TD3: Exceptions

Exercice 1. Exécution de code

Que fournit l'exécution du code suivant :

```
class Erreur extends Exception { public int num; }
class Erreur_d extends Erreur { public int code; }
class A
{ public A(int n) throws Erreur_d, Erreur
        { if (n==1) {
                Erreur_d e = new Erreur_d();
                e.num = 999; e.code = 12;
                throw e ;
            }
        }
}
public class Chemin1
{
    public static void main (String args[])
        { try{
                A a = new A(1) ;
                System.out.println ("apres creation a(1)");
            }
            catch (Erreur e){
                System.out.println ("** exception Erreur " + e.num);
            }
            System.out.println ("suite main") ;
            try{
                A b = new A(1);
                System.out.println ("apres creation b(1)");
            }
            catch (Erreur_d e){
                System.out.println ("** exception Erreur_d "+e.num+" "+e.code);
            }
            catch (Erreur e){
                System.out.println ("** exception Erreur " + e.num);
            }
        }
}
```

a. Quels résultats fournit ce programme ?

b. Que se passe-t-il

1. si l'on permute l'ordre des deux gestionnaires dans le second bloc try ?
2. si on remplace le throws Erreur_d, Erreur par throws Erreur_d dans le constructeur de A ?

Exercice 2. Retour sur les structures algébriques

Dans la continuité de l'exercice du TD2 sur les structures algébriques de $\mathbb{Z}$ et $\mathbb{Z}/p\mathbb{Z}$:

a. Écrire une classe représentant l'ensemble $\mathbb{Z}/p\mathbb{Z}$ comme un corps. Le constructeur devra en outre vérifier que le nombre p qui lui est donné en paramètre est choisi convenablement, et lever une exception si besoin.

b. Quels types d'exceptions pensez-vous pertinent de définir pour gérer les différentes erreures d'exécution possibles ?

Exercice 3. Classe d'entiers naturels

a. Réaliser une classe permettant de manipuler des entiers naturels (positifs ou nuls) et disposant :
- d'un constructeur à un argument de type int ; il générera une exception ErrConst si la valeur de son argument est négative ;
- de méthodes statiques de somme, de différence et de produit de deux naturels ; elles généreront respectivement des exceptions ErrSom, ErrDiff et ErrProd lorsque le résultat ne sera pas représentable ; la limite des valeurs des naturels sera fixée à la plus grande valeur du type int ;
- une méthode d'accès getN fournissant sous forme d'un int la valeur de l'entier naturel.

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.

b. 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.