Assignment 1

POLYNOMIAL CALCULATOR

Name: Șandor Cristian-Ionuț

Group: 30421

1. **Assignment objective**

Design and implement a polynomial calculator with a dedicated graphical interface through which the user can enter polynomials, select the operation to be performed (i.e. addition, subtraction, multiplication, division, derivative, integration) and display the result.

Secondary objectives:

* Design the project using the MVC manner and divide its functionalities in different packages;
* Designing algorithms in order to implement the polynomial operations;
* Provide a GUI for the polynomial calculator which is intuitive and easy to use;
* Testing the project using Junit

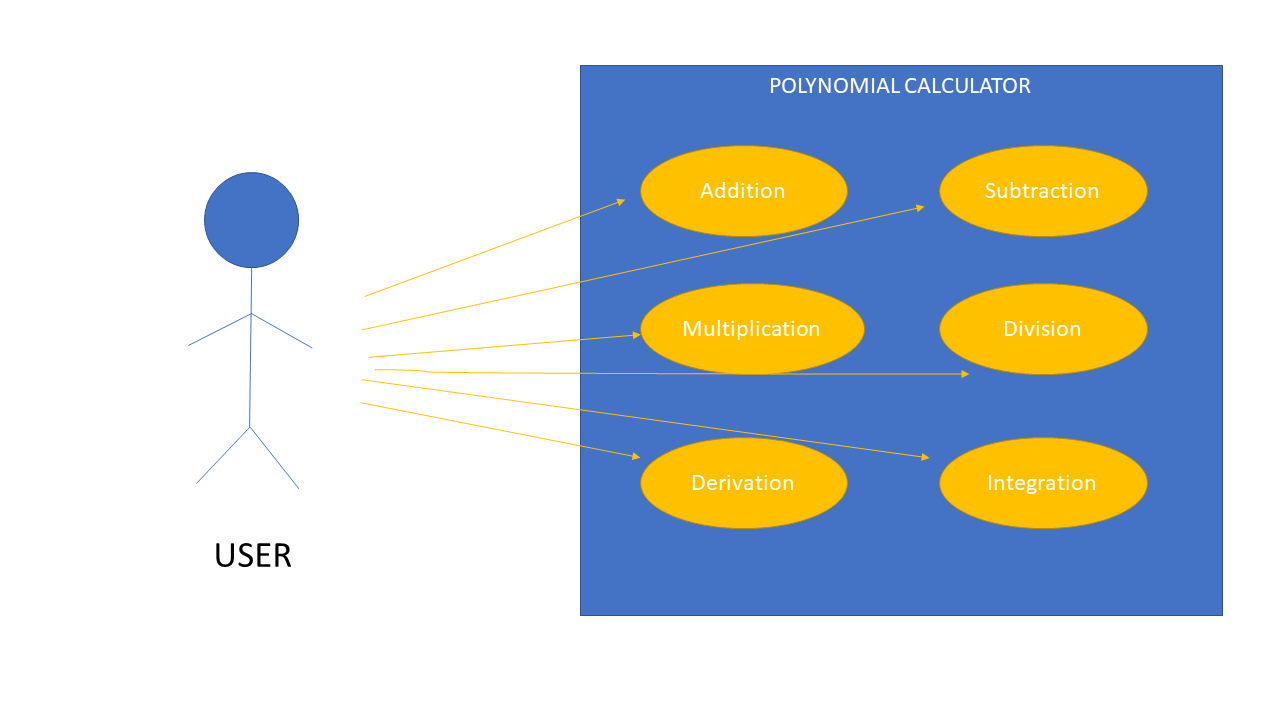
1. **Analyzing the problem, modelling, scenarios, use-cases**

It is necessary to correctly and efficiently implement all the operations for the proper use of the polynomial calculator: addition, subtraction, multiplication, division, derivation, integration. We also need to get the inputs which represent two strings and convert them in a format with which we can work easily during the operations and of course to take the result, convert it in a string and display it in the calculator’s window.

This project is modeled in a MVC manner. It is structured in three packages, model, view and controller and also, I have a package for the main program. The ‘model’ package contains classes in which we have all the necessary structures and algorithms which will help us make the operation for the polynomials. The view package contains a class in which I designed the GUI. Here are defined all the visual elements such as window, labels, buttons and text boxes. The control class contains a class which makes the link between the model package and the view package. Here we tell each button what to do and what to take and to display on screen.

A scenario represents a sequence of steps taken by the user to correctly interact with the application. The possible scenarios are described in the following section:

* The user makes the desired operation successfully. It writes the polynomials correctly and chooses the desired operation. The calculus is made without error and the result will be displayed in the result text box.
* The user does not succeed in making the desired operation. This could happen in two circumstances. In the first one, the polynomials inserted as input are not respecting the established format, in which case it will appear a new window which says “The inputs don't match the established format.”. The correct structure for the proper working of the application is: “sign” + “coefficient” + “x^” + “rank” + “sign” … (no spaces in between). The second situation which could not lead to the desired result is not inserting the polynomial in the first text box for the derivation and for the integration.



1. **Design**

The project is designed under a MVC pattern, having 3 separate components: Model, View and Controller. Model holds the Monomial, Polynomial and Operation classes, View holds the CalculatorView class and Controller holds the Controller class. The Controller class manipulates data stored under the Model package and interacts with the desired components of the graphical user interface, defined in the CalculatorView class.

The GUI class contains buttons for each operation (Addition, Subtraction, Multiplication, Division, Derivation, Integration), three text fields for the first polynomial, the second polynomial and the result of the chosen operation, three labels and a clear button. Each button triggers a listener and calls different methods, in order to obtain the desired output. The two strings taken as input are saved in Array Lists, converting the coefficients and the powers of the monomials in integer values. The said Array Lists are then taken as method parameters, leading to the expected result. The Monomial class contains information about the power & the coefficient of the element. The Polynomial class contains an Array List in which the monomials will be stored and a method for inserting a monomial in the polynomial. The Operations class contains the implementation of the requested operations.

The addition operation adds the coefficients of monomials of same power. If one polynomial contains a monomial with a rank which doesn’t appear in the other polynomial, that monomial will be simply added in the result polynomial.

The subtraction operation subtracts the monomials in the second polynomial from the monomials in the first one. It performs subtraction on monomials with the same rank and the remaining monomials are simply added to the result with the exception of those from the second polynomial where the coefficient is negated.

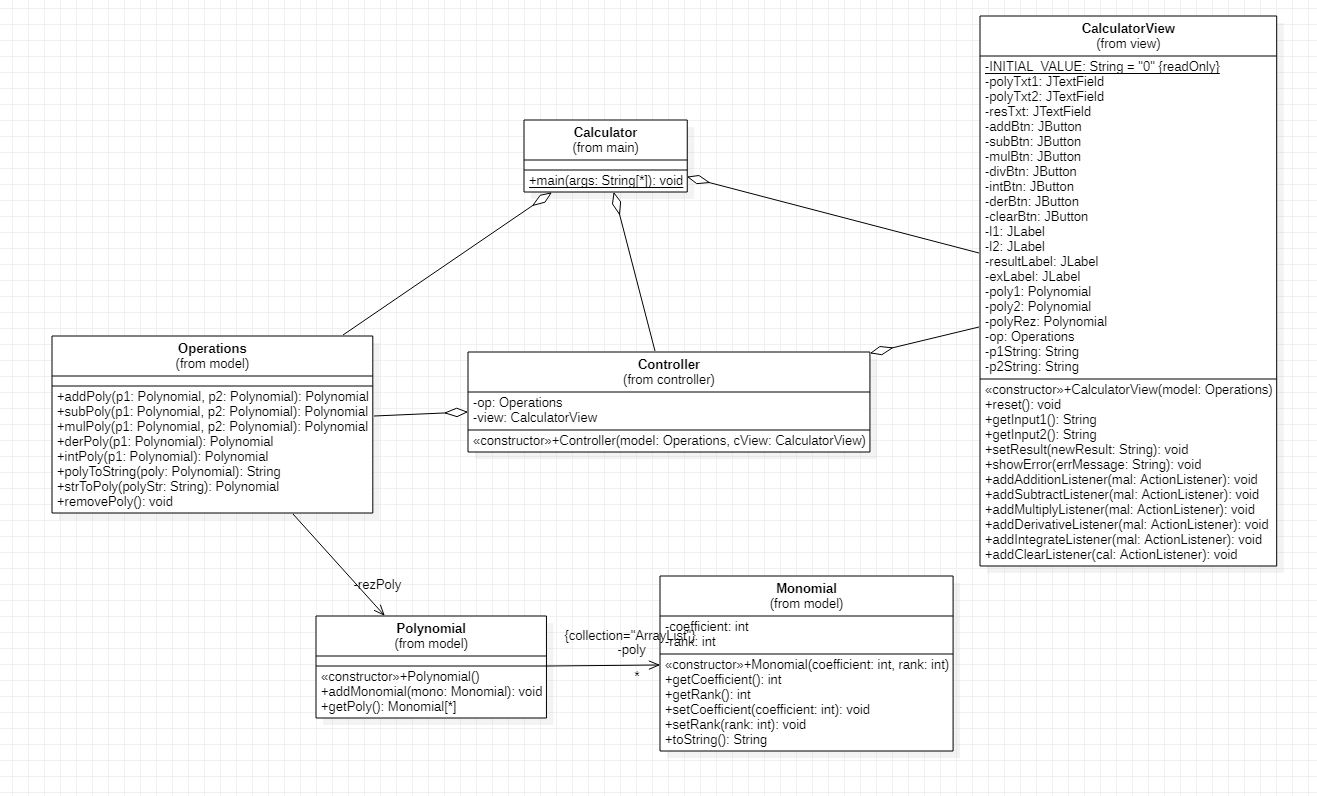
The derivation operation computes the derivative of each monomial. The derivative form is obtained by multiplying the coefficient of a monomial with its power, then decreasing the power.

The integration operation computes the integrated of each monomial. The integrated form is obtained by dividing the coefficient with the successor of the power, then increasing the power.

The multiplication operation is made rather simple. Each monomial from the first polynomial is multiplied with each monomial from the second polynomial. The multiplication between monomials is made by multiplying the coefficients and adding the ranks. After this, we reduce the terms by adding the monomials which have the same rank.

The algorithm of saving the monomials consists in splitting the string taken as input by sequences of: “sign” + “coefficient” + “x^” + “rank”. Everything before “x” is transformed in an integer and saved as the coefficient of the monomial and the number after “^” is also transformed in an integer but it is saved as the rank of the monomial. If the input polynomial doesn’t have the required format then the transformation would not be possible.

The buttons from the GUI will send the inputs to the controller where they will be transformed using the methods from model and then, still using the algorithms from the model, the result will be computed. This result is then transformed in a string and then is sent to the view in order to be displayed on the GUI.

This is the class diagram for the project with the polynomial calculator:

The data structure used for the implementation is the list and by using Array Lists I could easily manage the monomial objects and also recreate the operations.

1. **Implementation**

In the model package we find three classes each with its methods. First one is Monomial, which is the building block for our operations. This class is rather simple, it contains two variables which represent the coefficient and the rank of the monomial and besides that it also has a constructor and some getters and setters for the coefficient and rank (and a method toString() used for printing in console in the developing stage).

The second class is Polynomial. This class contains an ArrayList of monomial objects a constructor and a getter for the polynomial. Another method in this class is addMonomial which adds a monomial object to the polynomial and also it sorts in a descending way by rank all the monomials in the ArrayList after each addition of a monomial.

The third class is Operations. This class contains a polynomial array which is meant to represent the result polynomial. Besides this it contains all the methods necessary for each operation.

The first one is addPoly which takes two ArrayList of monomials as arguments and performs the addition. In the main while the algorithm check whether or not the first monomials from each array have the same rank. If that is true it sums those two monomials and add the result monomial in the result polynomial. If it is not true, in the result will be added only the one with the bigger rank. After each addition in the result polynomial, the monomial which was added is removed from its initial array. After the while loop finishes there are two additional while loops which check whether or not there are some remaining monomials in those two ArrayLists. In that case, all the monomials are added to the result.

The second method is subPoly which works basically on the same principle as the addition. The only difference is that the coefficients of the monomials are subtracted and besides that, if there are some remaining polynomials in the second array, they are added in the result polynomial with the coefficient negated.

The next method is mulPoly, which realizes the multiplication. While creating this algorithm I thought about the example of a multiplication made on paper. So, taking this in consideration, I used two for loops in order to multiply every monomial from the first polynomial with every monomial from the second polynomial by multiplying the coefficients and adding the ranks and I saved this intermediary result in an auxiliary array named “poly”. Then, for each rank I traversed the auxiliary array and added all the polynomials which have the selected rank then I added the result monomial in the final polynomial.

The derPoly method takes as a parameter only one polynomial. With a for each loop, the array is traversed and each monomial is derivated. The coefficient is multiplied by the rank and the rank is decreased by 1.

The intPoly method take also only one polynomial as parameter and by the same principle, with a for loop it traverses the array and integrates every monomial. The coefficient is divided by rank+1 and the rank is increased by 1.

The next method is polyToString which takes polynomial array as a parameter and it transforms it in a string.

The method strToPoly takes a string as a parameter and with the help of pattern matching it divides the string in monomials and after that, from each string piece it extracts the coefficient and the rank, creates a monomial with these values and then adds it to the result polynomial.

The view package contains only a class named CalculatorView. This class contains the definition of each graphical object: 3 JTextFields for inserting two inputs and displaying the output, and 7 buttons: addition, subtraction, multiplication, division, derivation, integration and clear. The constructor of this class sets the editing property of the output text field to false, sets the layout of the frame as a 4 line and 1 column grid and creates 4 panels: p1 - for the first text box, p2 – for the second text box, p3 – for the text box of the result and p4 – for the buttons. The class also contains methods for adding action listeners to the buttons, and definitions for buttons used with Junit.

The controller package has class named controller which realizes the link between the view and the model. Its constructor takes as parameters a variable of type Operations and one of type CalculatorView and creates listeners for each button. This class contains several inner classes, each one responsible for an action. Each one removes everything from the result variable and then gets the inputs from the text boxes, it converts them in arrays of monomials and then executes the specific operation for that action. The result is then converted in a string and sent to the GUI.

1. **Results**

Junit was used for testing the application. Each operation was tested once, excepting the division operation, whose remainder was not implemented.

Addition:

Input 1: 10x^2+15x^1-52

Input 2: 3x^2-7x^1

Expected output: +13x^2 +8x^1 -52

Output: +13x^2 +8x^1 -52

The test was successful.

Subtraction:

Input 1: 10x^2+15x^1-52

Input 2: 3x^2-7x^1

Expected output: +7x^2 +22x^1 -52

Output: +7x^2 +22x^1 -52

The test was successful.

Derivation:

Input : 10x^2+2x^1

Expected output: +20x^1 +2

Output: +20x^1 +2

The test was successful.

Integration:

Input : 10x^1+2

Expected output: +5x^2 +2x^1

Output: +5x^2 +2x^1

The test was successful.

Multiplication:

Input 1: 10x^2

Input 2: 3x^2-7x^1

Expected output: +30x^4 -70x^3

Output: +30x^4 -70x^3

The test was successful.

1. **Conclusion**

This assignment has given me a better understanding of the model-view-controller pattern, and has made me look up different tricks, how to split an array with respect to a certain condition, and how to apply Junit tests to my application. The application could be further improved by implementing a division algorithm and also by creating a better design for the GUI. In conclusion, this assignment has been a great opportunity to both learn new things and revise the information that I already had.

1. **Bibliography**

<http://stackoverflow.com/>

<http://zetcode.com/tutorials/javaswingtutorial/>

<https://www.leepoint.net/GUI/structure/40mvc.html>