JAVA

EXERCICES

Travail sur les exceptions

A. La division par zéro

La division par zéro crée une *ArithmeticException*.

1. *Créez une classe d'objet **MonEntier** qui contient une variable **monEntier** de type **int**. Écrivez le constructeur de cette classe.*
2. *Munissez cette classe de la méthode division. Cette méthode retourne le résultat de la division de la variable **monEntier** d'un objet **MonEntier** par la variable **monEntier** d'un autre objet **MonEntier** appelé diviseur. L'objet diviseur sera passé en paramètre d'entrée de la méthode division.*
3. *Testez votre méthode division sur deux entiers quelconques. Recommencez avec un diviseur égal à zéro. Que se passe-t-il ?*
4. *Dans la méthode division: testez le bloc qui effectue l'opération. Si une **ArithmeticException** est générée, affichez le message "Division impossible".*

N.B: Vous pouvez également afficher le message de l'exception en utilisant la méthode getMessage() :

```
System.out.println(variableException.getMessage());
```

B. Factorielle

1. Compilez et exécutez le programme Factorielle.java suivant en essayant successivement:

- de ne pas faire figurer sur la ligne de commande le paramètre attendu
- de mettre un paramètre non-entier
- de mettre un paramètre entier négatif
- de mettre l'entier 20

Dans les deux premiers cas, une exception est signalée. Dans les deux derniers cas le résultat est faux.

```
class Factorielle {

  public static void main (String[] args) {

     int i, nbEntiers=0, factorielle=1;
     int ancien;

     nbEntiers= Integer.parseInt(args[0]);
     for (i=2;i<= nbEntiers;i++) {
       ancien=factorielle;
       factorielle *=i;
     }
     System.out.println(" Voila la factorielle des "+ nbEntiers +
                        " premiers entiers : "+ factorielle );
    }
 }
```

2. Vous devez modifier le programme pour que, dans chacun des cas, l'erreur soit précisée à l'utilisateur.

Dans le premier cas, on souhaite que le programme affiche par exemple:

```
Indiquez le nombre d'entiers sur la ligne de commande
```

et si le paramètre indiqué est -4:

```
-4 est négatif: la factorielle d'un nombre négatif n'est pas définie
```

3. Dans les deux premiers cas, utilisez les blocs catch pour afficher le message d'erreur.

C. Une classe *ExpressionParenthesee*

On cherche, au cours de cet exercice, à vérifier si une expression est correctement parenthésée. Voici quelques exemples :

((a+b)-(c+d)) correct

a+(b+(c+d)) correct

((a+b)+c incorrect

((a+b)+c)) incorrect

)a+b(incorrect

1. *Écrire une classe* **ExpressionParenthesee** *qui est composée d'une String pour mémoriser l'expression. Ecrire un constructeur pour lequel on passe une chaîne de caractères en paramètre.*
2. *Encapsuler la* **String** *en privé et écrire une méthode* **String getString();**

3 cas peuvent se présenter :

a) L'expression est correcte

b) il y a des parenthèses ouvrantes en trop

c) il y a des parenthèses fermantes en trop

On va chercher à lever des exceptions en cas d'erreur. Ecrire 3 classes d'exceptions :

d) **ParentheseException** héritant de la classe Exception

e) **ParentheseOuvranteException** héritant de la classe **ParentheseException**

f) **ParentheseFermanteException** héritant de la classe **ParentheseException**

3. *Dans le constructeur de la classe* **ExpressionParenthesee***, écrire un algorithme qui vérifie si l'expression est correctement parenthésée ou qui déclenche une exception appropriée en cas d'erreur.*

4. *Ecrire une méthode main permettant de vérifier le fonctionnement de votre classe.*