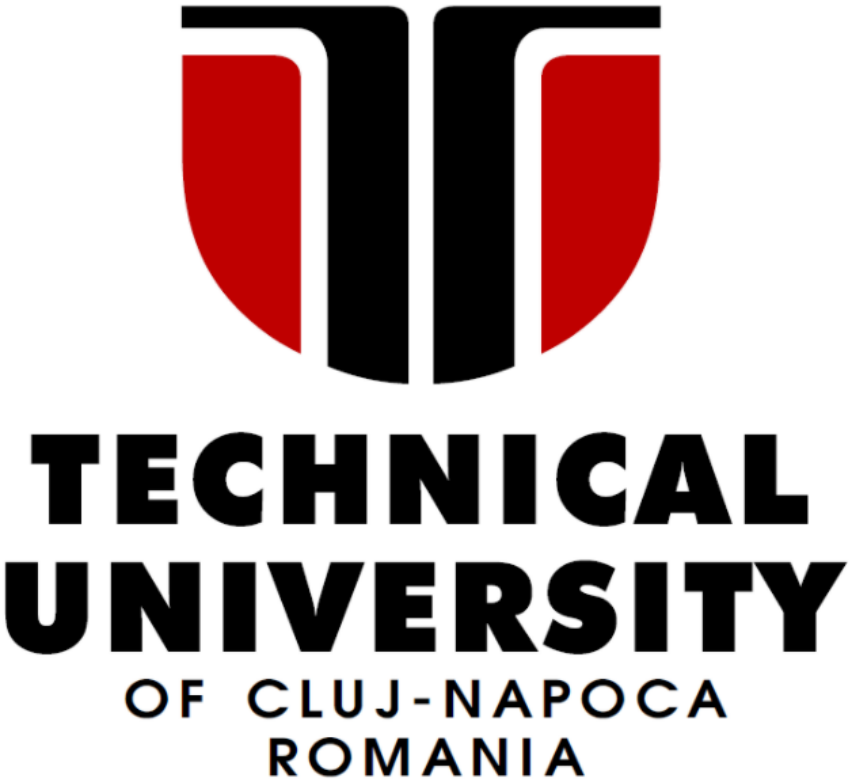
Technical University of Cluj-Napoca

Fundamental Programming Techniques

Laboratory assignment no. 1

Polynomial Calculator



Teacher: prof. Cristina Bianca Pop

Teacher Assistant: Andreea Valeria Vesa

Student: Dobner Szabolcs-Imre

Group:30424

1. Objective of the theme

The objective of the theme is to design a polynomial calculator, having the basic operations between two polynomials, having also the differentiation and integration of the polynomial included. The steps of creating such a polynomial calculator is designing the calculator and implementing the operations in such way that the operations work.

2. Problem analysis, modelling, scenarios, use cases

a) Problem analysis

In order to analyze the problem, we need to ask ourselves: What is a polynomial? Well ,Polynomial comes from poly- (meaning "many") and -nomial (in this case meaning "term") ... so it says "many terms". A polynomial can have the following elements: constants(coefficients), variables(such as X, Y, Z, etc.) and exponents (X^2). Its representation is the following:

$$P(x)= a_1x^n+ a_2x^{n-1}+a_3x^{n-2}+.....+a_nx^0$$

$A_{1....n}$-constants(coefficients)
n…..0-powers of x

So we know that polynomial means many terms, but this means that we need to know also what is a term in a polynomial. As mathematicians stated that a term in a polynomial is named a monomial. A monomial is also built by a constant(coefficient), variable and exponent. For example, $2x^5$ is a monomial. Its coefficient is 2 and the exponent(power) is 5. Having a sum of more monomials we get a polynomial ($3x^4 + 9x^1$ -3 is a polynomial). In the example shown in the parenthesis, this polynomial is a sum of 3 monomials. Although the last element has the exponent 0(meaning that the variable is not present either), it is still considered as a monomial.

With the help of the example given at how a polynomial is created generally we can perform operations on polynomials such as: addition, subtraction, multiplication, differentiation, integration and division.

b) Modelling

The user has 2 fields where he can introduce the polynomials in the graphical user interface. After that, the person can press any of the buttons below the input fields, there are 6 buttons for the 6 operations:

-Addition of two polynomials

-Subtraction of two polynomials

-Multiplication of two polynomials

-Division of two polynomials

-Differentiation of a polynomial(in this case, only the first field is considered)

-Integration of a polynomial(in this case, only the first field is considered)

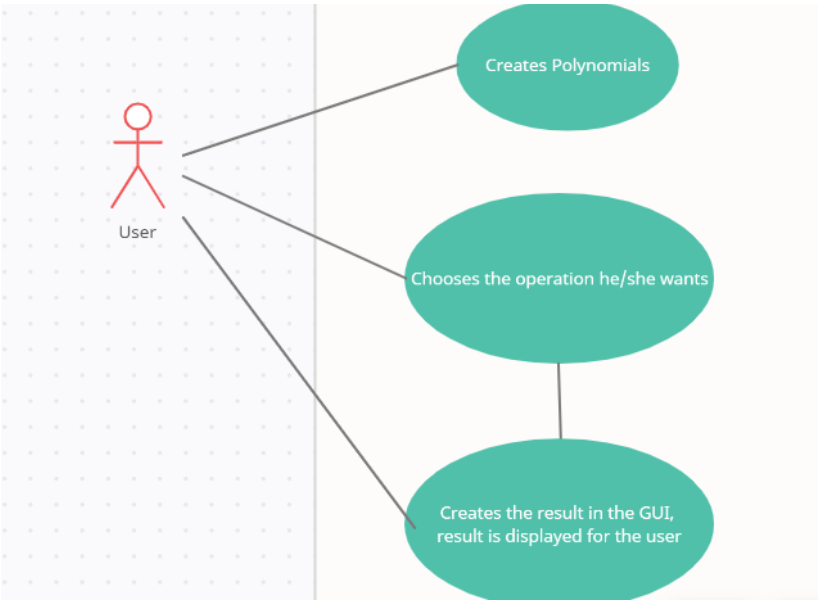
The result of the operation will be displayed below the buttons, according to the operation chosen. In the case of displaying the result, all operations except the integration will have the coefficients displayed as an integer number, and in the case of the integration the result's coefficients will be displayed as floating numbers.

c) Use cases and scenarios

Use cases are methodologies and processes used to review and analyze systems, like software platforms. You conduct use cases to help determine, interpret and organize the requirements and features in a system and how they get used. For example, a university developing new software for students to register for classes electronically might product use cases to show how the system works ideally and brainstorm potential misuses, issues or malfunctions that might occur.

The use cases in this application is in strong correlation with the user's steps. The graphical user interface is created in a rather friendly manner so the user gets all the information needed on how to write down the polynomials in order to have the operations working. For this, the user must pay attention to the instructions given in the GUI. After the polynomials are written down correctly, the user can do the operation he/she wants. If one of the polynomials or both polynomials are not written correctly, the GUI will display an error message regarding the input miswriting.

The use case diagram is the following:



In the choosing of operation he/she wants there are 6 possibilities of choosing:

-Addition of 2 polynomials

-Subtraction of 2 polynomials

-Multiplication of 2 polynomials

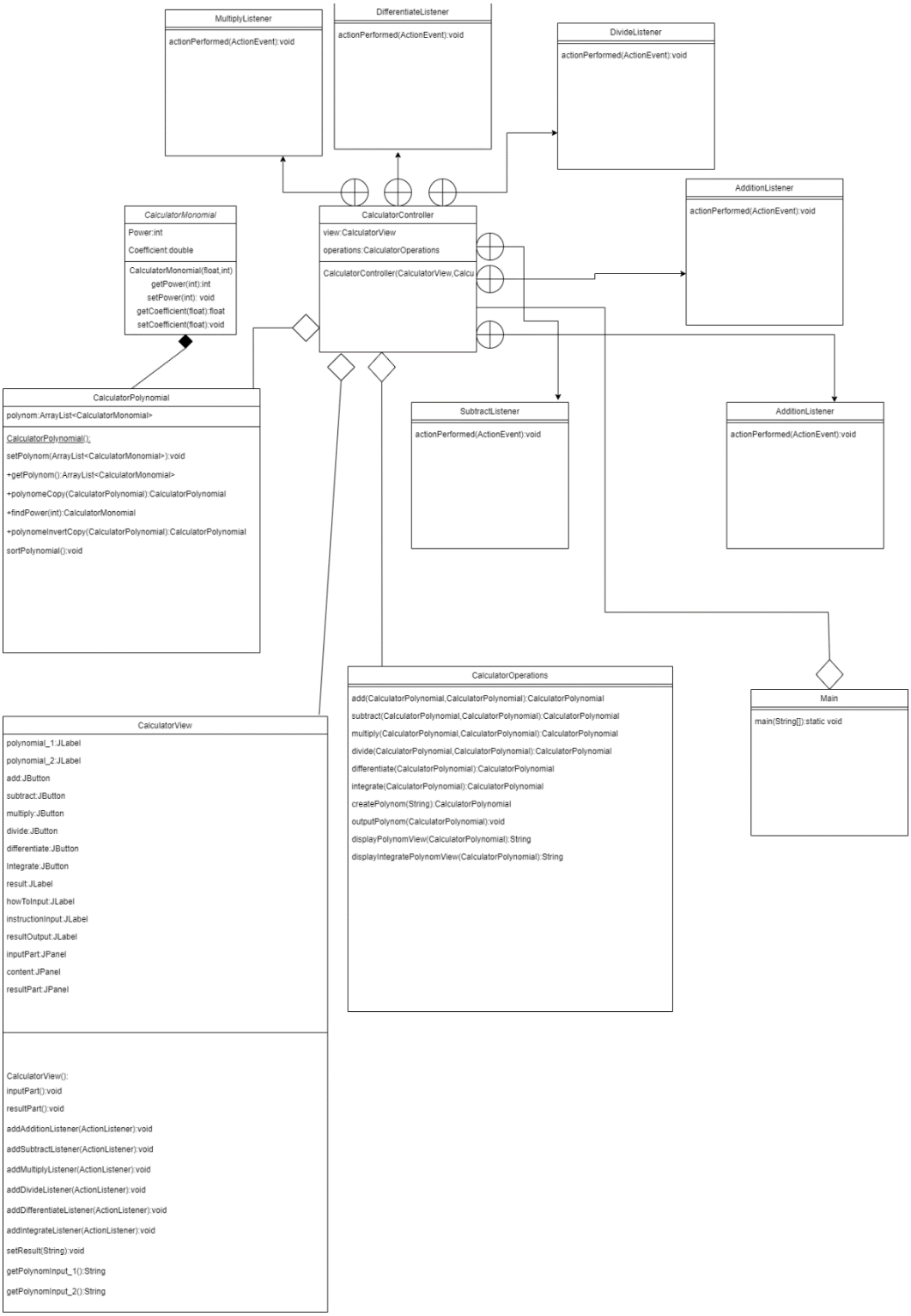
-Division of 2 polynomials

-Differentiation of a polynomial

-Integration of a polynomial

3. Projecting the application

a) Class diagram, package diagram



On this project I used the Model View Controller pattern of programming. The MVC architecture pattern turns complex application development into a much more manageable process. It allows several developers to simultaneously work on the application. The MVC pattern helped me break up the frontend and backend code into separate components. This way, it's much easier to manage and make changes to either side without them interfering with each other.

So on what components is MVC built on?

As its name suggests, MVC(Model View Controller) contains three parts: Model(the backend of the application which contains all the data logic), View(The GUI, the front-end of the application) and the Controller(the brains of the application which controls how data is displayed)

Another advantage of using the MVC pattern is the reusability of the code and allows to distinct the front-end from the back-end, therefore reducing time with debugging or code solving.

My project is also based on the Model View Controller pattern, I splitted these classes into three packages + the main(on which we run the GUI)

Model: Contains the brains of the application, the classes which model the problem and solve the operations requested by the user. In my application the model package includes three classes:

1) Monomial
2) Polynomial
3) Operations

View: Contains a single class which represents the Graphical User Interface

Controller: Makes the connection between the view and the model

b) Data structures used in the application

There are not many data structures used in the application due to its simplicity, primarily primitive data types were used: float and integer(for the monomial construction), but for the polynomial I used the ArrayList collection for the creation of multiple monomials together.

The reason why I used ArrayList instead of a classic array is that they have more freedom and is easier to use. Let's compare the two to understand better:

Firstly, the classical array needs to have its capacity specified whereas the ArrayList does not need to have that. This means that we use as much space as we need, not more than we need. Also, if we have a case where the array needs more space than its capacity, it results in an error because the array's capacity cannot be changed once declared initially, so it makes it harder to solve afterwards. ArrayList, on the other hand, has "infinite" capacity, does not have a capacity cap.

Another argument which makes ArrayList better than a classical array is that removing an element takes only 1 line of code comparing to multiple lines of code. There are multiple methods prewritten which makes only our job easier and less complex.

c) Used algorithms for the implementation of operations

There are many ways of conducting the operations on polynomials, it may be common that a lot of people made the algorithms in the same way as others, and this means that maybe I do not have a unique solution if compared to the others. For the addition, subtraction and multiplication I made the operation in such a way that I "pre-created" a number of instances which is equal to the highest power between the two polynomials. If we take for example polynomial 1: 3*X^3 +2*X^2 +1*X^1 +3 and 2*X^2 +7*X^1 +9, the

highest power between these two polynomials is 3, so I create 4 “empty instances” for the new polynomial (initially, I add power 0 and coefficient 0 to them). After that, I compare each instance from the first polynomial with the second one and if there is a case where the powers are the same, I change the result with the updated coefficient and power(this is only in the case of the multiplication).

In the case of the differentiation and integration, I only updated the coefficients and the power with respect to the operation and added them into the new polynomial.

In my opinion, another important algorithm that needs to be mentioned is the creation of the polynomial from a String. For this I need to split the polynomials into monomials and add them one by one into the polynomial. For the splitting I used the split() method which splits the String into different pieces. As a parameter of split() I specified the space as the splitting factor. After that, I get each monomial’s coefficient and power and add it to the polynomial. This is the main reason why I introduced in the GUI some explanations on how to input the polynomials in such a way the operations work.

4. Implementation

a) Class Design

By following the MVC pattern, my application has 3 parts(model, view, controller)+ the main function. In total I have 6 classes(if main is included too, otherwise only 5)

1) Model

In the model there is the back-end of the application, the logic applied to the program.

In this part, I have 3 classes implemented

a) Monomial class

As I mentioned earlier(2. Point, section a)), a polynomial is translated into “many terms”, and a polynomial is created by having more monomials together.

The monomial class has only 2 attributes: power and coefficient, having the type int and float respectively.

Constructor:

-public CalculatorMonomial(float coefficient, int power); this is the part where we initialize the monomial with the power and coefficient transmitted.

Methods:

- public float getCoefficient(): gets the coefficient of the monomial

- public void setCoefficient(float coefficient): sets a new value to the monomial’s coefficient

- public int getPower(): gets the power(degree) of the monomial

-public void setPower(int power): sets a new value to the monomial’s power(degree)

b) Polynomial class

This class has only 1 attribute: An ArrayList which has the type monomial(an array of monomials). It is important to mention that the order of the polynomial is descending, so the highest power is the first element of the ArrayList.

Constructor:

- public CalculatorPolynomial(): default constructor

Methods:

- public ArrayList<CalculatorMonomial> getPolynom(): gets the polynom

- public void setPolynom(ArrayList<CalculatorMonomial> polynom): sets a new polynom

- public CalculatorPolynomial polynomeCopy(CalculatorPolynomial p1): makes a copy of the polynom introduced in the method's argument

- public CalculatorMonomial findPower(int power): searches the entire polynomial for the desired power

- public CalculatorPolynomial polynomeInvertCopy(CalculatorPolynomial p1): makes a copy of the polynom but with inverted coefficients

- public void sortPolynomial(): orders the polynomial in such a way that the first monomial in the polynomial is the one with the highest power

c) Operation class

This class has no attributes and constructor, only methods. In this class we have the operations logic and solving.

Methods:

- public CalculatorPolynomial add(CalculatorPolynomial p1, CalculatorPolynomial p2): addition of two polynomials

- public CalculatorPolynomial subtract(CalculatorPolynomial p1, CalculatorPolynomial p2): subtraction of two polynomials

- public CalculatorPolynomial multiply(CalculatorPolynomial p1,CalculatorPolynomial p2): multiplication of two polynomials

- public CalculatorPolynomial divide(CalculatorPolynomial p1, CalculatorPolynomial p2): division of two polynomials

- public CalculatorPolynomial differentiate(CalculatorPolynomial p): differentiation of the first polynom

- public CalculatorPolynomial integrate(CalculatorPolynomial p): integration of the first polynom

- public CalculatorPolynomial createPolynom(String polynom): creation of the polynom

- public void outputPolynom(CalculatorPolynomial polynome): display the polynom in the console

- public String displayPolynomView(CalculatorPolynomial polynome): display the polynom in the GUI but with integer coefficients

- public String displayIntegratePolynomView(CalculatorPolynomial polynome): display the polynom in the GUI but with float coefficients

2) Controller

In the controller we have the link between the view and the model. In the controller we make the changes in the GUI.

Constructor:

public CalculatorController(CalculatorView view, CalculatorOperations operations): default constructor for the controller, additionally in the constructor there are the listeners added for each operation.

Methods(inner classes):

- public class AdditionListener implements ActionListener

- public class SubtractListener implements ActionListener

- public class MultiplyListener implements ActionListener

- public class DivisionListener implements ActionListener

- public class DifferentiateListener implements ActionListener

- public class IntegrateListener implements ActionListener

3) View

I created the GUI using java swing because I have no experience yet with javaFX. The class extends JFrame, I chose this because it was rather easy to determine its size, defaultCloseOperation and so on.

Constructor:

-public CalculatorView(): In this part I set the size and the defaultCloseOperation of the interface and set 2 parts for the GUI: the input part(input of polynomials + buttons and the result part along with the instructions on how to write the polynomials)

Methods:

- private void inputPart(): I create the input part of the GUI

- private void resultPart(): I create the result part of the GUI

- public String getpolynomInput_1(): gets the first polynomial's String

- public String getpolynomInput_2(): gets the second polynomial's String

- public void setResult(String results): sets the result in the GUI

The rest of the methods are strictly made for the controller, the button listeners for each operation (addition, subtraction, multiplication, division, differentiation, integration)

- public void addAdditionListener(ActionListener mal)

- public void addMultiplyListener(ActionListener mal)

-public void addDivideListener(ActionListener mal)

-public void addSubtractListener(ActionListener mal)

-public void addDifferentiateListener(ActionListener mal)

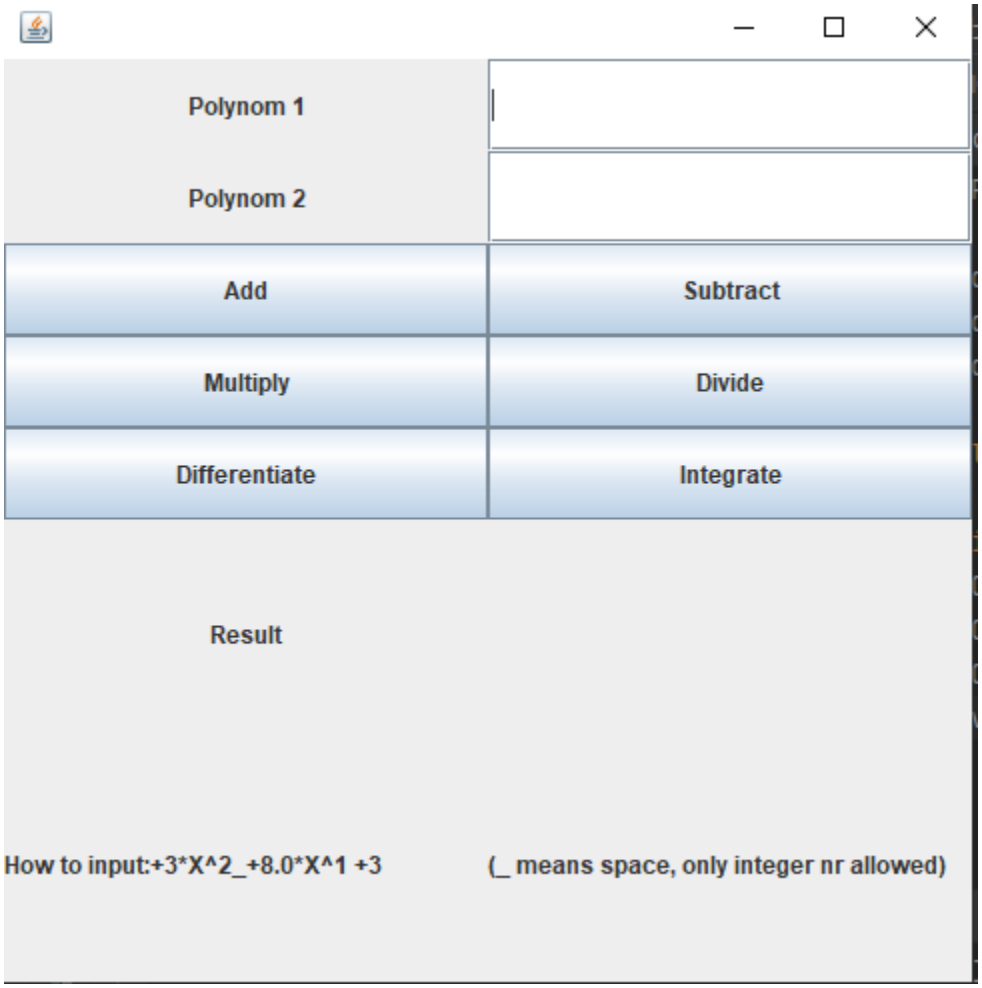
- public void addIntegrateListener(ActionListener mal)

4) Main

Contains the declaration of the object view, operations and controller and starts the GUI.

b) Graphical User Interface

The graphical user interface has the purpose of connecting the user with the application. The user can input some data and then choose any operation desired. The user has some obligations which needs to be respected, such as the input condition. There is an explanation on how to write down the polynomial in order to work.



After writing down the polynomials and choosing the operation desired, the result will be displayed on the right of the Result label. Important is to mention that in the case of integration, the numbers displayed in the GUI will be floating types whereas the rest will be integer types.

5. Results

The results of the operations is tested with the help of Junit testing, there are 5 methods created for the Junit: add, subtract, multiply, differentiate, integrate(I do not have divide since I did not manage to make it ☹)

What I test	Input Data	Expected Output	The effective result	Pass / Fail
Addition	+3*X^2 +2*X^1 +6; +3*X^1 +7	+3*X^2 +5*X^1 +13*X^0	+3*X^2 +5*X^1 +13*X^0	Pass
Subtraction	+3*X^2 +2*X^1 +6 ; +3*X^1 +7	+3*X^2 -1*X^1 -1*X^0	+3*X^2 -1*X^1 -1*X^0	Pass
Multiplication	+3*X^2 +2*X^1 +6 ; +3*X^1 +7	+9*X^3 +27*X^2 +32*X^1 +42*X^0	+9*X^3 +27*X^2 +32*X^1 +42*X^0	Pass
Differentiation	+3*X^2 +2*X^1 +6; +3*X^1 +7	+6*X^1 +2*X^0	+6*X^1 +2*X^0	Pass
Integration	+3*X^2 +2*X^1 +6; +3*X^1 +7	+1.5*X^3 +2.0*X^2 +6.0*X^1+C	+1.5*X^3 +2.0*X^2 +6.0*X^1+C	Pass

The use case diagram shows how to procedure of making the operations is made.

6. Conclusion, further improvements

In conclusion, this project taught me a lot of things: first things first is that never procrastinate, if you have time then start working on the projects because you won't have enough time to finish it, secondly, I consider this project as very educative, I learned a lot of new things about OOP which I did not know of. Thirdly, this project helped me understand that the OOP is not so simple and there are a lot of methods of implementing an application, the only thing which we need is time and focus.

As further improvements I would say that the division operation is the number one priority, although it is time consuming and hard to implement it, it is not impossible and it would bring a lot of improvements to me as a programmer, also the GUI looks too simplistic so the second thing which needs further improvement is the aesthetic of the GUI, so it looks more user friendly and colorful. Furthermore, in this project there were only 6 operations requested, but there can be more operations added into it such as: computing the roots of the polynomial, compute the polynomial if X has a certain value, display the polynomial on the screen.

7. Bibliography

-Fundamental Programming Techniques lectures
-OOP lectures
- https://creately.com for use case diagram

-draw.io for class diagram
- www.javatpoint.com
-stackoverflow.com