Les exceptions

1. Compréhension de programmes

Exercice 1 — *Résultat d'exécution*

Supposons que `message2()` déclenche une exception dans le bloc `try-catch` suivant :

```
try{
  message1();
  message2();
  message3();
}
catch( Exception1 e1 )
{...}
catch( Exception2 e2 )
{...}
message4();
```

1. La méthode `message3()` sera-t-elle exécutée ?
2. Si l'exception est propagée, mais n'est pas traitée, `message4()` sera-t-il exécuté ?
3. Si l'exception est traitée, `message4()` sera-t-il exécuté ?

Exercice 2 — *Affichage de programme*

Qu'affiche le programme suivant :

```
public class Test{
  public static void main( String[] args ){
    try{
      m();
      System.out.println( "après_appel_de_méthode()" ); }
    catch( RuntimeException e ){
      System.out.println("opération_sur_entiers_erronée");}
    catch( Exception e ){
      System.out.println("opération_sur_rationnels_erronée");}
 }

  static void m() throws Exception{
    Rational r1 = new Rational( 4,5 );
    Rational r2 = new Rational( 0,2 );
    // public Rational divide( Rational r ) throws Exception si le dénominateur de r =0
    Rational x = r1.divide( r2 );
    int i = 0;
    int y = 2/i;
    System.out.println("fin_d'exécution_de_méthode()");
  }
}
```

Exercice 3 — *Affichage de programme*

Qu'affiche le programme suivant :

```
public class Test{
  public static void main( String[] args ){
    try{
      m();
      System.out.println( "après_appel_de_méthode()" );}
    catch( RuntimeException e ){
      System.out.println("opération_sur_entiers_erronée");}
    catch( Exception e ){
      System.out.println("opération_sur_rationnels_erronée");}
  }
  static void m() throws Exception{
    try{
      Rational r1 = new Rational( 4,5 );
      Rational r2 = new Rational( 0,2 );
      Rational x = r1.divide( r2 );
      int i = 0;
      int y = 2/i;
      System.out.println("fin_d'exécution_de_méthode()");}
    catch( RuntimeException e ){
      System.out.println("opération_sur_entiers_erronée");}
    catch( Exception e ){
      System.out.println("opération_sur_rationnels_erronée");}
}}
```

Exercice 4 — *Résultat d'exécution*

Supposons que `message2()` déclenche une exception dans le bloc `try-catch` suivant :

```
try{
        message1();
        message2();
        message3();
}
catch( Exception1 e1 ){...}
catch( Exception2 e2 ){...}
catch( Exception3 e3 ){...}
finally{message4();}
message 5();
```

1. La méthode `message5()` sera-t-elle exécutée si l'exception n'est pas traitée ?
2. Si l'exception est de type 3, `message4()` sera-t-il exécuté ?
3. Si l'exception est de type 3, `message5()` sera-t-il exécuté ?

2. Analyse et Conception Orientée Objet

Exercice 5 — *Les piles*

Reprendre le code de la classe `Pile` vue en tp :

```
public class Pile {

        private int[] pile;
        private int nbElements=0;

        public Pile(int taille) {
```

```
            pile=new int[taille];
        }

        public boolean estVide() {
            return(nbElements==0);
        }

        public void empiler(int element) {
            pile[nbElements]=element;
            nbElements++;
        }

        public int depiler() {
            nbElements--;
            return pile[nbElements];
        }

        public int hautPile() {
            return pile[nbElements-1];
        }

        public boolean estPleine() {
            return(nbElements==pile.length);
        }
}
```

1. Définir deux classes d'exceptions `PilePleine` et `pileVide`.
2. Intégrer les à la classe `Pile`.
3. Proposer une méthode `main` qui traite ces exceptions.