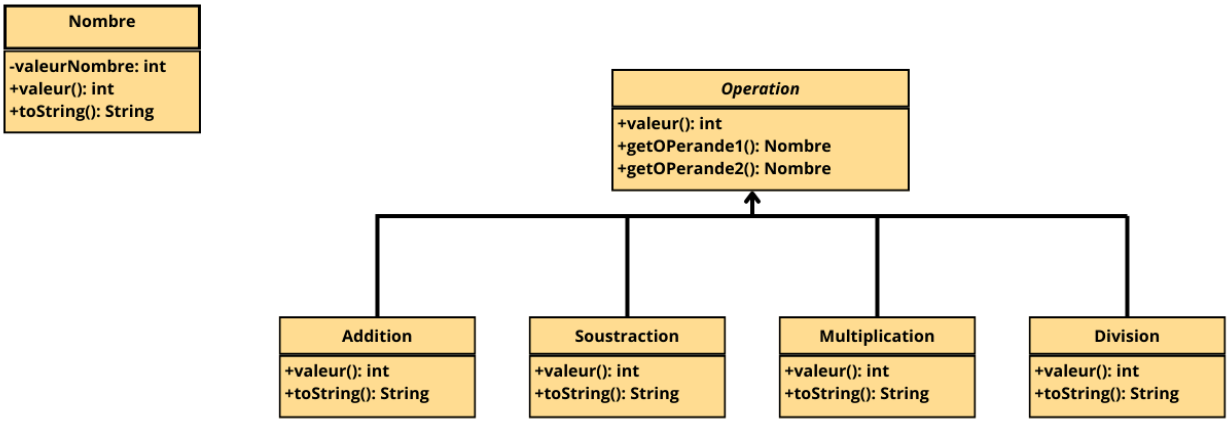
**PHAM Huy**
**MADRAZ Elies**

**Groupe : Zephyr**

# SAE R201 : Partie 1

## 1. UML



## 2. Java

Class Nombre :

```java
public class Nombre {
    private int valeurNombre;
    public Nombre(int valeur)
    {
        this.valeurNombre = valeur;
    }

    public int valeur()
    {
        return this.valeurNombre;
    }

    public String toString()
    {
        return "" + this.valeurNombre;
    }
```

Class Operation :

```java
public abstract class Operation {
    private Nombre operande1;
    private Nombre operande2;

    public Operation(Nombre operande1, Nombre operande2)
    {
        this.operande1 = operande1;
        this.operande2 = operande2;
    }

    public abstract int valeur();

    public Nombre getOperande1()
    {
        return this.operande1;
    }
    public Nombre getOperande2()
    {
        return this.operande2;
    }
}
```

Class Addition :

```java
public class Addition extends Operation {
    public Addition(Nombre operande1, Nombre operande2)
    {
        super(operande1, operande2);
    }

    public int valeur()
    {
        return getOperande1().valeur()+getOperande2().valeur();
    }

    public String toString()
    {
        return ""+getOperande1().valeur() + " + " + getOperande2().valeur();
    }
}
```

Class Soustraction :

```
1. public class Soustraction extends Operation {
2.     public Soustraction(Nombre operande1, Nombre operande2)
3.     {
4.         super(operande1, operande2);
5.     }
6.
7.     public int valeur()
8.     {
9.         return getOperande1().valeur()-getOperande2().valeur();
10.    }
11.
12.    public String toString()
13.    {
14.        return ""+getOperande1().valeur() + " - " + getOperande2().valeur();
15.    }
16. }
```

Class Multiplication :

```
1. public class Multiplication extends Operation {
2.     public Multiplication(Nombre operande1, Nombre operande2)
3.     {
4.         super(operande1, operande2);
5.     }
6.
7.     public int valeur()
8.     {
9.         return getOperande1().valeur()*getOperande2().valeur();
10.    }
11.
12.    public String toString()
13.    {
14.        return ""+getOperande1().valeur() + " * " + getOperande2().valeur();
15.    }
16. }
```

Class Division :

```
1. public class Division extends Operation {
2.     public Division(Nombre operande1, Nombre operande2) throws ArithmeticException
3.     {
4.         super(operande1, operande2);
5.         if (getOperande2().valeur() == 0)
6.             throw new ArithmeticException("Le dénominateur ne peut pas être zéro.");
7.     }
8.
9.     public int valeur()
10.    {
11.        return getOperande1().valeur()/getOperande2().valeur();
12.    }
13.
14.    public String toString()
15.    {
16.        return ""+getOperande1().valeur() + " / " + getOperande2().valeur();
17.    }
```

Class CalculatriceSimple :

```
1. public class CalculatriceSimple {
2.     public static void main(String[] args) {
3.         Nombre six = new Nombre(6);
4.         Nombre dix = new Nombre(10);
5.         Nombre zero = new Nombre(0);
6.         Nombre un = new Nombre(1);
7.         Operation s = new Soustraction(dix, six);
8.         System.out.println(s + " = " + s.valeur());
9.
10.        try {
11.            Operation div = new Division(six, zero);
12.                System.out.println(div + " = " +div.valeur());
13.        }
14.        catch(ArithmeticException e){
15.                System.out.println("il y a une erreur");
16.            }
17.
18.        Operation add= new  Addition(dix,un);
19.        Operation mul = new Multiplication(six,dix);
20.        System.out.println(add + " = " + add.valeur());
21.        System.out.println(mul + " = " + mul.valeur());
22.
23.    }
24. }
```