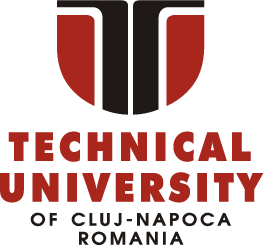
Technical University of Cluj-Napoca

Programming Techniques

Laboratory – Assignment 1

Polynomials Calculator

**

Teacher: prof. Ioan Salomie

Teacher Assistant: Cristina Bianca Pop

Student: Fleser Mihai

Group: E\_30424

1. Assignment Objective

Propose, design and implement a polynomial calculator with a dedicated graphical interface through which the user can insert polynomials, select the mathematical operation (i.e. addition, subtraction, division, derivative, integration) to be performed and view the result. Consider the polynomials of one variable and integer coefficients.

Secondary Objectives:

* Analyze the problem and identify requirements …………………...………………. Chapter 2
* Design the polynomial calculator …………………………………………………. Chapter 3
* Implement the polynomial calculator …………………………………………….. Chapter 4
* Test the polynomial calculator …………………………………………………….. Chapter 5

1. Problem analysis, modeling, scenarios, use cases
   1. *Analyzing the problem*

PROBLEM: Performing polynomial operations on paper is often difficult and time consuming.

A polynomial is an [expression](https://en.wikipedia.org/wiki/Expression_(mathematics)) that can be built from [constants](https://en.wikipedia.org/wiki/Constant_(mathematics)) and symbols called indeterminates or variables by means of [addition](https://en.wikipedia.org/wiki/Addition), [multiplication](https://en.wikipedia.org/wiki/Multiplication) and [exponentiation](https://en.wikipedia.org/wiki/Exponentiation) to a [non-negative integer](https://en.wikipedia.org/wiki/Non-negative_integer) power.

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

P = a0 + a1x + a2x^2 + a3x^3 +…+ anx^n

where a0, a1, a2, …, an are constants and x is the indeterminate. The word "indeterminate" means that x represents no particular value, although any value may be substituted for it.

The polynomial consists of a list of certain terms, also called monomials, for example 3x^2 is a monomial. The coefficient is 3, the indeterminate is x and the degree is 2. Forming a sum of several terms produces a polynomial, like the following one: 3x^2-5x+4, which has three terms with different exponents:  the first is degree two, the second is degree one, and the third is degree zero.

So, we can say that polynomial is composed of one or more monomials aixi with i in the range [0, n] where n is the degree of the polynomial.

This way of representing polynomials can be used to perform the most common operations on polynomials: addition, subtraction, multiplication, division, differentiation, and integration.

1. Modelling the problem

The user will be able to use the functions of the calculator by introducing two polynomials in the interface.

After that, he will have to press a specific button for each operation he wants to perform. The operations that my calculator allows are:

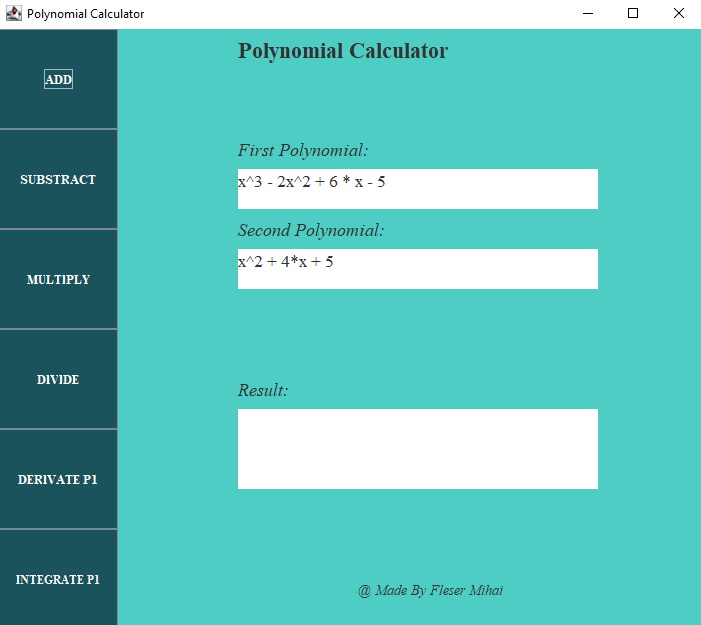
* Addition of two polynomials
* Subtraction of two polynomials
* Multiplication of two polynomials
* Division of two polynomials
* Differentiation of a polynomial
* Integration of a polynomial

The result of the chosen operation will be displayed in the interface. If the polynomials introduced are incorrect or are in an unaccepted form, the application will return an error together with a hint on how to solve it.

1. *Different scenarios and use cases*

Use cases are a technique for capturing, modelling and specifying the requirements of a system. A use case corresponds to a set of behaviors that the system may perform in interaction with its actors, and which produces an observable result that contribute to its goals. Actors represent the role that human users or other systems have in the interaction.

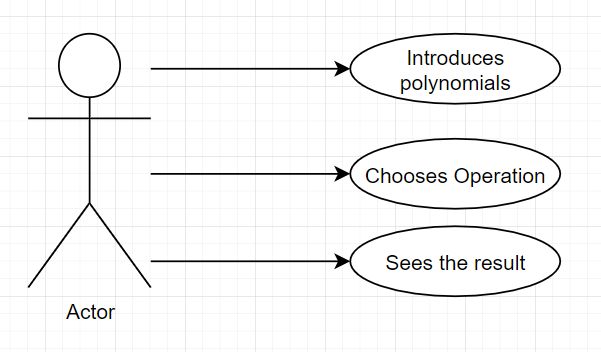
The use cases are strongly connected with the user steps. I tried to make the user interface as friendly as I could and the result looks like this:



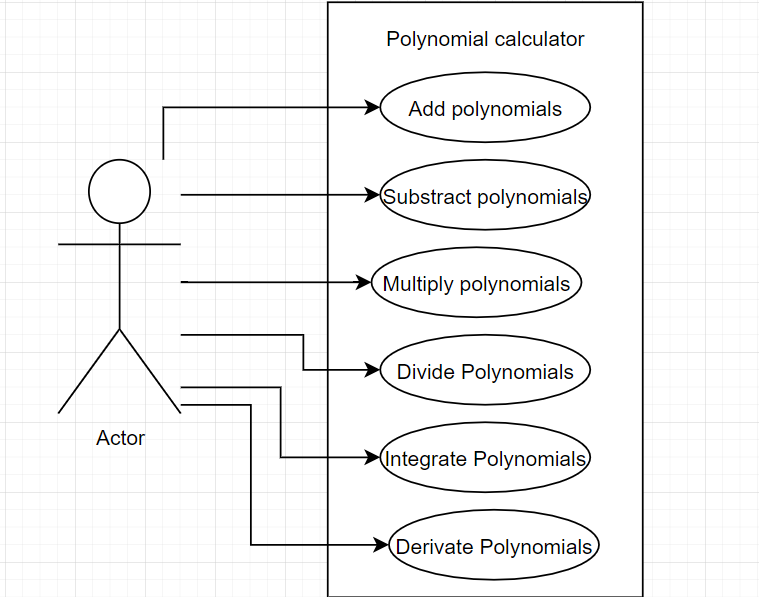
The user will introduce the two polynomials in the corresponding text fields. After that, he will press a button corresponding to which operation he wants to do. The result operation will appear in the result text field below.

The user must pay attention to follow the format of the polynomial, which appears in the GUI. If he does not follow this format, does not enter any polynomial or tries to divide with 0, an error will appear on the GUI together with a hint on how to solve it.

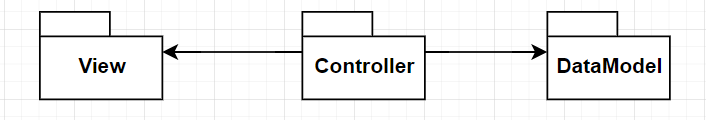
Use case Diagram:



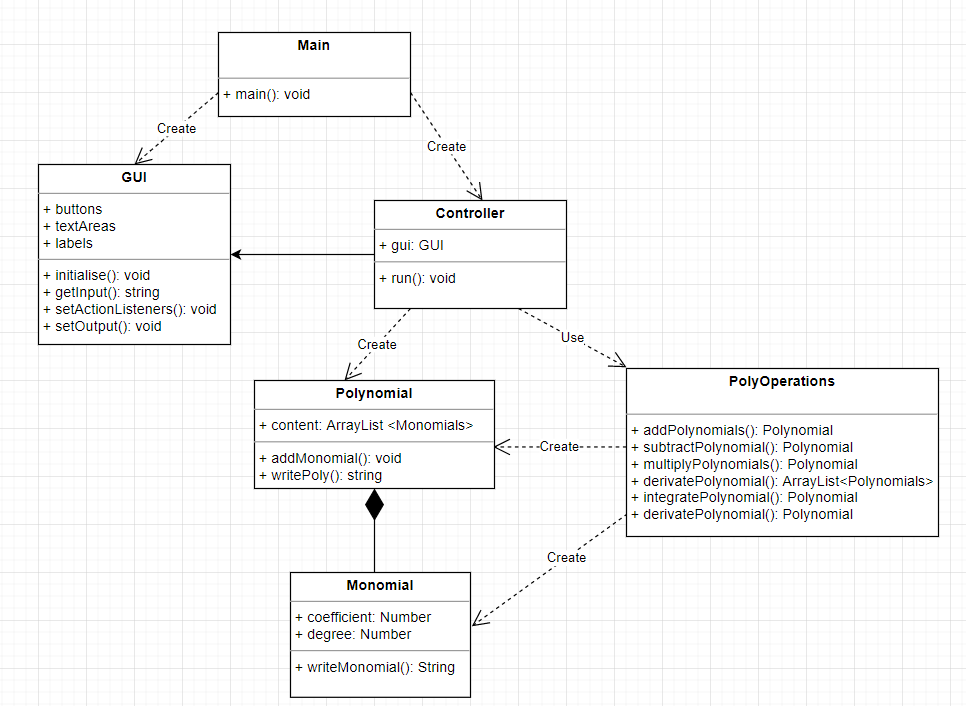
The use case presents the actor, which in our case is the user that interacts with the application. She/ He can perform several actions on the two chosen polynomials, such as addition, subtraction, multiplication, division, integration and differentiation.



* 1. Design
     + 1. *UML package diagram*

**

* 1. *UML class Diagram*

**

* 1. *Data Structures*

For my project, I have used the following data Structures: Number, ArrayList and String.

I have used Number for the coefficients and the degree of each Monomial. I decided to do this because, at input, the Polynomial are expressed with Integer coefficients, but if the user wants to integrate or divide, the output will often have float coefficients.

I decided to use ArrayList instead of classic arrays because ArrayList does not have a fixed length as normal arrays do, thus making them more memory efficient. Also, I do not have to worry about exceeding the length of the array.

I used String for writing the Polynomials from the internal program to the user or for receiving the input from the user and then translating it into a Polynomial inside my program.

* 1. *Design Decisions and Packages*

I have done my project with the Design Pattern named model-View-Controller.

**Model–view–controller** (usually known as **MVC**) is a software design pattern commonly used for developing user interfaces that divides the related program logic into three interconnected elements. This is done to separate internal representations of information from the way information is presented to and accepted from the user.

**The Model** represents the central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.

**The View** represents a collection of classes representing the elements in the user interface (all of the things the user can see and respond to on the screen, such as buttons, display boxes, etc.)

**The Controllers** receive input, usually as events that denote mouse movement, activation of mouse buttons or keyboard input. Events are translated to service requests, which are sent either to the model or to the view.

Therefore, I have created three packages containing one or more classes:

DataModel – the “brain” of the calculator, contains classes which store the data and perform different operations on it.

* Monomial class
* Polynomial class
* PolyOperations class

View – one class – The GUI class- interacts with the user.

Controller – one class – the Controller class: connects The Model with the GUI

The entry point of the application in the Main class, which is separated from the others packages.

1. *Class Design*

As I said earlier, my Program consists of 3 parts, following the MCV architecture.

The Data Model:

* Monomial class – has 2 fields, meaning the coefficient and the degree
* Polynomial class – consists of one or more Monomials. There does not exists two or more monomials in the same polynomial with the same degree. Between Polynomial and Monomial exists a one to many relationship as the Polynomial is composed of multiple monomials.
* PolyOperations class – takes one or two polynomials and performs addition, substraction, division, multiplication, derivation or integration on the polynomials returning the result.

The View:

* Gui class – interacts with the user, takes the input and shows the output

The Controller:

* Controller class – Handles the events that happens on the GUI, receives the input from the GUI and writes the output that the user wants to see. Communicates with both the Model and the GUI.

It has an one to one relationship with the GUI.

The main class is the start point of the application, just creates a new Controller which will do the rest.

1. *Algorithms*

* Addition: is done by adding the coefficients of the monomials with the same degree from the two polynomials.
* Subtraction: is done by subtracting the coefficients of the monomials with the same degree from the two polynomials.
* Multiplication: To multiply two polynomials, multiply each monomial in one polynomial by each monomial in the other polynomial, add the results and simplify if necessary.

**Example:** Consider the following two polynomials:

The result of multiplying the two polynomials is:

* Division: To divide two polynomials *P* and *Q*, the following steps should be performed:

**Step 1** - Order the monomials of the two polynomials *P* and *Q* in descending order according to their degree.

**Step 2** - Divide the polynomial with the highest degree to the other polynomial having a lower degree (let’s consider that *P* has the highest degree)

**Step 3** – Divide the first monomial of *P* to the first monomial of *Q* and obtain the first term of the quotient

**Step 4** – Multiply the quotient with *Q* and subtract the result of the multiplication from *P* obtaining the remainder of the division

**Step 5** – Repeat the procedure from step 2 considering the remainder as the new dividend of the division, until the degree of the remainder is lower than *Q*.

**Example:** Consider the following two polynomials:

The result of dividing the two polynomials is:

(X3 - 2\*X2 + 6\*X – 5) : (X2 – 1) = X – 2

-X3 + X

- 2\*X2 + 7\*X – 5

2\*X2 – 2

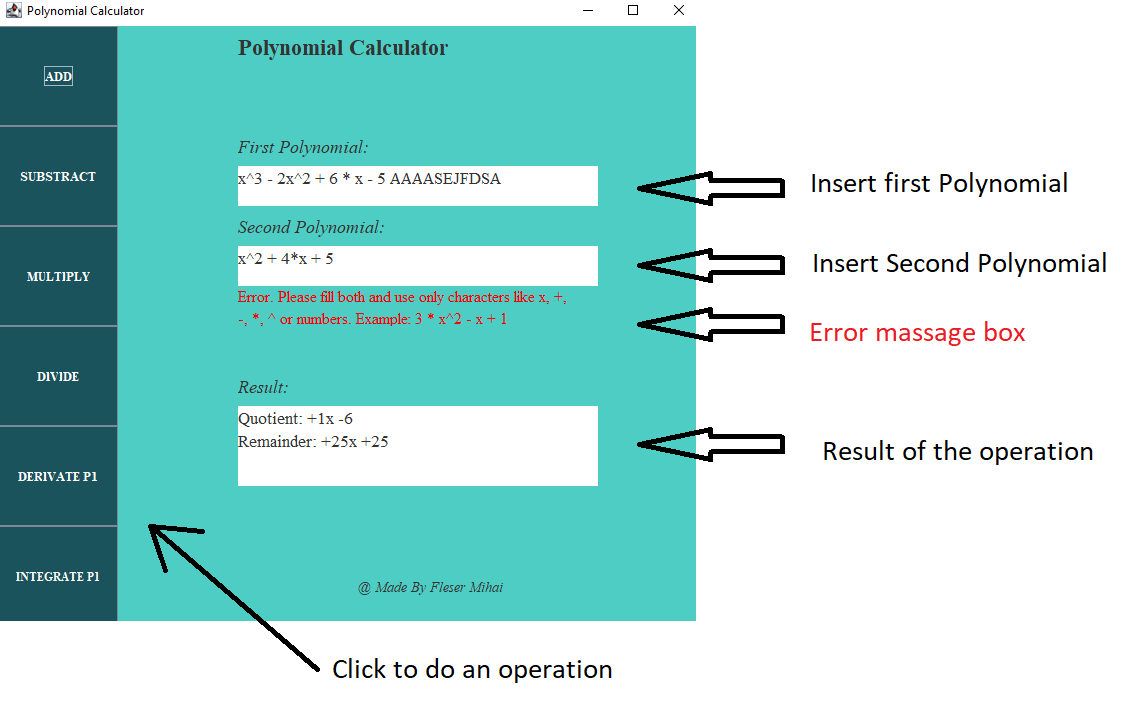
7\*X – 7 Quotient = X – 2; Remainder = 7\*X-7

* Derivative:
* Integration:

The integral of a polynomial P is defined as follows:

where:

1. *User Interface*



* 1. Implementation

Following the MVP pattern, I have three packages: DataModel ,View, Controller.

In the DataModel package, I have implemented the following classes:

* Monomial Class

Attributes:

* private Number coefficient
* private Number: degree

Constructors:

* public Monomial (Number coefficient, Number Degree) – sets the coefficient and degree
* public Monomial (String input) – takes a String as input and transforms it into a Monomial

Methods:

* public Number getCoefficient() – returns the coefficient
* public void setCoefficient(Number coefficient) – sets the coefficient
* public Number getDegree() – returns the degree
* public String writeMonomial() – returns the monomial as a String
* Polynomial Class

Attributes:

* private ArrayList<Monomial> content – contains the list of monomials

Constructors:

* public Polynomial(Polynomial p) – makes a new polynomial identical to p
* public Polynomial(String input) – takes the input as a String and makes a polynomial out of it

Methods:

* public void setContent(ArrayList content) – sets a new content for the Polynomial
* public ArrayList<Monomial> getContent() – returns the content
* public String writePoly() – returns the Polynomial as a String
* public void AddMonomial(Monomial m) – adds a new Monomial to the list inside the Polynomial
* PolyOperations Class

Methods:

* public static Polynomial addPolynomials(Polynomial p1, Polynomial p2) – adds 2 polynomials and returns the result
* public static Polynomial substractPolynomials(Polynomial p1, Polynomial p2) – subtracts 2 polynomials and returns the result
* public static Polynomial multiplyPolynomials(Polynomial p1, Polynomial p2) – multiplies 2 polynomials and returns the result
* public static ArrayList<Polynomial> dividePolynomials(Polynomial p1, Polynomial p2) – divides 2 polynomials and returns the quotient and the remainder
* public static Polynomial derivatePolynomial(Polynomial p1) – derivates the first polynomial and returns the result
* public static Polynomial integratePolynomial(Polynomial p1) – integrate the first polynomial and returns the result

In the Controller Package, I have implemented:

* Controller class

Attributes:

* Private GUI gui

Constructors:

* Public Controller(GUI gui) - receives a GUI

Methods:

* public String addPolyomials(String first, String second) – receives 2 strings, converts them to polynomials, makes the addition, then returns the result also as a string.
* public String substractPolynomials(String first, String second) – does substraction in the same manner
* public String multiplyPolynomials(String first, String second) – does the multiplication
* public String dividePolynomials(String first, String second) – does the division
* public String derivatePolynomial(String first) – does the derivation
* public String integratePolynomial(String first) – does the integration
* public void run() – the main method of the controller – adds listeners on the GUI buttons which read the input, check for errors and write the output for the user.

The main Class only creates a new Controller and then call the function controller.run()

In the View Package I have implemented:

* GUI class

Attributes:

* private JTextArea p1Text = new JTextArea();
* private JTextArea p2Text = new JTextArea();
* private JLabel errorLabel = new JLabel("");
* private JTextArea p3Text = new JTextArea();
* private JButton addition=new JButton("ADD");
* private JButton substract=new JButton("SUBSTRACT");
* private JButton multiply=new JButton("MULTIPLY");
* private JButton divide=new JButton("DIVIDE");
* private JButton derivate=new JButton("DERIVATE P1");
* private JButton integrate=new JButton("INTEGRATE P1");

Methods:

* public String getFirstPoly()
* public String getSecondPoly()
* public String getResultText()
* public void setResultText()
* public void setErrorText()
* The controller adds all the following Listeners on the GUI:
* public void setActionOnAdditonButton(ActionListener actionListener) -
* public void setActionOnSubtractionButton(ActionListener actionListener)
* public void setActionOnMultiplyButton(ActionListener actionListener)
* public void setActionOnDivideButton(ActionListener actionListener)
* public void setActionOnIntegrateButton(ActionListener actionListener)
* public void setActionOnDerivateButton(ActionListener actionListener)
* public void Initialise()
* This method initialize the Jframe and adds all the components to it.
* As components I have 6 buttons corresponding to each operation, a text area and a label for each polynomial and a label for the errors that might appear.
* The components are positioned on the Jframe using component.setBounds. Then I set their font and color.
* The input, in order to be parsed correctly, must be in the format [sign][coefficient][x^][degree]. If it doesn’t follow the format, the application will show an error message.
  1. Results – I used Junit for testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| What I test | Input Data | Expected Output | The effective result | Pass / Fail |
| Addition | x^3-2x^2+6x – 6;  x^2 - 1 | +1x^3 -1x^2 +6x -7 | +1x^3 -1x^2 +6x -7 | Pass |
| Subtraction | x^3-2x^2+6x – 6;  x^2 - 1 | +1x^3 -3x^2 +6x -5 | +1x^3 -3x^2 +6x -5 | Pass |
| Multiplication | x^3-2x^2+6x – 6;  x^2 - 1 | +1x^5 -2x^4 +5x^3 -4x^2 -6x +6 | +1x^5 -2x^4 +5x^3 -4x^2 -6x +6 | Pass |
| Division | x^3-2x^2+6x – 5;  x^2 - 1 | Q: +1x – 2  R: +7x -7 | Q: +1x – 2  R: +7x -7 | Pass |
| Differentiation | x^3-2x^2+6x – 6; | +3x^2 -4x +6 | +3x^2 -4x +6 | Pass |
| Integration | x^3-2x^2+6x – 6; | +0.25x^4 -0.6666667x^3 +3x^2 -6x | +0.25x^4 -0.6666667x^3 +3x^2 -6x | Pass |

* 1. Conclusions

I liked this project because it helped me to further develop my OOP knowledge, as the first semester we didn’t do too much. I learnt a lot from this application also regarding the best practices when working at a project.

For example, I have learnt that time management is very, very important, and besides all, it’s crucial to work at the project every week, not to let it until the last days. Also, I have learnt that it’s much easier if you organize the classes in packages with specific roles, rather then just coding without any scope. Actually, this is the most important thing that I learned, to organize the classes and the code, as I had to partially redo some of my code because it was poorly organized.

Future inprovements:

* One improvement will be to let the user choose what Polynomial he wants to Integrate or Derivate as, for now, he can only do this with the first Polynomial.
* Another one will be to calculate the roots for small degree polynomials
* Besides, another one will be to work with float coefficients and negative degrees.
* Being able to plot the polynomial on a graph will also be nice

* 1. Bibliography

1. Object-Oriented Programming - Lecture Slides of prof. Marius JOLDOS
2. Programming Techniques – Lectures of prof. Ioan SALOMIE
3. <https://docs.oracle.com/javase/tutorial/uiswing/>
4. <https://docs.oracle.com/javafx/2/get_started/jfxpub-get_started.htm>
5. <https://www.baeldung.com/javafx>
6. <https://docs.oracle.com/javase/tutorial/essential/regex/>
7. <https://www.baeldung.com/regular-expressions-java>
8. <https://www.vogella.com/tutorials/JUnit/article.html>
9. <https://stackoverflow.com/>
10. <https://en.wikipedia.org/wiki/Polynomial>
11. <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>