**Assignment 1 Documentation**

**-Polynomial Calculator-**

1. **Homework objective:**

Main objective:

The task is to design and implement a polynomial calculator having one variable an integer coefficients.

Secondary objectives:

-the implementation of a dedicated graphical user interface through which the user can enter polynomials, select the operation to be performed and display the result.

-the implementation of the mathematical operations such as: addition, subtraction, multiplication, division, derivation and integration.

-the code should be designed according to the Model View Controller architectural pattern and following the OOP style.

1. **Problem analysis, modelling, scenarios, use cases:**

Problem analysis:

We need to have a correct and efficient implementation of the mathematical operations of addition, subtraction, multiplication, division, derivation and integration for polynomials of one variable and with integer coefficients. Also, we have to respect the corresponding mathematical rules for each operation.

Modelling:

This application needs only two models, Polinom and Monom, since between the two classes there exists an aggregation relationship because every polynomial is made from one or more monoms.

Scenarios:

Scenarios are sequences of steps that represents the interaction between the system and an actor. The main scenarios are:

1. First scenario:
2. Identification summary:

Title: The polynomials are not introduced.

Summary: In this scenario, both polynomials are not introduced by the user.

1. Flow of events:

- the user starts the application

-the user do not introduce any polynomial

-the user choose the operation to be performed

-an error message is displayed, because of the absence of one or both of the polynomials

2. Second scenario:

a.) Identification summary:

Title: There are introduced characters different from those available.

Summary: In this scenario, the user introduces characters which differ from the ones that should be introduced, such as: x, digits from 0 to 9, +, -, ^.

1. Flow of events:

-the user starts the application

-the user introduces the polynomials in the corresponding textboxes, but also he introduces some invalid characters.

-the user choose the operation to be performed

-an error message is displayed, because of that invalid character

3. Third scenario:

1. Identification summary:

Title: The user open the application and successfully made a mathematical operation.

Summary: In this scenario, the user choose an operation and it takes place without any errors.

1. Flow of events:

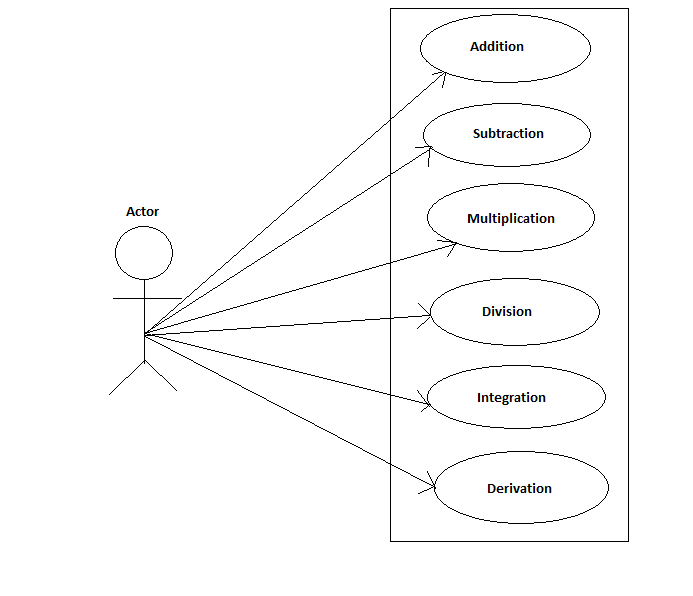
-the user starts the application

-the user introduce the first polynomial and the second one in the corresponding textboxes

-the user selects the operation to be performed

-the result is displayed without any errors

Use cases:



**3.Design:**

The application uses the format:

a1\*x^n + a2\*x^(n-1) + … + a(n-1)\*x + an, with float coefficients of 2 decimals, and integer powers.

The design of the application is based on the Model View Controller design pattern. The Model contains all the data-related logic that the user works with. The View contains all the UI logic of the application, representing the place where the user interacts with the application. The Controller acts as an interface between the Model and the View to process all the business logic and incoming requests, manipulate data using the Model and interact with the Views to render the final output, implementing the actions as a response to the user interaction.

Data structures:

The only data structure used for this assignment is the list. For the implementation of the List interface, it is used ArrayList.

Packages:

There are three packages: Controller, Model, View. In the Controller package, there is only the Control Class, which controls the actions taken by the user. In the Model package, there are main classes Monom and Polinom, and the rest of the auxiliary classes used for the mathematical operations. In the View package, there is only the GUI class, which consists of the UI logic for the application

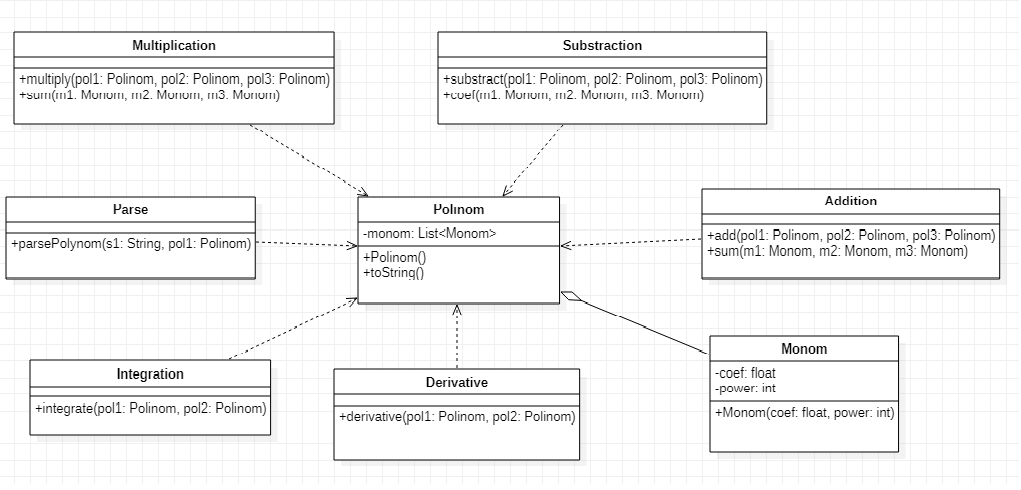
Class design:

The application is based on the main 2 classes: Polinom and Monom. Also, there are some auxiliary classes, such as Addition, Subtraction, Multiplication, Derivative, Integration and Parse. Every auxiliary class implements one mathematical operation, excepting the Parse class, which is used to get the input introduced by the user, expressed as a polynomial, not as a string.

* Monom : represents each element of the polynomial.
* Polinom: is represented as a set of one or more monoms.

The algorithms:

* For the addition operation, we used the merge algorithm. We start by comparing the powers of the first monom from each polynomial. If the two monoms have the same power, we create a new monom having the same power, but as a coefficient, there will be the sum of the two corresponding coefficients of the monoms, and we add the new monom to the solution, moving to the next monom in each polynomial. If one of the monoms has a greater power than the other one, we add to the solution the monom with the greater power, and also increase the index of the polynomial having the monom with a greater power. When a polynomial has no more monoms, but the other still has some, we keep adding the rest of the monomials to the solution.
* For the substraction, we used also the merge algorithm. We start by comparing the powers of the first monom from each polynomial. If the two monoms have the same power, we create a new monom having the same power, but as a coefficient, there will be the difference of the two corresponding coefficients of the monoms, and we add the new monom to the solution, moving to the next monom in each polynomial. If one of the monoms has a greater power than the other one, we add to the solution the monom with the greater power, and also increase the index of the polynomial having the monom with a greater power. When a polynomial has no more monoms, but the other still has some, we keep adding the rest of the monomials to the solution.
* For the multiplication, we multiply each monom form the first polynomial with each monom form the second polynomial, and add the resulting monoms into a new polynomial. After the multiplication, if there are monoms having the same power in the resulted polynomial, we add the coefficient to the ones having the same power.
* For the derivation, we used the mathematical formula: (a\*x^b)’ = a\*b\*x^(b-1).
* For the integration, we used the mathematical formula: (ax^n) = a/(n+1)x^(n+1).
* To get the two polynomials from the user, we have parsed the input.



User interface:

The user interface is designed as simple as possible, such as anyone will be able to use it. There are labels in the left of each textbox, for helping the user to put the right content in the right field. Also, there are button for every operation that can be performed, which have name on them which let the user know which button performs a certain operation.

**4.Implementation:**

We needed to implement the main classes Polinom and Monom, and also the auxiliary classes for each mathematical operation performed.

* Monom:

It represents the basic block for the problem. Here are stored the coefficient and the power of each term. The methods used are: the constructors, the setters and the getters.

* Polinom:

It is represented as an array list of class Monom. In the addition to the constructors and the setter and getter methods, this class contains also an overridden version of toString method, which transform into a string the array list of monoms.

* Addition:

The method implemented in this class is add(polynomial 1, polynomial2, polynomial3) which computes the addition of the two polynomials and store the result into a new polynomial. Also, there is a method sum(monomial m1, monomial m2, monomial m3) which computes the addition of the two monoms of the polynomials and store the result into a new monom. The merge algorithm is used to implement this method, which has been described in the algorithm description from section 3 of this documentation.

* Subtraction:

The method implemented in this class is sub(polynomial 1, polynomial 2, polynomial 3) which computes the subtraction of the two polynomials and store the result into a new polynomial. Also, there is a method sub(monomial m1, monomial m2, monomial 3) which computes the subtraction of the two monoms of the polynomials and store the result into a new monom. As the addition, the merge algorithm is used to implement this method, which has been described in the algorithm description from section 3 of this documentation.

* Multiplication:

The method implemented in this class is multiply(polynomial 1, polynomial 2, polynomial 3) which computes the multiplication of the two polynomials and store the result into a new polynomial. We begin by multiplying each monom from the first polynomial with each monom from the second polynomial and store the result into a new monom. After that, we add the new monom in the list of the new polynomial, then we sort the monoms of this polynomial in decreasing order and compare the powers of the monoms on the index i and i+1 in the list, merge the monoms that have the same power, using the method sum(monomial m1, monomial m2, monomial m3). After we merged the coefficients of the monoms having the same power, we verify if the new coefficient is different from 0, if it is, we set the monom in the array list, else we remove it from the list.

* Derivative:

The method implemented in this class is derivative(polynomial 1, polynomial 2) which computes for each monom its derivative form. This can be done by multiplying the coefficient with the power, then decrementing the power of each monom. We create a new monom, having a new coefficient and power, which will be added onto the list of the resulted polynomial.

* Integration:

The method implemented in this class is integrate(polynomial 1, polynomial 2) which computes for each monom the integration form. This can be done by incrementing the power and dividing the coefficient by the new power. We create a new monom, having a new coefficient and power, which will be added onto the list of the resulted polynomial.

* Parse:

The method implemented in this class is parsePolynom(string s, polynomial p) that receives the user input as a string s and parse it into a polynomial p, which will be used to calculate the result of a certain operation chosen by the user. Also, there is checked if the input contains only valid characters, or if the user does not introduce an input.

Graphical User Interface:

It has been implemented using Swing. It consist of three text fields and three labels corresponding to each text field, and 6 buttons corresponding to each operation and one button to clear the text fields. The structure of the panel view is:

-first label having the text ”First polynomial:” is followed by the first text field

-second label having the text “Second polynomial:” is followed by the second text field

-third label having the txt “Answer:” is followed by the third text field

-then, there are all of the buttons in this order: Add, Subtract, Multiply, Divide, Derive 1st p, Integrate 1st p, Clear.

All of the panel components are in the same row.

**5.Results:**

The application has been tested using Junit, and the test have been made for each implemented operation.

* For addition:

Input test 1:

Polynomial 1: **3x^2+2**

Polynomial 2: **5x-4**

Expected result: **3x^2+5x-2**

Our result: **3x^2+5x-2**

Input test 2:

Polynomial 1: **3x^3-4x^2-8x+2**

Polynomial 2: **5x^4-4**

Expected result: **5x^4+3x^3-4x^2-8x-2**

Our result: **5x^4+3x^3-4x^2-8x-2**

* For subtraction:

Input test 1:

Polynomial 1: **3x^2+2**

Polynomial 2: **5x-4**

Expected result: **3x^2-5x+6**

Our result: **3x^2-5x+6**

Input test 2:

Polynomial 1: **3x^3-4x^2-8x+2**

Polynomial 2: **5x^4-4**

Expected result: **-5x^4+3x^3-4x^2-8x+6**

Our result: **-5x^4+3x^3-4x^2-8x+6**

* For multiplication:

Input test 1:

Polynomial 1: **2x^2+2**

Polynomial 2: **x^2-2**

Expected result: **2x^4-2x^2-4**

Our result: **2x^4-2x^2-4**

Input test 2:

Polynomial 1: **2x^4+2x^3-2x**

Polynomial 2: **x^3+2x^2+3x**

Expected result: **2x^7+6x^6+10x^5+4x^4-4x^3-6x^2**

* For derivation:

Input test 1:

Polynomial 1: **3x^2+2**

Expected result: **6x**

Our result: **6x**

Input test 2:

Polynomial 1: **3x^3-4x^2-8x+2**

Expected result: **9x^2-8x-8**

Our result: **9x^2-8x-8**

* For integration:

Input test 1:

Polynomial 1: **3x^2+2**

Expected result: **x^3+2x**

Our result: **x^3+2x**

Input test 2:

Polynomial 1: **3x^3-4x^2-8x+2**

Expected result: **0.75x^4-1.33x^3-4x^2+2x**

Our result: **0.75x^4-1.33x^3-4x^2+2x**

**6.Conclusions:**

This assignment has been a combination of the mathematical rules and operations with the implementation of an algorithm that can do all of them. Also, styling the code in the MVC pattern was an interactive part of the project. This lab work has been an opportunity to learn and use the java swing for creating the GUI, even though the user interface created for this project can be improved after that. Moreover, the OOP concepts were tested which was a nice exercise for practicing and for a better understanding of some concepts.

**7.Biblyography:**

<https://www.geeksforgeeks.org/comparator-interface-java/?fbclid=IwAR0gKiOSccxDDjlVaJa6sLoMQ-7g3Pc6uovT2Ru0TXPamyxzjV9FyqsW4T0>

<https://www.javatpoint.com/java-swing>

<https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm>