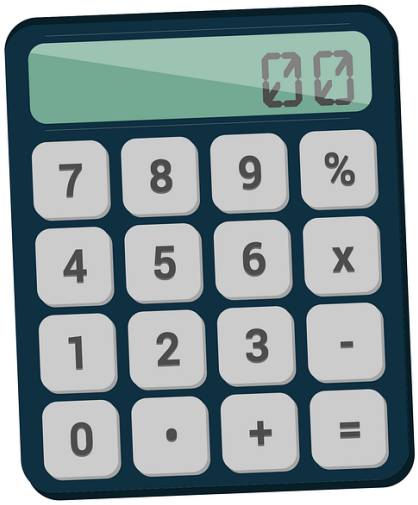
2021-2022

# SAE 2.01 – Développement d'une application

Enseignant : Mr Bouthinon

## Ma Calculatrice



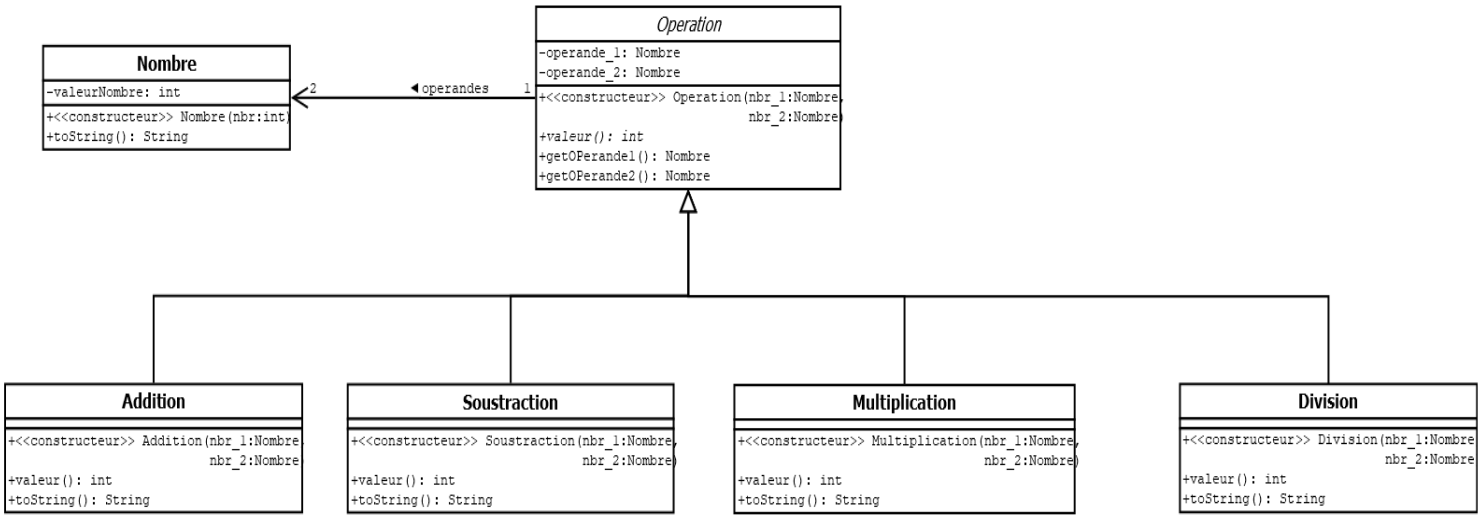

Fatih FIDAN & Tamij SARAVANAN

fatih.fidan@edu.univ-paris13.fr
tamijanebane.saravanan@edu.univ.paris13.fr

# Partie I : opérations simples

L'objectif du projet est de modéliser en UML et coder en java une calculatrice effectuant des opérations sur des nombres entiers.
On dispose des opérations suivantes : l'addition (+), la soustraction (-), la multiplication (*) et la division entière (/) de deux nombres entiers

## I. UML




## II. JAVA

**Operation.java :**

```
public abstract class Operation {

	private Nombre operande_1;
	private Nombre operande_2;

	public Operation(Nombre nbr_1, Nombre nbr_2) {
		this.operande_1 = nbr_1;
		this.operande_2 = nbr_2;
	}

	public abstract int valeur();
	public Nombre getOPerande1() {return this.operande_1;}
	public Nombre getOPerande2() {return this.operande_2;}


	//La méthode valeur ci-dessous nous a permis de tester la class Operation sans qu'il soit abstrait
	/*public int valeur() {
		return 0;
	}*/

}
```

**Nombre.java :**

```
public class Nombre {

	private int valeurNombre;

	public Nombre(int nbr) {
		this.valeurNombre = nbr;
	}

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

	public String toString() {
		return "Le nombre choisi est : " + this.valeurNombre;
	}

}
```

**Addition.java :**

```
public class Addition extends Operation{

	public Addition(Nombre nbr_1, Nombre nbr_2) {
		super(nbr_1, nbr_2);
	}

	public int valeur() {
		return this.getOPerande1().valeur() + this.getOPerande2().valeur();
	}

	public String toString() {
		return "(" + this.getOPerande1().valeur() + " + " + this.getOPerande2().valeur() + ")" ;
	}

}
```

**Soustraction.java :**

```
public class Soustraction extends Operation{

	public Soustraction(Nombre nbr_1, Nombre nbr_2) {
		super(nbr_1, nbr_2);
	}

	public int valeur() {
		return getOPerande1().valeur() - getOPerande2().valeur();
	}

	public String toString() {
		return "(" + getOPerande1().valeur() + " - " + getOPerande2().valeur() + ")" ;
	}
```

```
}
```

**Multiplication.java :**

```
public class Multiplication extends Operation{

	public Multiplication(Nombre nbr_1, Nombre nbr_2) {
		super(nbr_1, nbr_2);
	}

	public int valeur() {
		return this.getOPerande1().valeur() * this.getOPerande2().valeur();
	}

	public String toString() {
		return "(" + this.getOPerande1().valeur() + " x " + this.getOPerande2().valeur() + ")" ;
	}

}
```

**Division.java :**

```
public class Division extends Operation{

	public Division(Nombre nbr_1, Nombre nbr_2) throws ArithmeticException {
		super(nbr_1, nbr_2);
		if(nbr_2.valeur()==0) {
			throw new ArithmeticException ("--------->	Vous ne pouvez pas divisé par 0 !
<---------");
		}
	}

	//Première version de valeur() avant d'ajouter une 'Exception'
	/*
	public int valeur() {
		if(this.getOPerande2().valeur()==0) {
			System.out.println("Vous ne pouvez pas diviser par 0");
		}
		return this.getOPerande1().valeur() / this.getOPerande2().valeur();
	}
	*/

	public int valeur() {
		return this.getOPerande1().valeur() / this.getOPerande2().valeur();
	}

	public String toString() {
		return "(" + this.getOPerande1().valeur() + " / " + this.getOPerande2().valeur() + ")" ;
	}

}
```

**CalculatriceSimple.java :**
**- Cette class contient une méthode main qui nous a permis de tester tous les class précédents décrits.**

```
public class CalculatriceSimple {

	public static void main(String[] args) {


		System.out.println("---------->   Code du sujet de la Partie I   <----------");
		//Code du sujet de la Partie I
		Nombre six = new Nombre(6) ;
		Nombre dix = new Nombre(10) ;

		Operation s = new Soustraction(dix,six) ;
		System.out.println(s + " = " + s.valeur()) ; // doit afficher : (10 – 6) = 4

		System.out.println("\n---------->   Code écrit par Fatih & Tamij   <----------");
		//Code écrit par Fatih et Tamij
		Nombre zero = new Nombre(0);
		Nombre deux = new Nombre(2);
		Nombre huit = new Nombre(8);

		Operation PLUS = new Addition(huit, deux);
		System.out.println(PLUS + " = " + PLUS.valeur());

		Operation MOINS = new Soustraction(huit, deux);
		System.out.println(MOINS + " = " + MOINS.valeur());

		Operation FOIS = new Multiplication(huit, deux);
		System.out.println(FOIS + " = " + FOIS.valeur());

		try {
			Operation DIVISERsans0 = new Division(huit,deux);
			System.out.println(DIVISERsans0 + " = " + DIVISERsans0.valeur()) ;
		} catch (Exception e) {
			System.out.println("\nIl y a une erreur !\nLa voici : " + e);
		}

		try {
			Operation DIVISEavec0 = new Division(huit,zero);
			System.out.println(DIVISEavec0 + " = " + DIVISEavec0.valeur()) ;
		} catch (Exception e) {
			System.out.println("\nIl y a une erreur !\nLa voici : " + e);
		}



	}

}
```

# Information sur les classes

**Nombre.java :**
- Cette class contient la base de notre calculatrice avec 1 constructeur champ à champ (qui prend en paramètre un nombre de type 'int').
- Il possède 2 méthodes qui sont valeur (qui retourne la valeur du nombre) et toString (qui retourne une chaîne de caractères avec la valeur du nombre).

**Operation.java :**
- Cette class est abstraite, il possède un constructeur champ à champ (qui prend en paramètre 2 nombres de types 'Nombre' précédemment créés avec le class Nombre).
- Il possède 3 méthodes différentes les deux premiers, dont getOPerande1 et getOPerande2 (il retourne chacun le Nombre des opérandes), enfin il y a la méthode valeur de type int qui est abstrait.

**Addition.java :**
- Cette class héritée de la class Opération, il possède un constructeur champ à champ (qui prend en paramètre 2 nombres de type 'Nombre').
- Il possède 2 méthodes différentes les premières sont la méthode valeur (qui retourne une addition) et la deuxième est la méthode toString (qui retourne l'addition en question sous forme de chaîne de caractères).

**Soustraction.java :**
- Cette class héritée de la class Opération, il possède un constructeur champ à champ (qui prend en paramètre 2 nombres de type 'Nombre').
- Il possède 2 méthodes différentes les premières sont la méthode valeur (qui retourne une soustraction) et la deuxième est la méthode toString (qui retourne la soustraction en question sous forme de chaîne de caractères).

**Multiplication.java :**
- Cette class héritée de la class Opération, il possède un constructeur champ à champ (qui prend en paramètre 2 nombres de type 'Nombre').
- Il possède 2 méthodes différentes les premières sont la méthode valeur (qui retourne une multiplication) et la deuxième est la méthode toString (qui retourne la multiplication en question sous forme de chaîne de caractères).

**Division.java :**
- Cette class héritée de la class Opération, il possède un constructeur champ à champ (qui prend en paramètre 2 nombres de type 'Nombre') et qui crée une ArithmeticException si le deuxième nombre en paramètre est égal à 0, car effectivement il est impossible de diviser par 0.
- Il possède 2 méthodes différentes les premières sont la méthode valeur (qui retourne une division) et la deuxième est la méthode toString (qui retourne la division en question sous forme de chaîne de caractères).

# BONUS

**Test.java :**
**- Cette class contient une méthode main avec les tests réalisés pour chaque class afin de voir s'il fonctionnait bien correctement.**

```
public class Test {

	public static void main(String[] args) {

		//		Test de la class Nombre
		System.out.println("----------->		Test de la class 'Nombre'		<----------
");
		Nombre nbr1 = new Nombre(10);
		Nombre nbr2 = new Nombre(5);
		Nombre zero = new Nombre(0);
		//Test valeur()
		System.out.println(nbr1.valeur());
		System.out.println(nbr2.valeur());
		//Test toString() (ajout espace au debut et à la fin)
		System.out.println(nbr1.toString());
		System.out.println(nbr2.toString());


		/*
		//		Test de la class Operation (afin de tester Operation nous avons enlever la
methode abstrait)
		System.out.println("\n----------->		Test de la class 'Operation'		<----------
");
		Operation op1 = new Operation(nbr1, nbr2);
		//Test getOPerande1 & getOPerande2
		System.out.println(op1.getOPerande1());
		System.out.println(op1.getOPerande2());
		//Test valeur() (il est censer envoyer un 0)
		System.out.println(op1.valeur());
		*/


		//Test de la class Addition
		System.out.println("\n----------->		Test de la class 'Addition'		<----------
");
		Operation addition = new Addition(nbr1, nbr2);
		//Test toString() & valeur()
		System.out.println(addition.toString()+ " = " + addition.valeur());


		//Test de la class Soustraction
		System.out.println("\n----------->		Test de la class 'Soustraction'		<--------
-------");
		Operation soustraction = new Soustraction(nbr1, nbr2);
		//Test toString() & valeur()
		System.out.println(soustraction.toString()+ " = " + soustraction.valeur());
```

```
		//Test de la class Multiplication
		System.out.println("\n---------------->		Test de la class 'Multiplication'		<----------");
		Operation multiplication = new Multiplication(nbr1, nbr2);
		//Test toString() & valeur()
		System.out.println(multiplication.toString()+ " = " + multiplication.valeur());


		//Test de la class Division
		System.out.println("\n---------------->		Test de la class 'Division'		<----------");
		Operation division = new Division(nbr1, nbr2);
		//Test toString() & valeur()
		System.out.println(division.toString()+ " = " + division.valeur());


		//Test de la class Division #Test Exception
		System.out.println("\n---------------->		Test de la class 'Division'		<----------");
		try {
			Operation divisionE = new Division(nbr1, zero);
			//Test toString() & valeur()
			System.out.println(divisionE.toString()+ " = " + divisionE.valeur());
		} catch(ArithmeticException e) {
			System.out.println("Voici l'erreur : " + e);
		}


	}

}
```

**TableMultiplication.java :**
**- Cette class contient une méthode main avec toutes les tables de multiplication qui sont calculées avec notre class Multiplication.**

```
public class TableMultiplication {

	public static void main(String[] args) {

		Nombre un = new Nombre(1) ;
		Nombre deux = new Nombre(2) ;
		Nombre trois = new Nombre(3) ;
		Nombre quatre = new Nombre(4) ;
		Nombre cinq = new Nombre(5) ;
		Nombre six = new Nombre(6) ;
		Nombre sept = new Nombre(7) ;
		Nombre huit = new Nombre(8) ;
		Nombre neuf = new Nombre(9) ;
		Nombre dix = new Nombre(10) ;

		//Table de 1
		System.out.println("\n-------->    Table de 1    <--------");
		for(int i=0; i<=10; i++) {
			Nombre j = new Nombre(i) ;
			Operation table1 = new Multiplication(un,j) ;
			System.out.println(table1 + " = " + table1.valeur());
		}

		//Table de 2
		System.out.println("\n-------->    Table de 2    <--------");
		for(int i=0; i<=10; i++) {
			Nombre j = new Nombre(i) ;
			Operation table2 = new Multiplication(deux,j) ;
			System.out.println(table2 + " = " + table2.valeur());
		}

		//Table de 3
		System.out.println("\n-------->    Table de 3    <--------");
		for(int i=0; i<=10; i++) {
			Nombre j = new Nombre(i) ;
			Operation table3 = new Multiplication(trois,j) ;
			System.out.println(table3 + " = " + table3.valeur());
		}

		//Table de 4
		System.out.println("\n-------->    Table de 4    <--------");
		for(int i=0; i<=10; i++) {
			Nombre j = new Nombre(i) ;
			Operation table4 = new Multiplication(quatre,j) ;
			System.out.println(table4 + " = " + table4.valeur());
		}
```

```
		//Table de 5
		System.out.println("\n-------->    Table de 5    <--------");
		for(int i=0; i<=10; i++) {
			Nombre j = new Nombre(i) ;
			Operation table5 = new Multiplication(cinq,j) ;
			System.out.println(table5 + " = " + table5.valeur());
		}

		//Table de 6
		System.out.println("\n-------->    Table de 6    <--------");
		for(int i=0; i<=10; i++) {
			Nombre j = new Nombre(i) ;
			Operation table6 = new Multiplication(six,j) ;
			System.out.println(table6 + " = " + table6.valeur());
		}

		//Table de 7
		System.out.println("\n-------->    Table de 7    <--------");
		for(int i=0; i<=10; i++) {
			Nombre j = new Nombre(i) ;
			Operation table7 = new Multiplication(sept,j) ;
			System.out.println(table7 + " = " + table7.valeur());
		}

		//Table de 8
		System.out.println("\n-------->    Table de 8    <--------");
		for(int i=0; i<=10; i++) {
			Nombre j = new Nombre(i) ;
			Operation table8 = new Multiplication(huit,j) ;
			System.out.println(table8 + " = " + table8.valeur());
		}

		//Table de 9
		System.out.println("\n-------->    Table de 9    <--------");
		for(int i=0; i<=10; i++) {
			Nombre j = new Nombre(i) ;
			Operation table9 = new Multiplication(neuf,j) ;
			System.out.println(table9 + " = " + table9.valeur());
		}

		//Table de 10
		System.out.println("\n-------->    Table de 10    <--------");
		for(int i=0; i<=10; i++) {
			Nombre j = new Nombre(i) ;
			Operation table10 = new Multiplication(dix,j) ;
			System.out.println(table10 + " = " + table10.valeur());
		}
	}
}
```