

## TP\_TD N° 5 : Gestion des exceptions

### Exercice1 :

Classe Opérations	Programme principale
<pre> <b>public final class</b> Opérations { <b>public static double</b> division ( <b>double</b> p, <b>double</b> q) {     <b>return</b> p / q; } <b>public static double</b> somme( <b>double</b> p, <b>double</b> q) {     <b>return</b> p+q; } } </pre>	<pre> <b>import</b> java.util.Scanner; <b>public class</b> Test {     <b>public static void</b> main(String[] args)     {         <b>double</b> a, b, res;         Scanner <b>clavier</b> = <b>new</b> Scanner(System.<b>in</b>);         System.<b>out</b>.println("Enter le premier opérande");         a = <b>clavier</b>.nextDouble();         System.<b>out</b>.println("Enter le deuxième opérande");         b = <b>clavier</b>.nextDouble();         res = Opérations.<b>division</b>(a, b);         System.<b>out</b>.println("le résultat de " + a + " divisé par " + b + " est " + res);         System.<b>out</b>.println("Fin du programme"); }     } } </pre>

### Questions :

- Créer un nouveau projet Java nommé GestionExceptions.
- Dedans ce projet, créer un package nommé Exceptions
- Dedans ce package, créer deux classes nommées respectivement « Operations » et « Test » comme mentionné ci-dessus.
- Quels sont les types d'exceptions qu'on peut rencontrer dans ce programme ?  
.....  
.....
- Essayer de tester l'exécution de ce programme avec des exemples qui représentent des exceptions, puis noter ce que vous avez remarqué.  
.....  
.....
- Cette fois, on vous demande de lever et traiter ces exceptions pour l'objectif de les rendre plus informatives et aussi pour éviter l'arrêt immédiat de l'exécution du programme principale.

### Exercice2 :

- Créer un nouveau projet Java nommé EntiersNaturels.
- Dedans ce projet, créer un package nommé EntNaturel.
- Réaliser une classe *EntNat* permettant de manipuler des entiers naturels (positifs ou nuls). Cette classe disposera tout simplement :

- d'un constructeur à un argument de type *int* qui générera une exception personnalisée de type *ErrorConst* lorsque la valeur reçue de son argument est négative.
- d'une méthode *getN* fournissant sous forme d'un *int*, la valeur encapsulée dans un objet de type *EntNat*.
- *ErrorConst* est une classe à définir avec un champ *valeur* destiné à conserver la valeur avec laquelle on a tenté de construire à tort un entier naturel.
- Écrire un petit programme d'utilisation qui traite l'exception *ErrorConst* en affichant un message et en interrompant l'exécution.

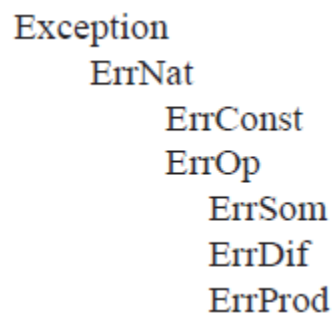
### **Exercice 3 :**

Reprendre l'exercice précédent, puis ajouter à la classe *EntNat* :

- de méthodes statiques de somme, de différence et de produit de deux naturels (de type *EntNat*); elles généreront respectivement des exceptions *ErrSom*, *ErrDiff* et *ErrProd* lorsque le résultat ne sera pas représentable ou un résultat de type *EntNat* dans le cas contraire; la limite des valeurs des naturels sera fixée à la plus grande valeur du type *int*;

**N.B:** On s'arrangera pour que toutes les classes exception dérivent d'une classe *ErrNat* et pour qu'elles permettent à un éventuel gestionnaire de récupérer les valeurs ayant provoqué l'exception.

La hiérarchie des classes d'exception se présentera comme suit :



La classe *ErrOp* servira de base aux exceptions liées à des opérations arithmétiques (somme, différence ou produit) ; elle possèdera deux champs de type *int* représentant les valeurs des deux opérandes de l'opération.

Écrire deux exemples d'utilisation de la classe :

- l'un se contentant d'intercepter sans discernement les exceptions de type dérivé de *ErrNat*.
  - l'autre qui explicite la nature de l'exception en affichant les informations disponibles.
- Les deux exemples pourront figurer dans deux blocs *try* d'un même programme.