

Cours n° 6

Les flots d'entrées-sorties



Sommaire

1. Flots d'entrées/sorties

1.1. Gestion d'un système de fichiers

1.2. Flots d'octets

1.2.1. Flots d'octets en sortie

1.2.2. Flots d'octets en entrée

1.3. Flots texte

1.3.1. Flots texte en sortie

1.3.2. Flots texte en entrée

Définition

Canaux susceptibles de transmettre de l'information sous forme d'octets

Clavier, écran

Systèmes de fichiers (disque dur, disquette, distants)

Réseau (socket, internet)

Périphérique de son et d'image (webcam, microphone, ...)

Normalisation des méthodes de lecture et d'écriture

En Java, deux types de flots

Flots d'octets et flots texte en format unicode

Implémentation en Java

Package « Java.io »

10 interfaces, 50 classes, 16 exceptions

Gestion d'un système de fichiers (File)

Système de représentation d'un chemin

Gestion des répertoires et des fichiers

création, destruction, test d'existence, permissions

Gestion des entrées/sorties binaires (InputStream/OutputStream)

Lecture/écriture d'octets, de tableaux d'octets, de types primitifs et d'objets

Ajout de tampon, de compression, de tables de transcodage

Gestion des entrées/sorties textes (Reader/Writer)

Lecture/écriture de textes

Classe File

Représentation abstraite du chemin d'accès d'un fichier

Format indépendant du système d'exploitation (windows, unix, ...)

Méthodes de conversion

`String` `getAbsolutePath()` `URL` `toURL()`

Parcours dans l'arborescence

`File[]` `ListRoots()` systèmes de fichiers accessibles

`String` `getParent()` `File` `getParentFile()` `File[]` `listFiles()`

Propriétés

`boolean` `exists()` `boolean` `isDirectory()` `boolean` `isFile()`

`long` `length()` `long` `lastModified()`

`boolean` `canRead()` `boolean` `canWrite()` `boolean` `isHidden()`

Création/destruction

`boolean` `createNewFile()` `boolean` `mkdir()` `boolean` `makedirs()`

`boolean` `setReadOnly()` `boolean` `delete()`

1.1. GESTION D'UN SYSTEME DE FICHIERS - EXEMPLE

Recherche d'un fichier dans une arborescence

```
package cours06;
import java.io.*;

public class GestionFile {
    public static File findFirst(File f, String nomFichier) {
        File fres = null; int i;
        File[] tf = f.listFiles(); // liste des fichiers du
        // répertoire
        for (i = 0; i < tf.length; i++) {
            fres = tf[i];
            if (fres.isDirectory() == true) {
                fres = findFirst(tf[i], nomFichier);
                if (fres != null) break; // sortie de boucle
            }
            else
                if (fres.getName().equals(nomFichier) == true)
                    break; // sortie de boucle
        }
        if (i == tf.length) return null; // fichier non trouvé
        else return fres; // fichier trouvé
    }
}
```

1.1. GESTION D'UN SYSTEME DE FICHIERS - EXEMPLE

Test de la méthode FindFirst()

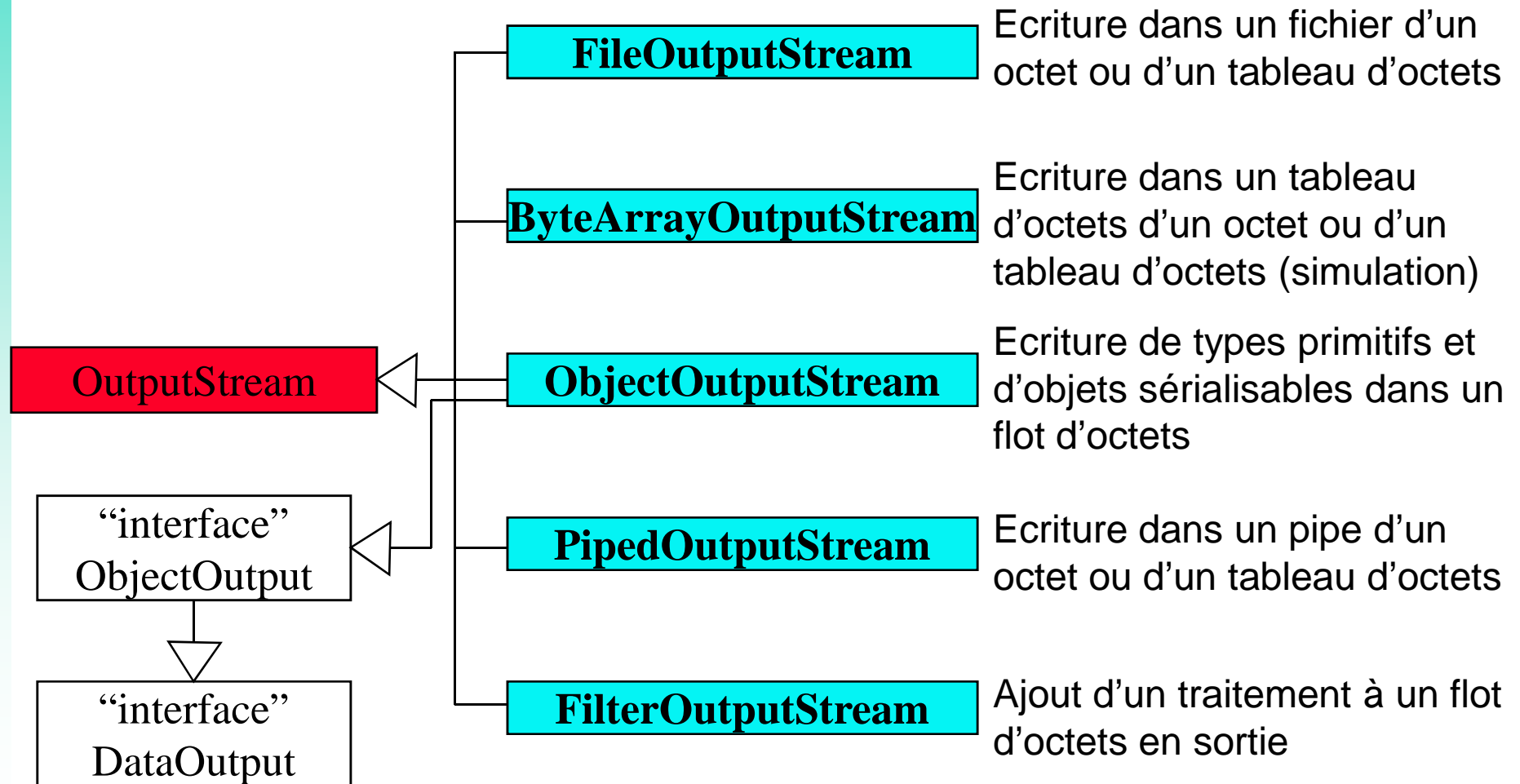
```
package cours06; import java.io.*; import util.*;

public class testGestionFile {
    public static void main(String[] args) {
        String nR = Keyboard.getString("Répertoire de recherche : ");
        String nF = Keyboard.getString("Fichier à rechercher : ");
        File fres = GestionFile.findFirst(new File(nR), nF);
        if (fres != null) {
            System.out.println("Chemin absolu du fichier : ");
            System.out.println(fres.getAbsolutePath());
        }
        else System.out.println("Aucune occurrence du fichier...");
    }
}
```

```
Répertoire de recherche : d:/
Fichier à rechercher : Date.java
Chemin absolu du fichier :
d:\... \cours04\Date.java
```

Classe abstraite « OutputStream »

Cinq classes dérivées



Classe `FileOutputStream`

Deux constructeurs

Ouverture en écriture d'un fichier

`FileOutputStream`(`File`)

`FileOutputStream`(`String`)

Écriture d'octets dans un fichier

`write`(`byte`) Écriture d'un octet

`write`(`byte[]`) Écriture d'un tableau d'octets

`write`(`byte[]`, `int`, `int`) Écriture d'une partie d'un tableau d'octets

Fermeture du Fichier

`void close()`

..

Classe **ObjectOutputStream**

Un constructeur

ObjectOutputStream(**OutputStream**)

OutputStream à choisir parmi {FileOutputStream, ByteArrayOutputStream, PipedOutputStream, FilterOutputStream}

Écriture de types primitifs dans un flot d'octets

void writeBoolean(**boolean**)

void writeByte(**byte**)

void writeBytes(**String**)

void writeChar(**char**)

void writeChars(**String**)

void writeUTF(**String**)

void writeShort(**short**)

void writeInt(**int**)

void writeLong(**long**)

void writeFloat(**float**)

void writeDouble(**double**)

Écriture d'objets sérialisables dans un flot d'octets

Objet implémentant l'interface **Sérializable** ou dont tous les attributs sont des objets sérialisables

void writeObject(**object**)

1.2.1. FLOTS D'OCTETS EN SORTIE – EXEMPLE 1

Ecriture d'un objet dans d'un fichier binaire (1/2)

```
package cours06;
import java.io.*; import cours04.Date;

public class Date3 extends Date implements Serializable {
    static final long serialVersionUID = 176;
    public Date3(int j, int m, int a) {super(j,m,a);}
    public Date3() {super(0,0,0);}

    public void Sauvegarder(File f, boolean ajout) {
        try {
            FileOutputStream fos = new FileOutputStream(f, ajout);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(this);
            System.out.println("date sauvegardée");
            oos.close();
        }
        catch (IOException e) { e.printStackTrace(); }
    }
}
```

1.2.1. FLOTS D'OCTETS EN SORTIE – EXEMPLE 1

Ecriture d'un objet dans d'un fichier binaire (2/2)

```
package cours06;
import java.io.*;import util.Keyboard;

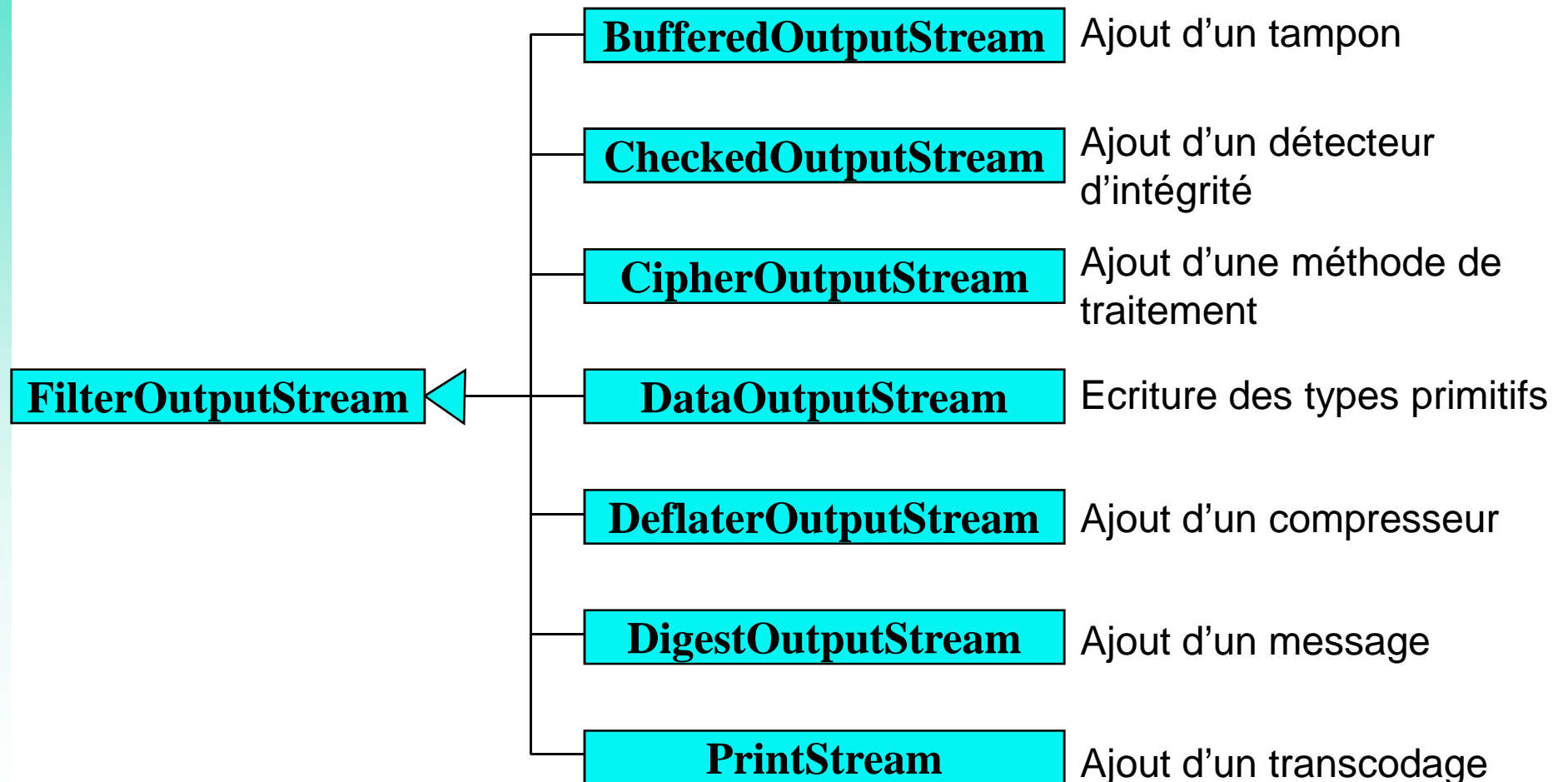
public class testDate3 {

    public static void main(String[] args) {
        File f = new File(Keyboard.getString("Fichier à créer : "));
        Date3 today = new Date3(10, 11, 2005);
        Date3 fête = new Date3(14, 7, 2006);
        while (today.CompareTo(fête) == true) {
            today.Incrementer();
            today.Sauvegarder(f, true);
        }
    }
}
```

Fichier à créer : cours06/FS_Date
date sauvegardée
date sauvegardée
...

Arborescence de la classe « **FilterOutputStream** »

Sept classes dérivées



Ecriture dans d'un fichier binaire compressé

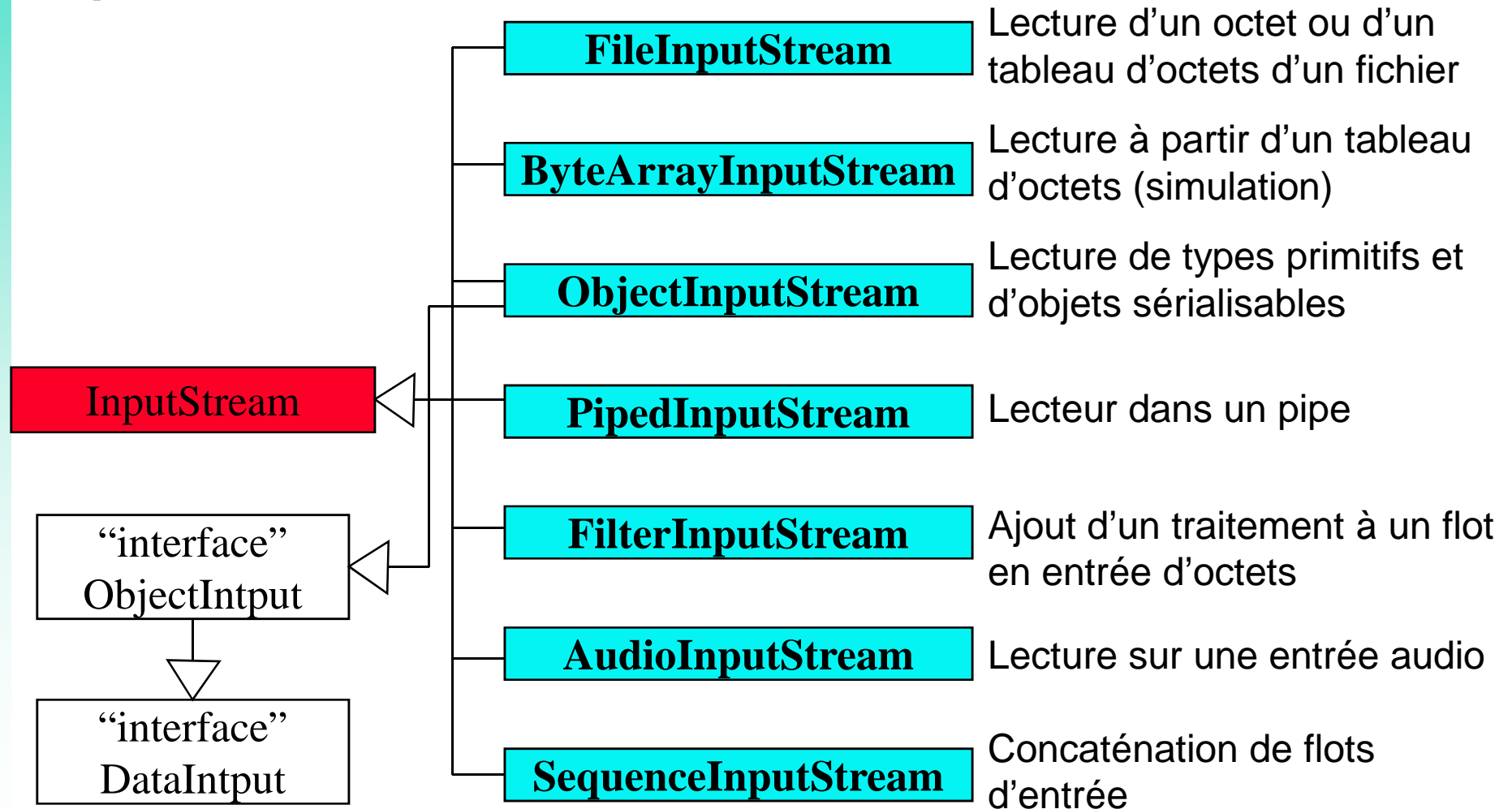
```
package cours06; import java.io.*; import java.util.zip.*;
import util.Keyboard;
public class testEcrZip {
    public static void main(String[] args) {
        try {
            File f = new File(Keyboard.getString("Nom du fichier
                                                    compressé : "));

            FileOutputStream fos = new FileOutputStream(f);
            DeflaterOutputStream dos = new DeflaterOutputStream(fos);
            DataOutputStream fdos = new DataOutputStream(dos);
            fdos.writeUTF("Le président de la Serbie-Monténégro
                           s'excuse pour la guerre en Bosnie.");
            fdos.writeUTF("Le Monde");
            fdos.close();
            System.out.println("Fin d'écriture de fichier");
        }
        catch (IOException e) { e.printStackTrace(); }
    }
}
```

Nom du fichier compressé :cours06/S_Zip
Fin d'écriture de fichier

Classe abstraite « InputStream »

Sept classes dérivées



1.2.2. FLOTS D'OCTETS EN ENTREE – EXEMPLE 2

Lecture d'un fichier compressé

```
package cours06;
import java.io.*; import java.util.zip.*;
import util.Keyboard;
public class testLctZip {
    public static void main(String[] args) {
        String d;
        try {
            File f = new File(Keyboard.getString("Nom du fichier
                                                    compressé : "));

            FileInputStream fis = new FileInputStream(f);
            InflaterInputStream is = new InflaterInputStream(fis);
            DataInputStream idis = new DataInputStream(is);
            while ((d=idis.readUTF()) != null){
                System.out.println(d);
            }
            idis.close();
        }
        catch (EOFException e) { System.out.println("Fin de lecture
                                                    de fichier"); }
        catch (IOException e) { e.printStackTrace(); }
    }
}
```

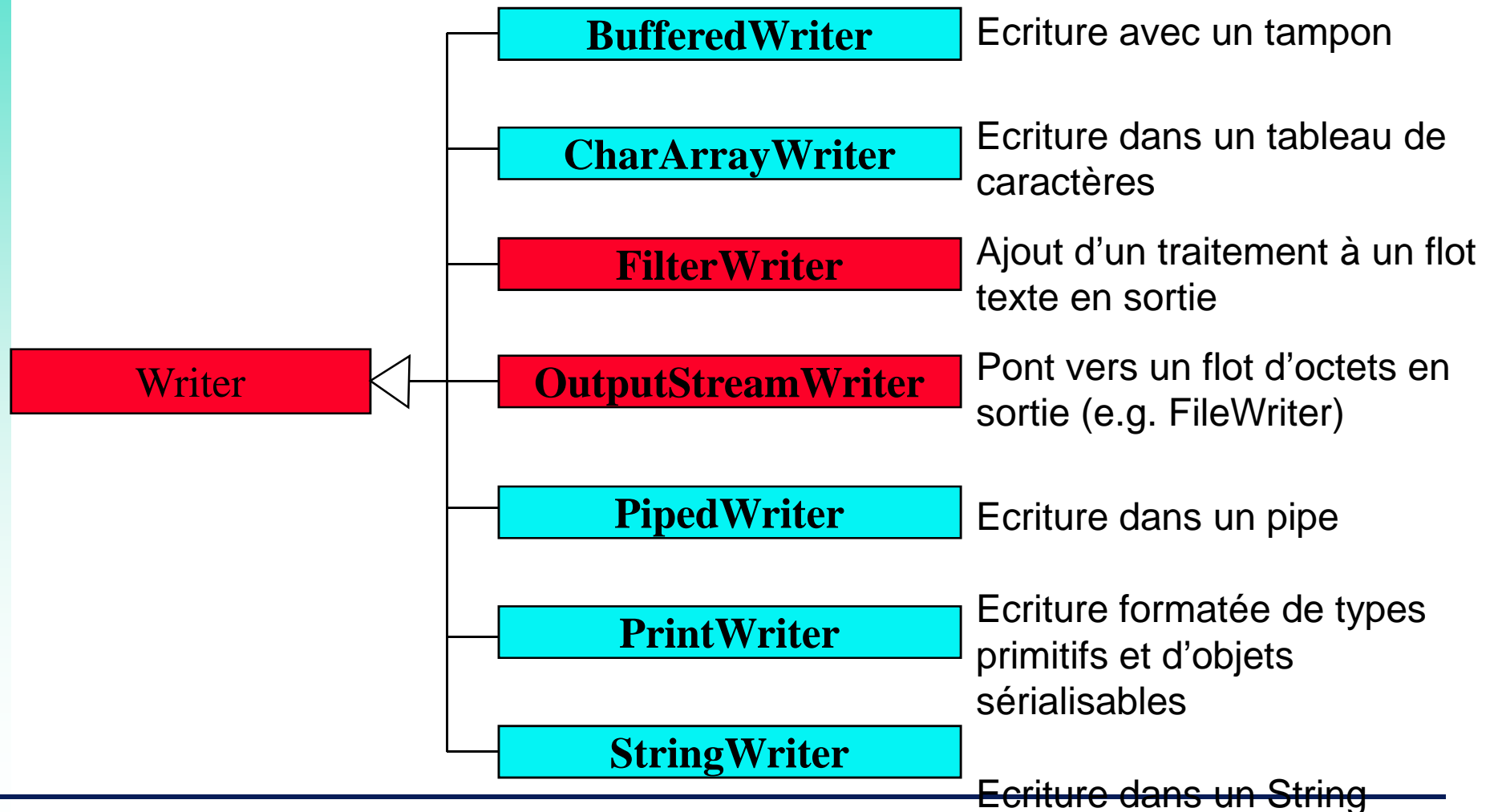

1.2.2. FLOTS D'OCTETS EN ENTREE – EXEMPLE 2

Lecture d'un fichier compressé**Exécutions de testLctZip**

```
Nom du fichier compressé : cours06/S_Zip  
Le président de la Serbie-Monténégro s'excuse pour la  
guerre en Bosnie.  
Le Monde  
Fin de lecture de fichier
```

Classe abstraite « Writer »

Sept classes dérivées



Classe « **PrintWriter** »

Quatre constructeurs de base

PrintWriter(**File** file) à partir d'un fichier

PrintWriter(**String** fileName) à partir d'un nom de fichier

PrintWriter(**OutputStream** out) à choisir parmi { **FileOutputStream**,
ByteArrayOutputStream, **PipedOutputStream**, **FilterOutputStream**}

PrintWriter(**Writer** out) à choisir parmi { **Buffered Writer**, **CharArrayWriter**,
PipedWriter, **StringWriter** }

Deux constructeurs avec transfert possible en fin de ligne

PrintWriter(**OutputStream** out, **boolean** autoFlush)

PrintWriter(**Writer** out , **boolean** autoFlush)

Deux constructeurs avec choix du jeu de caractères

PrintWriter(**File** file, **String** charset)

PrintWriter(**String** fileName, **String** charset)

Classe « PrintWriter »

Ecriture formatée dans un flot texte

```
void print (boolean b)
void print(char c)      void print(char[] s)
void print(double d)    void print(float f)
void print(int i)       print(long l)
void print(String s)
void print(Object obj)
```

Ecriture formatée avec passage à la ligne dans un flot texte

```
void println (boolean b) void println(char c) void println(char[] s)
void println(double d) void println(float f) void println(int i) println(long l)
void println(String s) void println(Object obj)
```

1.3.1. FLOTS TEXTE EN SORTIE - EXEMPLE

Ecriture d'un fichier texte

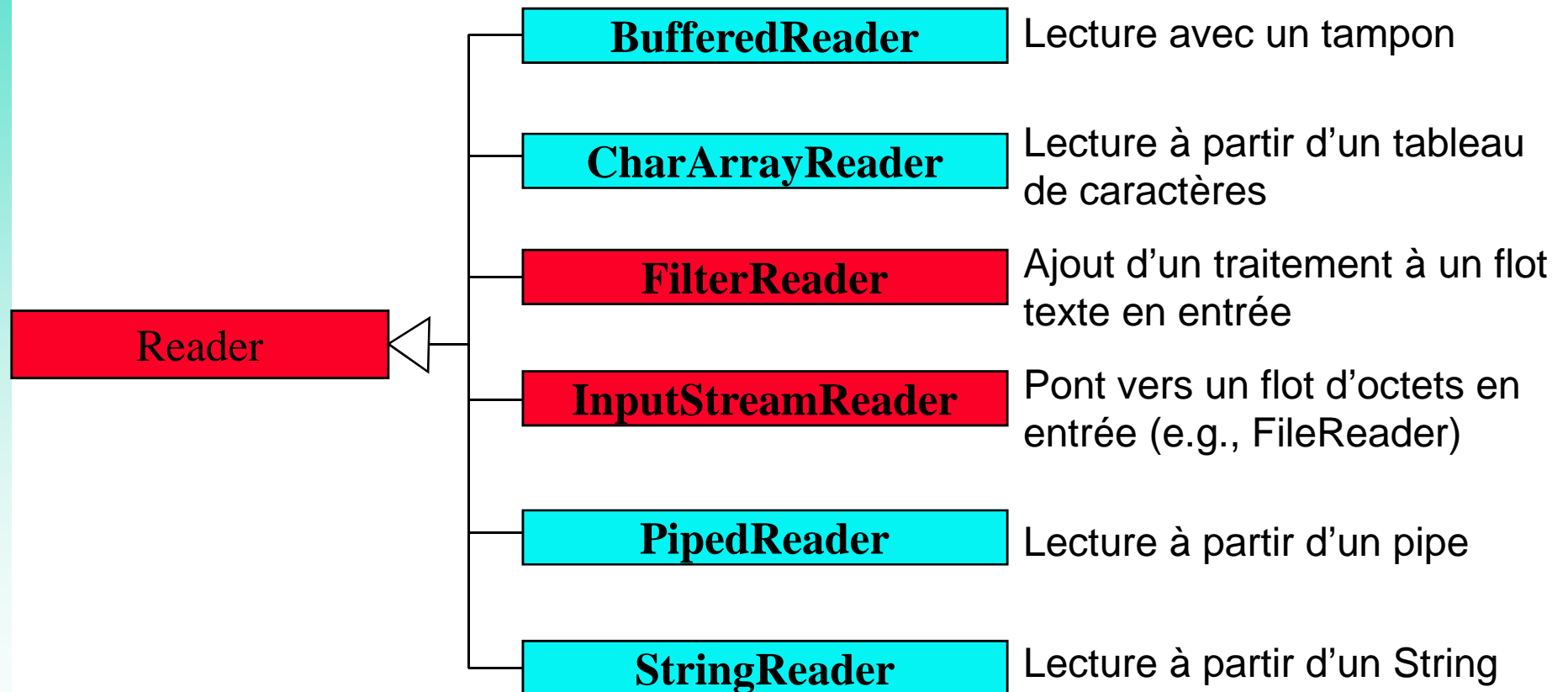
```
public class testEcrTxt {  
    public static void main(String[] args) {  
        String ligne;  
        try {  
            File f = new File(Keyboard.getString("Nom du fichier  
                                                    à écrire : "));  
  
            PrintWriter pr = new PrintWriter(f);  
            pr.print("Hello, ");  
            pr.println("you...");  
            pr.close();  
            System.out.println("Fin d'écriture fichier");  
        }  
        catch (IOException e) { e.printStackTrace(); }  
    }  
}
```

Fichier FS_Txt :
Hello, you...

Nom du fichier à écrire : cours06/FS_Txt
Fin d'écriture fichier

Classe abstraite « Reader »

Six classes dérivées



1.3. 2.FLOTS TEXTE EN ENTREE – EXEMPLE 1

Lecture-clavier robuste

```
package util;
import java.io.*;
/**
 * Bibliothèque des entrées sécurisées clavier
 * Lecture robuste des variables de type natif
 * (byte, short, int, float, double, char)
 * et de variables de type String
 * Toutes les méthodes sont des méthodes de classe
 */
public class KeyboardSecure {
    /**
     * Lecture d'une ligne saisie au clavier
     * @return La chaîne de caractères saisie au clavier
     */
    private static String getLine() {
        String ligne_lue = null;
        try {
            InputStreamReader lecteur = new InputStreamReader(System.in);
            BufferedReader entree = new BufferedReader(lecteur);
            ligne_lue = entree.readLine();
        }
        catch (IOException e) { System.exit(0); }
        return ligne_lue;
    }
}
```

1.3.2. FLOTS TEXTE EN ENTREE – EXEMPLE 1

Lecture-clavier robuste

```
/**
 * Lecture d'une chaîne de caractères saisie au clavier
 * @param mess    Message d'invite d'entrée au clavier
 * @return        La chaîne lue au clavier
 */
public static String getString(String mess) {
    System.out.print(mess);
    return getLine();
}

/**
 * Lecture d'un caractère saisi au clavier
 * @param mess    Message d'invite à entrer un caractère au clavier
 * @return        Le caractère lu au clavier
 */
public static char getChar(String mess) {
    System.out.print(mess);
    return getLine().charAt(0);
}
```


1.3.2. FLOTS TEXTE EN ENTREE – EXEMPLE 1

Lecture-clavier robuste

```
/**
 * Lecture robuste d'un int saisi au clavier
 * @param messMessage d'invite à entrer un int au clavier
 * @return L'int lu au clavier
 */
public static int getInt(String mess) {
    System.out.print(mess);
    int entier=Integer.MIN_VALUE;
    while (true) {
        try { entier = Integer.parseInt(getLine()); break; }
        catch (NumberFormatException e){
            System.out.println("Erreur de format...");
            System.out.print(mess);
        }
    }
    return entier;
}
```

Squelette identique pour getByte, getShort, getFloat, getDouble (cf. KeyboardSecure.java)

1.3.2. FLOTS TEXTE EN ENTREE – EXEMPLE 1

Lecture-clavier robuste

```
package cours06;
import util.*;
/** Driver de test de la bibliothèque KeyboardSecure */
public class LectureRobuste {

    public static void main(String[] args) {
        String ligne = KeyboardSecure.getString("Lecture d'une ligne
                                                : ");
        System.out.println("Ligne lue au clavier : " + ligne);

        char caractere = KeyboardSecure.getChar("Lecture d'un
                                                caractère : ");
        System.out.println("Caractère lu au clavier : " + caractere);

        int entier = KeyboardSecure.getInt("Entier : ");
        System.out.println("Entier lu au clavier : " + entier);

        float flottant = KeyboardSecure.getFloat("Flottant : ");
        System.out.println("Flottant lu au clavier : " + flottant);
    }
}
```

1.3.2. FLOTS TEXTE EN ENTREE – EXEMPLE 1

Lecture-clavier robuste

Exécutions de LectureRobuste

```
Lecture d'une ligne : Driver de test de KeyboardSecure
Ligne lue au clavier : Driver de test de KeyboardSecure
Lecture d'un caractère : azerty
Caractère lu au clavier : a
Entier : deux
Erreur de format...
Entier : 34.5
Erreur de format...
Entier : 345
Entier lu au clavier : 345
Flottant : pi
Erreur de format...
Flottant : 3.14
Flottant lu au clavier : 3.14
```

Formats d'entrée des réels
(float, double) :

3	3.0
3f	3.0
314e-2	3.14
314E-2	3.14

Classe « `BufferedReader` »

Deux constructeurs

`BufferedReader(Reader)` `BufferedReader(Reader, int)`

Reader à choisir parmi {`CharArrayReader`, `FileReader`, `PipedReader`, `StringReader`}

Lecture de caractères dans un flot texte

`int read()` Lecture d'un caractère

`int read(char[], int, int)` Lecture de caractères et stockage dans un tableau

`String readLine()` Lecture d'une ligne de textes

`long skip(long n)` sauter n caractères

1.3.2. FLOTS TEXTE EN ENTREE – EXEMPLE 1

Lecture d'un fichier texte

```

package util;
import java.io.*;
public class tstLctTxt {
    public static void main(String[] args) {
        String ligne;
        try {
            File f = new File(Keyboard.getString("Nom du fichier
                                                    à lire : "));

            FileReader fr = new FileReader(f);
            BufferedReader br = new BufferedReader(fr);
            while ((ligne=br.readLine()) != null) {
                System.out.println(ligne);
            }
            br.close();
        }
        catch (IOException e) { e.printStackTrace(); }
    }
}

```

Nom du fichier à lire : cours06/FS_Date
 xœcðøIU((:¼²83%5⁻D!%U!'Q!8μ()3U×7?⁻ äðÊ¼Ã+Ó(òŠõs+'K(?ªók(
 @ŠòKS(ŠRRóœð(ó2Sõ8€fõnn

1.3.2. FLOTS TEXTE EN ENTREE – EXEMPLE 2

Lecture d'un fichier texte

```
Nom du fichier à lire : cours06/testDate3.java
package cours06;

import java.io.*;
import util.Keyboard;

public class testDate3 {

    public static void main(String[] args) {
        File f = new File(Keyboard.getString("Nom du fichier à créer :"));

        Date3 today = new Date3(10, 11, 2005);
        Date3 fête = new Date3(14, 7, 2006);
        while (today.CompareTo(fête) == true) {
            today.Incrementer();
            today.Sauvegarder(f, true);
            System.out.println("date sauvegardée");
        }
    }
}
```

1.3.2. FLOTS TEXTE EN ENTREE – EXEMPLE 3

Lecture d'un fichier texte formaté

```
package cours06;
import java.io.*; import util.Keyboard;

public class testLctTxtFmt {
    public static void main(String[] args) {
        String ligne;
        try {
            File f = new File(Keyboard.getString("Nom du fichier
                                                    à lire : "));

            FileReader fr = new FileReader(f);
            BufferedReader br = new BufferedReader(fr);
            while ((ligne=br.readLine()) != null) {
                StringTokenizer st = new StringTokenizer(ligne,
                                                         "[:\n\t]");

                while (st.hasMoreTokens())
                    System.out.print(st.nextToken());
                System.out.print("\n");
            }
            br.close();
        }
        catch (IOException e) { e.printStackTrace(); }
    }
}
```

1.3.2. FLOTS TEXTE EN ENTREE – EXEMPLE 3

Lecture d'un fichier texte formaté**Exécution de testLctTxtFmt**

Fichier formaté *TestToken* :

```
[14; 7; 1789]  
[11; 11; 1918]  
[8; 5; 1945]
```

Nom du fichier à lire : **TestToken**

```
14 7 1789  
11 11 1918  
8 5 1945
```