Cours7
Exceptions et fichier texte

Année 2024-2025 J-F. Kamp - IUT de Vannes

Les exceptions

Les erreurs sont signalées par des exceptions

```
class SimplesTableau {

  void principal() {
    afficherTab ( null );
  }

  void afficherTab(int[] tab)
throws Exception {

    if ( tab == null ) {
      throw new Exception
("Erreur:tableau inexistant");
    } else {
      …
    }
  }

}
```

Les erreurs sont capturées

```
class SimplesTableau {

  void principal() {
    try {
      afficherTab ( null );
    }
    catch ( Exception e ) {
      Sop ( e.getMessage() );
    }
  }

  void afficherTab(int[]
tab)throws Exception {
    if ( tab == null ) {
      throw new Exception
("Erreur:tableau inexistant");
    } else {
      …
    }
  }
}
```

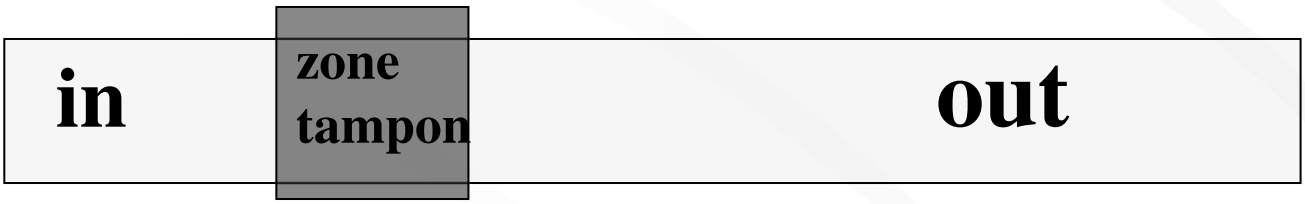
Les entrées/sorties textuelles

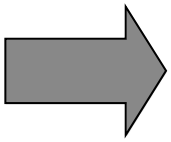
Les flux I/O génériques

Les entrées/sorties s'effectuent en Java avec des *Stream*.

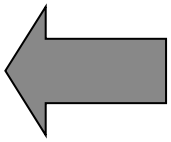
Un *Stream* ou flot de données est une séquence d'octets (ou de bytes) avec un début et une fin.

Ce *Stream* peut être dirigé vers la sortie (*out*) ou vers l'entrée (*in*).





Ecrire (in => out)



Lire (out => in)

Paquetage java.io

En java un paquetage (50 classes environ) est dédié exclusivement aux entrées/sorties :

java.io

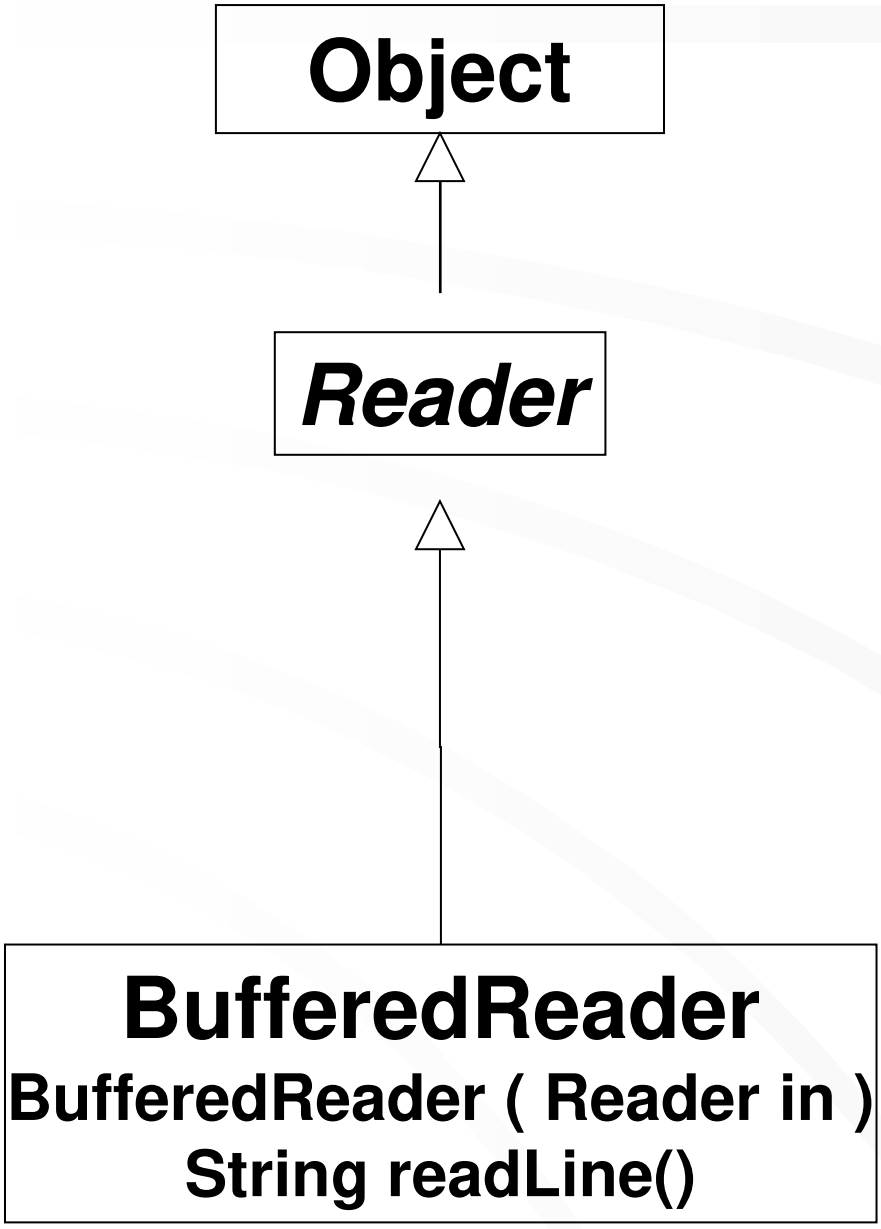
Ce paquetage est organisé en une arborescence d'héritage dont les classes :

- ***java.io.Reader*** **: flux en lecture d'une collection de caractères**
- ***java.io.Writer*** **: flux en écriture d'une collection de caractères**

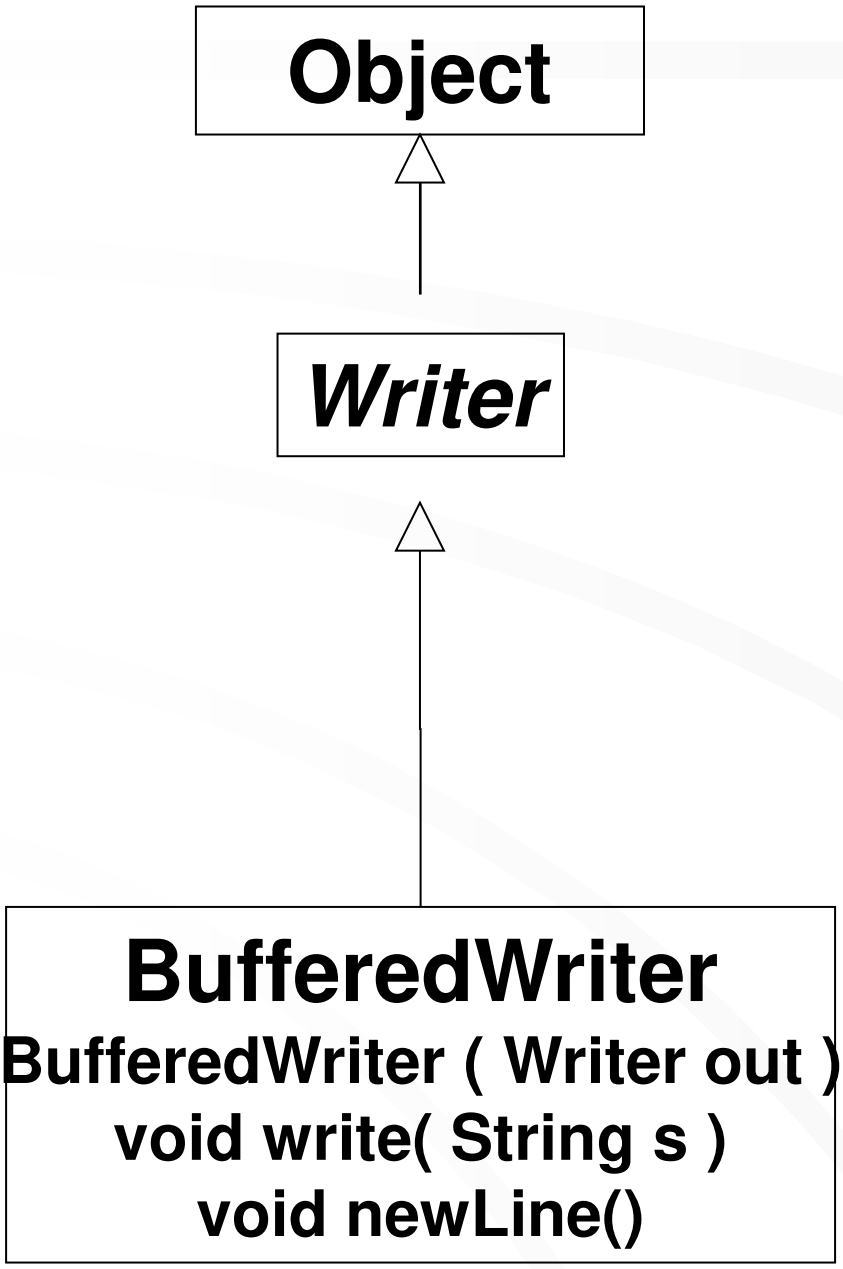
⇒ il faudra donc importer ces classes

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

Arborescence d'héritage (input texte)



Arborescence d'héritage (output texte)



Fichier texte / fichier binaire

Sur le disque, on ne trouve que du binaire :

0100110001100101000010000111000 0 …

SI fichier texte ALORS

La suite binaire = une suite de caractères ASCII dont :

- **des caractères de fin de ligne < CR >**
- **un caractère de fin de fichier < EOF >**

Exemple :

Le petit chat.<CR>
Encore le petit chat,<CR>
etc.etc.<CR>
etc.etc.<CR>
Finalement le petit chat.<EOF>

Fichier texte / fichier binaire

Un fichier texte est un fichier qui, lorsqu'il est chargé par un éditeur de texte (*geany*, *atom*, *visual studio*, *notepad*++, …), devient LISIBLE.

Exemple :

- **un fichier *.java***
- **un fichier *.html***
- **un script *unix***
- **n'importe quel texte tapé au clavier**

```
// cette classe est enregistrée dans un fichier texte
// SimplesTableau.java, elle est donc lisible par un
// éditeur
class SimplesTableau {

        // point d'entrée
        void principal() {…}
}
```

Lecture d'un fichier texte

Soit l'information
01001100011001010000100001110000

Pour lire une ligne, il faut donc :

- **découper par paquets de 8 bits (octets)**
  01001100 / 01100101 / 00001000 / 01110000

- **transformer octet –> caractère**
  ASCII76 / ASCII101 / ASCII8 / ASCII112

  L / e/ '\b' / p ('\b'=backspace)

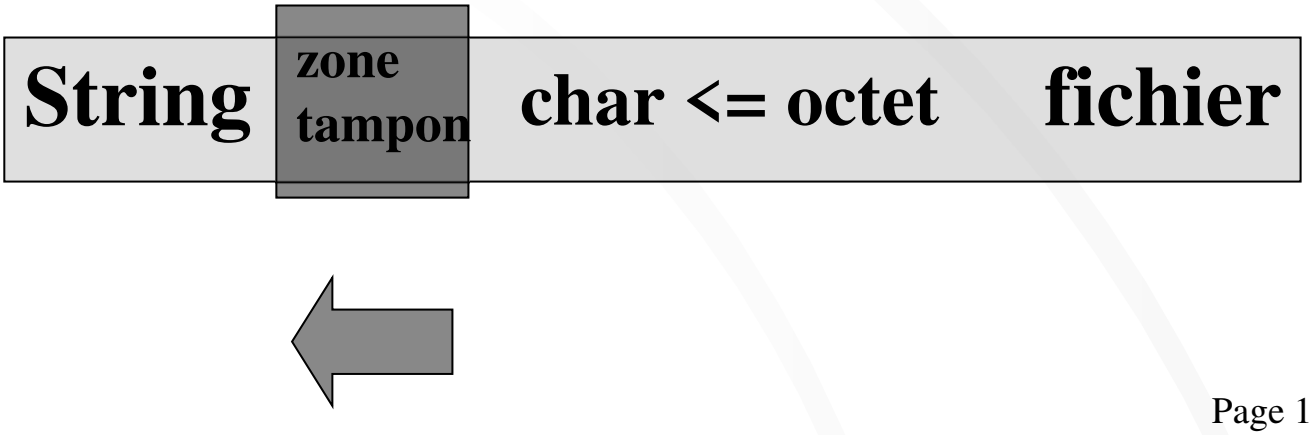
- **assembler les caractères pour former la ligne jusqu'à rencontrer le caractère <CR>**

Lecture fichier texte en Java

Une lecture efficace de lignes de texte dans un fichier en *Java* nécessite l'utilisation de 2 objets : *FileReader* et *BufferedReader*.

- **rôle de l'objet *FileReader* : lit les octets un par un et les transforme en caractères**

- **rôle de l'objet *BufferedReader* : regroupe les caractères pour former une ligne de texte**



Lecture fichier texte en Java

Quel objet est capable d'ouvrir un flux sur un fichier ?

FileReader

constructeur : ***FileReader (String nomFich)***

Quel objet est capable de lire une ligne de texte ?

BufferedReader

constructeur : ***BufferedReader (Reader input)***

méthode ***String readLine()***

Lecture fichier texte en Java

```
void lecture ( String nomFichier ) {
        boolean eof = false;
        String str;
        BufferedReader tampon;
        FileReader file;

        file = new FileReader ( nomFichier );
        tampon = new BufferedReader ( file );

        while ( ! eof ) {
                str = tampon.readLine();
                if ( str == null ) {
                        eof = true;
                }
                else {
                        System.out.println( str );
                }
        }

        tampon.close();
}
```

Lecture fichier texte en Java

MAIS il y a des exceptions à gérer…

FileReader

constructeur : ***FileReader (String nomFich)***

est susceptible de lancer ***FileNotFoundException***

BufferedReader

constructeur : ***BufferedReader (Reader input)***

méthode ***String readLine()***

est susceptible de lancer ***IOException***

Lecture fichier texte en Java

```
void lecture ( String nomFichier ) {
        boolean eof = false; String str;
        BufferedReader tampon; FileReader file;

        try {
                tampon = new BufferedReader ( new
FileReader ( nomFichier ) );

                while ( ! eof ) {
                        str = tampon.readLine();

                        if ( str == null ) { eof = true; }
                        else {
                                System.out.println( str );
                        }
                }
                tampon.close();
        }
        catch (FileNotFoundException e) {
                 System.out.println ( e.getMessage() );
        }
        catch (IOException e) {
                 System.out.println ( e.getMessage() );
        }
}
```

Décodage d’une chaîne

But : on récupère une chaîne de caractères et on voudrait extraire une par une les données qui composent la chaîne.

Exemple :
01 / TER / Vannes / Redon

séparateur = "/" =>

01
TER
Vannes
Redon

Décodage d'une chaîne

String possède une méthode qui extrait les informations et les renvoie sous forme d'un tableau de _String_ en UNE SEULE opération.

```
…
String[] lesInfos;
// lecture d'une ligne de texte du fichier
String ligne = in.readLine();
// on suppose ici que «ligne» contient
// "01 / TER / Vannes / Redon"

// Extraction des éléments en UNE SEULE opération
lesInfos = ligne.split ( "/" );

// «lesInfos» est un tableau de String qui contient :
//        en 0 : "01 "
//        en 1 : " TER "
//        en 2 : " Vannes "
//        en 3 : " Redon"
//
// et ensuite on peut facilement enlever les espaces
// en début et en fin avec la méthode trim() :
String info = lesInfos[1].trim(); // contient "TER"
```

Ecriture dans un fichier texte

On procède à l'inverse de la lecture.

1. **Codage de l'information en caractères ASCII.**

   Exemple : entier 23
   = '2' '3'
   = ASCII50 ASCII51 (décimal)
   = 00110010 00110011 (binaire)

2. **Regroupement des caractères pour former des lignes.**

   00110010/00110011/<CR>

3. **Ecriture de la ligne dans le fichier**

Ecriture dans un fichier texte

```
void ecriture ( String nomFichier, String ligne ) {
        FileWriter file;
        BufferedWriter tampon;
        PrintWriter out;

        try {
                file = new FileWriter ( nomFichier );
                tampon = new BufferedWriter ( file );
                out = new PrintWriter ( tampon );

                out.println ( ligne );
                out.close();
        }

        catch ( IOException e ) {
                 System.out.println ( e.getMessage() );
        }
}
```