

Laboratory Work - Homework 1

Name: Bologa Marius Vasile

Group: 30425

1. Project objectives

Task: Propose, design and implement a system for polynomial processing. Consider the polynomials of one variable and integer coefficients. The application needs to be capable of performing operations on polynomials with integer coefficients. The application should perform the addition of two polynomials, the subtraction of two polynomials, the multiplication of two polynomials, the division of two polynomials, polynomial integration and polynomial differentiation.

To achieve this objective, we use object-oriented programming principles, applied programming language Java using Eclipse IDE. The application works in a basic way, that user retrieves data from the keyboard via a Graphical User Interface and apply different operations, depending on the user's desire. The results of the operations will be outputted in a user friendly way.

2. Problem analysis, modeling scenarios, use cases

2.1 Problem analysis

The analysis of a problem starts from examining the real model or the model we confront with in the real world and passing the problem through a laborious process of abstractization. Hence we identify our problem domain and we try to decompose it in modules easy to implement. As a start we take our knowledge in mathematics and we have that: a **polynomial** is an expression consisting of variables and coefficients which only employs the operations of addition, subtraction, multiplication, and non-negative integer exponents.

A polynomial in a single indeterminate x can always be written (or rewritten) in the form

$$a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0,$$

In order to be able to design and build a software program that performs and satisfies all of the specifications and requirements presented above it is very important to understand the notion of polynomial, degree and coefficient. Then it is essential to decompose our program and to try to discover the data structures, classes, methods and algorithms needed. Another challenge was to create an interface as user friendly as possible. For this I used a Frame where I added buttons for adding a polynomial, a combo box for operations and text areas for input/output.

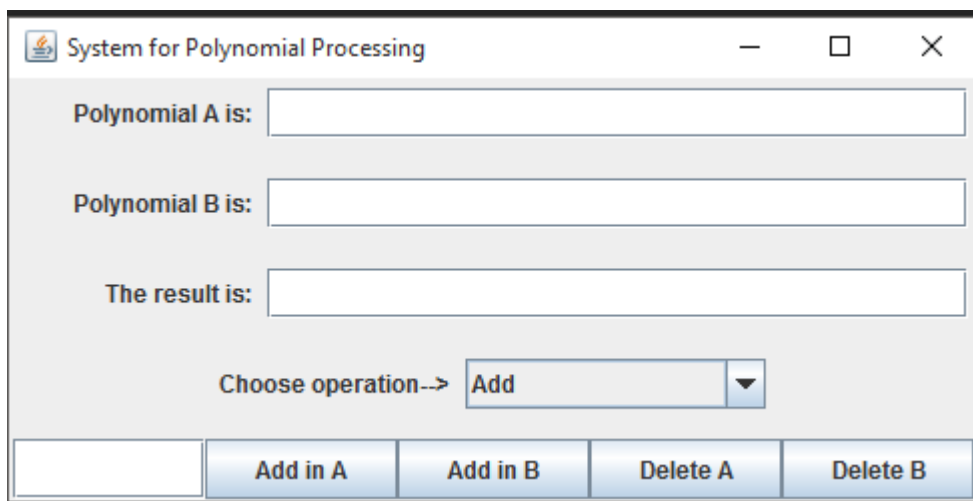
2.2 Modeling

First of all, we need to find some means to store the polynomial. The specification says that the user introduces the coefficients from the keyboard. This means that the easiest way to represent a polynomial is through its coefficients and degree. In this way processing the polynomials with the above named operations is quite easy.

Although the requirements state that the input coefficients are integers, performing the division of 2 polynomials or computing the primitive of a polynomial outputs a polynomial with real coefficients. That is why I used a `TypeOfCoefficient` with integer coefficients for addition, subtraction, and multiplication, and real (floating point type -> double) coefficients for division and integration. Also the coefficient of the largest degree of the polynomial is always different from 0. So I started to put myself more questions. Is this implementation efficient if I use arrays to store my polynomials or should I use `ArrayList`? I decided upon using arrays instead of `ArrayList`, it seems simpler to implement. The program should store the coefficients in an array of coefficients; coefficients that can be, integer or double and the degree can be very easily stored in an integer variable.

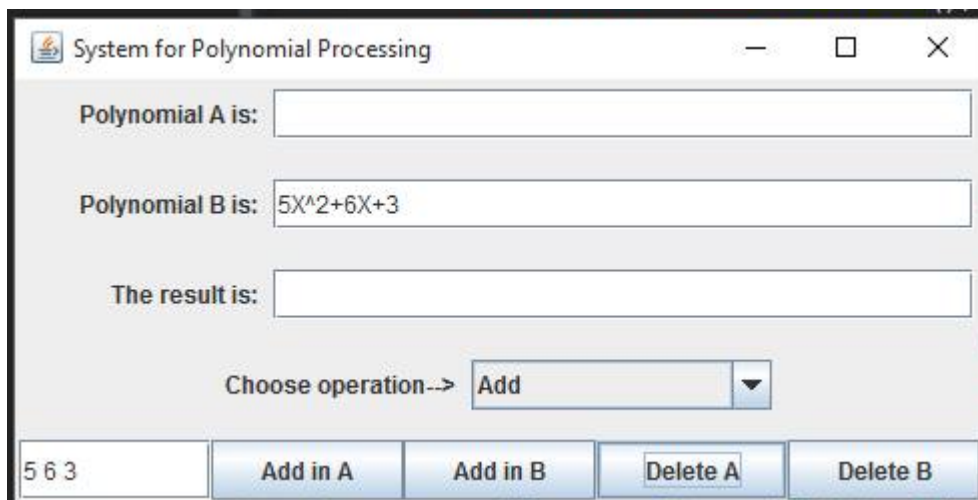
2.3 Scenario and Use cases

The use cases are strongly related to the user. How is it better to design the interface in order to be as user-friendly as possible? Hence, I have decided upon the following model:



The user introduces the coefficients of the polynomial he wants to introduce. He starts introducing the coefficients. After he introduces the coefficients in the text field coeff, the user must press the button Add in A in order to form the polynomial. The user introduces 0 for the elements of the polynomial that are missing. The same operation is repeated for the second polynomial. After that he chooses an operation from the combobox, that allows the user to choose the following options:

addition, subtraction, multiplication, integration of A and B, differentiation of A and B, multiplication by scalar, evaluate the polynomial at a certain value and find if the two polynomials are equal. Once an option is performed the result is displayed in the `JTextField` labeled by „The result is:” being the resulting polynomial. Moreover if a user wants to change the polynomials he can press the buttons Delete A or Delete B in order to introduce other polynomials (This fact is shown in the next picture).

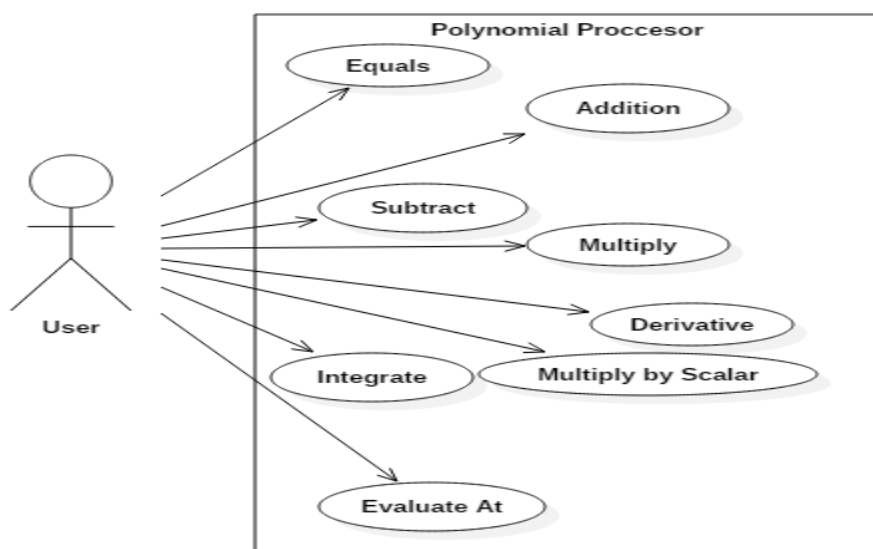


3. Projection and Implementation

In what the implementation is concerned this project was developed in Eclipse and it was only tested in this environment. However the program should maintain its portability. Concerning the code implementation I did not make use of laborious algorithms, but I have rather stayed faithful to the classical algorithms of computing polynomials learned in high school.

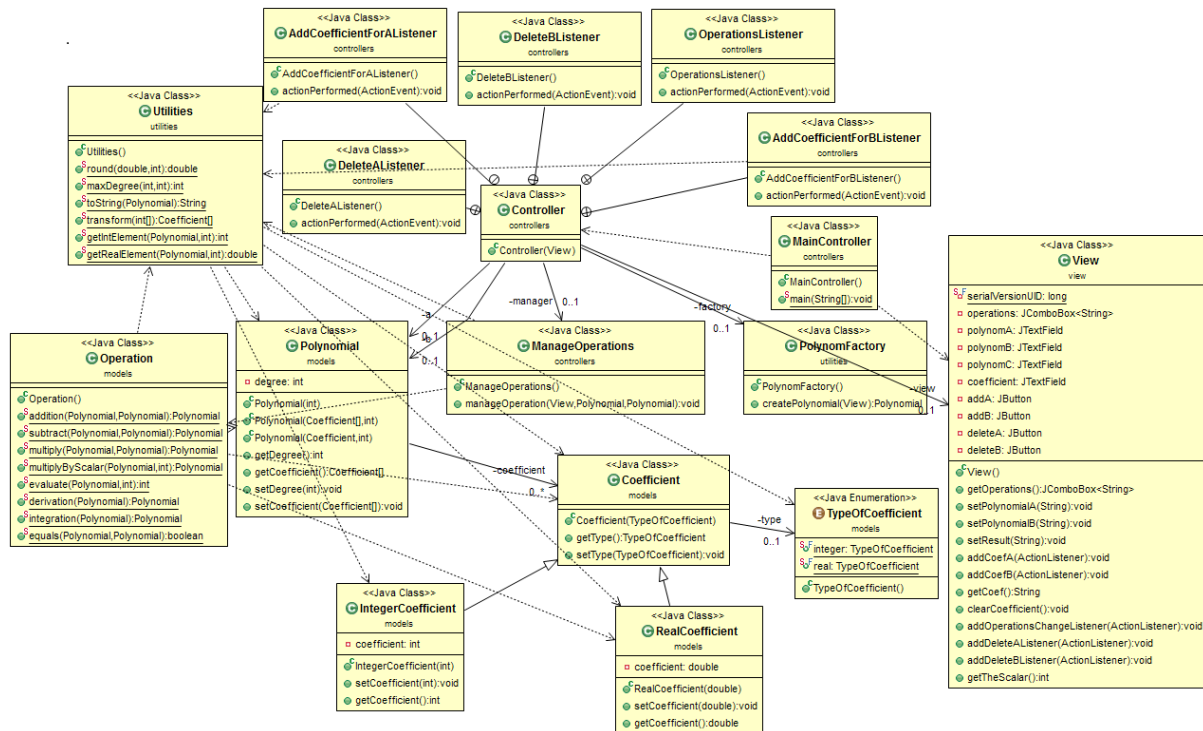
3.1 UML diagrams

3.1.1 Use case diagram



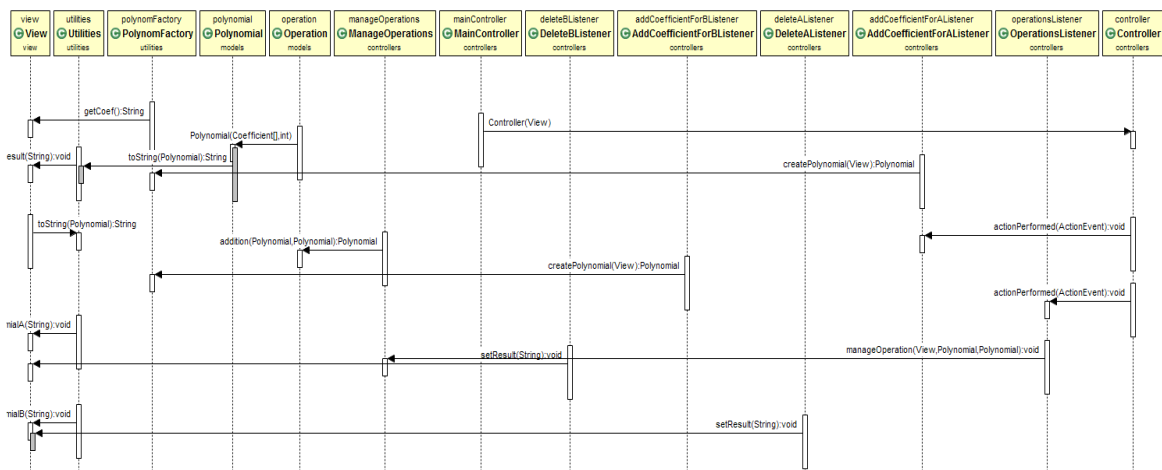
The use case diagram is nothing but the system functionalities written in an organized manner, presenting the actor, which is the user and the operations provided by the system for the user.

3.1.2 Class diagram



It can be seen that I have used inheritance for the two types of coefficients, RealCoefficient and IntegerCoefficient. I have also used the association between Controller and the View or PolynomFactory. Also added dependency between Operation and Coefficient or MainController and View.

3.1.3 Sequence diagram



The diagram can be easily explained. We run the application from MainController, and that implies to create a new controller. The controller responds to any input from the user. Its attribute is to handle all the actions that are performed on a button in the GUI. When we press the button Add in A, then the Polynom factory is called, and gets the coefficient from the text area and transforms the inputs into the mathematical representation, through the method toString(). The area reserved for the polynomial A is set now through the function setPolynomialA(). The same action is performed also for the B polynomial.

If the user puts at the input the wrong data then he might want to change the data. The user needs to press the button Delete A, by using this the Controller takes the command and clears the text field where the polynomial A was stored. To introduce another value the user will give another input.

After the user ensures that he has introduced the right data, he chooses one of the operations, then the controller is called. On his turn the controller asks for ManageOperation to fulfill its oath. For example the user chooses the addition. Then the two polynomials that were introduced by the user are sent as a parameter for the addition function, and this addition function returns a polynomial. It is needed a polynomial to be created using the constructor. The polynomial calls for the toString() method to get its own mathematical representation, after that the polynomial obtained is printed out in the result area.

3.2 Data structures

The data structures used at this problem are either primitive data types such as integers or floats or more complex objects such as array of coefficients or new created objects such as Polynomial, Coefficient or Operations. The class Coefficient was introduced for representing a type of coefficient that can be of two types: integer or double. This coefficient represents the main or the most important part of the program. All the operations are actually computed on the coefficients of the polynomial. So we needed to make a difference between the two ways it can be represented. I chose to have a RealCoefficient and an IntegerCoefficient, representing two classes that are the sub classes of the class Coefficient. Again the object Polynomial has been introduced in order to obtain an array of coefficients and a degree in order to form together a polynomial. Last but not least, the object Operations is intended to store any kind of operation implemented.

3.3 Class projection

Class projection refers mainly to how the model was thought, how the problem was divided in sub-problems, each sub-problem representing more or less the introduction of a new class. First I will start by mentioning exactly how my problem was divided into packages and afterwards each package with its own classes. I begin by creating the four packages I used: the first one being called „view” the second one being called „model”, the third one



„utilities” and the fourth one „controllers”. I named them intuitively because the first one handles the interface part, the part that deals with the user, the second one handles the implementation part, the third package is the one that provides useful functions, in order to make the algorithms in a modularized way and the fourth one, is the one that contains the main class, and the controller of the interface. I will begin with the interface, then I will continue with the models, utilities and controller part.

3.4 Interface(view)

View.java

-extends JFrame class , with which we made the GUI of the application

-some elements of the interface are: labels , text fields, buttons and a combo box for operations

this.setPreferredSize(new Dimension(500, 250))-sets the width of the frame to a specific size of 500, and the height to 250;

private JComboBox<String> operations- this is the combobox, used to store the operations that can be applied to the polynomials, and it contains 11 items like: addition, subtraction, multiply, evaluate at, etc.

private JTextField polynomA, polynomB, polynomC, coefficient- represents the text fields where the coefficients are inserted(coefficient), and the polynomial are the text fields where we get a mathematical representation of the polynomial.

private JButton addA, addB, deleteA, deleteB- the buttons are used in order to form a polynomial, and display it on the text field area, or to delete an existing polynomial, just to introduce another from the keyboard.

The constructor of this class is used for designing the frame. The frame's layout is based on a GridLayout of 1 column and 5 rows containing 4 panels. I will also only mention the methods used in this class as they will be described further at the Algorithms section. Therefore, the methods used in this class are:

public JComboBox<String> getOperations()-method to get the operation from the combo box

public void setPolynomialA(String result)-method to set the text to the JTextField related to polynomial A

public void setPolynomialB(String result))-method to set the text to the JTextField related to polynomial B

public void setResult(String result))-method to set the text to the JTextField related to polynomial resulted after computing operations

public void addCoefA(ActionListener listener)- method to get the mathematical representation of the polynomial A(add an action listener to the button)

public void addCoefB(ActionListener listener))- method to get the mathematical representation of the polynomial B(add an action listener to the button)

public String getCoef()- method for getting the text from the JTextField of the coefficients

public void clearCoefficient()-clear the JTextField area related to the polynomials



public void addOperationsChangeListener(ActionListener listener)- adds action listener to the combo box

public void addDeleteAListener(ActionListener listener)-adds action listener to the button

public void addDeleteBListener(ActionListener listener)-adds action listener to the button

public int getTheScalar() –method to get a JoptionPane , in order to introduce the scalar

3.5 Models

TypeOfCoefficient.java

```
public enum TypeOfCoefficient {  
    integer, real  
}
```

This class, its actually an enum, that represents the main types of coefficients that a polynomial can have:integer or real;

Coefficient.java

private TypeOfCoefficient type: a coefficient can have a type, type represented in the enum, presented above.

IntegerCoefficient.java

this class extends the class Coefficient so it has a type and its type is integer

private int coefficient;

public void setCoefficient()- used to set the coefficient

public void getCoefficient()-used to get the coefficient

RealCoefficient.java

this class extends the class Coefficient so it has a type and its type is real

private double coefficient;

public void setCoefficient()- used to set the coefficient

public void getCoefficient()-used to get the coefficient

Both of the above classes the same kind of constructor:

```
public Integer(Real)Coefficient(int/double coefficient) { //gets as a parameter a double/int, and sets the instance  
    super(TypeOfCoefficient.integer); //variable. The constructor calls the super class  
    this.coefficient = coefficient; //constructor that has as a parameter a type int/double  
}
```


Polynomial.java

This is actually the core class, of this application. The class models the real object polynomial, that have two instance variables:

private Coefficient[] coefficient- this is an array of coefficients, representing the coefficients of the polynomial
private int degree- this is the degree of the polynomial

It also have several constructors, but the most important is:

```
public Polynomial(Coefficient[] coeff, int d) {  
    coefficient = new Coefficient[d + 1];  
    for (int i = 0; i < d + 1; i++) {  
        coefficient[i] = coeff[i];  
    }  
    degree = d;  
}
```

The constructor receives as a parameter an array of coefficients and a degree, and sets the array of coefficients and the degree of a polynomial.

Also in this class are accesors and mutators, in order to set the coefficients and the degree.

Operations.java

This class is used for implementing the actual operations that are to be performed on polynomials.

public static Polynomial addition(Polynomial a, Polynomial b)-returns the resulting polynomial at addition

public static Polynomial subtract(Polynomial a, Polynomial b)-returns the resulting polynomial at subtration

public static Polynomial multiply(Polynomial a, Polynomial b)-returns the resulting polynomial at multiplication

public static Polynomial differentiation(Polynomial a)-returns the resulting polynomial at differentiation

public static Polynomial integration(Polynomial a)-returns the resulting polynomial at integration

public static Polynomial multiplyByScalsr(Polynomial a, int x)-returns the resulting polynomial multiplied by the scalar x

public static int evaluate(Polynomial a, int x)-returns the evaluation of the polynomial at the point x

public static boolean equals(Polynomial a,Polynomial b)-returns true if the polynomials are equals, false if not

3.6 Utilities

PolynomFactory.java

This class contains only one method „public Polynomial createPolynomial()” that gets the coefficients from the text area, and add them to the array of coefficients and in the same time sets the degree of the polynomial. The method returns a polynomial.

Utilities.java

The methods that are in thos class are used in order to make the work with polynomials and the computation of the operations easier.

Public static double round()- it truncates the value of a double, in such a way we have only two decimals after the decimal point

Public static int maxDegree()- get the maximum between two values

Public static int/double getInt/RealElement()-gets an element from a certain position from the array of coefficients

Public static toString()- the most important method in this class - costructs the mathematical representation of the polynomial.

3.7 Controllers

ManageOperations.java

The method in this class, gets as arguments two polynomials, and a view, and it cheks the value that is in combo box. If an element in the combo box is clicked then this method calls the methods in order to compute the result according to the option choosed by the user.

MainController.java

```
public static void main(String[] args) {  
    View view = new View();  
    new Controller(view);  
    view.setVisible(true);  
}
```

This class was intoduced in order to launch the application. So in the main method is created a view and also a controller for that view. Finally the frame is set visible.

Controller.java

The constructor of this class:

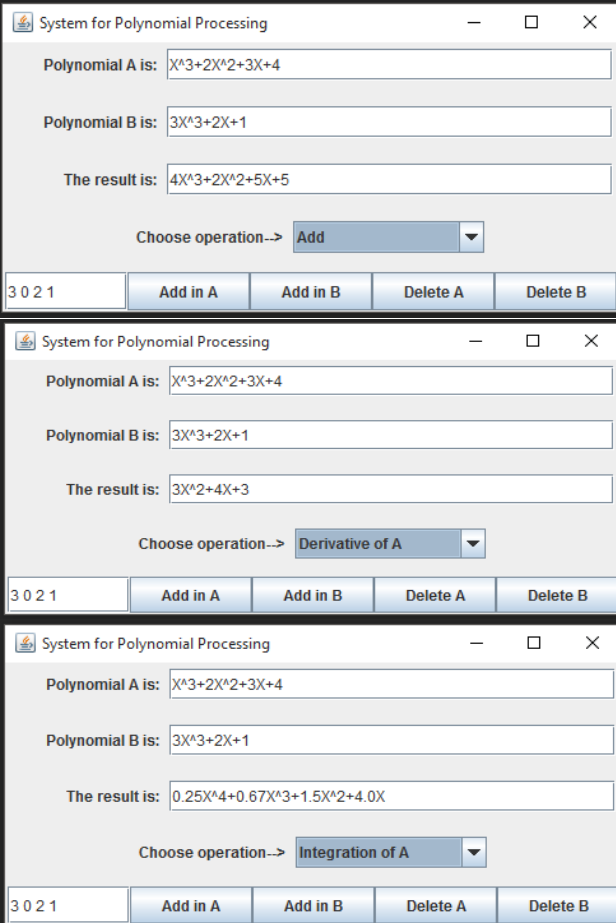
```
public Controller(View view) {  
    this.view = view;//set the instance variable view to the view(param)  
    this.view.addOperationsChangeListener(new OperationsListener());  
    this.view.addCoefA(new AddCoefficientForAListener());  
    this.view.addCoefB(new AddCoefficientForBListener());  
    this.view.addDeleteAListener(new DeleteAListener());  
    this.view.addDeleteBListener(new DeleteBListener());  
}
```

More than that, for the JFrame that we created in the class View, we now add ActionListener and we handle the events. To do that, we create classes OperationListener, AddCoefficientForAListener, AddCoefficientForBListener, DeleteAListener, DeleteBListener that implements ActionListener. This implies the implementation of the method actionPerformed.

```
public class OperationsListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        manager.manageOperation(view, a, b);
    }
}
```

4. Testing

The program has been tested on more polynomials, such that certain conditions were checked for different operation.



The application window 'System for Polynomial Processing' shows three different states:

- Top Screenshot:** Polynomial A is X^3+2X^2+3X+4 , Polynomial B is $3X^3+2X+1$. The result is $4X^3+2X^2+5X+5$. The 'Choose operation' dropdown is set to 'Add'.
- Middle Screenshot:** Polynomial A is X^3+2X^2+3X+4 , Polynomial B is $3X^3+2X+1$. The result is $3X^2+4X+3$. The 'Choose operation' dropdown is set to 'Derivative of A'.
- Bottom Screenshot:** Polynomial A is X^3+2X^2+3X+4 , Polynomial B is $3X^3+2X+1$. The result is $0.25X^4+0.67X^3+1.5X^2+4.0X$. The 'Choose operation' dropdown is set to 'Integration of A'.

$$A: X^3+2X^2+3X+4$$

$$B: 3X^3+2X+1$$

$$\text{Expected result: } 4X^3+2X^2+5X+5$$

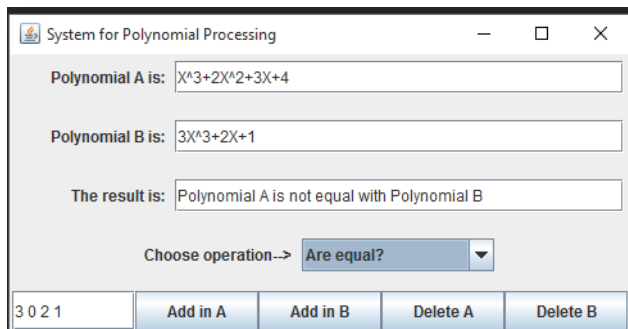
$$\text{Input: } X^3+2X^2+3X+4$$

$$\text{Expected result is: } 3X^2+4X+3$$

$$\text{Input: } X^3+2X^2+3X+4$$

$$\text{Expected result is:}$$

$$0.25X^4+0.67X^3+1.5X^2+4.0X$$



A: X^3+2X^2+3X+4

B: $3X^3+2X+1$

Expected result: Polynomial A is not equal with Polynomial B

4.1 Algorithms

The algorithms used are mathematical ones that perform polynomial operations. The algorithms of these operations deal only with the coefficients of the polynomial. For example if we add 2 polynomials of different size, we create a new polynomial of a size equal to the maximum of the sizes of the two polynomials received by the method as an argument. When this operation we ensure there will be no overflow. Then we pass through the first polynomial to add the coefficients to the new formed polynomial. The same operation is repeated for the second polynomial. At the end we create a polynomial with the new created array of coefficients, that would be the result of the method.

There are implemented algorithms that perform addition, subtraction and multiplication of 2 polynomials, an algorithm that performs the scalar multiplication of a polynomial with a scalar number. Besides these algorithms that describe operations that take two operands, there is also implemented an algorithm that computes the derivative of a polynomial and one that computes a primitive of a polynomial. Some of these algorithms have a low complexity, addition, subtraction, scalar multiplication, derivative and computing primitive having an $O(n)$ complexity, while the multiplication of two polynomials has an $O(n^2)$ complexity.

5. Results

The application is a user friendly and useful application to perform basic polynomial operations such as: addition, subtraction, multiplication, division, differentiation and integration. Output results of the application are actually the results of operations performed on the data the user provided as input, and is displayed in a text field just below the input data, as can be seen from the picture that shows the graphical user interface. The results are presented in the form of string into a text field that is editable, user can copy the text string from that field and it can be used wherever the user wants. Even though being limited, this application can be considered as being a helpful tool that can be used when dealing with such polynomial operations.



6. Conclusion and future developments

Achieving such a program may be hard both in terms of algorithms and graphical structure. Also this project was required mathematical knowledge, but no more than the properties of polynomials and operations for them. Attention to details and conditions is important, especially the way we read or write a polynomial. Each method has a high degree of difficulty, but not without a solution. Analysis, design and modeling are important steps to achieve a successful final product.

The interface could be upgraded by adding a menu. The most interesting, but also the hardest task could be to plot graphs for polynomial functions also to add some extra operations like finding the root. It could be really useful. Another important feature would be the possibility to save data from computations for further use.

7. Bibliography

<http://stackoverflow.com/>

<https://ro.wikipedia.org/wiki/Polinom>

<http://docs.oracle.com/>

<http://users.utcluj.ro/~jim/OOPE/>

