2017

Informatique - Aide TP2
Exceptions. Fichiers textes.

But du TP : Lire et écrire dans des fichiers textes. Utiliser les exceptions. Utiliser la classe StringTokenizer.

1 Les exceptions

Une *exception* se produit lorsque il y a une erreur telle qu'une division par zéro, une tentative de lecture d'un fichier qui n'existe pas ... Le traitement des exceptions est destiné à éviter un arrêt du programme et à permettre à l'utilisateur de corriger son erreur.

En Java une exception est une classe qui est instanciée lorsqu'une erreur s'est produite. Il existe une classe `Exception` dont dérivent toutes les autres exceptions parmi lesquelles on peut citer `IOException`, `ArrayIndexOutOfBoundsException`, ...

Si l'appel d'une méthode lance une exception on peut traiter cette exception, ou la relancer pour qu'elle soit traitée à un niveau supérieur.

Traitement d'une exception

Voici la version la plus simple du traitement d'une exception.

```
try {
  < instructions >
}
catch ( Exception e ){
  e.printStackTrace() <ou autre message d'erreur>
}
```

Les instructions sont protégées. En cas d'erreur, le traitement est transféré au bloc catch.

Relancer une exception

Il suffit d'indiquer dans l'en-tête de la méthode, que l'exception est relancée.

```
public void lire() twrows Exception {
 < instructions >
}
```

Création d'une exception

Si on a écrit un constructeur de la classe *tableau* lisant un ensemble de vecteurs, voici comment on peut écrire un constructeur pour la classe *Matrice*, qui va contrôler que tous les vecteurs ont même nombre d'éléments.

```
public Matrice(String nf) throws Exception {
  super(nf) ; // appel au constructeur ancêtre
  if (! testColonnesEgales() ){
    throw ( new Exception(nf+" n'est pas une matrice !"));
  }
}
```

2 Fichiers textes

Attention : pour utiliser les fonctions qui suivent, il faut importer des classes avec la clause :

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

Lecture dans un fichier texte

Voici une façon simple de lire dans un fichier texte, ligne par ligne, en utilisant un buffer qui limite les accès disque.

```
String line ;
FileReader fr=new FileReader("fichier.dat");
BufferedReader bf=new BufferedReader(fi);
String line=bf.readLine();
while (line!=null){
   <traitement de la ligne i du fichier>
}
bf.close();fr.close();
```

Attention toutes les fonctions de lecture/écriture sur disque lancent des exceptions ; il faut donc soit les traiter (try{ } catch() { }), ou bien les relancer (throws).

Ecriture dans un fichier texte

La démarche est semblable à la lecture.

```
FileWriter fich =new FileWriter(nfw);
BufferedWriter bufwrite = new BufferedWriter(fich);
...
bufwrite.write( ... ); // écriture dans le fichier
...
bufwrite.close(); fich.close();
```

3 La classe StringTokenizer

Attention à la clause import :

```
import java.util.StringTokenizer ;
```

Soit la chaîne `s="1.2 2.3 3.4"`, contenant des nombres séparés par des espaces. La classe *StringTokenizer*, sert à découper une telle chaîne en éléments (tokens), qui pourront être ensuite interprétés comme des réels.

Les principales méthodes sont :

```
int countTokens() // compte le nombre d'éléments
String nextToken() // donne l'élément suivant
boolean hasMoreTokens() // vrai si on n'est pas à la fin
```

Par exemple :

```
s="1.2 2.3 3.4" ;
StringTokenizer st=new StringTokenizer(s);
while (st.hasMoreTokens()){
  System.out.println(st.nextToken());
}
```

Les séparateurs par défaut sont les espaces, les tabulations, les fins de ligne. Il est possible de préciser d'autres séparateurs tels que le point-virgule, lors de l'appel du constructeur (voir doc. Java).