Master 2 LITL
Programmation pour le TAL (SLT0905V)
Fichiers, exceptions et tableaux associatifs

Franck Sajous/CLLE-ERSS

http://fsajous.free.fr/

La classe java.io.File

Constructeur (un parmi d'autres)

```
File (String pathname)
```

→ `File myCorpus = new File ("chemin/nomfichier.txt");`

Séparateur

```
public static final String separator
```

vaut \ sous Windows, / sous Unix

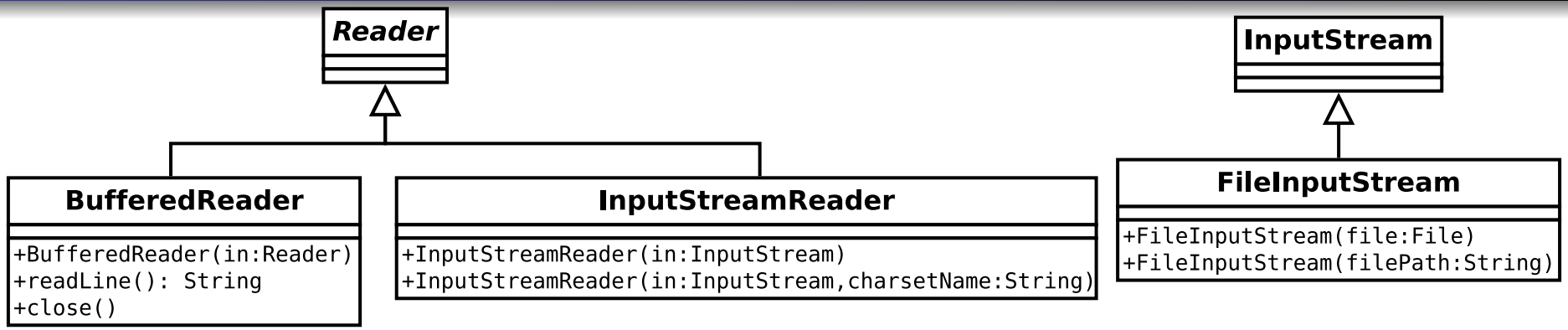
→ `File myCorpus = new File ("chemin" + File.separator + "nomfichier.txt");`

Quelques méthodes

```
public boolean canRead()
public boolean canWrite()
public boolean exists()
public boolean isDirectory()
public File[] listFiles()
```

→ voir la doc !

Lire un fichier ligne à ligne



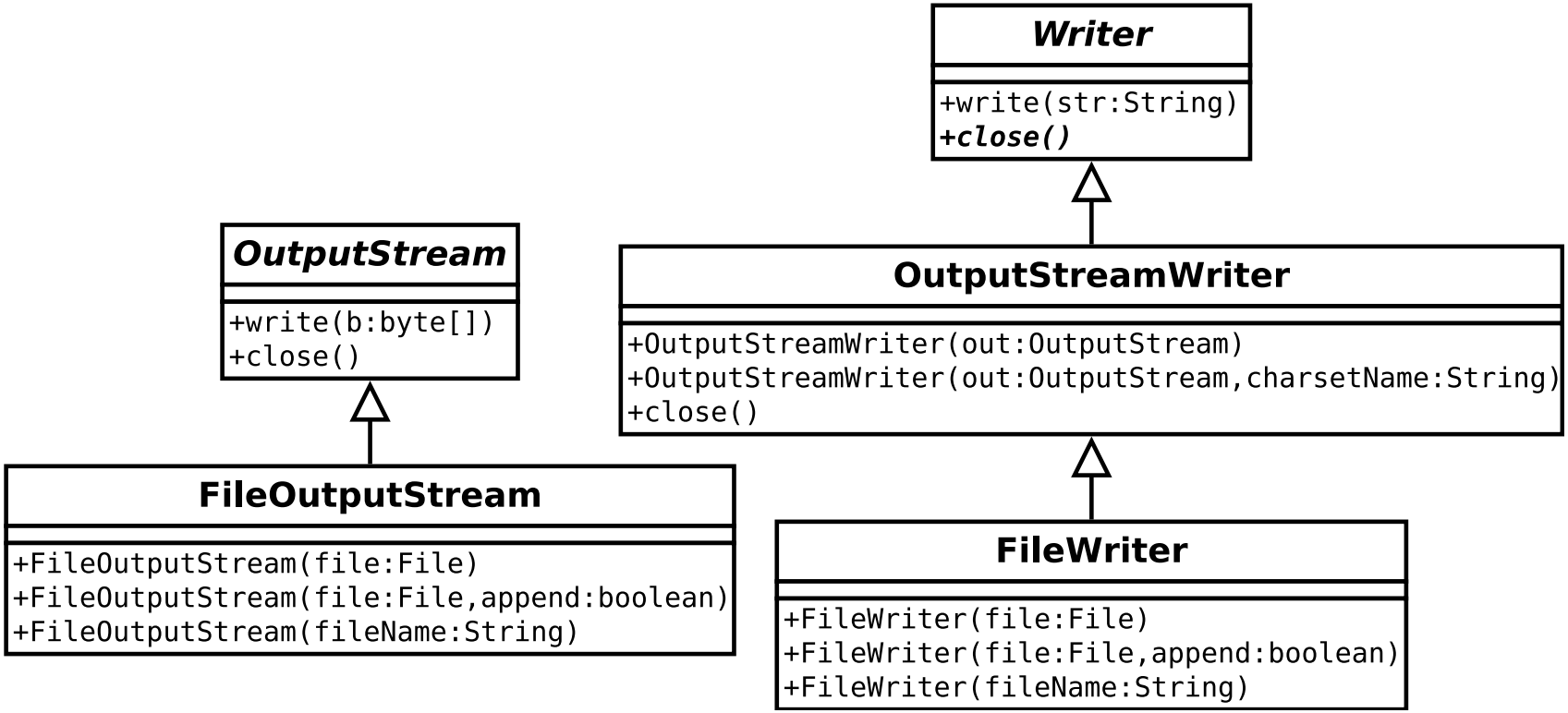

```java
import java.io.*;

BufferedReader bReader = new BufferedReader(
    new InputStreamReader(
        new FileInputStream("/tmp/corpus.tal"), "UTF-8")
    )
);

String line = null;
while ((line = bReader.readLine()) != null)
{
  // do sth
  System.out.println (line); // or sth else...
}

bReader.close ();
```

Écrire dans un fichier




```
import java.io.*;

OutputStreamWriter osWriter = new OutputStreamWriter(
        new FileOutputStream("/tmp/sortie.out"),"UTF-8"
    );

osWriter.write ("test ecriture\n");
osWriter.write ("dans fichier\n");
osWriter.close ();
```

Gestion d'erreurs : les exceptions

Compilation/Exécution

Un certain nb d'erreurs potentielles sont repérées à la compilation (e.g. utilisation d'une variable qui n'a jamais été déclarée)
D'autres erreurs surviennent à l'exécution :

- un utilisateur saisit une année de naissance incorrecte (contenant des lettres) et on essaie de l'utiliser comme un nombre
- un programme essaie de se connecter à un site web, mais la connexion réseau dysfonctionne, etc.
- tentative de lecture d'un fichier effacé, division par zéro

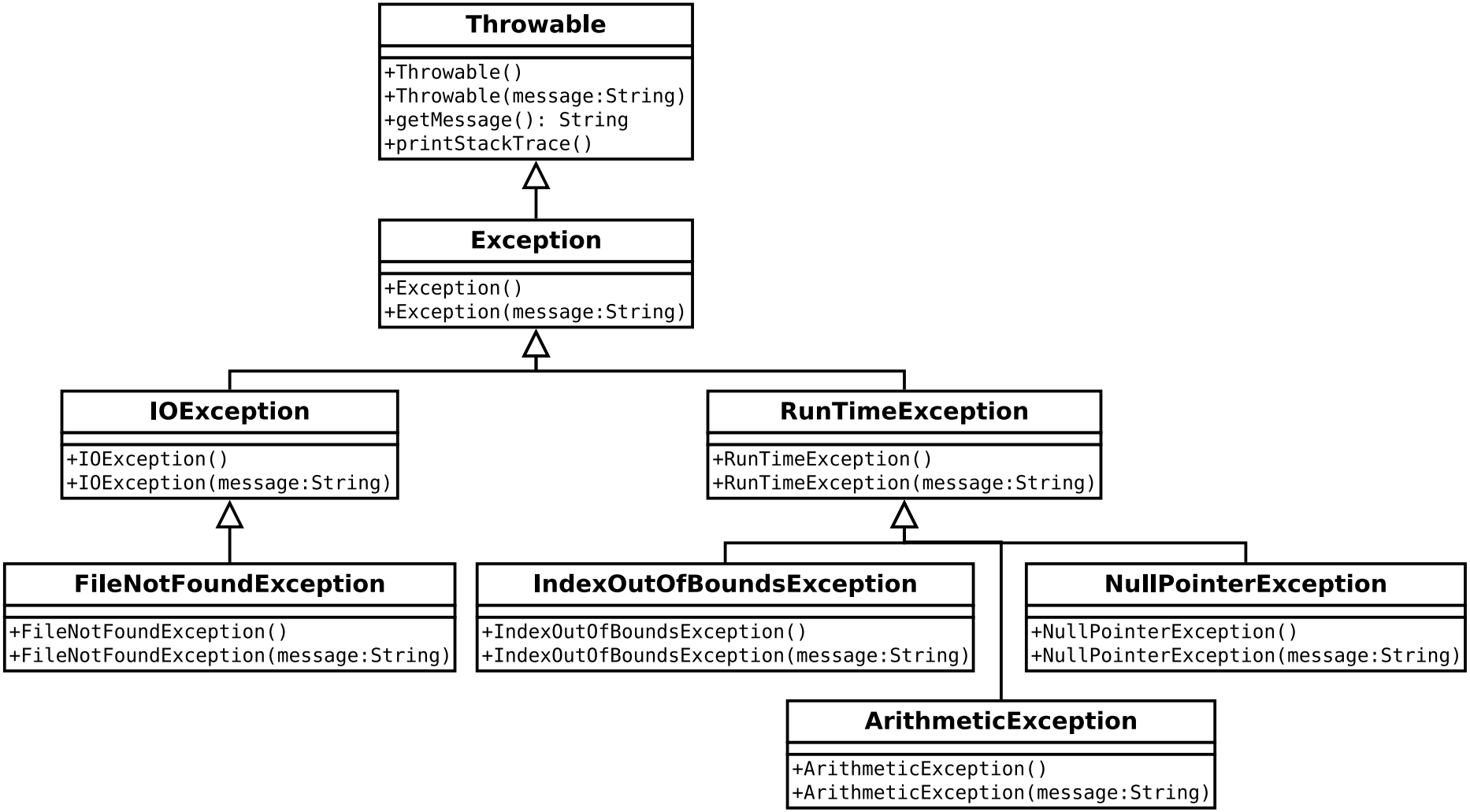
On "sait" où (dans le code) ces erreurs peuvent survenir

Deux possibilités :

- soit *intercepter* l'erreur et déclencher un traitement particulier
- soit déléguer ce traitement et laisser l'erreur se propager

Vous en avez déjà rencontré : affichage de messages "Array Out of Bound Exception", "Null Pointer Exception", etc.

La classe Exception et quelques classes dérivées



Savoir quand une exception risque de se produire

```
FileInputStream fis = new FileInputStream("/tmp/corpus.tal");
```

Savoir quand une exception risque de se produire

```
FileInputStream fis = new FileInputStream("/tmp/corpus.tal");
```

java.io

Class FileInputStream

java.lang.Object
  java.io.InputStream
    java.io.FileInputStream

All Implemented Interfaces:

Closeable, AutoCloseable

```
public class FileInputStream
extends InputStream
```

A `FileInputStream` obtains input bytes from a file in a file system. What files are available depends on the host environment.

`FileInputStream` is meant for reading streams of raw bytes such as image data. For reading streams of characters, consider using `FileReader`.

Since:

JDK1.0

See Also:

`File`, `FileDescriptor`, `FileOutputStream`, `Files.newInputStream(java.nio.file.Path, java.nio.file.OpenOption...)`

Constructor Summary

Constructors

Constructor and Description
`FileInputStream(File file)` Creates a `FileInputStream` by opening a connection to an actual file, the file named by the `File` object `file` in the file system.
`FileInputStream(FileDescriptor fdObj)` Creates a `FileInputStream` by using the file descriptor `fdObj`, which represents an existing connection to an actual file in the file system.
`FileInputStream(String name)` Creates a `FileInputStream` by opening a connection to an actual file, the file named by the path name `name` in the file system.

Savoir quand une exception risque de se produire

```
FileInputStream fis = new FileInputStream("/tmp/corpus.tal");
```

java.io

Class FileInputStream

java.lang.Object
  java.io.InputStream
    java.io.FileInputStream

FileInputStream

```
public FileInputStream(File file)
                throws FileNotFoundException
```

Creates a FileInputStream by opening a connection to an actual file, the file named by the File object file in the file system. A new FileDescriptor

First, if there is a security manager, its checkRead method is called with the path represented by the file argument as its argument.

If the named file does not exist, is a directory rather than a regular file, or for some other reason cannot be opened for reading then a FileNotFoundExce

Parameters:

file - the file to be opened for reading.

Throws:

FileNotFoundException - if the file does not exist, is a directory rather than a regular file, or for some other reason cannot be opened for reading.

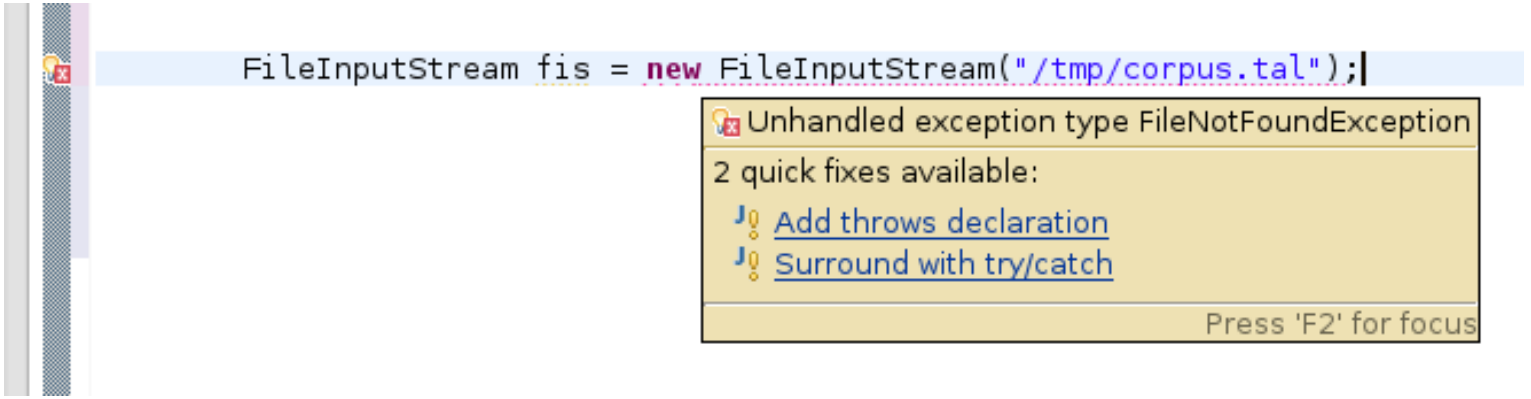
SecurityException - if a security manager exists and its checkRead method denies read access to the file.

See Also:

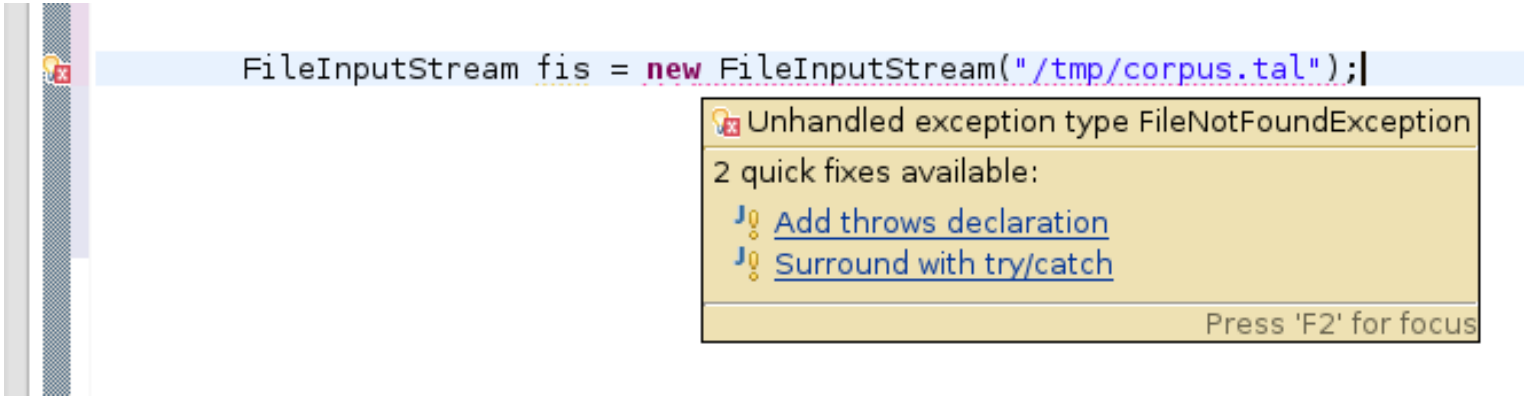
File.getPath(), SecurityManager.checkRead(java.lang.String)

Constructor and Description
FileInputStream(File file) Creates a FileInputStream by opening a connection to an actual file, the file named by the File object file in the file system.
FileInputStream(FileDescriptor fdObj) Creates a FileInputStream by using the file descriptor fdObj, which represents an existing connection to an actual file in the file system.
FileInputStream(String name) Creates a FileInputStream by opening a connection to an actual file, the file named by the path name name in the file system.

Obligation d'intercepter les exceptions : bloc try/catch



Obligation d'intercepter les exceptions : bloc try/catch

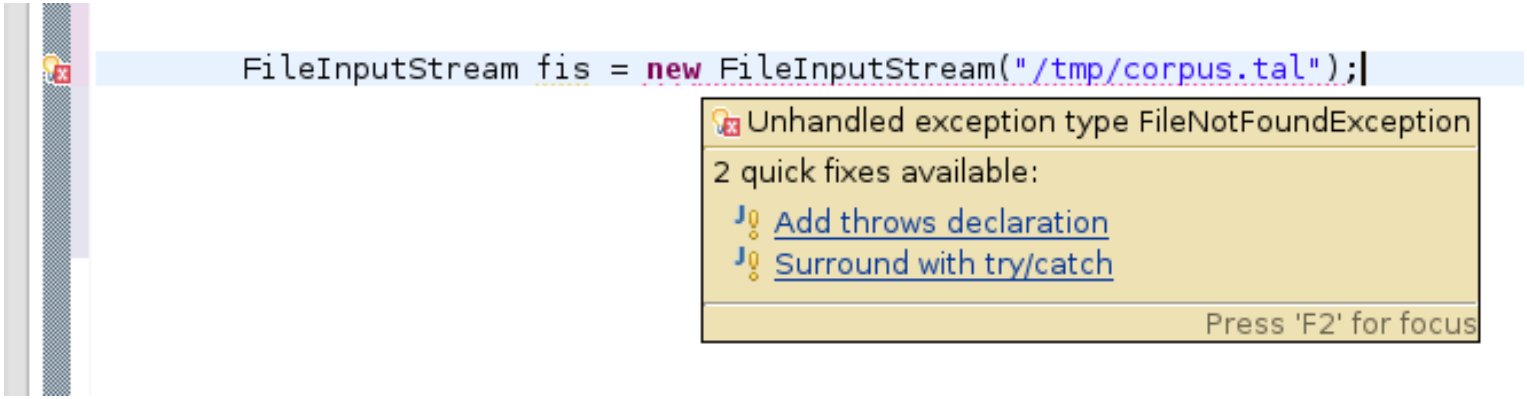



```
FileInputStream fis;

try
{
    fis = new FileInputStream("/tmp/corpus.tal");
    // traitement du fichier
}
catch (FileNotFoundException e)
{
    System.out.println ("Le fichier n'existe pas");
    System.exit(-1);
}

System.out.println ("Le fichier a bien été traité");
```

Obligation d'intercepter les exceptions : bloc try/catch



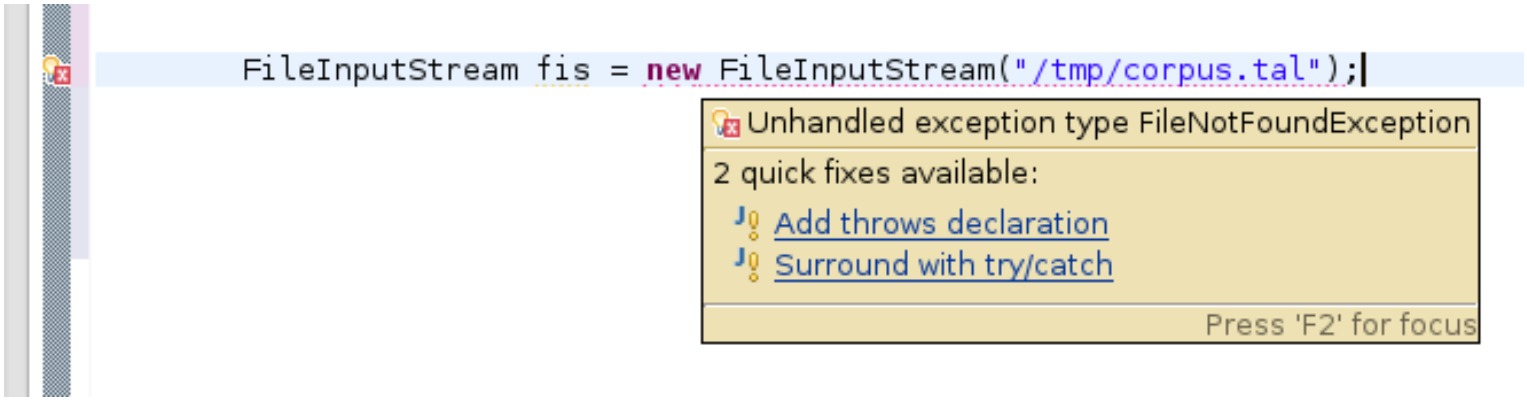

```
FileInputStream fis;

try
{
    fis = new FileInputStream("/tmp/corpus.tal");
    // traitement du fichier
}
catch (FileNotFoundException e)
{
    System.out.println ("Le fichier n'existe pas");
    System.exit(-1);
}

System.out.println ("Le fichier a bien été traité");
```

→ terminaison propre du programme

Intercepter les exceptions, éventuellement les gérer




```
FileInputStream fis;

try
{
   fis = new FileInputStream("/tmp/corpus.tal");
}
catch (FileNotFoundException e)
{
   System.out.println ("Le fichier n'existe pas.");
   System.out.println ("Utilisation du fichier par défaut.");
   loadDefaultFile();
}

System.out.println ("Le fichier a bien été traité");
```

→ traitement alternatif

Intercepter plusieurs types d'exceptions : catch multiples

```
int freqSum = 0;
String line = null;
try
{
   BufferedReader bReader = new BufferedReader(
           new InputStreamReader(
                   new FileInputStream("/tmp/frequencies.txt")
           )
       );
   while ((line = bReader.readLine()) != null)
   {
       freqSum += Integer.parseInt(line);
   }
}
catch (IOException ioe)
{
   System.out.println ("Pb lecture fichier.");
   System.exit(-1);
}
catch (NumberFormatException  nfe)
{
   System.out.println(line  + " - Mauvais format de fréquence !");
  // ignorer la ligne et continuer
}

System.out.println ("Total Freq=" + freqSum);
```

catch multiples : interception spécifique → générique

Catchez les exceptions par ordre de spécificité

```
InputStreamReader reader = null;

try
{
   reader = new InputStreamReader(
          new FileInputStream("/tmp/" + Integer.parseInt(args[1]))
          );
   System.out.println(reader.ready());
}
catch (FileNotFoundException fnfe)
{
   // TODO: handle exception
}
catch (IOException ioe)
{
   // TODO: handle exception
}
catch (NumberFormatException nfe)
{
   // TODO: handle exception
}
catch (Exception e)
{
   // TODO: handle exception
}
```

Lancer une exception

```
public class Token
{
   public void relemmatize ()
   {
      if (lemma.equals("_")) {
         if (getPOS().equals("ADV"))
             setLemma(getWordForm());
         else if (getPOS().equals("NC")) {
             if (getWordForm().endsWith("s"))
                 setLemma(
                     getWordForm().substring(0, getWordForm().length()- 1)
                 );
             else {
                 // autre heuristique de lemmatistion
             }
         }
         else if (getPOS().startsWith("V")) {
             // heuristique de lemmatisation des verbes
         }
         else if (getPOS()...) {
            /* traitement de tous les
               autres POS connus */
         }
         else {
            // ????
         }
      }
   }
}
```

```
Token myToken = sent.getToken(i);
myToken.relemmatize();
```

Lancer une exception

```
public class Token
{
   public void relemmatize ()
   {
      if (lemma.equals("_")) {
         if (getPOS().equals("ADV"))
             setLemma(getWordForm());
         else if (getPOS().equals("NC")) {
             if (getWordForm().endsWith("s"))
                 setLemma(
                     getWordForm().substring(0, getWordForm().length()- 1)
                 );
             else {
                 // autre heuristique de lemmatistion
             }
         }
         else if (getPOS().startsWith("V")) {
             // heuristique de lemmatisation des verbes
         }
         else if (getPOS()...) {
            /* traitement de tous les
               autres POS connus */
         }
         else {
            throw new Exception (
                 "Unexpected POS: " + getPOS()
                 );
         }
      }
   }
}
```

```
Token myToken = sent.getToken(i);
myToken.relemmatize();
```

Lancer une exception

```
public class Token
{
   public void relemmatize () throws Exception
   {
      if (lemma.equals("_")) {
         if (getPOS().equals("ADV"))
             setLemma(getWordForm());
         else if (getPOS().equals("NC")) {
             if (getWordForm().endsWith("s"))
                 setLemma(
                     getWordForm().substring(0, getWordForm().length()- 1)
                 );
             else {
                 // autre heuristique de lemmatistion
             }
         }
         else if (getPOS().startsWith("V")) {
             // heuristique de lemmatisation des verbes
         }
         else if (getPOS()...) {
            /* traitement de tous les
               autres POS connus */
         }
         else {
            throw new Exception (
                 "Unexpected POS: " + getPOS()
                 );
         }
      }
   }
}
```

```
Token myToken = sent.getToken(i);
myToken.relemmatize();
```

Lancer une exception

```
public class Token
{
   public void relemmatize () throws Exception
   {
      if (lemma.equals("_")) {
         if (getPOS().equals("ADV"))
             setLemma(getWordForm());
         else if (getPOS().equals("NC")) {
             if (getWordForm().endsWith("s"))
                 setLemma(
                     getWordForm().substring(0, getWordForm().length()- 1)
                 );
             else {
                 // autre heuristique de lemmatistion
             }
         }
         else if (getPOS().startsWith("V")) {
             // heuristique de lemmatisation des verbes
         }
         else if (getPOS()...) {
             /* traitement de tous les
                autres POS connus */
         }
         else {
             throw new Exception (
                 "Unexpected POS: " + getPOS()
                 );
         }
      }
   }
}
```

```
Token myToken = sent.getToken(i);

try
{
   myToken.relemmatize();
}
catch (Exception e)
{
   System.out.println(e.getMessage ());
   // traitement eventuel de l'erreur
}
```

Intercepter les exceptions. . . ou pas

```java
public class DependenciesBuilder
{
   public void lookForNmodAdj ()
   {
      for (int i = 0; i < sentence.getNbToken(); i++)
      {
         Token depToken = sentence.getToken(i);

         try
         {
             depToken.relemmatize();
         }
         catch (Exception e)
         {
             // tant pis, pas grave
         }

         if (depToken.getPOS().equals("ADJ")
                 && depToken.getDepLabel().equals("mod"))
         {
             Token govToken = sentence.getToken(
                                 depToken.getGovIndex() - 1);
         }
      }
   }
}
```

```java
public class Token
{
   public void relemmatize ()
                    throws Exception
   {

   }
}
```

```java
DependenciesBuilder depBuilder
            = new DependenciesBuilder();
TalismaneReader2 reader
            = new TalismaneReader2("/tmp/smallCorpus.tal");

while (reader.hasMoreSentence())
{
    Sentence sent = reader.getSentence();
    depBuilder.setSentence (sent);
    depBuilder.lookForNmodAdj();
}
```

Intercepter les exceptions. . . ou pas

```
public class DependenciesBuilder
{
   public void lookForNmodAdj ()
   {
      for (int i = 0; i < sentence.getNbToken(); i++)
      {
          Token depToken = sentence.getToken(i);



              depToken.relemmatize();









          if (depToken.getPOS().equals("ADJ")
                  && depToken.getDepLabel().equals("mod"))
          {
              Token govToken = sentence.getToken(
                                  depToken.getGovIndex() - 1
                                  );
          }
      }
   }
}
```

```
public class Token
{
   public void relemmatize ()
                  throws Exception
   {

   }
}
```

```
DependenciesBuilder depBuilder
            = new DependenciesBuilder();
TalismaneReader2 reader
            = new TalismaneReader2("/tmp/smallCorpus.tal");

while (reader.hasMoreSentence())
{
    Sentence sent = reader.getSentence();
    depBuilder.setSentence (sent);
    depBuilder.lookForNmodAdj();
}
```

Intercepter les exceptions. . . ou pas

```java
public class DependenciesBuilder
{
   public void lookForNmodAdj () throws Exception
   {
      for (int i = 0; i < sentence.getNbToken(); i++)
      {
          Token depToken = sentence.getToken(i);



              depToken.relemmatize();









          if (depToken.getPOS().equals("ADJ")
                  && depToken.getDepLabel().equals("mod"))
          {
              Token govToken = sentence.getToken(
                                  depToken.getGovIndex() - 1
                                  );
          }
      }
   }
}
```

```java
public class Token
{
   public void relemmatize ()
                    throws Exception
   {

   }
}
```

```java
DependenciesBuilder depBuilder
            = new DependenciesBuilder();
TalismaneReader2 reader
            = new TalismaneReader2("/tmp/smallCorpus.tal");

while (reader.hasMoreSentence())
{
    Sentence sent = reader.getSentence();
    depBuilder.setSentence (sent);
    depBuilder.lookForNmodAdj();
}
```

Intercepter les exceptions. . . ou pas

```java
public class DependenciesBuilder
{
   public void lookForNmodAdj () throws Exception
   {
      for (int i = 0; i < sentence.getNbToken(); i++)
      {
          Token depToken = sentence.getToken(i);



              depToken.relemmatize();









          if (depToken.getPOS().equals("ADJ")
                  && depToken.getDepLabel().equals("mod"))
          {
              Token govToken = sentence.getToken(
                                  depToken.getGovIndex() - 1);
          }
      }
   }
}
```

```java
public class Token
{
   public void relemmatize ()
                  throws Exception
   {

   }
}
```

```java
DependenciesBuilder depBuilder
            = new DependenciesBuilder();
TalismaneReader2 reader
            = new TalismaneReader2("/tmp/smallCorpus.tal");

while (reader.hasMoreSentence())
{
    Sentence sent = reader.getSentence();
    depBuilder.setSentence (sent);
    try
    {
        depBuilder.lookForNmodAdj();
    }
    catch (Exception e)
    {
        // TODO: handle exception
    }
}
```

Créer votre propre classe d'exception

```
public class UnexpectedPOSException extends Exception {
   public UnexpectedPOSException () {
       super();
   }

   public UnexpectedPOSException (String message) {
       super(message);
   }
}
```

```
try
{
   // ...
}
catch (UnexpectedPOSException unkPOS)
{
   System.out.println("POS inconnu");
}
catch (Exception e)
{
   // traitement d'autres exceptions
}
```

Créer votre propre classe d'exception

```
public class UnexpectedPOSException extends Exception {
   private String pos;
   private Token token;

   public UnexpectedPOSException () {
       super();
   }

   public UnexpectedPOSException (String message) {
       super(message);
   }

   public UnexpectedPOSException (String message, String pos) {
       super(message);
       this.pos = pos;
   }

   public UnexpectedPOSException (String message, String pos, Token token) {
       this(message, pos);
       this.token = token;
   }
   public String getPos(){
       return pos;
   }
   public Token getToken(){
       return token;
   }
}
```

```
try
{
   // ...
}
catch (UnexpectedPOSException unkPOS)
{
   System.out.println("POS inconnu");
}
catch (Exception e)
{
   // traitement d'autres exceptions
}
```

Créer votre propre classe d'exception

```java
public class UnexpectedPOSException extends Exception {
   private String pos;
   private Token token;

   public UnexpectedPOSException () {
       super();
   }

   public UnexpectedPOSException (String message) {
       super(message);
   }

   public UnexpectedPOSException (String message, String pos) {
       super(message);
       this.pos = pos;
   }

   public UnexpectedPOSException (String message, String pos, Token token) {
       this(message, pos);
       this.token = token;
   }
   public String getPos(){
       return pos;
   }
   public Token getToken(){
       return token;
   }
}
```

```java
try
{
   // ...
}
catch (UnexpectedPOSException unkPOS)
{
   System.out.println("POS inconnu : " + unkPOS.getPos());
   Token tok = unkPOS.getToken();
   tok.setLemma(tok.getWordForm());
}
catch (Exception e)
{
   // traitement d'autres exceptions
}
```

Tableaux associatifs

Définition

ensemble de paires <clé, valeur>.
En Java : everything is object! (always!)
Les clés et les valeurs sont donc des objets !

Une implémentation parmi d'autres : la classe `java.util.TreeMap` ($\rightarrow$ voir la doc !)

Hachage

Attention, l'utilisation du terme *hachage* pour désigner un tableau associatif est un abus de langage... que l'on peut s'autoriser (si c'est en connaissance de cause)

Tableaux associatifs : exemples

```
TreeMap miscHash = new TreeMap();

miscHash.put("maCle", "maValeur");
miscHash.put("maCle2", new Integer (123));
miscHash.put("maCle3", new PointFixe (3, 5));
miscHash.put("maCle4", new Double (3.14157));

Object obj = miscHash.get("maCle4");
System.out.println(obj.toString());
System.out.println(((Double) obj).intValue());
```

Tableaux associatifs : exemples

```
TreeMap<Object, Object> miscHash = new TreeMap<Object, Object>();

miscHash.put("maCle", "maValeur");
miscHash.put("maCle2", new Integer (123));
miscHash.put("maCle3", new PointFixe (3, 5));
miscHash.put("maCle4", new Double (3.14157));

Object obj = miscHash.get("maCle4");
System.out.println(obj.toString());
System.out.println(((Double) obj).intValue());
```

Tableaux associatifs : exemples

```
TreeMap<Object, Object> miscHash = new TreeMap<Object, Object>();

miscHash.put("maCle", "maValeur");
miscHash.put("maCle2", new Integer (123));
miscHash.put("maCle3", new PointFixe (3, 5));
miscHash.put("maCle4", new Double (3.14157));

Object obj = miscHash.get("maCle4");
System.out.println(obj.toString());
System.out.println(((Double) obj).intValue());
```

Les clés et les valeurs sont des objets...

Les valeurs peuvent avoir des types hétérogènes.

Les clés doivent être du même type et implémenter l'interface *Comparable*

(cette affirmation est une simplification dont nous nous contenterons)

Tableaux associatifs : exemples

Typage

Cf. `List liste = new ArrayList ();`

$\Leftrightarrow$ `List<Object> liste = new ArrayList<Object> ();`

vs. `List<Type> liste = new ArrayList<Type> ();`

```
TreeMap<String, String> lemmas = new TreeMap<String, String> ();

lemmas.put("coda", "coder");
lemmas.put("coderiez", "coder");
lemmas.put("coderont", "coder");
// ...
lemmas.put("recodes", "recoder");
lemmas.put("recodent", "recoder");

System.out.println(
            lemmas.get ("coderont").toUpperCase()
        );
```

Tableaux associatifs : exemples

Typage

Cf. `List liste = new ArrayList ();`

⇔ `List<Object> liste = new ArrayList<Object> ();`

vs. `List<Type> liste = new ArrayList<Type> ();`

```
TreeMap<String, String> lemmas = new TreeMap<String, String> ();

lemmas.put("coda", "coder");
lemmas.put("coderiez", "coder");
lemmas.put("coderont", "coder");
// ...
lemmas.put("recodes", "recoder");
lemmas.put("recodent", "recoder");

System.out.println(
            lemmas.get ("coderont").toUpperCase()
        );
```

Attention !

Est-on sûr que *coderont* est bien une clé du tableau associatif ?
Sinon, que se passe-t-il ?

Tableaux associatifs : exemples

```
TreeMap<String, Integer> frequencies = new TreeMap<String, Integer>();

Integer currentFreq = frequencies.get("mot");
currentFreq++;
frequencies.put("mot", currentFreq);
```

Tableaux associatifs : exemples

```
TreeMap<String, Integer> frequencies = new TreeMap<String, Integer>();

Integer currentFreq = frequencies.get("mot");
currentFreq++;
frequencies.put("mot", currentFreq);
```

NullPointerException

Tableaux associatifs : exemples

```
TreeMap<String, Integer> frequencies = new TreeMap<String, Integer> ();


Integer currentFreq = frequencies.get("mot");

if (currentFreq == null)
    currentFreq = new Integer (1);
else
    currentFreq++;

frequencies.put("mot", currentFreq);
```

Tableaux associatifs : parcours

Principes

Voir dans l'API les méthodes `keySet()`, `values()` et la classe `Set`.
`keySet()` renvoie l'ensemble (`java.util.Set`) des clés.
Comme `List`, `Set` implémente la méthode `iterator()` qui renvoie une instance de `Iterator<Type>` (avec Type = type des clés du tableau associatif).

```
TreeMap<String, Integer> frequencies = new TreeMap<String, Integer>();

frequencies.put("mot1", 123);
frequencies.put("mot2", 456);

Iterator<String> it = frequencies.keySet().iterator();

while (it.hasNext())
{
    String mot = it.next();
    Integer freq = frequencies.get(mot);
    System.out.println("Frequence de " + mot
                                        + " = " + freq.toString());
}
```

Tableaux associatifs : parcours

Principes

Voir dans l'API les méthodes `keySet()`, `values()` et la classe `Set`.
`keySet()` renvoie l'ensemble (`java.util.Set`) des clés.
Comme `List`, `Set` implémente la méthode `iterator()` qui renvoie une instance de `Iterator<Type>` (avec Type = type des clés du tableau associatif).

```
TreeMap<String, Integer> frequencies = new TreeMap<String, Integer>();

frequencies.put("mot1", 123);
frequencies.put("mot2", 456);

Iterator<String> it = frequencies.keySet().iterator();

while (it.hasNext())
{
    String mot = it.next();
    Integer freq = frequencies.get(mot);
    System.out.println("Frequence de " + mot
                                        + " = " + freq.toString());
}
```

Ne faites pas de présupposition sur l'ordre d'itération sur les clés !