Polynomials

Student:
Suciu Mihai
Group: E_30422

1.Objective

The main objective of this homework was to develop a Java application capable of implementing and performing operations on polynomials having integer coefficients. The software should be able to perform addition, subtraction, multiplication, division, differentiation and integration. The program must have an intuitive and “user friendly” graphical interface.

This problem can be split as follows:

- creating the Monom and Polynomial classes, where Polynomial is a list of Monoms
- creating the graphical interface
- converting the strings (given in the text fields) in the Monoms to work with them
- create different methods for every mathematical operation
- display the result

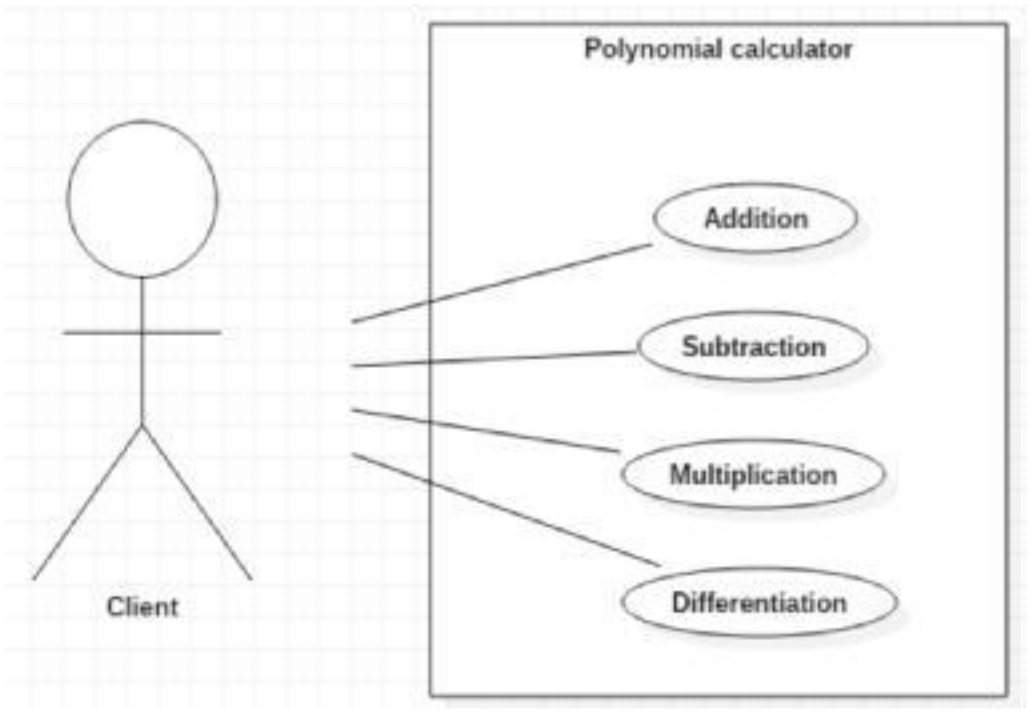
2.Problem analysis

In mathematics, a polynomial is an expression consisting of variables, called indeterminates, and coefficients that involves only the operations of addition, subtraction, multiplication, and non-negative integer exponents. An example of a polynomial of a single indeterminate (or variable), x, is x2 − 4x + 7, which is a quadratic polynomial. Polynomials appear in a wide variety of areas of mathematics and science. For example, they are used to form polynomial equations, which encode a wide range of problems, from elementary word problems to complicated problems in the sciences; they are used to define polynomial functions, which appear in settings ranging from basic chemistry and physics to economics and social science; they are used in calculus and numerical analysis to approximate other functions. In advanced mathematics, polynomials are used to construct polynomial rings and algebraic varieties, central concepts in algebra and algebraic geometry.

We will implement this problem from a general user point of view. That is with no different roles in the utilization. The application will be a simple calculator that works on polynomials.

We will introduce two polynomials, P1(X) and P2(X), load them and perform the following operations: addition, subtraction, multiplication and differentiation.

Use-Case diagram:



Actor: Client

Case: **without** error

a) The client writes the polynomials without errors e.g.: 2x^5+3 and 7x^4+3x
b) Press “Save”
c) The client chooses the operation and presses the corresponding button
d) The result appears under the buttons, near the “Result:” label
e) The client presses again “Save” button and chooses another operation (jump at d)
f) The client changes the polynomials without errors (jump at a)

Actor: Client

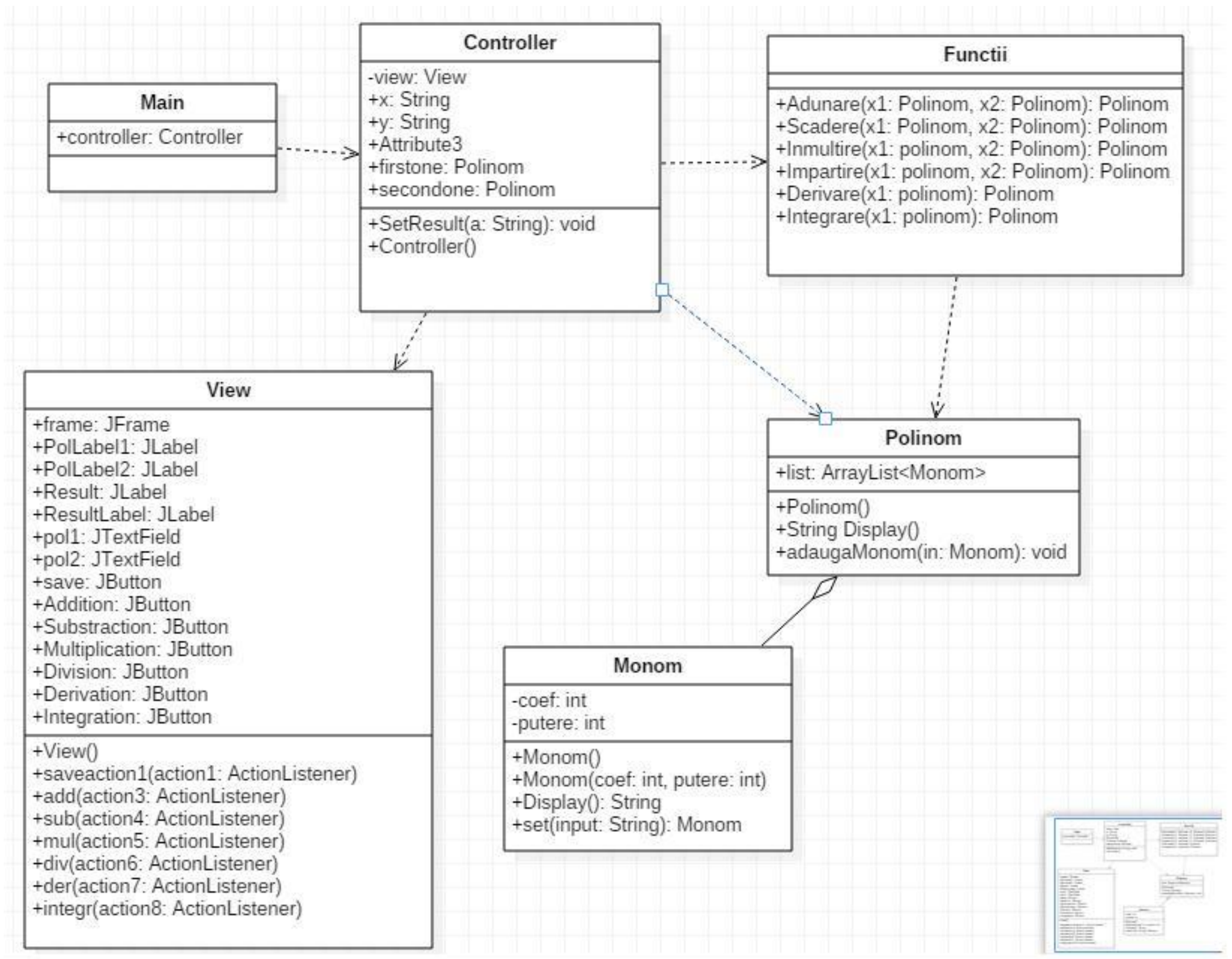
Case: **with** error

a) The client writes the polynomials with one or more errors
b) Press “Save”
c) The application checks the input and doesn’t work because the input’s structure is bad

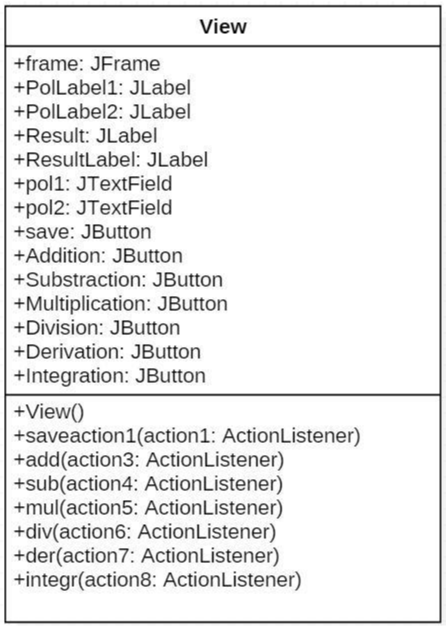
Assumptions:

-The client will know the structure of the polynomial

3. Design


Main
+controller: Controller
Controller
-view: View
+x: String
+y: String
+Attribute3
+firstone: Polinom
+secondone: Polinom
+SetResult(a: String): void
+Controller()
Functii
+Adunare(x1: Polinom, x2: Polinom): Polinom
+Scadere(x1: Polinom, x2: Polinom): Polinom
+Inmultire(x1: polinom, x2: Polinom): Polinom
+Impartire(x1: polinom, x2: Polinom): Polinom
+Derivare(x1: polinom): Polinom
+Integrare(x1: polinom): Polinom
View
+frame: JFrame
+PolLabel1: JLabel
+PolLabel2: JLabel
+Result: JLabel
+ResultLabel: JLabel
+pol1: JTextField
+pol2: JTextField
+save: JButton
+Addition: JButton
+Substraction: JButton
+Multiplication: JButton
+Division: JButton
+Derivation: JButton
+Integration: JButton
+View()
+saveaction1(action1: ActionListener)
+add(action3: ActionListener)
+sub(action4: ActionListener)
+mul(action5: ActionListener)
+div(action6: ActionListener)
+der(action7: ActionListener)
+integr(action8: ActionListener)
Polinom
+list: ArrayList<Monom>
+Polinom()
+String Display()
+adaugaMonom(in: Monom): void
Monom
-coef: int
-putere: int
+Monom()
+Monom(coef: int, putere: int)
+Display(): String
+set(input: String): Monom

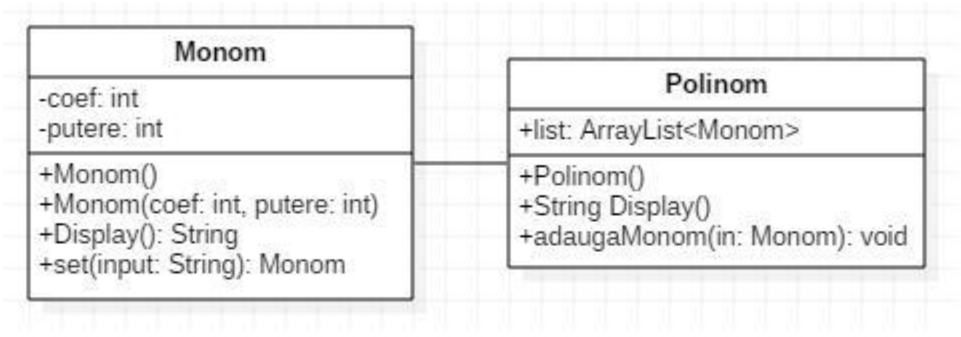
The projection of this application started with the user interface. I will present below View class with its attributes and methods.



(uml diagram of View class)

The first two labels are used to show where the user must insert the polynomials, the labels with the result name are used to show where will be the result and the result itself, the pol1 and pol2 text fields are used to read the polynomials, the button "save" is used to save the polynomials and the rest buttons are corresponding for each operation that the client wants to use. After this I choose the position for all components.

The second step I implemented 2 classes representing the Monom and Polynomial.

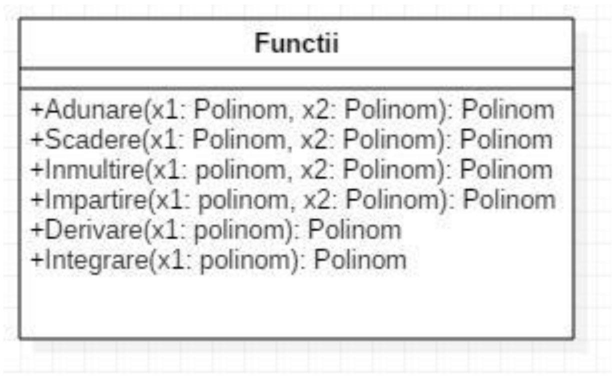


(uml diagrams of Monom and Polynomial classes)

In class Monom I have the attributes coef and putere which represent the coefficient and the power of each term, two constructors (one with default data and one with a specific coefficient and power), one method for display the result and the last method named set is to transform the String in to the Monom object.

In class Polinom I have a list of Monoms, one constructor, one method which displays one String and the last method("adaugaMonom") is to insert the Monoms in to the list.

The third step was to implement a class with all the operations requested. This class is named "Functii" and here is its uml diagram:


Functii
+Adunare(x1: Polinom, x2: Polinom): Polinom
+Scadere(x1: Polinom, x2: Polinom): Polinom
+Inmultire(x1: polinom, x2: Polinom): Polinom
+Impartire(x1: polinom, x2: Polinom): Polinom
+Derivare(x1: polinom): Polinom
+Integrare(x1: polinom): Polinom

(uml diagram of Functii class)

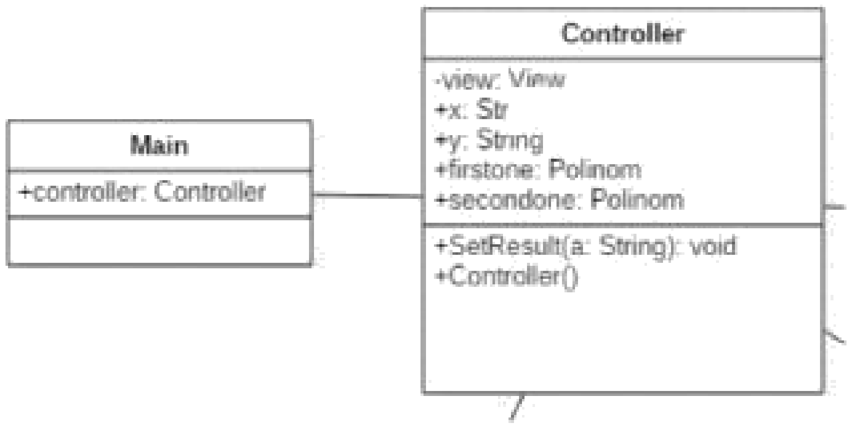
First method represents the addition and return a Polynomial. We have two Polynomials as parameters and I used the method from Polynomial class named "adaugaMonom" which add the coefficients if the Monoms have the same power.

Second method makes the subtraction and return a Polynomial. Here I negated all the coefficients of the second Polynomial and I used the addition method.

The third method returns the product and we have two parameters and the result Polynomial. To calculate the multiplication, I follow the first list of Monoms and multiply every term with all terms of the second list like this: multiply the coefficients and add the powers.

The method of differentiation one Polynomial has one parameters and returns a Polynomial witch is the result.

The last step was to implement the main and the controller classes.



The Main class contains an attribute of type Controller which construct the application and the Controller contains few attributes for data input (x, y, firstone, secondone), an attribute for View and the method to put the result in to the label

4. Implementation

a) Classes (important methods)

The first class implemented was the Monom class. Here I have two constructors (for default data and with specific coefficient and power):

```
public Monom() {
            this(1, 1);
      }
```

```
public Monom(int coef, int putere) {
       this.coef = coef;
       this.putere = putere;
}
```

And one display function which transforms the monoms in Strings:

```
public String display() {
              String mon = "";
              if (this.coef == 0) {
                    return "0";
              } else if (this.coef == 1) { if
                    (this.putere == 0) {
                          mon = mon + "+" + String.valueOf(this.coef);
                    } else {
                          mon = mon + "+";
                    }
                } else if (this.coef == -1) {
                     if (this.putere == 0) {
                          mon = mon + String.valueOf(this.coef);
                    } else {
                          mon = mon + "-";
                    }
              } else if (this.coef < 0) {
                    mon = mon + String.valueOf(this.coef);
```

```
            } else if (this.coef > 0) {
                    mon = mon + "+" + String.valueOf(this.coef);
            }
            if (this.putere == 1) {
                    mon = mon + "x";
            } else if (this.putere > 1) {
                    mon = mon + "x^" + String.valueOf(this.putere);
            }
            return mon;
    }
```

After this display, I implemented a method which transforms the String from the input in to an object, Monomial type.

```
public Monom set(String input) {
            this.putere = 0; this.coef = 0;
            String aux[];
            if (input.contains("^")) {
                    aux = input.split("x\\^");
                    if (aux[0].length() == 0) {
                            this.coef = 1;
                    } else
                            this.coef = Integer.parseInt(aux[0]);
                    if (aux[1].length() == 0) {
                            this.putere = 1;
                    } else {
                            this.putere = Integer.parseInt(aux[1]);
                    }
            }else {
                    if(input.contains("x")) {
                            this.putere = 1;
                            aux = input.split("x");
                            if(aux.length == 1) {this.coef = Integer.parseInt(aux[0]);
                            }else this.coef = 1;
                    }else {
                            this.putere = 0;
                            this.coef = Integer.parseInt(input);}
            }
            return this;
    }
```

The second class was Polinom, which is a list of Monoms. Here I have a constructor which takes a String, split it (by "+" and "-") in to Monoms and add it in the list.

```
public Polinom(String input) {
            String pmon[], nmon[];
            Monom auxiliar;     int i = 0;
            pmon = input.split("\\+");
            for (String aux : pmon) {
                    nmon = aux.split("\\-");i = 0;
                    for (String aux1 : nmon) {
                            auxiliar = new Monom();
                            if (aux1.length() == 0)
                                    i = 1;       continue;}
                    if (i == 0) {
                          auxiliar.set(aux1);   i = 1;
                    } else {
                          auxiliar.set(aux1);
                          auxiliar.setCoef(auxiliar.getCoef() * -1);}
                    this.adaugaMonom(auxiliar);}
             }
       }
```

The last method in Polynomial class is “adaugaMonom” which allows you to add in the list one monom following a certain logic: if the list has no elements I add the monom in the top of the list, and if the list contains some elements I try to sort it descending by powers.

```
public void adaugaMonom(Monom in) {
             if (list.size() == 0 || in.getPutere() > list.get(0).getPutere())
                   { list.add(0, in);
             } else if (list.get(list.size() - 1).getPutere() > in.getPutere()) {
                   list.add(list.size(), in);
             } else {
                   for (int i = 0, j = 1; i < list.size(); i++, j++) { if
                         (list.get(i).getPutere() > in.getPutere() &&
                            list.get(j).getPutere() < in.getPutere())
                         { list.add(j, in);
                         } else if (list.get(i).getPutere() == in.getPutere()) {
                   list.get(i).setCoef(in.getCoef() + list.get(i).getCoef());
                         }
                   }
             }
       }
```

The next step was to implement the operations. Here, for addition, i used the method which add a monom in the list because if the powers are the same, the coefficients are gathered.

```
public static Polinom adunare(Polinom x,Polinom x1)
             { for(Monom aux : x.getList()) {
                   x1.adaugaMonom(aux) ; }
             return x1;
       }
```

For the next operation, subtraction, I used the addition method but firstly I negated all the coefficients of the second polynomial.

```
public static Polinom scadere(Polinom x, Polinom y)
             { for(Monom aux:y.getList() ) {
                   aux.setCoef(aux.getCoef()*-1);
                   x.adaugaMonom(aux);
             }
             return x;
       }
```

For the multiplication we go through the first polynomial and we multiply each monom by the second polynomial.

```
public static Polinom inmultire(Polinom x, Polinom y) {
		Polinom z = new Polinom("");
		for(Monom aux:x.getList()) {
			for(Monom aux1:y.getList()) {
				Monom aux2 = new Monom();
				aux2.setCoef(aux.getCoef()*aux1.getCoef());
				aux2.setPutere(aux.getPutere()+aux1.getPutere());
				z.adaugaMonom(aux2);
			}
		}
		return z;
	}
```

For the differentiation I multiplied each coefficient with its power, and I decreased the power by 1.

```
public static Polinom derivare(Polinom x) {
		for(Monom aux:x.getList()) {
			aux.setCoef(aux.getCoef()*aux.getPutere());
			aux.setPutere(aux.getPutere()-1);
		}
		return x;
	}
```

The last class before the main is Controller. Here I have one method to set the result in Label :

```
Public void SetResult(String a) {
			view.ResultLabel.setText(a);
		}
```

After this, is the constructor, which contains all the actionListeners for the buttons. The first one is for the Save and the rest are for the operations which are the same, only the method called is different.

```
public Controller() {
		this.view = new View();
		this.view.saveaction1(e -> {
			x = view.pol1.getText();
			firstone = new Polinom(x);
			y = view.pol2.getText();
			secondone = new Polinom(y);
		});
		this.view.add(e -> {
			SetResult(Functii.adunare(firstone, secondone).display());
		});
		this.view.sub(e -> {
			SetResult(Functii.scadere(firstone, secondone).display());
		});
		this.view.mul(e -> {
			SetResult(Functii.inmultire(firstone, secondone).display());
		});
		this.view.der(e -> {
```

```
                SetResult(Functii.derivare(firstone).display());
        });
    }
```

b) The user interface

First, I have one Frame and two textfields with their labels (included in frame and set with setBounds)

```
public JFrame frame = new JFrame("Polynomials");

public JTextField pol1 = new JTextField();
frame.add(pol1) ;
pol1.setBounds(30, 60, 200, 30);

public JTextField pol2 = new JTextField();
frame.add(pol2);
pol2.setBounds(30, 150, 200, 30);

JLabel PolLabel1 = new JLabel(" First Polynomial: ");
frame.add(PolLabel1);
PolLabel1.setBounds(30, 30, 150, 20);

JLabel PolLabel2 = new JLabel(" Second Polynomial: ");
frame.add(PolLabel2);
PolLabel2.setBounds(30, 120, 150, 20);
```

The rest are buttons, I will put here only the attributes without bounds because the way is the same as the textfields and labels.

```
JButton save = new JButton("* Save *");
JButton Addition = new JButton("Addition   ( + ) ");
JButton Subtraction = new JButton("Subtraction   ( - ) ");
JButton Multiplication = new JButton("Multiplication   ( x )");
JButton Division = new JButton("Division   ( : )");
JButton Derivation = new JButton("Derivation  ");
JButton Integration = new JButton("Integration");
```

5. Testing

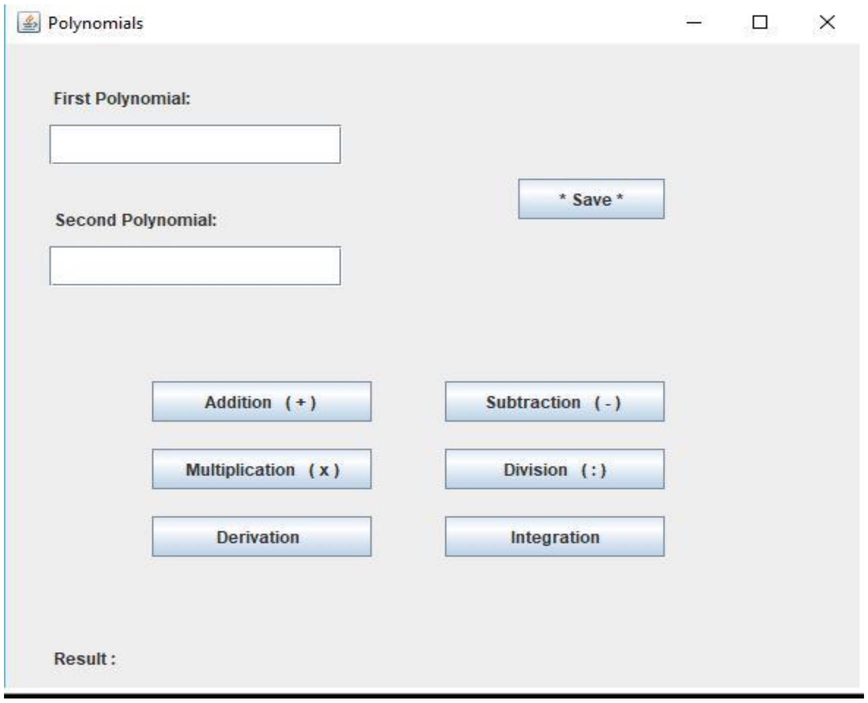
I tested all implemented operations, and all are correct.

What is tested	Input	Expected output	Output	Result (pass / fail)
Addition	-3x^2 +4x +3 & 5x^2 -2x +1	-2x^2 +2x +2	-2x^2 +2x +2	Pass
Subtraction	-3x^2 +4x +3 & 5x^2 -2x +1	-8x^2 +6x +4	-8x^2 +6x +4	Pass
Multiplication	2x^2 +4 & 2x^2 -4	4x^4 -16	4x^4 -16	Pass
Differentiation	3x^4 +2x^2 +1	12x^3 +4x	12x^3 +4x	Pass

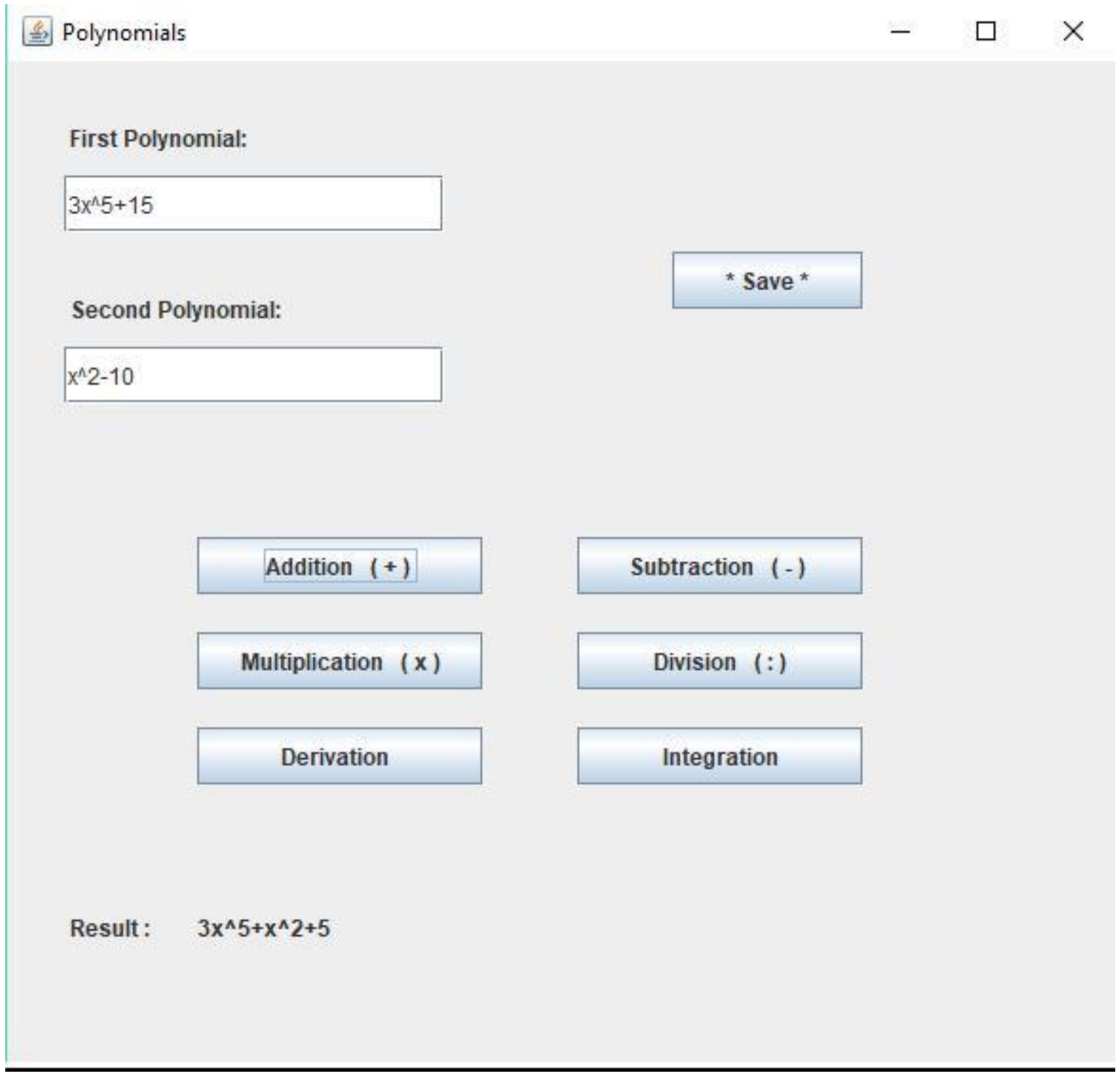
Here is the test unit of adding operation:

```
@Test public void testAdd() {
        Polinom a = new Polinom( "-3x^2+4x+3" );
        Polinom b = new Polinom( "5x^2-2x-1" ); Polinom
        c=new Polinom( "2x^2+2x+2" ); Polinom d=new
        Polinom(Functii.adunare(a,b) );
        assertEquals(c.display(), d.display() ); }
```

6. Results



You need to write the polynomials, press “Save” and after you choose one operation and press the corresponding button. If we write “3x^5+15” and “x^2-10” we will receive the expected answer, 3x^5+x^2+5:

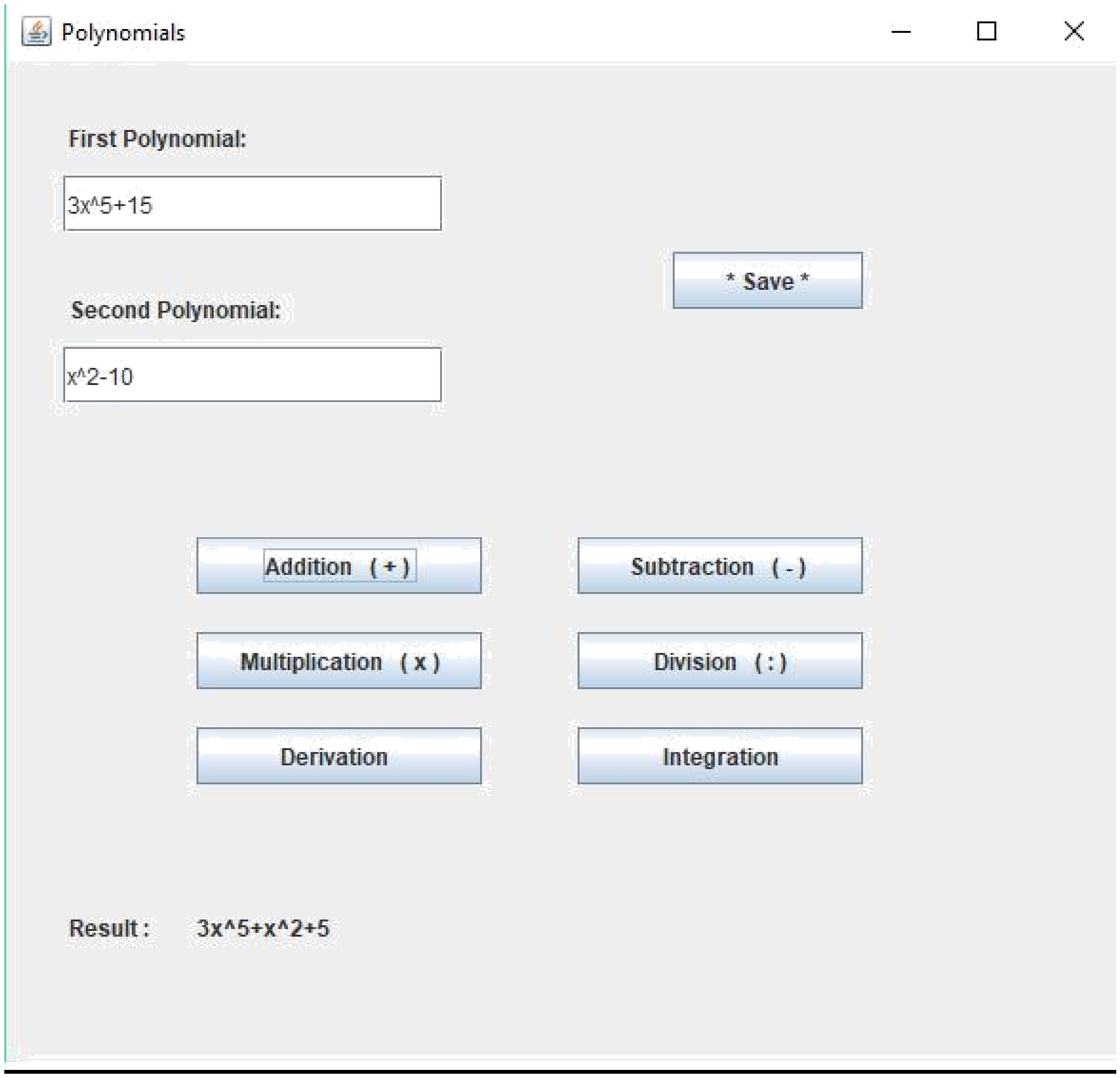


First Polynomial: 3x^5+15

Second Polynomial: x^2-10

Result : 3x^5+x^2+5

If you need another operation, you need to press “Save” again and choose the second one. For example, if you choose Derivation, the first polynomial will be calculated.



7. Conclusions

During the design of this project I improved my abilities in interfaces, I understand the lists and I think that the lists are more efficient than the arrays. Before this I learned how to perform a test unit and . Moreover, I have reminded the operations on polynomials.

Further development: the input polynomials to have real coefficients and implement the division and integration operations.

8.Bibliography

- https://en.wikipedia.org/wiki/Polynomial
- http://www.mkyong.com/unittest/maven-and-junit-example/