Notions abordées : compréhension du mécanisme des exceptions et mis en place d'exception dans des applications.

Exercice 1 – Les exceptions

Que fait le programme suivant ?

```
public class MonException extends Exception {
    public MonException(String message) {
        super(message);
    }
}

public class Generateur {
    private int val;
    public Generateur(int val) {
        this.val = val;
    }

    /**
     * la méthode uneMethode génère une exception de la classe MonException
     * si l'entier passé en paramètre est égal à la valeur de l'attribut val
     */

    public void uneMethode(int i) throws MonException {
        if(i==val)
            throw new MonException("Oups, l'exception est levee");
    }
}


public class ExempleSimple2 {

    public void premierTest(Generateur gen) {
        System.out.println("Premier test:");
        System.out.println("-------------");
        int v=42;
        try {
            System.out.println("un");
            gen.uneMethode(v);
            System.out.println("deux");
        } catch(MonException e) {
            System.out.println("trois " + e.getMessage());
        }
        System.out.println("quatre");
    }

    public void secondTest(Generateur gen) {
        System.out.println("\nSecond test:");
        System.out.println("------------");
        int v = 1;
        try {
            System.out.println("un");
            gen.uneMethode(v);
            System.out.println("deux");
```

```
        } catch(MonException e) {
            System.out.println("trois " + e.getMessage());
        }
        System.out.println("quatre");
    }


    public void troisiemeTest(Generateur gen) throws MonException {
        System.out.println("\nTroisieme test:");
        System.out.println("---------------");
        int v=42;
        System.out.println("un");
        gen.uneMethode(v);
        System.out.println("deux");
    }

    public void quatriemeTest(Generateur gen) throws MonException {
        System.out.println("\nQuatrieme test:");
        System.out.println("---------------");
        int v=2;
        System.out.println("un");
        gen.uneMethode(v);
        System.out.println("deux");
    }

    public void cinquiemeTest(Generateur gen) throws MonException {
        System.out.println("\nCinquieme test:");
        System.out.println("---------------");
        int v = 42;
        try {
            System.out.println("un");
            gen.uneMethode(v);
            System.out.println("deux");
        } catch(MonException e) {
            System.out.println("trois " + e.getMessage());
            throw e;
        }
        finally {
            System.out.println("trois bis");
        }
        System.out.println("quatre");
    }

    public void sixiemeTest(Generateur gen) throws MonException
    {
        System.out.println("\nSixieme test:");
        System.out.println("-------------");
        int  v = 2;
        try {
            System.out.println("un");
            gen.uneMethode(v);
            System.out.println("deux");
```

```
        } catch(MonException e) {
            System.out.println("trois " + e.getMessage());
            throw e;
        } finally {
            System.out.println("trois bis");
        }
        System.out.println("quatre");
    }

    public static void main(String[] args) throws MonException{
        Generateur gen = new Generateur(42);
        ExempleSimple2 exemple = new ExempleSimple2();
        exemple.premierTest(gen);
        exemple.secondTest(gen);
        try {
            exemple.troisiemeTest(gen);
        } catch(MonException e) {
            System.out.println("cinq " + e.getMessage());
        }
        try {
            exemple.quatriemeTest(gen);
        } catch(MonException e) {
            System.out.println("cinq " + e.getMessage());
        }
        try {
            exemple.cinquiemeTest(gen);
        } catch(MonException e) {
            System.out.println("cinq "+ e.getMessage());
        }
        try {
            exemple.sixiemeTest(gen);
        } catch(MonException e) {
            System.out.println("cinq "+ e.getMessage());
        }


        exemple.troisiemeTest(gen);



    }

}
```

Exercice 2

Le programme suivant demande à l'utilisateur de saisir au clavier un indice et un nombre. L'interface `Scanner` permet de récupérer dans le programme l'indice de la case dans un tableau d'entiers et un entier. Il effectue ensuite la division du contenu de la case désignée par l'indice par l'entier saisi.

```
import java.util.*;

class ModifTableau {
```

```
    public static void aff(int t[])
    {
        for (double d:t) System.out.print(d+" : ");
        System.out.println();
    }

    public static void main(String arg[]){
        double d[]= new int[]{3, 4, 5, 1, 7};

        aff(d);

        Scanner sc = new Scanner(System.in);
        System.out.println("Veuillez saisir l'indice de l'élément à
         ↪ traiter:");
        int indice = sc.nextInt();

        System.out.println("Veuillez saisir le diviseur :");
        int diviseur = sc.nextInt();
        d[indice]/=diviseur;

        aff(d);
    }
}
```

Beaucoup de problèmes peuvent se passer au cours de l'exécution de ce programme :

— L'indice saisi peut être en dehors des indices possibles pour le tableau.

— Les valeurs saisies au clavier peuvent être incorrectes.

— L'entier saisi peut être un zéro.

Modifiez le programme afin qu'il ne se termine que lorsque l'une opération correcte aura pu se faire. Vous utiliserez les exceptions :

— `ArithmeticException`

— `ArrayIndexOutOfBoundsException`

— `java.util.InputMismatchException`

Exercice 3

Le programme suivant effectue une division entre deux nombres de type primitif `double`.

```
public class Divzero {

    public static void main (String arg[]){
        Div a=new Div(1,2);
        System.out.println (a.division());

        Div b=new Div(1,0);
        System.out.println (b.division());

    }
}
```

```
class Div {
    double a,b;
    public Div (double a, double b){
        this.a = a;
        this.b = b;
    }

    public double division(){
        return a/b;
    }
}
```

L'exécution donne le résultat suivant :

```
$ java Divzero
0.5
Infinity
```

On souhaite lever une exception pour la division par zéro. Proposez une solution.

Exercice 4 – Exceptions dans les constructeurs

`Personne` est une classe avec deux attributs privées `String nom` et `int age`. Écrire un constructeur `public Personne (String n, int a)` qui propage des exceptions de type `IllegalArgumentException` lorsque le nom `n` est `null` ou lorsque l'age `a` est négatif. Utiliser ce constructeur dans une méthode `main` pour contrôler les appels `new Personne ("Charles", 4)` et `new Personne ("Charline", -11)` et `new Personne (null,11)`. Afficher les erreurs éventuelles lors de l'exécution du constructeur. L'exécution doit produire par exemple, l'affichage suivant :

```
Première personne
Voici : Charles a 4 ans.
Seconde personne
Création de personne ratée !! java.lang.IllegalArgumentException: age négatif !
```

ou

```
Première personne
Voici : Charles a 4 ans.
Seconde personne
Création de personne ratée !! java.lang.IllegalArgumentException: pas de nom !
```