FICHE 7

TRAITEMENT DES EXCEPTIONS

(Fiches Java)

1 traiter une exception qui a été déclanchée

Lorsque qu'on programme une méthode A qui fait appel à une méthode B pouvant lancer une exception on a deux façons de faire. Soit on ne teste pas cette exception dans le corps de A. On la laisse passer pour être traitée par une autre méthode et donc A propage le lancement de cette exception. Ou bien on la traite dans le corps de la méthode A en écrivant le code à appliquer si cette exception se déclanche.

1.1 On la laisse passer

La méthode A doit alors avoir dans sa déclaration la mention

throws Exception
Par exemple dans la méthode main de la classe suivante remarquez la mention **throws FonctionException**

```
public class Test15{
  public static void main(String[] args)
  throws FonctionException{
     char a,b;
     for (a='A'; a<= 'Z';a++){
       for (b='A'; b <='Z'; b++){

         String S1= String.valueOf(a);
 String S2= String.valueOf(b);

         String S="ESS"+S1+S2+"(x)=x+3";
 System.out.println(S);
         Fonction F=new Fonction(S);
```

```
      }
    }

    String S="f(x)=x+3*x";
  }
}
```

Si toutes les méthodes appelantes laissent passer, c'est l'interprête qui finira par la traiter.

1.2 On réagit

Pour réagir il faut englober l'appel de la méthode B dans un boc d'instruction "try catch finally".

```
import java.io.*;

class FahrToCelsius  {

  public static void main (String args[]) {

    double fahr, celsius;
    double lower, upper, step;

    lower = 0.0;    // lower limit of temperature table
    upper = 300.0;  // upper limit of temperature table
    step  = 20.0;   // step size

    fahr = lower;

    try {

      FileOutputStream fout =  new FileOutputStream("test.out");

      // now to the FileOutputStream into a PrintStream
      PrintStream myOutput = new PrintStream(fout);

      while (fahr <= upper) {  // while loop begins here
        celsius = 5.0 * (fahr-32.0) / 9.0;
        myOutput.println(fahr + " " + celsius);
```

```
        fahr = fahr + step;
      } // while loop ends here

    }  // try ends here
    catch (IOException e) {
      System.out.println("Error: " + e);
      System.exit(1);
    }

  } // main ends here

}
```

2 Programmer ses propres exceptions

Voici une classe qui définit des exceptions liées à un analyseur syntaxique d'expressions fonctionnelles.

```
package rr.modulog.syntaxe;

public class FonctionException extends Exception {

  private static String[] tabledesmessages = {
  "Expression correcte",
  "Il manque une parenthèse ouvrante",
  "Il manque une parenthèse fermante",
  "Un nom de variable est incorrect",
  "Il y a des caractères en trop",
  "La chaîne d'entrée est vide",
  "Il y a un nombre non valide",
  "Le nombre de variables ne correspond pas à la définition",
  "Le nom de la fonction existe déjà dans la table",
  "Il manque un signe = ",
  "Un nom de variable est déjà utilisé",
  "Un nom de variable est incorrect",
  "Le nom de la fonction est incorrect",
  "La table de fonctions est pleine",
  "La fonction est utilisée"
  };
```

```
  public FonctionException(int numero) {
    super(tabledesmessages[numero]);
  }
}
```

Et voici une méthode qui lance des exceptions lors d'une analyse.

```
  private static void NonV1(String S)
  throws FonctionException{
    int vanum=0;
    if ((S.charAt(Ind+1) < 'A') || (S.charAt(Ind+1) > 'Z')) {
      throw new FonctionException(11);
    }
    else {
      incrementeI();
      char C=S.charAt(Ind);
      vanum=vanum+1;
      if (getvarlist(C)!=0){
        throw new FonctionException(10);
      }
      else{
        setvarlist(C,vanum);
        while (S.charAt(Ind+1)==','){
          incrementeI();
          if ((S.charAt(Ind+1) < 'A') || (S.charAt(Ind+1) > 'Z')) {
            throw new FonctionException(11);
          }
          else{
            incrementeI();
            C=S.charAt(Ind);
            vanum=vanum+1;
            if (getvarlist(C)!=0){
               throw new FonctionException(10);
            }
            else{
              setvarlist(C,vanum);
            }
          }
        }
        Fct_Nb_Var=vanum;
        if (S.charAt(Ind +1)!=')'){
```

```
        throw new FonctionException(2);
      }
      else{
        incrementeI();
      }
    }
  }
}
```

3 Remarque sur les fonctions arithmétiques

Les fonctions arithmétiques ne lancent pas d'exceptions. Une division par 0 par exemple ne provoque pas d'exception. Les résultats des opérations sont conformes à la norme IEEE-754. Les résultats peuvent être par exemple NAN (not a number) ou $+\infty$ ou $-\infty$.