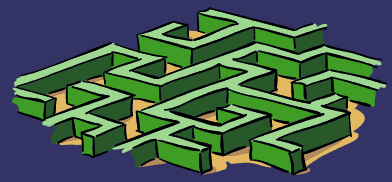
Déclenchement et traitement d'une exception.

Réaliser une classe EntNat permettant de manipuler des entiers naturels (positifs ou nuls).

Pour l'instant, cette classe disposera simplement :

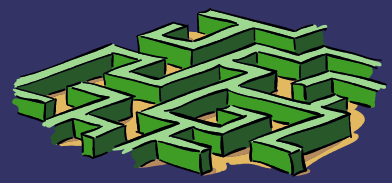
- D'un constructeur à un argument de type int qui générera une exception de type ErrConst (type de classe à définir) lorsque la valeur reçue ne conviendra pas,
- D'une méthode getN fournissant sous forme d'un int, la valeure encapsuléedans un objet de type EntNat.

Écrire un petit programme d'utilisation qui traite l'exception ErrConst en affichant un message et en interrompant l'exécution.



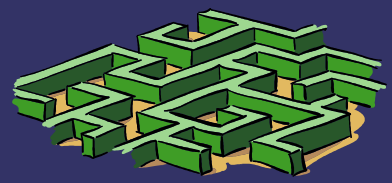
Solution : class ErrConst.

```
Calss EntNat
        {
        private int n;
        public EntNat (int n) throws ErrConst
              {
              if (n<0) throw newErrConst() ;
              this.n = n;
              }
        public int getN() {return n;)
        }
class ErrConst extends Exception
        {}
```



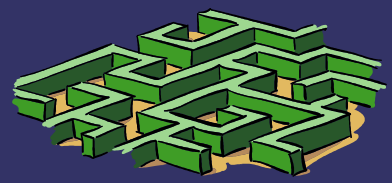
Solution : Programme d'utilisation

```
Public class TesEntNat
        {
        public static void main {string args[])
                {
                try
                        {
                        EntNat n1 = new EntNat (20);
                        system.out.println("n1 = " + n1.getN( )) ;
                        EntNat n2 = new EntNat (-12);
                        system.out.println("n2 = " + n2.getN( )) ;
                        }
                        catch (ErrConst e)
                                {
                                system.out.println("***erreur construction *** ");
                                system.exit (-1) ;
                                }
                        }
                }
```



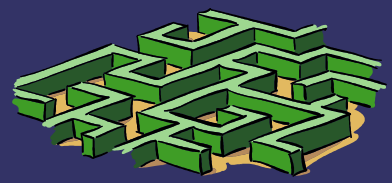
Transmission d'informations au gestionnaire.

Adapter EntNat de l'exercice precedant et le programme d'utilisation de maniére à disposer dans le gestionnaire d'exception du type ErrConst de la valeur fournie à tort au constructeur.



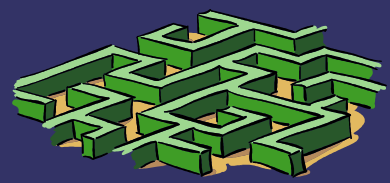
Solution : class ErrConst.

```
Calss EntNat
        {
        private int n;
        public EntNat (int n) throws ErrConst
              {
              if (n<0) throw newErrConst(n) ;
              this.n = n;
              }
        public int getN() {return n;)
        }
class ErrConst extends Exception
        {
        private int valeur;
        public ErrConst (int valeur) {this.valeur = valeur ;}
        public int getValeur {return valeur;}
        }
```



Solution : Programme d'utilisation

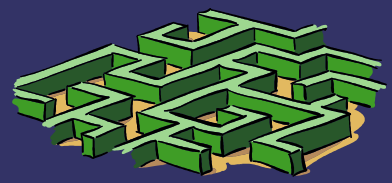
```
Public class TesEntNat
        {
        public static void main {string args[])
                {
                try
                        {
                        EntNat n1 = new EntNat (20);
                        system.out.println("n1 = " + n1.getN( )) ;
                        EntNat n2 = new EntNat (-12);
                        system.out.println("n2 = " + n2.getN( )) ;
                        }
                        catch (ErrConst e)
                                {
                                system.out.println("tentative de construction avec :  " + e.getValeur( ) );
                                system.exit (-1) ;
                                }
                        }
                }
```



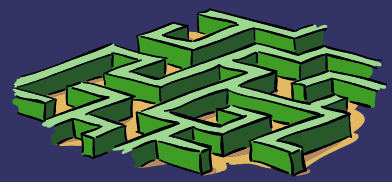
Cheminement d'exceptions

Que produit le programme suivant lorsqu'on lui fournit en donnée :

- La valeur 0,
- La valeur 1,
- La valeur 2,

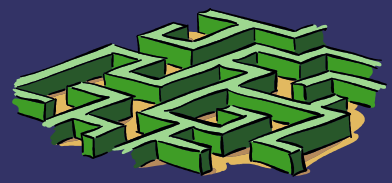


```
class Except extends Exception
        {
        private int n;
        public Except (int n) {this.n = n ;}
        }
public class chemin
        {public static void main {string args[])
                {int n;
                system.out.println("donnez un entier : ") ; n = Clavier.lireInt() ;
                try
                        {system.out.println("début du premier bloc try ") ;
                        if (n!=0) throw new Except (n);
                        system.out.println("fin du premier bloc try ") ;
                        }
                catch (Except e)
                        {system.out.println("catch 1 –  n = " + e.n) ; }
                system.out.println("suite du programme") ;
                try
                        {system.out.println("début du second bloc try ") ;
                        if (n!=1) throw new Except (n);
                        system.out.println("fin du second bloc try ") ;
                        }
                catch (Except e)
                        {system.out.println("catch 2 –  n = " + e.n) ; system.exit(-1); }
                system.out.println("fin du programme") ;
                }
        }
```



Bloc finally - Que fournit le programme suivant :

```
Class Except extends exception
{}
public class Finally
        {public static void f (int n)
                {try
                        {if(n!=1) throw new Except ( ) ;
                        }
                catch (Except e)
                        { System.out.println("catch dans f – n = " + n) ;
                                return ;
                        }
                finally
                        {system.out.println("dans finally -n = " +n) ;
                        }
                }
        }
public static void main (String args[ ])
        {
        f(1);
        f(2);
        }
```



Synthèse : entiers naturels.

Réaliser une classe permettant de manipuler ds entiers naturels (positifs ou nuls) et disposant :

- D'un constructeur à un argument de type int qui générera une exception de type ErrConst (type de classe à définir) lorsque la valeur de son argument est négative.
- D'une méthode statique de somme, de différence et de produit de deux naturels ; elles générerons 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'accés getN fournissant sous forme d'un int la valeur de l'entier naturel.

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