streams privés tels que SocketInputStream et SocketOutputStream de java.net, seulement accessibles par le biais d'opérations sur les "Sockets"

InputStream et OutputStream

<abstract>

InputStream
read () : int read (b : byte[]) : int read (b : byte[], off : int, len : int) : int mark (readlimit : int) : void skip (n : long) : long close () : void reset () : void available () : int markSupported () : boolean

OutputStream
close () : void flush () : void write (b : int) : void write (b : byte[]) : void write (b : byte[], off : int, len : int) : void

Class InputStream

```
public abstract int read()
                          throws IOException
```

Lit le prochain octet du flot.
Retourne -1 si la fin du flot est atteinte (ctrl Z sous DOS).

```
public int read(byte[] b)
                          throws IOException
```

Lit jusqu'à *b.length* octets depuis ce flot vers le tableau b.
Retourne le nombre d'octets lus.

```
public void close() throws IOException
```

Ferme ce flot en entrée et libère les ressources associées

Lecture ascii

```
import java.io.*;

public class SimpleIn {
  static public void main (String[] args) throws IOException {
    int charRead;
    System.out.println(System.in.getClass());
    while ((charRead = System.in.read()) != -1)
            System.out.println(charRead);
   }
}
```

Class OutputStream

```
public abstract void write(int b) throws IOException
```

Ecrit l'octet (fournit comme int) sur le flot en sortie

```
public void write(byte[] b)
                          throws IOException
```

Ecrit *b.length* octets depuis le tableau d'octets spécifié vers le flot en sortie.

```
public void flush() throws IOException
```

Force l'écriture du contenu d'éventuels buffers

```
public void close() throws IOException
```

Ferme ce flot en sortie et libère les ressources associées

Ecriture Ascii

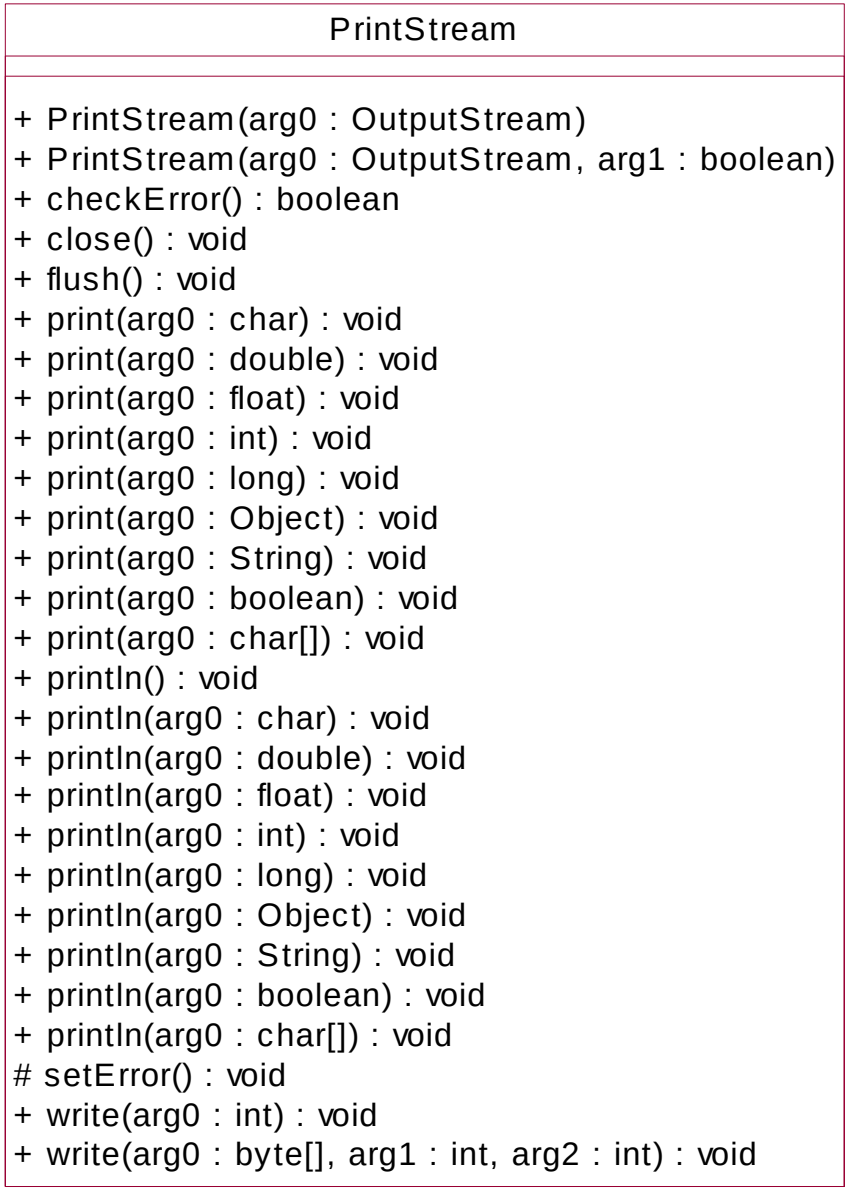
```
import java.io.*;
public class SimpleOut {
  public static void main (String[] args) throws IOException {
    for (int i = 0; i < args.length; ++ i) {
      println (args[i]);
    } }
  public static void println (String msg) throws IOException {
      for (int i = 0; i < msg.length (); ++ i)
              System.out.write (msg.charAt (i) & 0xff);
     System.out.write ('\n'); System.out.flush ();
  }
}
```

ByteArray(In/Out)putStream

ByteArrayInputStream
read () : int read (b : byte[], off : int, len : int) : int mark (markpos : int) : void skip (n : long) : long reset () : void ByteArrayInputStream (buf : byte[]) : void ByteArrayInputStream (buf : byte[], offset : int, length : int) : void available () : int

ByteArrayOutputStream
size () : int reset () : void write (b : int) : void write (b : byte[], off : int, len : int) : void ByteArrayOutputStream () : void ByteArrayOutputStream (size : int) : void toByteArray () : byte[] toString () : String toString (hibyte : int) : String writeTo (out : OutputStream) : void

PrintStream



Exemple

```
public class ByteArrayOutputTest {
  static public void main (String[] args) throws IOException {
    ByteArrayOutputStream byteArrayOut = new ByteArrayOutputStream ();
    byte[] buffer = new byte[16];
    int numberRead;
    while ((numberRead = System.in.read (buffer)) > -1)
         byteArrayOut.write (buffer, 0, numberRead);
    System.out.println ("Read " + byteArrayOut.size () + " bytes.");
    System.out.write (byteArrayOut.toByteArray ());
    for (int i = 0; i < args.length; ++ i) {
         System.out.println("Written to " + args[i] + ".");
         FileOutputStream fileOut = new FileOutputStream (args[i]);
         byteArrayOut.writeTo (fileOut);
         fileOut.close ();
    }    byteArrayOut.close ();}
}
```

Exemple

Ecrire un programme qui lit des caractères au clavier et les insère dans une chaîne passée en paramètre avant que d'afficher à nouveau l'ensemble.

(Sans utiliser la classe PrintStream)

Corrigé

```
import java.io.*;

public class InputOutput {
    byte c ;
    ByteArrayOutputStream intermediaire=new ByteArrayOutputStream();
    String chaine;

    public static void main(String args[]) throws IOException{
            new InputOutput().run(args);
    }
    void run(String args[]) throws IOException{
            int i=0;
            chaine = args[0];
            while((c=(byte)System.in.read())>=0){
                intermediaire.write(c);
                if(i<chaine.length()) intermediaire.write(chaine.charAt(i++));
                }
            intermediaire.writeTo(System.out);
    }
}
```

Piped(In/Out)putStream

PipedInputStream
read () : int read (b : byte[], off : int, len : int) : int close () : void PipedInputStream () : void receive (b : int) : void receive (b : byte[], off : int, len : int) : void available () : int receivedLast () : void PipedInputStream (src : PipedOutputStream) : void **connect (src : PipedOutputStream) : void**

PipedOutputStream
close () : void flush () : void write (b : int) : void write (b : byte[], off : int, len : int) : void PipedOutputStream () : void PipedOutputStream (snk : PipedInputStream) : void **connect (snk : PipedInputStream) : void**

File(In/Out)putStream

FileInputStream
read () : int read (b : byte[]) : int read (b : byte[], off : int, len : int) : int skip (n : long) : long close () : void finalize () : void available () : int readBytes (b : byte[], off : int, len : int) : int **FileInputStream (file : File)** : void open (name : String) : void **FileInputStream (name : String)** : void **getFD () : FileDescriptor** **FileInputStream (fdObj : FileDescriptor)** :

FileOutputStream
close () : void write (b : int) : void write (b : byte[]) : void write (b : byte[], off : int, len : int) : void finalize () : void FileOutputStream (file : File) : void open (name : String) : void writeBytes (b : byte[], off : int, len : int) : void FileOutputStream (name : String) : void FileOutputStream (name : String, append : boolean) : void openAppend (name : String) : void getFD () : FileDescriptor FileOutputStream (fdObj : FileDescriptor) : void

Exemple

Ecrire un programme qui recopie un fichier texte dans un autre fichier texte.

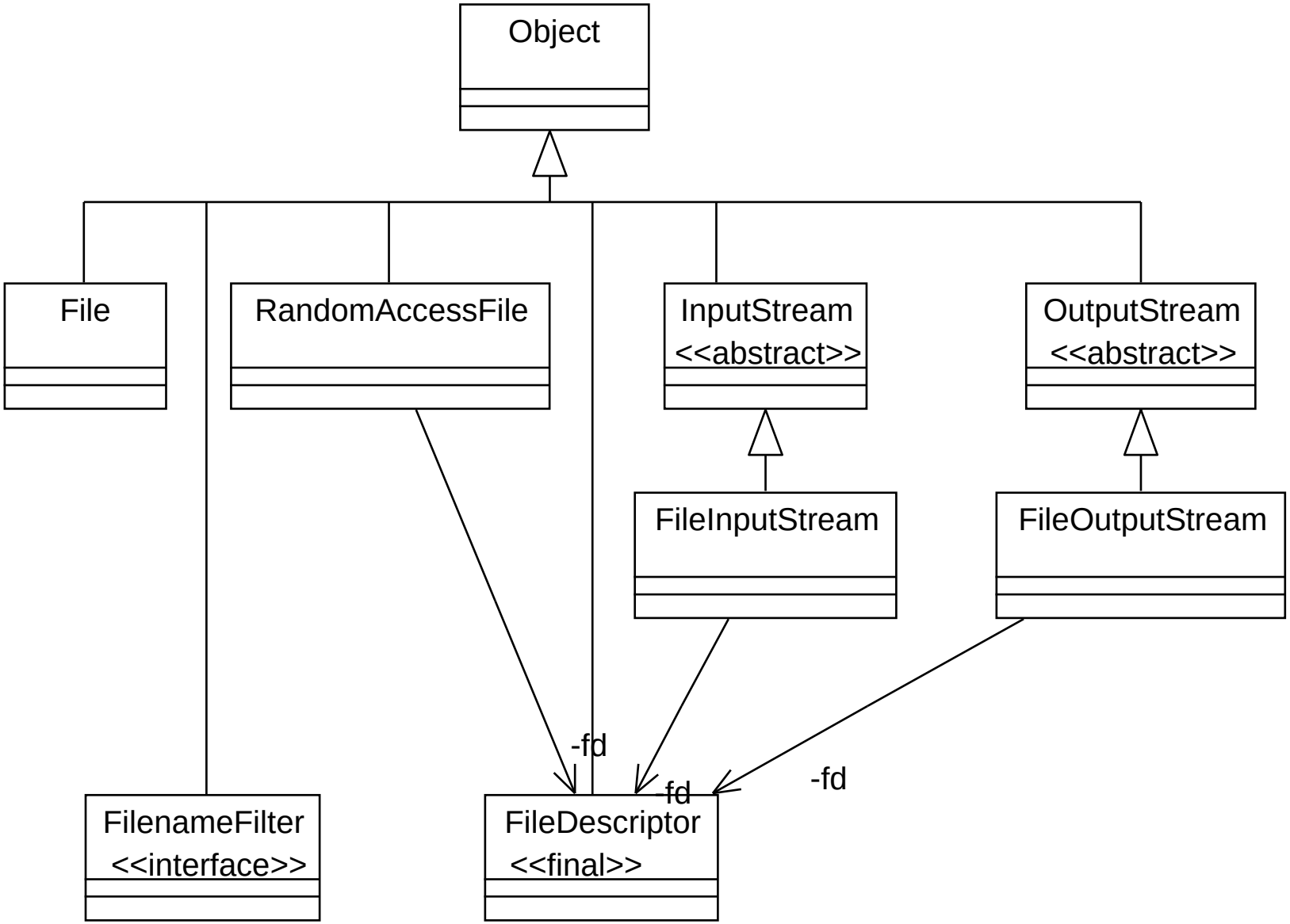
Les deux noms de fichiers seront passés en paramètres.

Corrigé

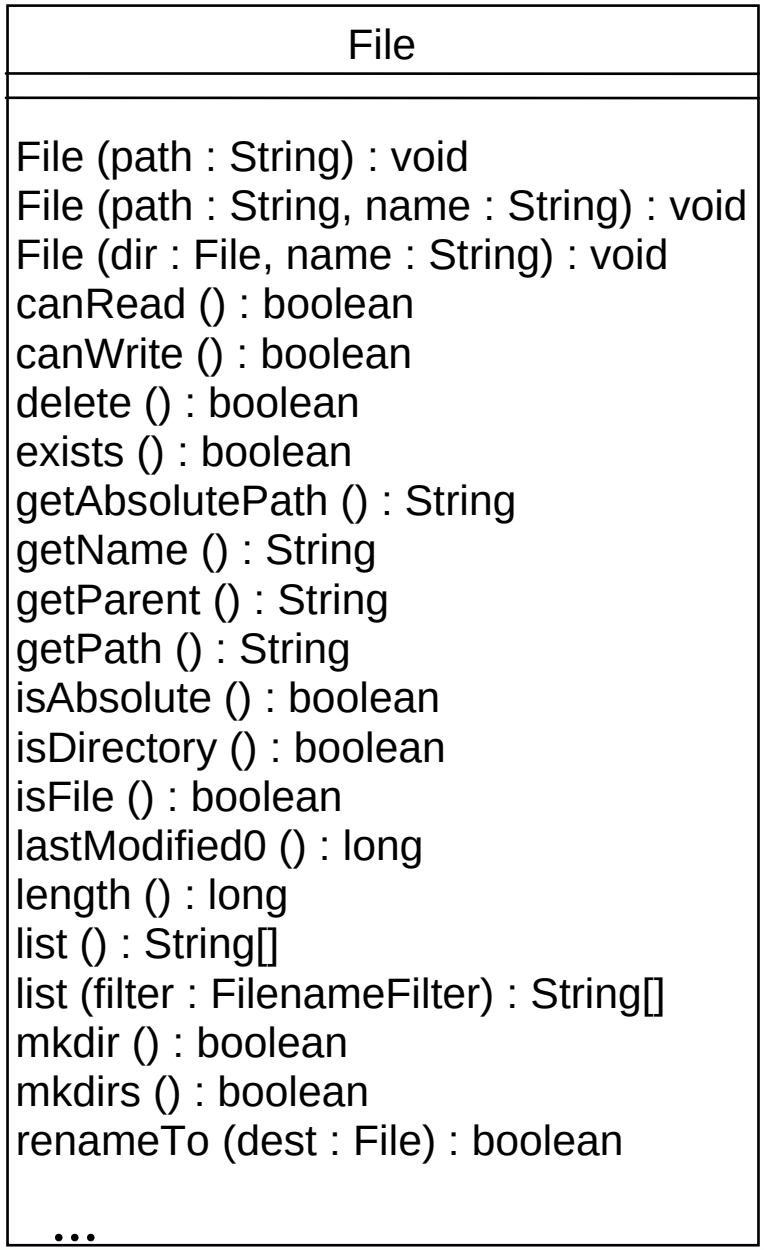
```
import java.io.*;

public class Files {

    public static void main(String args[]) throws IOException{
            FileInputStream in = new FileInputStream(args[0]);
            FileOutputStream out = new FileOutputStream(args[1]);
            byte buffer[] = new byte[256];
            int n;
            while((n=in.read(buffer))>=0) out.write(buffer,0,n);
            out.close();
            in.close();
    }
}
```

- Le constructeur
  `FileOutputStream(String nom, boolean append)`
  permet d'ajouter à la fin du fichier
- Sinon, le contenu du fichier est effacé à la création du flot

Streams et fichiers



class File


File
File (path : String) : void
File (path : String, name : String) : void
File (dir : File, name : String) : void
canRead () : boolean
canWrite () : boolean
delete () : boolean
exists () : boolean
getAbsolutePath () : String
getName () : String
getParent () : String
getPath () : String
isAbsolute () : boolean
isDirectory () : boolean
isFile () : boolean
lastModified0 () : long
length () : long
list () : String[]
list (filter : FilenameFilter) : String[]
mkdir () : boolean
mkdirs () : boolean
renameTo (dest : File) : boolean
...

Exemple

Créer un programme qui prend en argument un nom de répertoire et édite ses fichiers, leur nature et les droits associés.

Corrigé 1

```
import java.io.*;  // d'après Java programming Explorer

class ListDir {

  public static void main (String args[]) {
    int MAXFILES = 100; // maximun files per directory
    if (args.length < 1) { System.out.println("Usage: Java listDir dirName");
      return;
    }
    File dirName = new File(args[0]);
    if (!dirName.exists()) {System.out.println(args[0] +
                                    " does not exist on file system");
      return; }

    if (!dirName.isDirectory())
            {System.out.println(args[0] + " is not a directory");return; }
    System.out.print("Files in directory: " + dirName.getAbsolutePath() + "\n\n\n");
    String fileArr[] = new String[MAXFILES];

    /* suite ....*/
  }
}
```

Corrigé1 (suite)

```
try {fileArr = dirName.list();}
   catch (Exception e) { System.out.println("Error listing contents of " + dirName);
          System.exit(0);
   }
   for (int i = 0; i < fileArr.length; i++) {
     File filename = new File(args[0] + System.getProperty("file.separator") + fileArr[i]);
     System.out.println("  Filename: " + fileArr[i]);
     if (filename.exists()) {
          if (filename.isFile()) { System.out.println("    Is a file");}
          if (filename.isDirectory()) {System.out.println("    Is a directory"); }
          if (filename.canWrite()) { System.out.println("    You have write permission"); }
          if (filename.canRead()) { System.out.println("    You have read permission"); }
     }
     else {System.out.println("    Does not exist");}
   System.out.println("----------------------------");
   }
  }
}
```

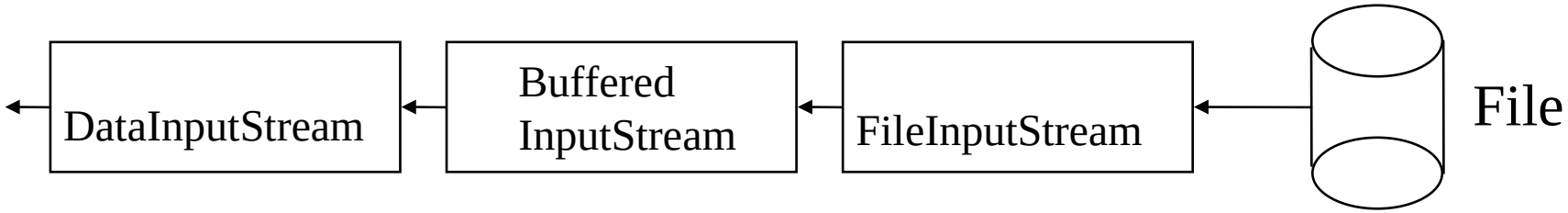
Filtres : Etendre les Streams

Les fonctionnalités des streams sont minimales

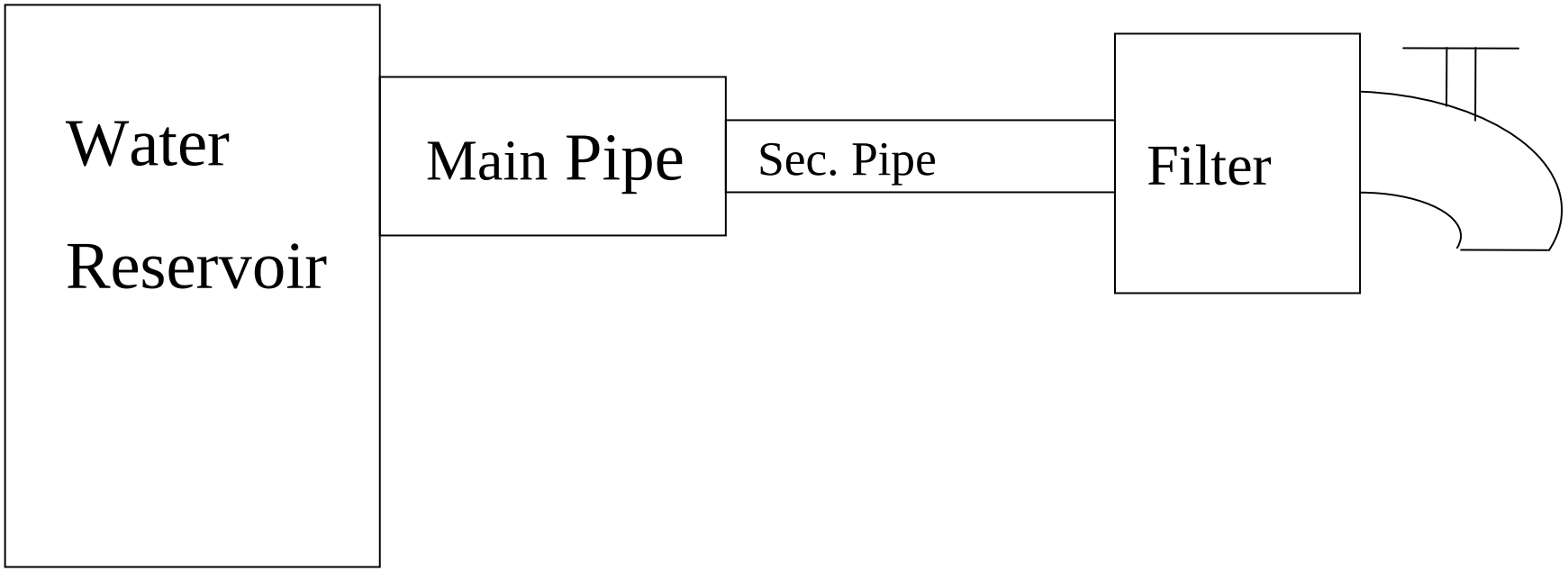
On peut les étendre par héritage, mais alors il faut choisir d'étendre tel ou tel des Streams ou bien de rajouter une classe pour chacun.

Java utilise les filtres

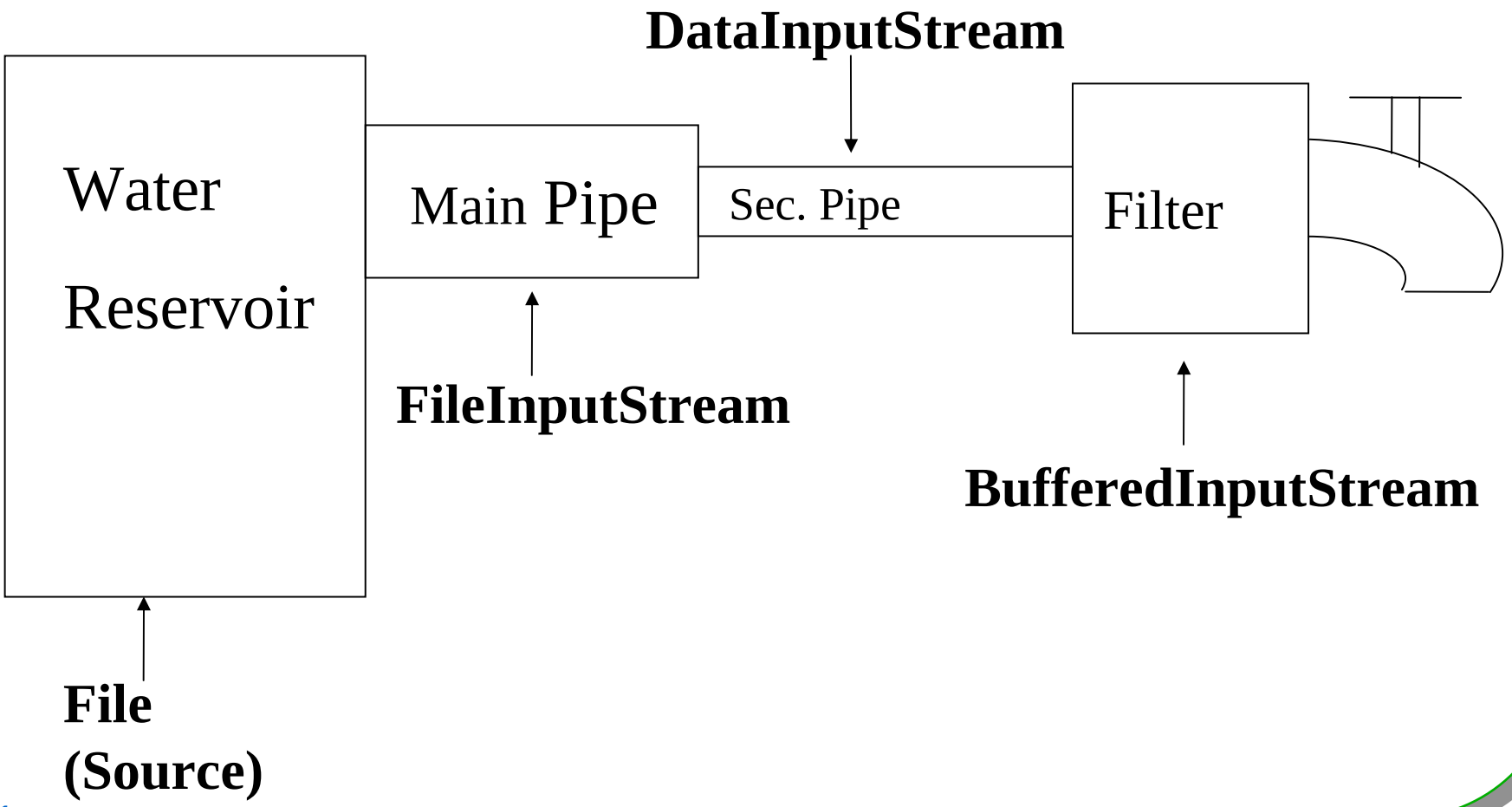
Les filtres i.e. le patron décorateur



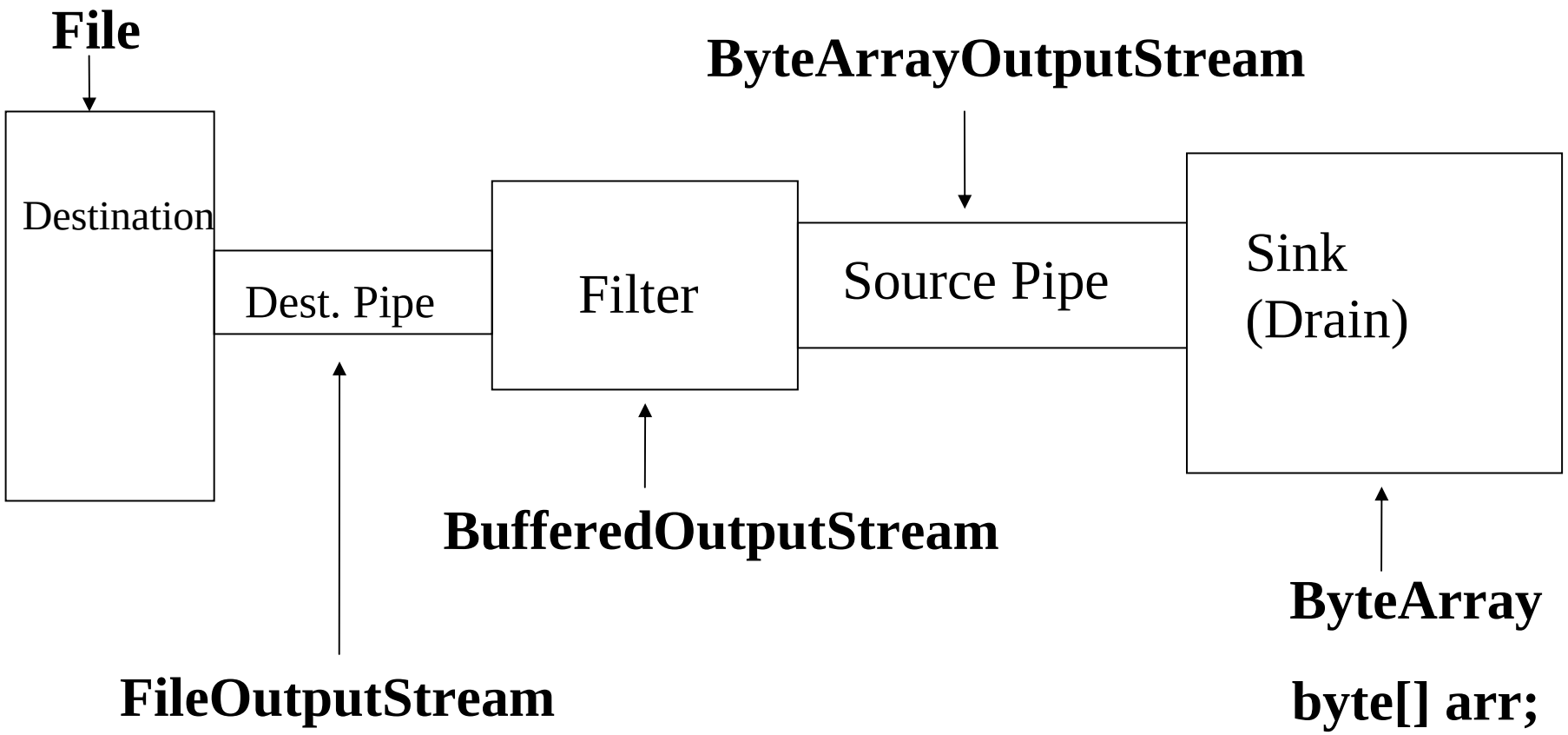
Plomberie de base !



Lecture au fil de l'eau

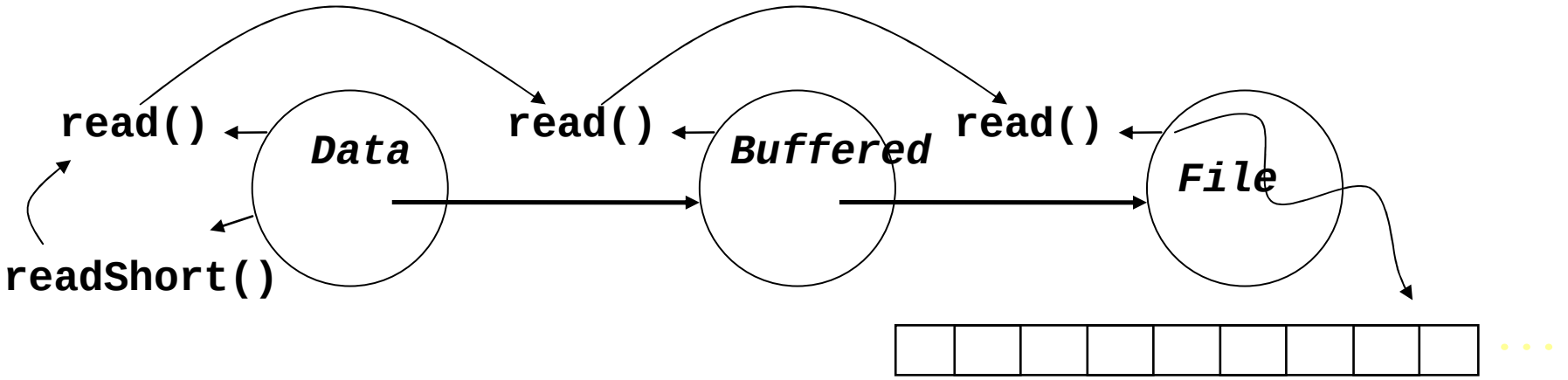


Dans l'autre sens



Chaîner les Streams

```
try {
   DataInputStream input = new DataInputStream(
      new BufferedInputStream(
         new FileInputStream(args[0])));
} catch (FileNotFoundException fnfe) {
   // ...
}
```



Pattern Decorateur

- **idée:**
  - **Toutes les classes implémentent les mêmes fonctions de base**
  - **Chaque décorateur ajoute une fonctionnalité supplémentaire (“decorate”)**

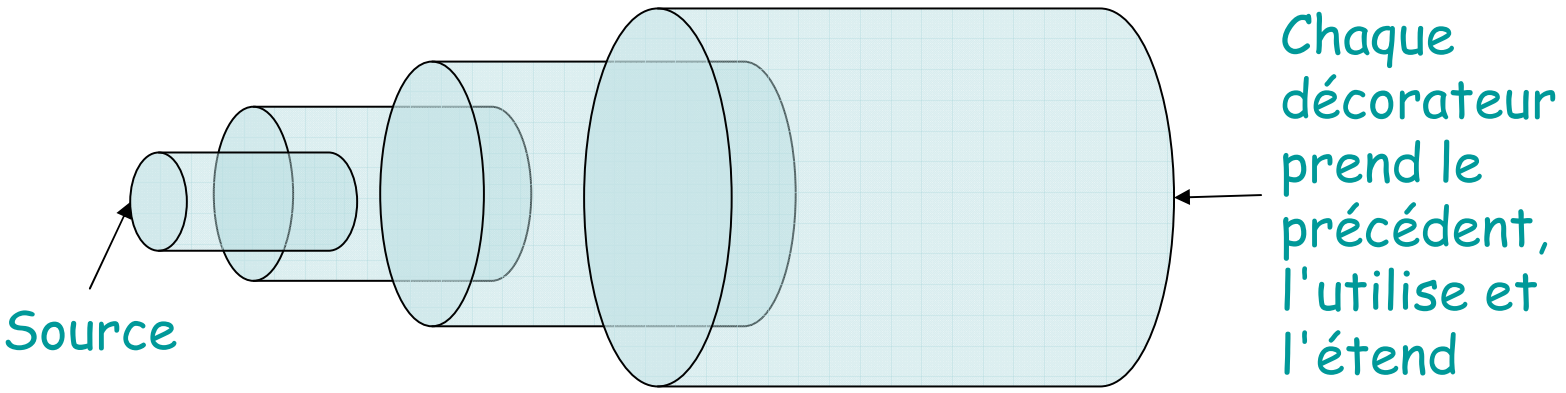
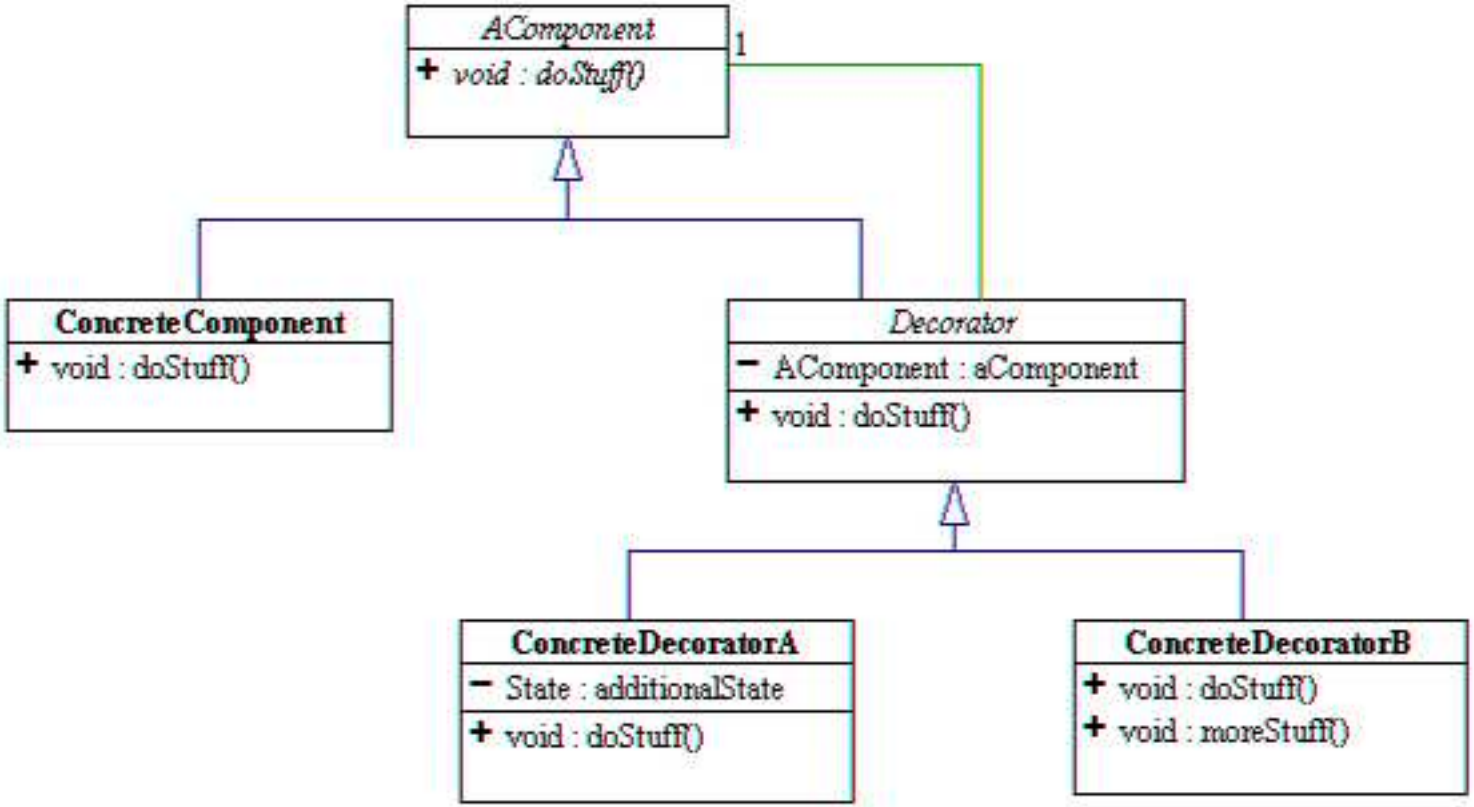
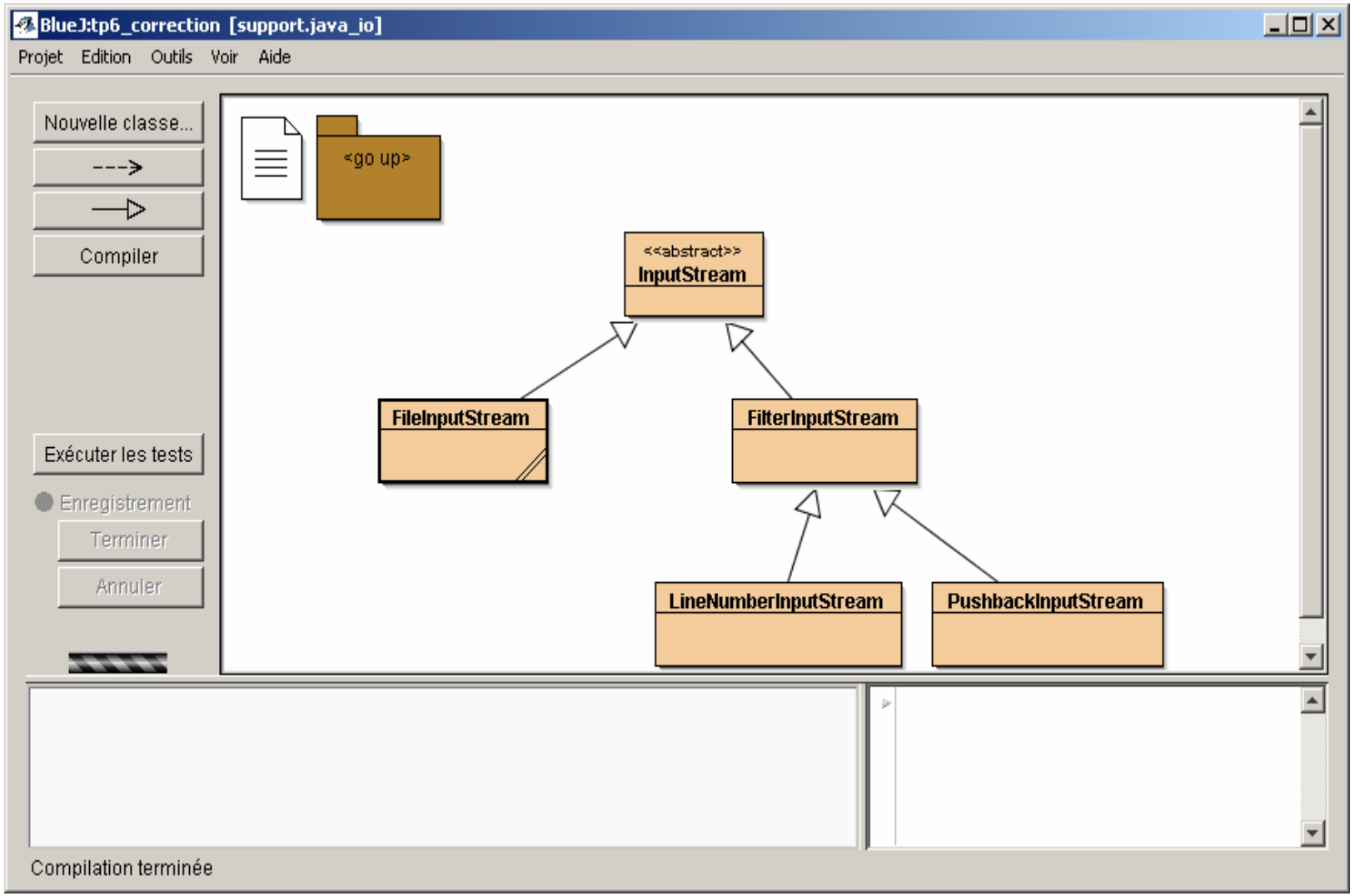


diagramme Decorateur



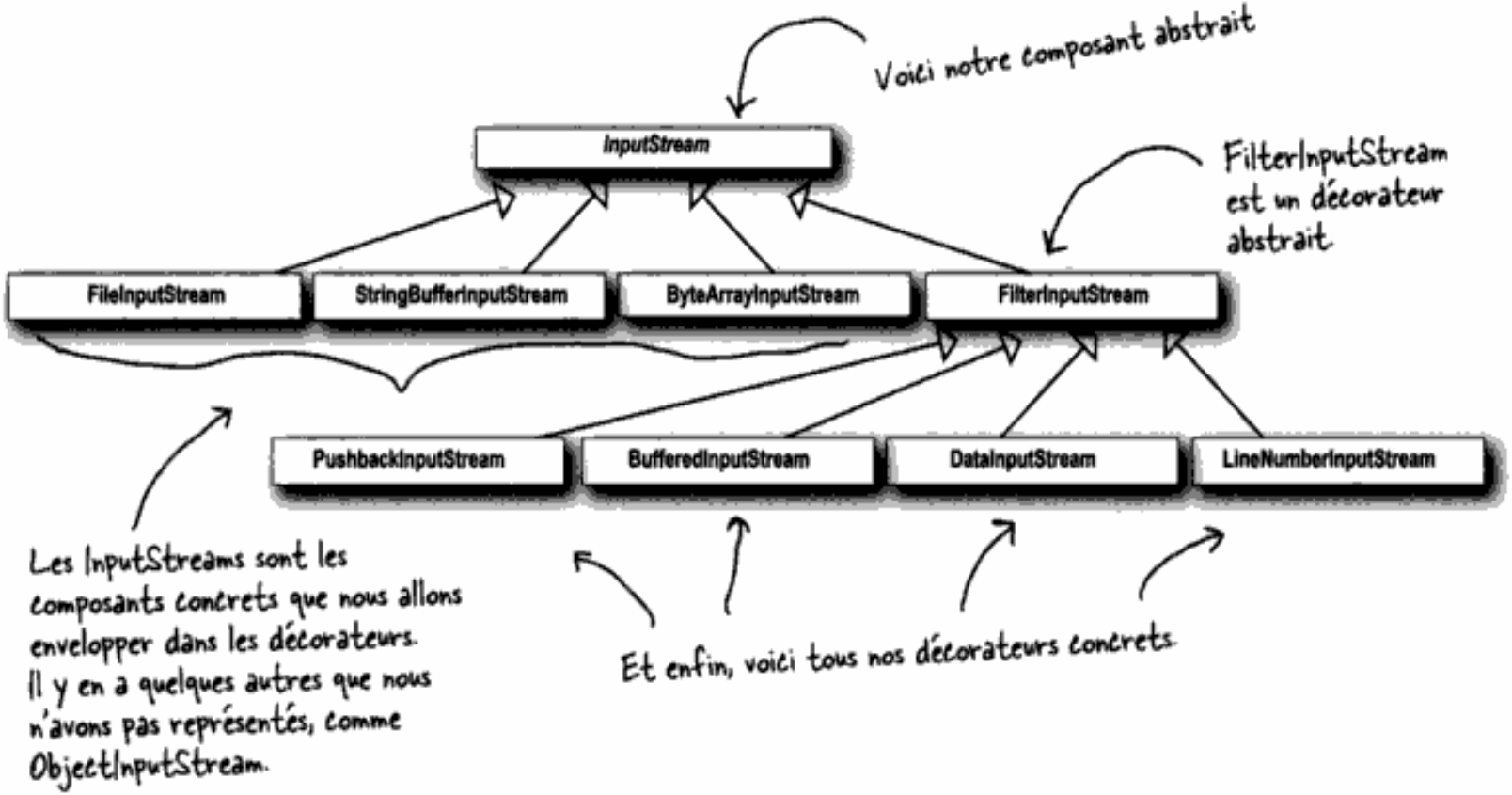
java.io



- **Le Décorateur est bien là**

Extrait de java : tête la première

Décoration des classes de java.io



Utilisation

```
InputStream is = new LineNumberInputStream(
                        new FileInputStream(
                            new File("LectureDeFichierTest.java")));
```

Reader, récidive

```
LineNumberReader r =
        new LineNumberReader(
            new FileReader(new File("README.TXT")));
```

- `System.out.println(r.readLine());`
- `System.out.println(r.readLine());`

java.net.Socket

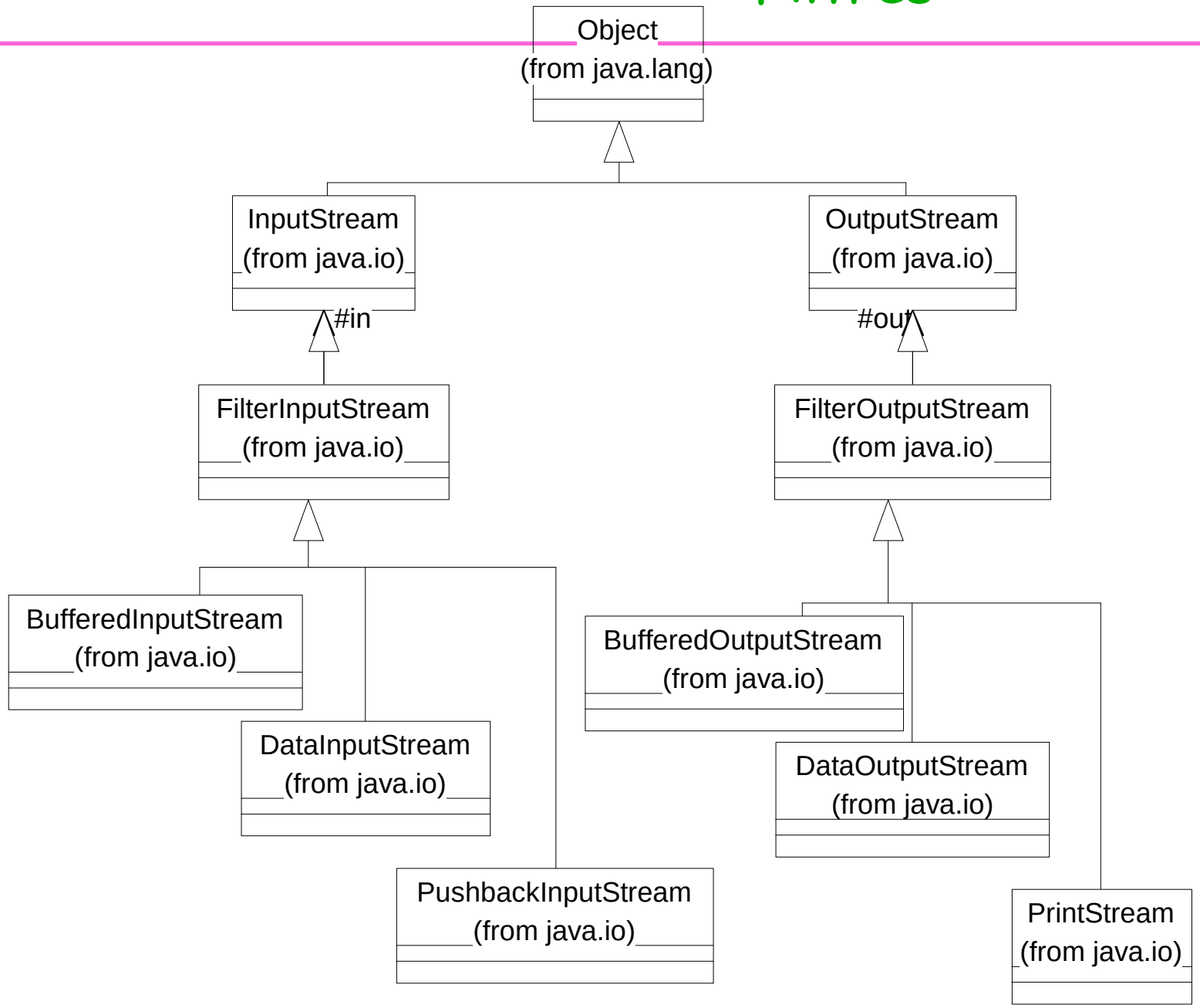
```
Socket socket = new Socket("vivaldi.cnam.fr", 5000);


 BufferedReader in =
   new BufferedReader(
      new InputStreamReader(socket.getInputStream()));


    System.out.println(in.readLine());
```

I/O decorators

- La fonction de base est la lecture/écriture d'octets
- Sources sont les périphériques ou d'autres sources comme :
  - Files
  - Sockets
  - Byte arrays
  - Pipes
  - Strings

- Les décorations sont:
  - Reading/writing primitive (int, char, float…)
  - Reading/writing objects
  - Compressing / decompressing
  - Buffering
    - Reading lines of text
  - Printing
    - Writing lines
  - Filtering

Filtres

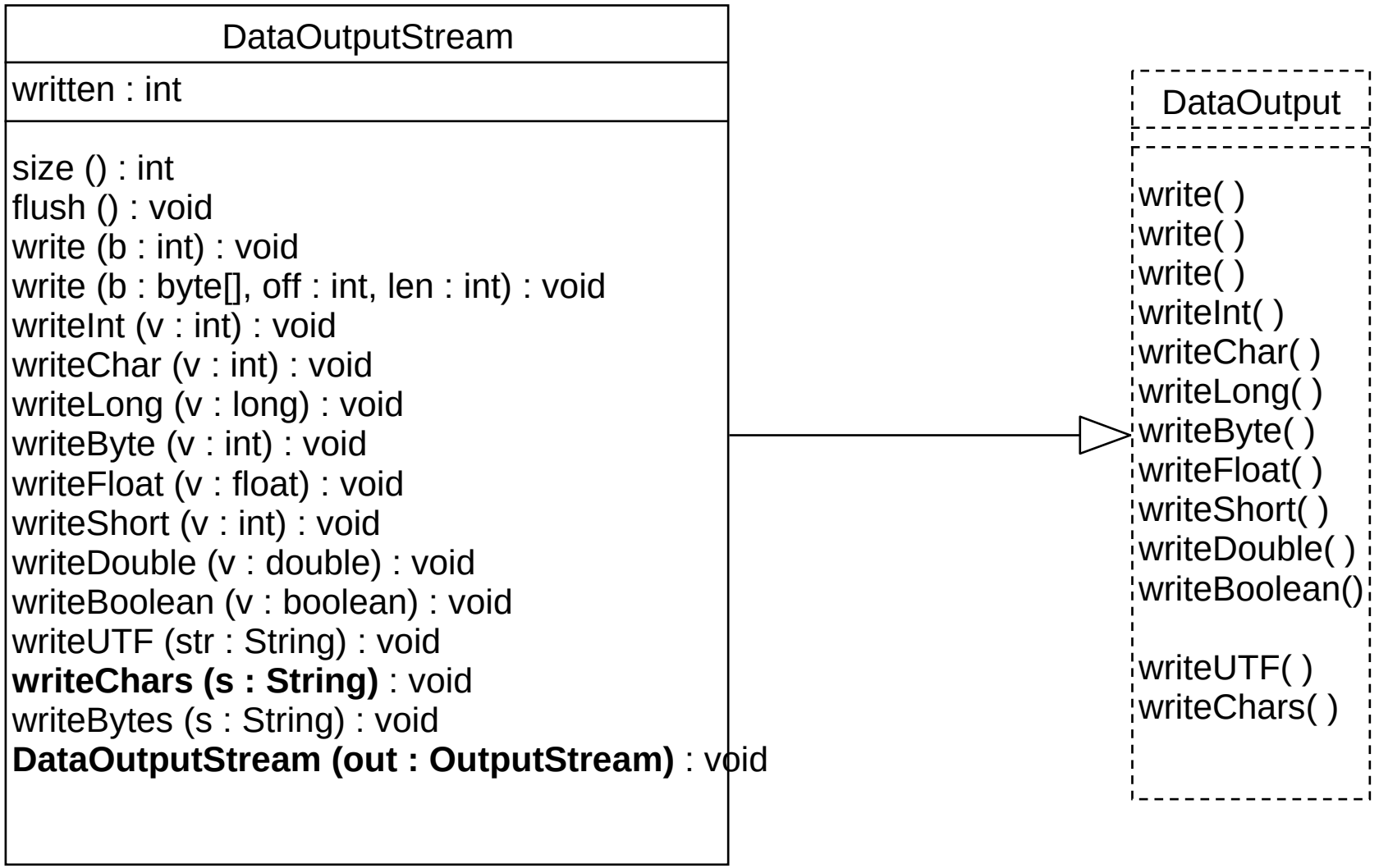

Object
(from java.lang)
InputStream
(from java.io)
#in
OutputStream
(from java.io)
#out
FilterInputStream
(from java.io)
FilterOutputStream
(from java.io)
BufferedInputStream
(from java.io)
DataInputStream
(from java.io)
PushbackInputStream
(from java.io)
BufferedOutputStream
(from java.io)
DataOutputStream
(from java.io)
PrintStream
(from java.io)

Buffered(In/Out)putStream

BufferedInputStream
protected buf : byte[] protected pos : int protected count : int protected markpos : int protected marklimit : int
read () : int read (b : byte[], off : int, len : int) : int private fill () : void mark (readlimit : int) : void skip (n : long) : long reset () : void available () : int markSupported () : boolean **BufferedInputStream (in : InputStream)** : void **BufferedInputStream (in : InputStream, size : int)** :

BufferedOutputStream
protected buf : byte[] protected count : int
flush () : void write (b : int) : void write (b : byte[], off : int, len : int) : void flushBuffer () : void BufferedOutputStream (out : OutputStream) : void BufferedOutputStream (out : OutputStream, size : int) : void

DataOutputStream

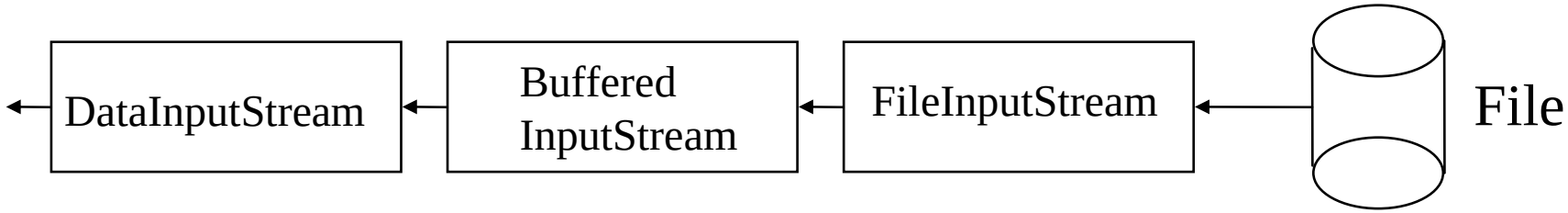


```
DataOutputStream dos =
   new DataOutputStream(
      new BufferedOutputStream(
         new FileOutputStream("fichier")));
dos.writeDouble(12.5);
dos.writeUTF("Dupond");
dos.writeInt(1254);
dos.close();
```

PushbackInputStream

PushbackInputStream
protected buf : byte[] protected pos : int
read () : int read (b : byte[], off : int, len : int) : int **unread (b : int)** : void unread (b : byte[]) : void unread (b : byte[], off : int, len : int) : void available () : int markSupported () : boolean PushbackInputStream (in : InputStream) : void PushbackInputStream (in : InputStream, size : int) : void

Déclaration des filtres




```
DataInputStream f = new DataInputStream(new BufferedInputStream(new FileInputStream(NomFich)));
```

String s = f.readLine(); Deprecated cf BufferedReader !!!

Exemple

Sélectionner un fichier texte le lire ligne à ligne et mettre celles ci à la fin d'un autre fichier.

Corrigé

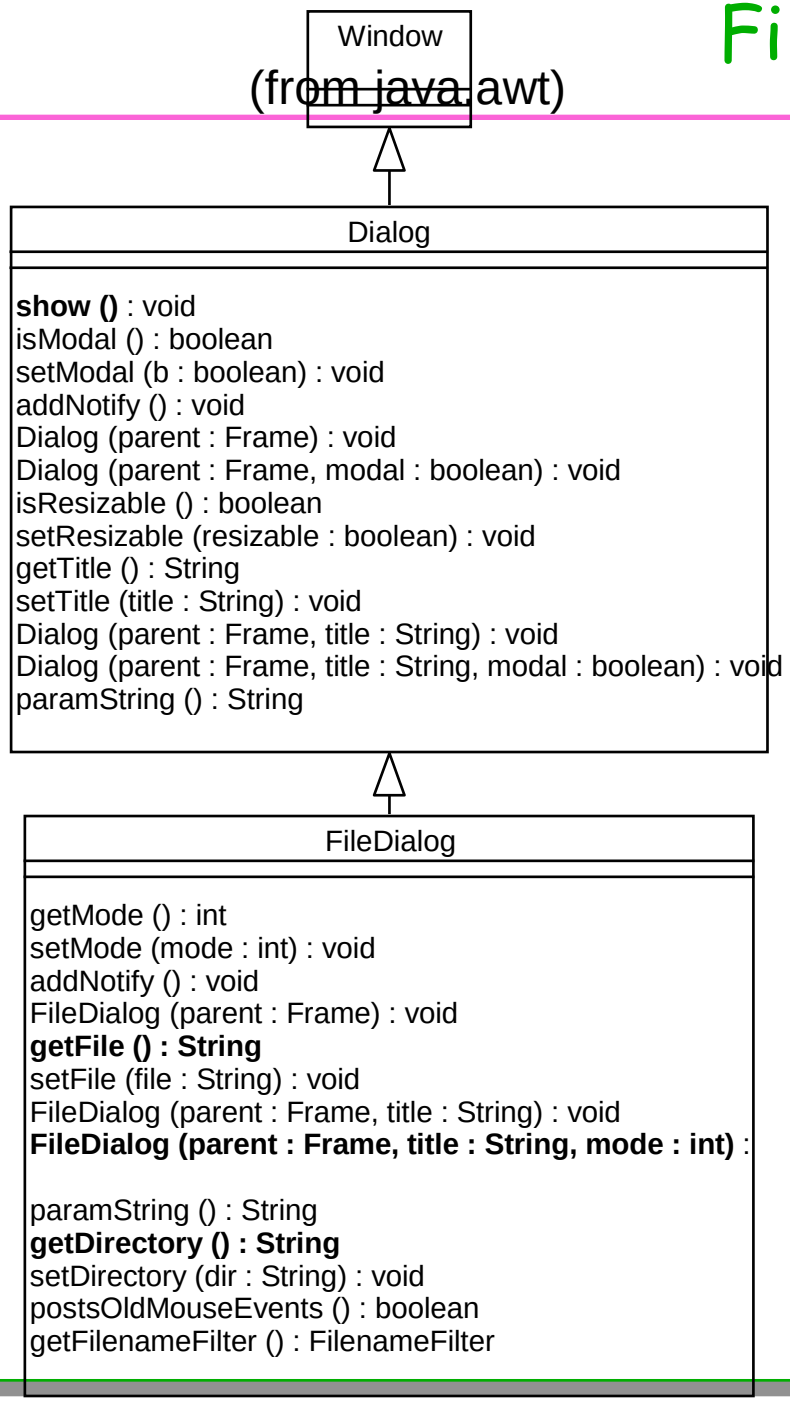
```
import java.io.*;
import java.awt.*;

public class Random extends Frame{
    public static void main(String args[]) throws IOException{
            new Random().run(args); }
    public void run(String args[])throws IOException{
            FileDialog fd= new FileDialog(this);
            fd.setMode(FileDialog.LOAD); fd.setVisible(true);
            String nom=fd.getDirectory() + "/" + fd.getFile();
               BufferedReader iFile = new BufferedReader(
                        new InputStreamReader(new FileInputStream(nom)));
            RandomAccessFile oFile = new RandomAccessFile(args[0],"rw");
            oFile.seek(oFile.length());
            String ligne;
            while((ligne=iFile.readLine())!=null){
               oFile.writeBytes(ligne);    oFile.writeBytes("\r\n");
            }
            oFile.close();
    }}
```

Autre filtre : Zip

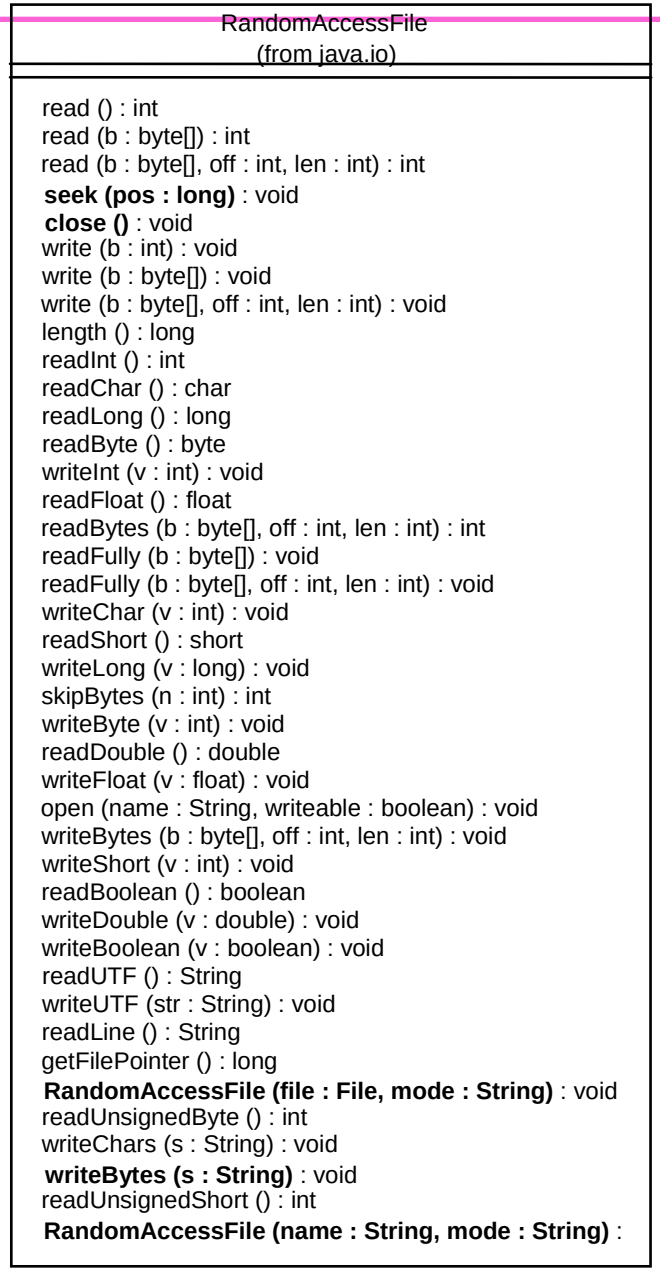
java.util.zip.ZipInputStream **pour lire des fichier “zippés” (inverse : j**ava.util.zip.**ZipOutputStream)**

```
ZipInputStream zin = new ZipInputStream
            (new FileInputStream(“names.zip”));
ZipEntry ze = zin.getNextEntry();
if (ze.getNextEntry() != null) {
     DataInputStream dzin = new DataInputStream(zin);
     float s = dzin.readFloat();
}
```

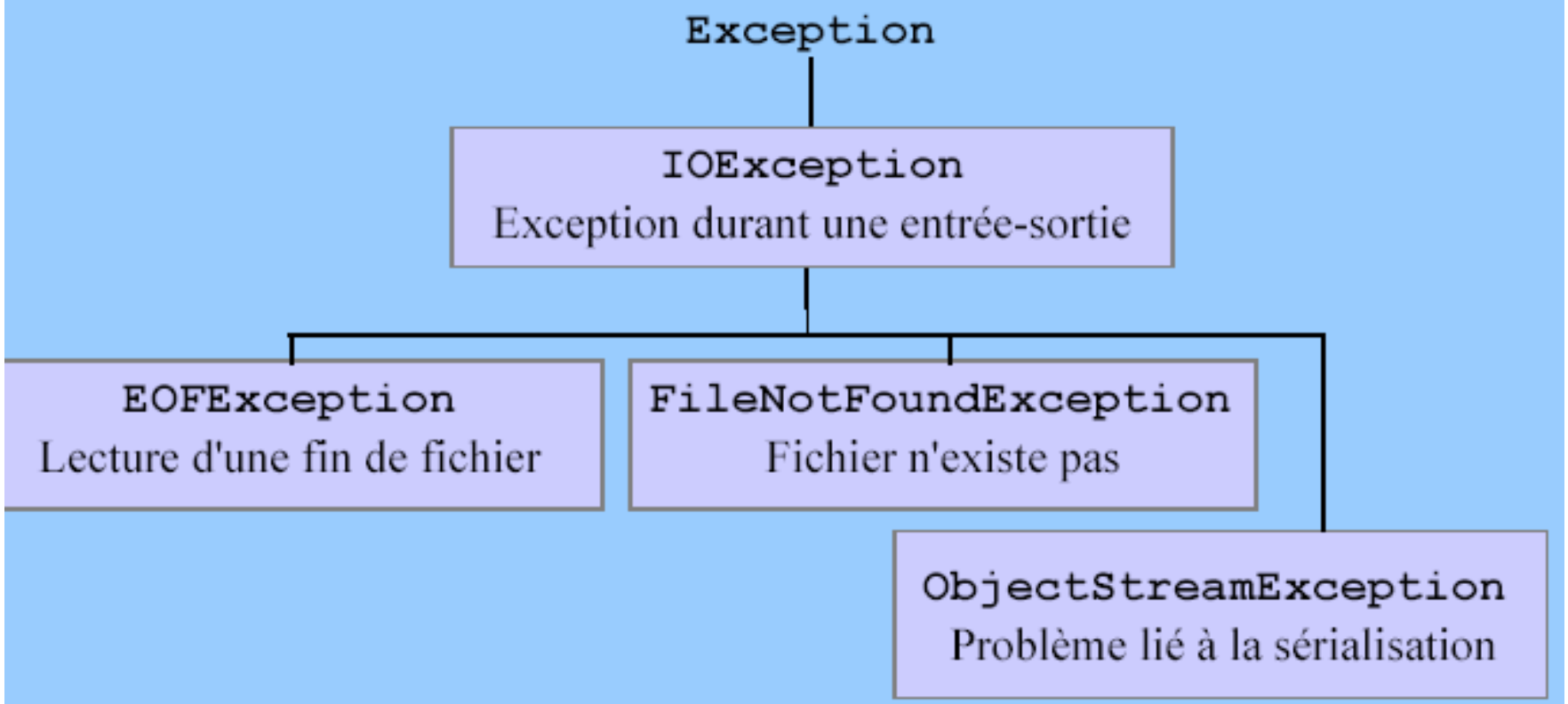
FileDialog


Window
(from java.awt)
Dialog
show () : void
isModal () : boolean
setModal (b : boolean) : void
addNotify () : void
Dialog (parent : Frame) : void
Dialog (parent : Frame, modal : boolean) : void
isResizable () : boolean
setResizable (resizable : boolean) : void
getTitle () : String
setTitle (title : String) : void
Dialog (parent : Frame, title : String) : void
Dialog (parent : Frame, title : String, modal : boolean) : void
paramString () : String
FileDialog
getMode () : int
setMode (mode : int) : void
addNotify () : void
FileDialog (parent : Frame) : void
getFile () : String
setFile (file : String) : void
FileDialog (parent : Frame, title : String) : void
FileDialog (parent : Frame, title : String, mode : int) :
paramString () : String
getDirectory () : String
setDirectory (dir : String) : void
postsOldMouseEvents () : boolean
getFilenameFilter () : FilenameFilter

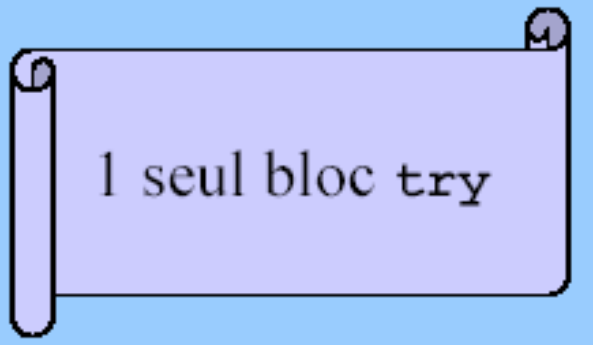
RandomAccessFile



Principales exceptions liées aux entrées-sorties



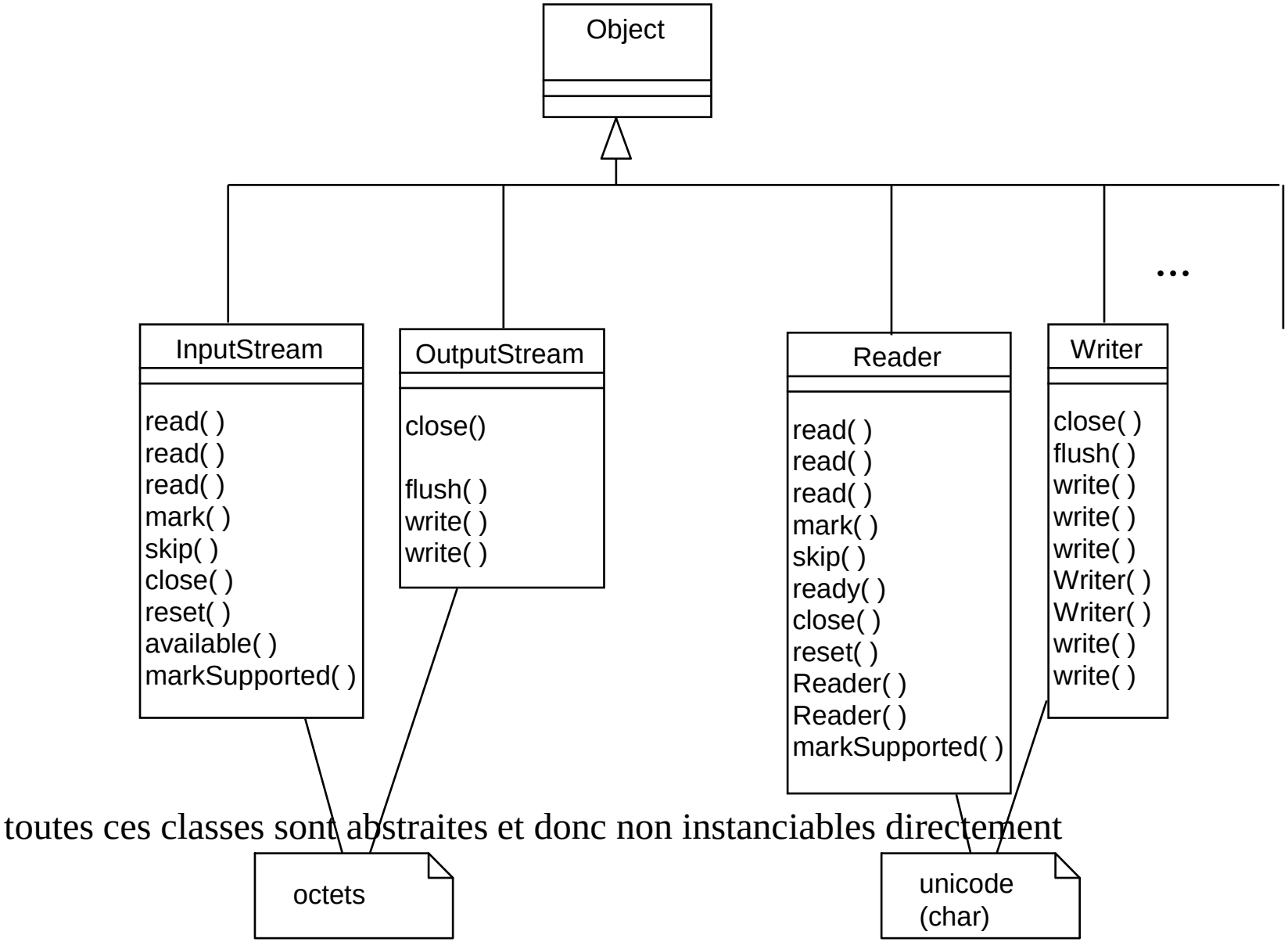
```
DataInputStream dis;
try {
   dis =
     new DataInputStream(new FileInputStream("fichier"));
   while (true) {
     double d = dis.readDouble();
     . . .
   }
}
catch(FileNotFoundException e) { . . }
catch(EOFException e) {}
catch(IOException e) { . . . }
finally {
   if (dis != null)
     try {dis.close();} catch (IOException) {…}
}
```



Texte *vs* données binaires

- **On distingue souvent les données textuelles des autres types de données**
- **Les sources sont dites 'fichiers texte' ou 'fichiers binaires'**
- **Les fichiers binaires utilisent des formats différents et sont affichés par des plugins spécifiques.**
- **La structure des fichiers texte est plus simple. Elle utilise cependant des encodages différents.**

Binaire *vs* caractère

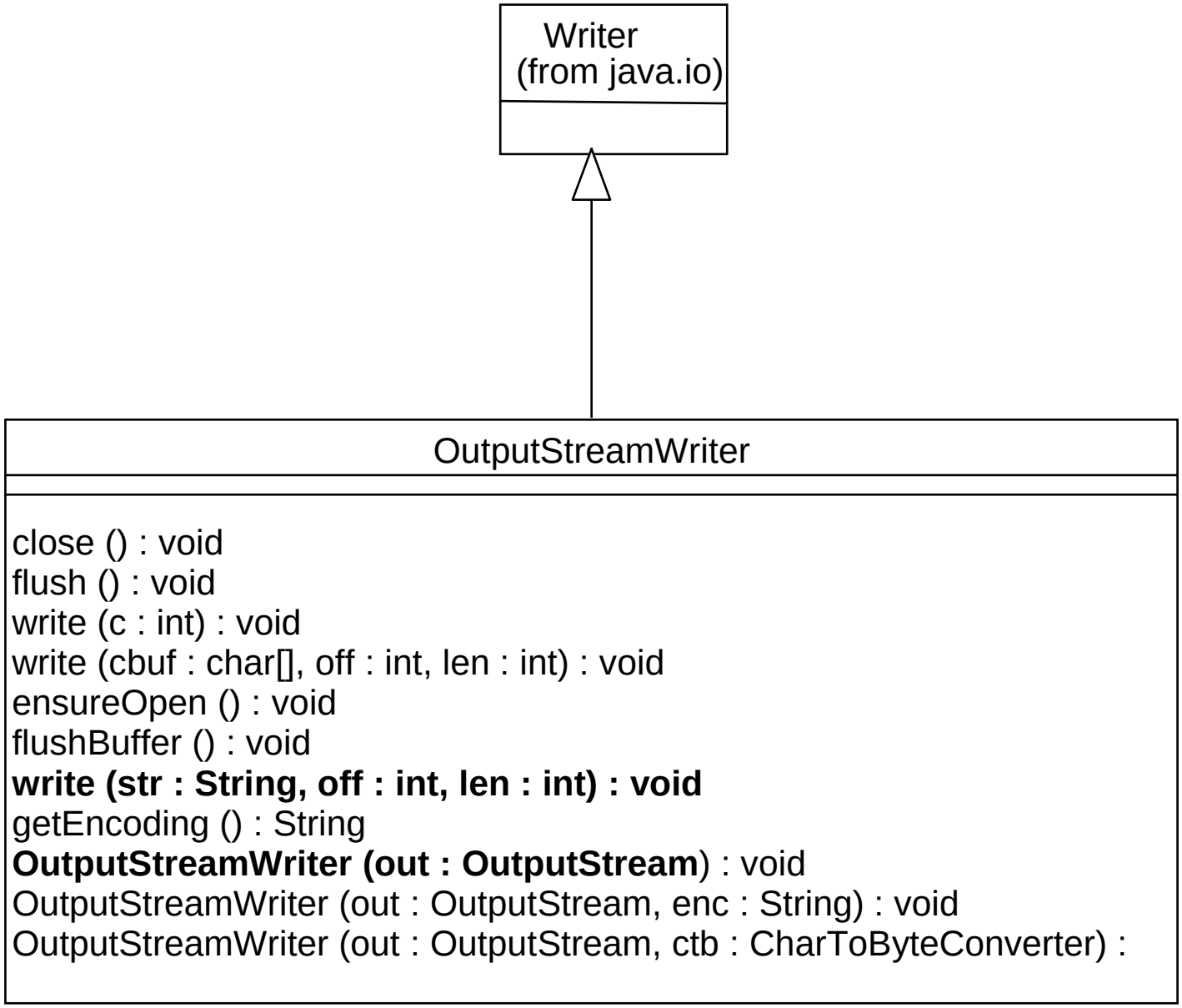


toutes ces classes sont abstraites et donc non instanciables directement

Octets et Caractères

OutputStream	Writer
InputStream	Reader
FileOutputStream	FileWriter
FileInputStream	FileReader
ByteArrayOutputStream	CharArrayWriter
ByteArrayInputStream	CharArrayReader
	StringWriter
	StringReader
BufferedOutputStream	BufferedWriter
BufferedInputStream	BufferedReader
	PrintWriter
DataOutputStream	
DataInputStream	
	OutputStreamWriter
	OutputStreamReader
ObjectInputStream	
ObjectOutputStream	

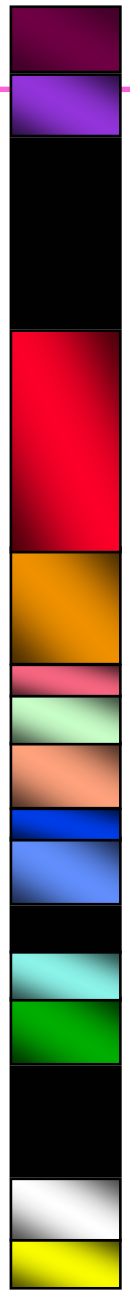
Reader et InputStream Writer et OutputStream



Encodage des caractères

US-ASCII	!	33
IBM-EBCDIC	!	90
Iso Latin1	é	232
UTF8	é	195 168

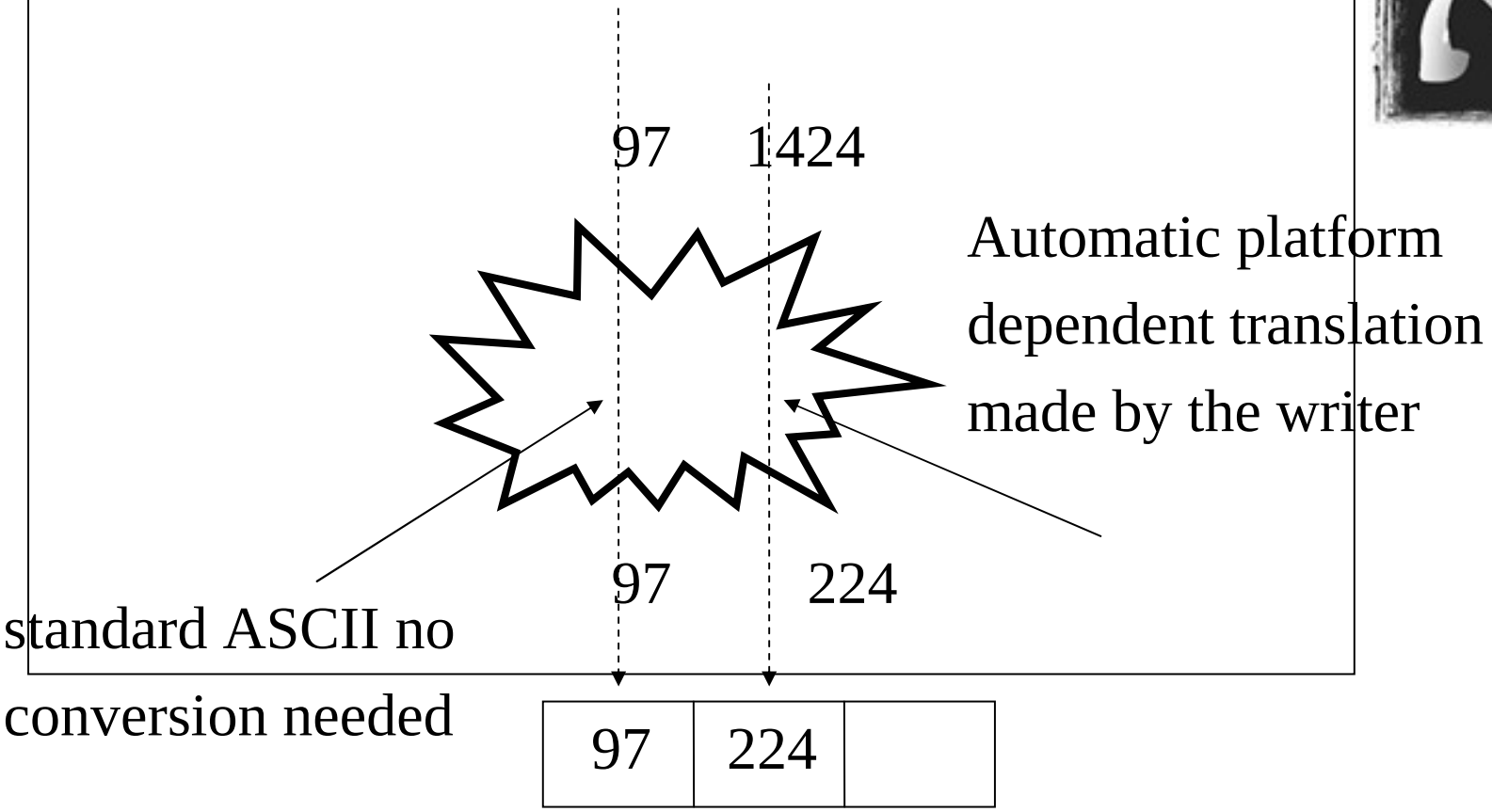
Unicode™ / ISO 10646



- 16-bit international character encoding
- Windows 2000 uses Unicode version 2.0

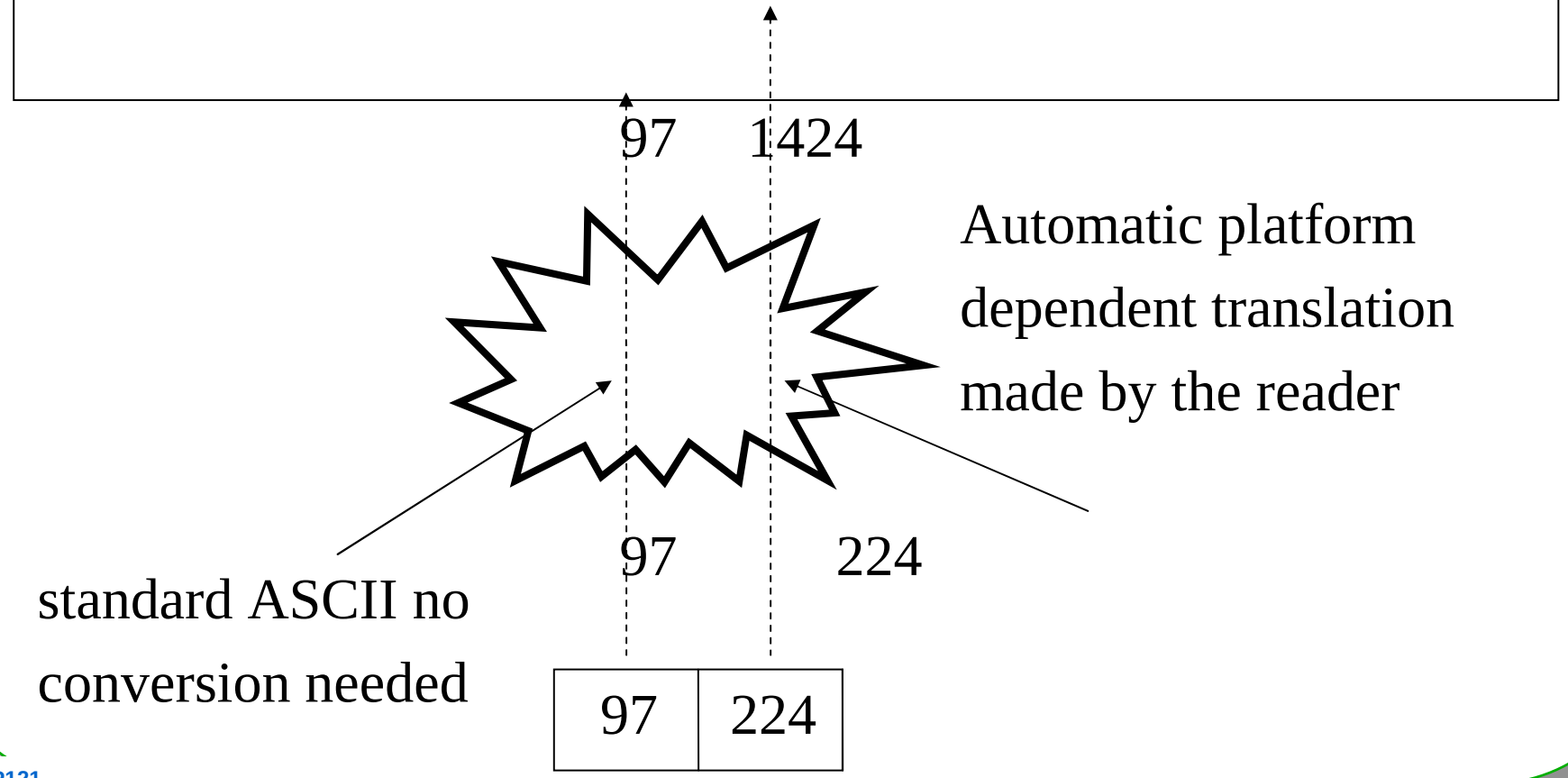
Readers & Writers

```
Writer writer = new FileWriter("mail.txt");
writer.write('a');
writer.write('\u0590'); // Aleph
```



Readers & Writers

```
Reader reader = new FileReader("mail.txt");
char c = reader.read();        // c = 'a'
c = reader.read();             // c = '\u0590'
```



Conversion de codage

```
public class Convert {
  public static void main (String[] args) throws IOException {
    if (args.length != 4) throw new IllegalArgumentException
                ("Convert <srcEnc> <source> <dstEnc> <dest>");
    FileInputStream fileIn = new FileInputStream (args[1]);
    FileOutputStream fileOut = new FileOutputStream (args[3]);
    InputStreamReader inputStreamReader= new InputStreamReader (fileIn, args[0]);
    OutputStreamWriter outputStreamWriter=new OutputStreamWriter(fileOut,args[2]);
    char[] buffer = new char[16];
    int numberRead;
    while ((numberRead = inputStreamReader.read (buffer)) > -1)
          outputStreamWriter.write (buffer, 0, numberRead);
    outputStreamWriter.close ();
    inputStreamReader.close ();
  }
}
>java Convert  UTF8 fichier.un latin1 fichier.deux
```

Class java.io.Reader

```
public abstract int read(char[] buffer,
    int offset, int length) throws IOException
```

lit ‘length’ caractères dans le ‘buffer’ en démarrant depuis ‘offset’, retourne le nombre de caractères lus.

```
public void close() throws IOException
```

- **D’autres méthodes c.f. la javadoc**

Class java.io.Writer

```
public void write(int c) throws IOException
```

Ecriture d'un caractère, un entier.

```
public void write (char[] buffer)
    throws IOException
```

un tableau de caractères.

Example: Reader Writer

```
import java.io.*;

// This class reads a text file and writes it into
// another text file after converting all letters to
// uppercase.
// Usage: java ToUpper <source> <target>
class ToUpper {

    public static void main(String[] args) {
        if (args.length!=2) {
            System.err.println(“Invalid usage.”);
            return;
        }
        String sourceName = args[0];
        String targetName = args[1];
```

Example (cont.)

```
try {
        Reader reader = new FileReader(sourceName);
        Writer writer = new FileWriter(targetName);
        int c;

        while ((c=reader.read())!=-1) {
            c = Character.toUpperCase((char)c);
            writer.write(c);
        }
       reader.close();         writer.close();
    } catch (IOException ioe) {
        System.err.println("Copying failed.");
    }
}
```

Résumé:
Data Processing Streams

Process	Character Streams	Byte Streams
Buffering	BufferedReader BufferedWriter	BufferedInputStream BufferedOutputStream
Filtering	FilterReader, FilterWriter	FilterInputStream, FilterOutputStream
Converting between bytes and characters	InputStreamReader OutputStreamReader	
Concatenation		SequenceInputStream
Object Serialization		ObjectInputStream, ObjectOutputStream
Data Conversion		DataInputStream, DataOutputStream
Counting	LineNumberReader	LineNumberInputStream
Peeking Ahead	PushbackReader	PushbackInputStream
Printing	PrintWriter	PrintStream

Adéquation Objet / Fichier, base de données

- **Une instance de classe persistante ?**
  - **Sérialisation ? tout java**
  - **Fichier au format CSV ?**
    - **Une ligne : une instance**
  - **Base de données ? SQL, Relationnelle,**
    - **une ligne : une instance**

Un objet persistant

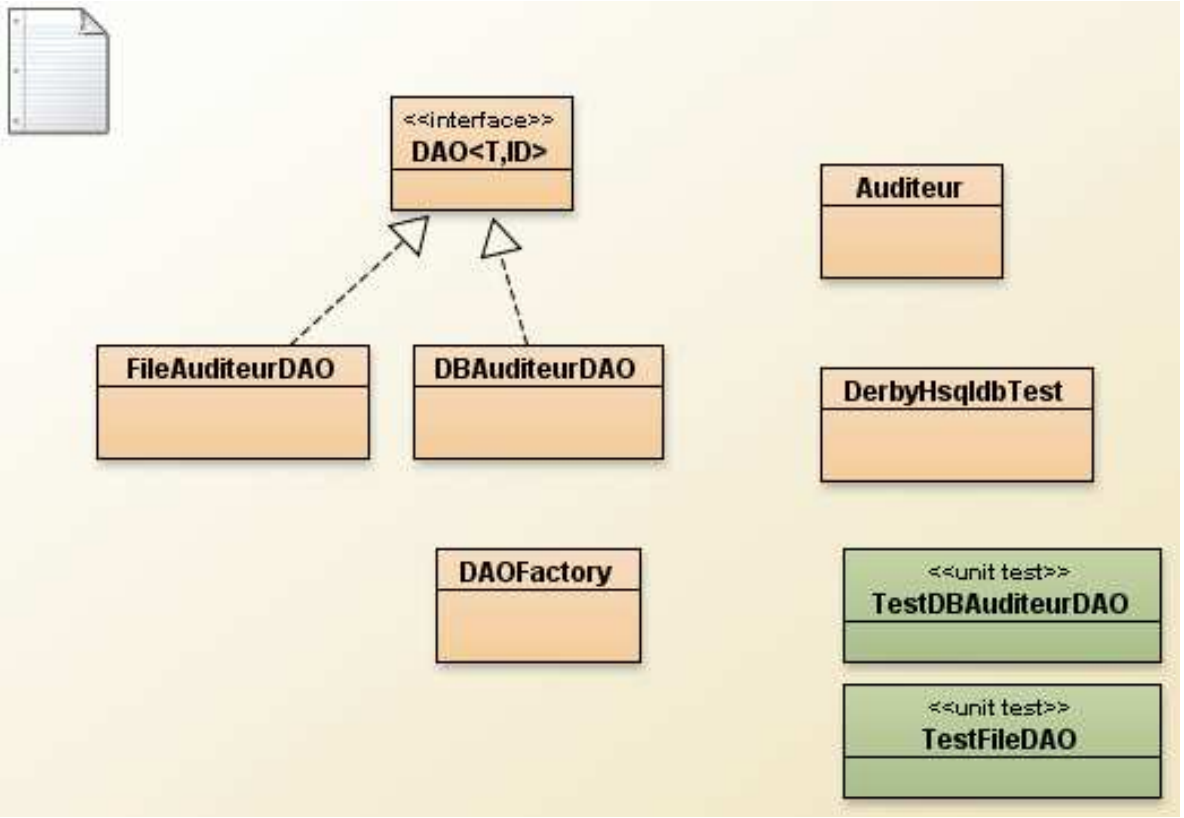
- **Comment le rendre persistant ?**
  - **Tout en assurant un couplage faible**
    - **Comment s'affranchir du support, du format**
  - **Le patron DAO**
    - **Data Access Object**
    - **Propose les 4 opérations CRUD**
      - **Create Retrieve Update Delete**
  - http://best-practice-software-engineering.ifs.tuwien.ac.at/patterns/dao.html

DAO<T, ID> / CRUD

```
/**
 *  T   pour Le type souhaité
 *  ID  pour l'identifiant
 *  CRUD pour create/retrieve/update/delete
 *       findAll la liste de tous les éléments
*/
public interface DAO<T,ID>{

  public void create(T t) throws Exception;

  public T retrieve(ID id) throws Exception;

  public void update(T t) throws Exception;

  public void delete(ID id) throws Exception;

  public List<T> findAll() throws Exception;

}
```

http://best-practice-software-engineering.ifs.tuwien.ac.at/patterns/dao.html

DAO + DAOFactory, exemple



- **Auditeur**
  - La classe métier
- **DAOFactory**
  - L'utilisateur sélectionne le DAO à l'exécution, en fichier ou en base
- **FileAuditeurDAO, DBAuditeurDAO**
  - En fichier ou en base de données (ici derby ou hsqldb)

Un usage du dao (Fichier comme BD)

```
public void testCreate()throws Exception{
    try{
        for(int i=0;i<=100;i=i+5){
            Auditeur a = new Auditeur("nom"+i,"prenom"+i);
            int id = dao.create(a);
        }
    }catch(Exception e){
        fail(" Exception ! ");
    }
}

public void testUpdate()throws java.lang.Exception{
    for(Auditeur auditeur : dao.findAll()){
        auditeur.setNom(auditeur.getNom()+"xx");
        dao.update(auditeur);
    }
    for(Auditeur auditeur : dao.findAll()){
        assertTrue(auditeur.getNom().endsWith("xx"));
    }
}
```

- **Abstraction réussie !**

FileAuditeurDAO

- **Persistance des auditeurs**
  - **Hypothèse**
    - **Chaque auditeur : un fichier**
    - **Ce fichier doit être unique**
      - **Un identifiant, un nombre comme discriminant**
    - **Dans un répertoire dédié**
  - **DAO/CRUD**
    - **Create : ajout d'un nouveau fichier**
    - **Update : mis à jour du contenu de ce fichier**
    - **Retrieve : recherche du fichier, en retour un auditeur**
    - **Delete : suppression du fichier**

FileAuditeurDAO, Create

```java
public Integer create(Auditeur a) throws Exception{
  int id = getAndIncId();
  a.setId(id);
  String fileName = basePath + id + "_" + a.getNom() + "_" + a.getPrenom() + ".txt";
  PrintWriter pw = new PrintWriter(new FileWriter(fileName));
  pw.write(a.toString()+"\n");
  pw.flush();
  pw.close();
  return id;
}
```

- **Ici, un fichier pour chaque auditeur**
  - **Dans un répertoire dédié (un choix d'implémenteur)**

  - **Chaque fichier est numéroté : l'identifiant (id) de l'auditeur**

FileAuditeurDAO, Retrieve, update

```
public Auditeur retrieve(Integer id) throws Exception{
  BufferedReader br = new BufferedReader(new FileReader(getFile(id)));
  Auditeur auditeur = Auditeur.parseAuditeur(br.readLine());
  br.close();
  return auditeur;
 }

public void update(Auditeur a) throws Exception{
  PrintWriter pw = new PrintWriter(getFile(a.getId()));
  pw.write(a.toString()+"\n");
  pw.flush();
  pw.close();
 }
```

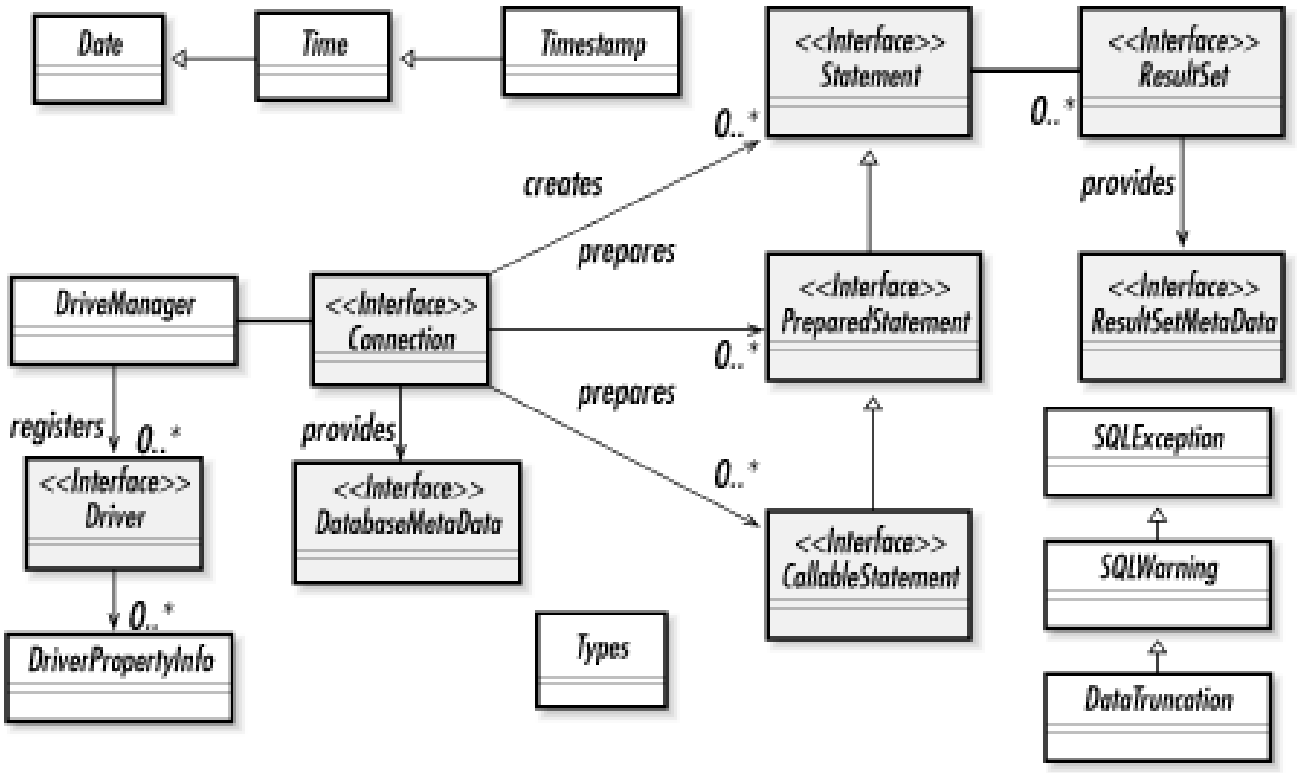
- **Opérations sur fichier**
  - **De la classe Auditeur**
    - **String toString()**
    - **Auditeur Auditeur.parseAuditeur(String a)**

FileAuditeurDAO, Delete, FindAll

```
public void delete(Integer id) throws Exception{
  getFile(id).delete();
}

public List<Auditeur> findAll() throws Exception{
  List<Auditeur> list = new ArrayList<Auditeur>();
  File[] files = new File(basePath).listFiles();
  for(File file : files){
    try{
      if(file.getName().endsWith(".txt")){
        BufferedReader br = new BufferedReader(new FileReader(file));
        list.add(Auditeur.parseAuditeur(br.readLine()));
        br.close();
      }
    }catch(Exception e){
    }
  }
  return list;
 }
```

- **La liste des fichiers de vient la liste des auditeurs**

JDBC Intro



- http://java.boot.by/ocpjp7-upgrade/ch02s05.html
- **Explication**

Mise en oeuvre

- **Chargement du pilote et obtention d'une connexion**

```
//private static String dbURL = "jdbc:derby://localhost:1527/myDB;create=true;user=me
private static String derbyURL = "jdbc:derby:auditeurs_derby;create=true";
private static String hsqldbURL = "jdbc:hsqldb:file:auditeurs_hsqldb;shutdown=true";

public static void derbyTest() throws Exception{
    Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
    Connection conn = DriverManager.getConnection(derbyURL);

public static void hsqldbTest() throws Exception{

    Class.forName("org.hsqldb.jdbc.JDBCDriver");
    Connection conn = DriverManager.getConnection(hsqldbURL);
```

Statement, Création de la table

- **Statement**
  - **executeUpdate(** *Commande SQL* **)**

```
private static void createTable(Connection conn) throws Exception{
     Statement stmt = conn.createStatement();
     String sql = "CREATE TABLE AUDITEURS " +
"(id INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY (START WITH 1,INCREMENT BY 1), " +
                  "nom VARCHAR(255) not NULL, " +
                  "prenom VARCHAR(255), " +
                  "PRIMARY KEY ( id ))";

if(stmt != null){
      stmt.executeUpdate(sql);
      stmt.close();
    }
```

Nb : `"(id INTEGER NOT NULL GENERATED BY DEFAULT AS IDENTITY (START WITH 1,INCREMENT BY 1), "`
comme `id INTEGER NOT NULL AUTOINCREMENT`

Statement,Lecture de la table

- **Statement**
  - **ResultSet rs = executeQuery(*Commande SQL*)**

```
Statement stmt = null;
     String query = "SELECT * from AUDITEURS";

     try {
         stmt = con.createStatement();
         ResultSet rs = stmt.executeQuery(query);
         while (rs.next()) {
             String nom = rs.getString("NOM");
             String prenom = rs.getString("PRENOM");
             System.out.println(nom + "\t" + prenom);
         }
     } finally {
         if (stmt != null) { stmt.close(); }
     }
```

DBAuditeurDAO, Preparation…

```
public class DBAuditeurDAO implements DAO<Auditeur,Integer>{
  private PreparedStatement createStmt;
  private PreparedStatement retrieveStmt;
  private PreparedStatement updateStmt;
  private PreparedStatement deleteStmt;
  private PreparedStatement findAllStmt;
  private PreparedStatement lastIdStmt;

  public DBAuditeurDAO(String url) throws Exception{
    Connection conn = DriverManager.getConnection(url);
    prepareStatement(conn);
  }

  public DBAuditeurDAO(String url, String user, String pw) throws Exception {
    Connection conn = DriverManager.getConnection(url, user, pw);
    prepareStatement(conn);
  }
```

- **PrepareStatement**
  - **Une commande SQL à trous**

DBAuditeurDAO, Preparation suite

```
private void prepareStatement(Connection conn)throws Exception{
  createStmt = conn.prepareStatement("INSERT INTO AUDITEURS (nom, prenom) VALUES (?, ?)");
  retrieveStmt = conn.prepareStatement("SELECT id, nom, prenom FROM AUDITEURS WHERE id = ?");
  updateStmt = conn.prepareStatement("UPDATE AUDITEURS SET nom=?,prenom=? WHERE id = ?");
  deleteStmt = conn.prepareStatement("DELETE FROM AUDITEURS WHERE id = ?");

  findAllStmt = conn.prepareStatement("SELECT * FROM AUDITEURS");
  lastIdStmt = conn.prepareStatement("SELECT MAX(id) FROM AUDITEURS");
}
```

```
public Integer create(Auditeur a) throws Exception{
   createStmt.setString(1, a.getNom());
   createStmt.setString(2, a.getPrenom());
   createStmt.executeUpdate();
```

DBAuditeurDAO, Create

```
public Integer create(Auditeur a) throws Exception{
  //this.createStmt = conn.prepareStatement("INSERT INTO AUDITEU
  this.createStmt.setString(1, a.getNom());
  this.createStmt.setString(2, a.getPrenom());
  createStmt.executeUpdate();
  ResultSet result = lastIdStmt.executeQuery();
  int id = -1;
  if(result.next()){
    id = result.getInt(1);
    a.setId(id);
  }
  return id;
}
```

- **L'identifiant s'est auto incrémenté,**
  - **a.setId(id);**

DBAuditeurDAO, Retrieve, update

```
public Auditeur retrieve(Integer id) throws Exception{
  //this.retrieveStmt = conn.prepareStatement("SELECT id,
  retrieveStmt.setInt(1, id);
  ResultSet result = retrieveStmt.executeQuery();
  Auditeur auditeur = null;
  if(result.next()){
    auditeur.setId(id);
    auditeur.setNom(result.getString("nom"));
    auditeur.setPrenom(result.getString("prenom"));
  }
  return auditeur;
}
public void update(Auditeur a) throws Exception{
  //this.updateStmt = conn.prepareStatement("UPDATE
  updateStmt.setString(1, a.getNom());
  updateStmt.setString(2, a.getPrenom());
  updateStmt.setInt(3, a.getId());
  updateStmt.executeUpdate();
}
```

FileAuditeurDAO, Delete, FindAll

```
public void delete(Integer id) throws Exception{
  //this.deleteStmt = conn.prepareStatement("DELETE
  deleteStmt.setInt(1, id);
  deleteStmt.executeUpdate();
}

public List<Auditeur> findAll() throws Exception{
  //this.findAllStmt = conn.prepareStatement("SELEC
  List<Auditeur> liste = new ArrayList<Auditeur>();
  ResultSet result = findAllStmt.executeQuery();
  while(result.next()){
    Auditeur a = new Auditeur();
    a.setId(result.getInt("id"));
    a.setNom(result.getString("nom"));
    a.setPrenom(result.getString("prenom"));
    liste.add(a);
  }
  return liste;
}
```

Démonstration

Java New I/O - NIO

- **Un autre support**
  - **Cf. http://gfx.developpez.com/tutoriel/java/nio/**

```
in = new FileInputStream("fichier").getChannel();
int size = (int) in.size();
bytes = in.map(FileChannel.MapMode.READ_ONLY, 0, size);
char c = (char) bytes.get();
```

```
ByteBuffer bytes = ByteBuffer.allocateDirect(1024);
FileChannel in = new FileInputStream("fichier").getChannel();
int read = in.read(bytes);
char c = (char) bytes.get();
```