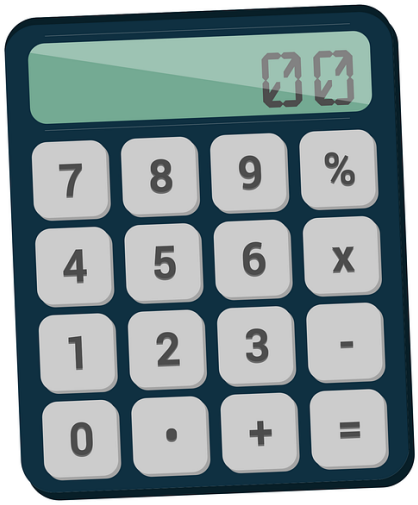
2021-2022

SAE 2.01 – Développement d'une application

Enseignant : Mr Bouthinon

Ma Calculatrice



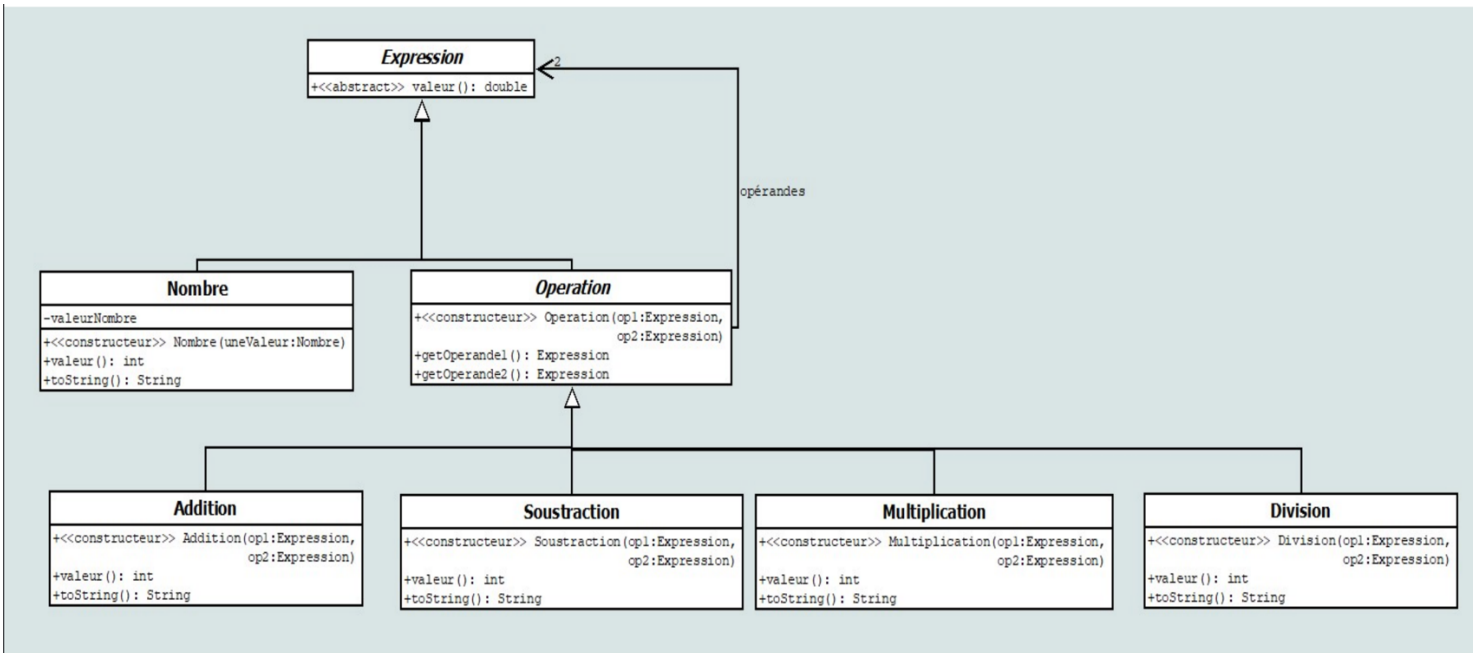
Fatih FIDAN & Tamij SARAVANAN

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

Partie II : Sensibilisation à la récursivité

Dans cette seconde partie nous considérons que la calculatrice peut effectuer des opérations composées. Une opération est composée lorsque qu'au moins l'une des deux opérandes est-elle même une opération. L'arbre ci-dessous représente l'opération (3+4) / 5 où l'opérande de gauche est l'opération 3+4.

I. UML



On observe que les Nombres et les Operations sont des Expressions. On constate aussi que les 2 opérandes d'une Operation sont de type Expression modélisant ainsi le fait qu'une opérande peut être un Nombre ou une Operation. Dans ce diagramme les classes Expression et Operation sont abstraites. Dans le programme java une Expression sera soit un Nombre soit une opération concrète c'est à dire une Addition, une Soustraction, une Multiplication ou une Division.

II. JAVA

Expression.java :

```
public abstract class Expression {

	public abstract double valeur();

	/* C'est lignes de code nous ont servis à tester notre class 'Operation' qui herite de
'Expression'
	public abstract Expression getOPerande1();
	public abstract Expression getOPerande2();
	*/
}
```

Nombre.java :

```
public class Nombre extends Expression {

	private double valeurNombre;

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

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

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

}
```

Operation.java :

```
public abstract class Operation extends Expression {

	private Expression operande_1;
	private Expression operande_2;

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

	public abstract double valeur();

	/* Cette ligne de code nous a permis de tester la class 'Operation' qui herite de 'Expression'
	public double valeur() {
		return 0;
	}
	*/

	public Expression getOPerande1() {return this.operande_1;}
	public Expression getOPerande2() {return this.operande_2;}

}
```

Addition.java :

```
public class Addition extends Operation{

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

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

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

}
```

Soustraction.java :

```
public class Soustraction extends Operation{

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

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

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

}
```

Multiplication.java :

```
public class Multiplication extends Operation{

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

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

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

}
```

Division.java :

```
public class Division extends Operation{

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

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

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

}
```

Calculatrice.java :

```
public class Calculatrice {

	public static void main(String[] args) {
		//Les nombres :
		Expression zero = new Nombre(0);
		Expression un = new Nombre(1);
		Expression deux = new Nombre(2);
		Expression trois = new Nombre(3);
		Expression quatre = new Nombre(4);
		Expression cinq = new Nombre(5);
		Expression six = new Nombre(6);
		Expression sept = new Nombre(7);
		Expression huit = new Nombre(8);
		Expression neuf = new Nombre(9);
		Expression dix = new Nombre(10);
		Expression dixSept = new Nombre(17);


		System.out.println("---------->  Code du sujet de la Partie II  <----------");
		//Code du sujet de la Partie II
		Expression s = new Soustraction(dixSept, deux) ;
		Expression a = new Addition(deux, trois) ;
		Expression d = new Division(s, a) ;
		System.out.println(d + " = " + d.valeur()) ; // affiche ((17 - 2) / (2 + 3)) = 3
```

```
System.out.println("\n----------> Code écrit par Fatih & Tamij <----------");
//Code écrit par Fatih et Tamij
Expression PLUS = new Addition(dixSept, deux);
System.out.println(PLUS + " = " + PLUS.valeur());

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

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

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

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


System.out.println("\n----------> opérations composées <----------");
Expression OC_calc_2_1 = new Addition(sept, trois);
Expression OC_calc_2_2 = new Soustraction(huit, cinq);
Expression OC_calc_2_3 = new Soustraction(dix, un);
Expression OC_calc_2_4 = new Addition(deux, six);
Expression OC_calc_2_5 = new Multiplication(OC_calc_2_1, OC_calc_2_2);
Expression OC_calc_2_6 = new Multiplication(OC_calc_2_3, OC_calc_2_4);
Expression OC_calc_2_Res = new Division(OC_calc_2_5, OC_calc_2_6);
System.out.println(OC_calc_2_Res + " = " + OC_calc_2_Res.valeur());


Expression OC_calc_3_1 = new Addition(un, deux);
Expression OC_calc_3_2 = new Addition(un,sept);
Expression OC_calc_3_3 = new Multiplication(OC_calc_3_1, OC_calc_3_2);
Expression OC_calc_3_Res = new Multiplication(OC_calc_3_3, dix);
System.out.println(OC_calc_3_Res + " = " + OC_calc_3_Res.valeur());

Expression OC_calc_4_1 = new Multiplication(deux,trois);
Expression OC_calc_4_2 = new Multiplication(quatre, deux);
Expression OC_calc_4_Res = new Addition(OC_calc_4_1, OC_calc_4_2);
System.out.println(OC_calc_4_Res + " = " + OC_calc_4_Res.valeur());


Expression OC_calc_5_1 = new Multiplication(dix,dix);
Expression OC_calc_5_2 = new Multiplication(dix,sept);
Expression OC_calc_5_3 = new Division(OC_calc_5_1, deux);
Expression OC_calc_5_4 = new Division(OC_calc_5_2, deux);
Expression OC_calc_5_Res = new Soustraction(OC_calc_5_3, OC_calc_5_4);
System.out.println(OC_calc_5_Res + " = " + OC_calc_5_Res.valeur());
```

```
		Expression OC_calc_6_1 = new Addition(quatre, deux);
		Expression OC_calc_6_2 = new Soustraction(un, dix);
		Expression OC_calc_6_Res = new Soustraction(OC_calc_6_1, OC_calc_6_2);
		System.out.println(OC_calc_6_Res + " = " + OC_calc_6_Res.valeur());


		Expression OC_calc_7_1 = new Soustraction(un, dix);
		System.out.println(OC_calc_7_1+ " = " + OC_calc_7_1.valeur());


		Expression OC_calc_8_1 = new Soustraction(six, dix);
		Expression OC_calc_8_2 = new Soustraction(quatre, neuf);
		Expression OC_calc_8_Res = new Division(OC_calc_8_1, OC_calc_8_2);
		System.out.println(OC_calc_8_Res+ " = " + OC_calc_8_Res.valeur());


		Expression OC_calc_9_1 = new Soustraction(neuf, deux);
		Expression OC_calc_9_2 = new Soustraction(un, neuf);
		Expression OC_calc_9_Res = new Multiplication(OC_calc_9_1, OC_calc_9_2);
		System.out.println(OC_calc_9_Res+ " = " + OC_calc_9_Res.valeur());


		Expression OC_calc_10_1 = new Soustraction(six, cinq);
		Expression OC_calc_10_2 = new Soustraction(quatre, quatre);
		Expression OC_calc_10_Res = new Multiplication(OC_calc_10_1, OC_calc_10_2);
		System.out.println(OC_calc_10_Res+ " = " + OC_calc_10_Res.valeur());


		Expression OC_calc_11_1 = new Multiplication(trois,six);
		Expression OC_calc_11_2 = new Multiplication(sept,sept);
		Expression OC_calc_11_3 = new Division(OC_calc_11_1, quatre);
		Expression OC_calc_11_4 = new Division(OC_calc_11_2, deux);
		Expression OC_calc_11_Res = new Soustraction(OC_calc_11_3, OC_calc_11_4);
		System.out.println(OC_calc_11_Res + " = " + OC_calc_11_Res.valeur());

		try {
			Expression OC_calc_1_1 = new Soustraction(six, quatre);
			Expression OC_calc_1_2 = new Soustraction(OC_calc_1_1, deux);
			Expression OC_calc_1_3 = new Addition(neuf, dixSept);
			Expression OC_calc_1_Res = new Division(OC_calc_1_3, OC_calc_1_2);
			System.out.println(OC_calc_1_Res + " = " + OC_calc_1_Res.valeur());
		} catch (Exception e) {
			System.out.println("\nIl y a une erreur !\nLa voici : " + e);
		}


	}

}
```

Information sur les classes

Expression.java
- Cette class est abstraite, il ne possède pas de constructeur.
- Il possède une méthode valeur de type double et qui est abstraite.

Nombre.java :
- Cette class hérite de la class Expression et 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 et hérite de la class Expression, il possède un constructeur champ à champ (qui prend en paramètre 2 nombres de types 'Expression' précédemment créés avec le class Nombre qui hérite d'Expression).
- 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 double 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 'Expression').
- 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 'Expression').
- 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 'Expression').
- 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 'Expression') 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).

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

BONUS

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

Test.java :

```
public class Test {

	public static void main(String[] args) {

		//		Test de la class Nombre qui herite de 'Expression'
		System.out.println("---------------->		Test de la class 'Nombre' qui herite de 'Expression'		<----------------");
		Expression nbr1 = new Nombre(10);
		Expression nbr2 = new Nombre(5);
		Expression zero = new Nombre(0);
		//Test valeur()
		System.out.println(nbr1.valeur());
		System.out.println(nbr2.valeur());
		//Test toString() (ajout espace au debut et □ 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' qui herite de 'Expression'		<----------------");
		Expression 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 qui herite de 'Expression'
		System.out.println("\n---------------->		Test de la class 'Addition' qui herite de 'Expression'		<----------------");
		Expression addition = new Addition(nbr1, nbr2);
		//Test toString() & valeur()
		System.out.println(addition.toString()+ " = " + addition.valeur());


		//Test de la class Soustraction qui herite de 'Expression'
		System.out.println("\n---------------->		Test de la class 'Soustraction' qui herite de 'Expression'		<----------------");
```

```
		Expression soustraction = new Soustraction(nbr1, nbr2);
		//Test toString() & valeur()
		System.out.println(soustraction.toString()+ " = " + soustraction.valeur());


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


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


		//Test de la class Division qui herite de 'Expression' / #Test Exception
		System.out.println("\n---------------->		Test de la class 'Division'		<----------------
");
		try {
			Expression 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);
		}



	}

}
```