TP_TD N° 5 : Gestion des exceptions

Exercice1 :

Classe Opérations	Programme principale

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

Questions :

1. Créer un nouveau projet Java nommé GestionExceptions.
2. Dedans ce projet, créer un package nommé Exceptions
3. Dedans ce package, créer deux classes nommées respectivement « Operations » et « Test » comme mentionné ci-dessus.
4. Quels sont les types d'exceptions qu'on peut rencontrer dans ce programme ?
   ……
   ……
5. Essayer de tester l'exécution de ce programme avec des exemples qui représentent des exceptions, puis noter ce que vous avez remarqué.
   ……
   ……
6. 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 un type 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.

Exrecice 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 :

Exception
ErrNat
ErrConst
ErrOp
ErrSom
ErrDif
ErrProd

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.