**ASSIGNMENT 1 DOCUMENTATION**

**POLYNOMIAL CALCULATOR**

**Name: Zarnescu Radu**

**Group: 30423**

**Laboratory Assistant: Antal Marcel**

Contents

[1. Assignment Objective 3](#_Toc476131445)

[2. Problem Analysis 3](#_Toc476131446)

[3. Design 6](#_Toc476131449)

[3.1. Data Structures 6](#_Toc476131451)

[3.2. UML Diagram 7](#_Toc476131452)

[4. Implementation 7](#_Toc476131454)

[4.1. Class Description 7](#_Toc476131453)

[4.2. Algorithms 9](#_Toc476131453)

[5. Testing 10](#_Toc476131454)

[6. Conclusions and Further Improvements 13](#_Toc476131455)

[7. Bibliography 13](#_Toc476131456)

1. **Assignment Objective**

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, multiplication, division, derivative, integration) to be performed and view the result.

1. **Problem Analysis**

In mathematics, a polynomial is an expression consisting of variables (also called indeterminates) and coefficients. The operations which can be applied on polynomials are addition, subtraction, multiplication, division, derivation and integration.

A use case is a written description of how users will perform tasks on your website.  It outlines, from a user’s point of view, a system’s behavior as it responds to a request. Each use case is represented as a sequence of simple steps, beginning with a user's goal and ending when that goal is fulfilled.

Basic Flow: Computing an operation on two polynomials:

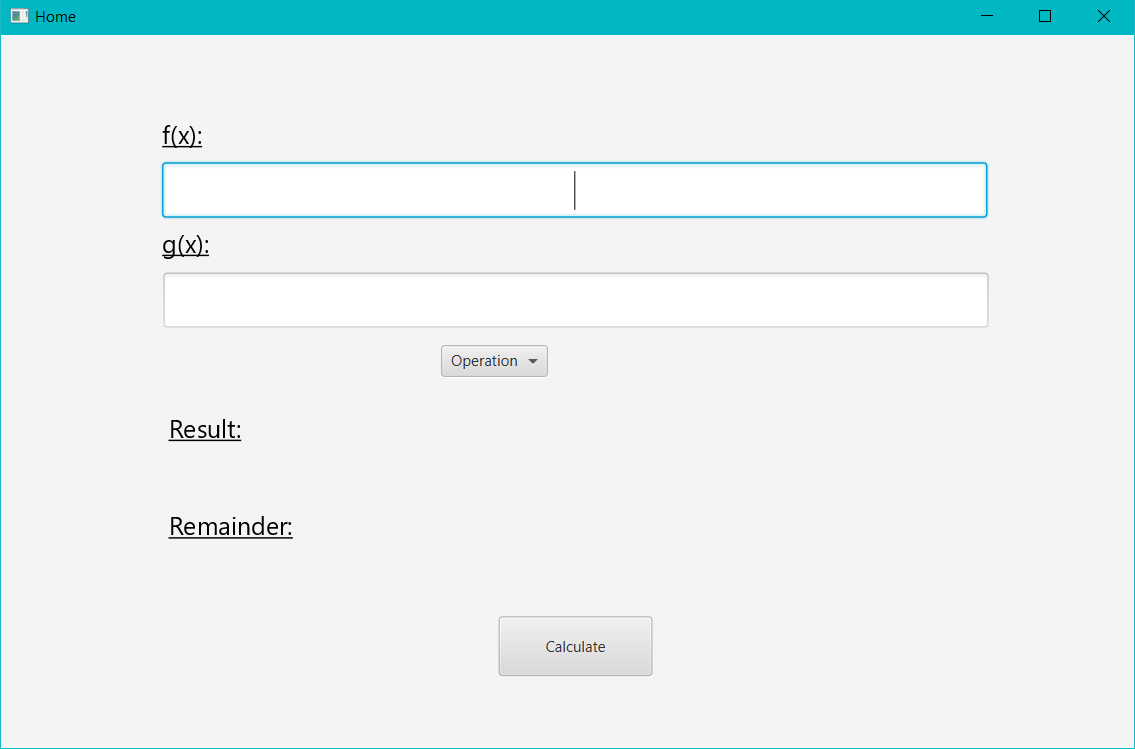
* Actor: User
* Description: The main success scenario is the situation where the user gives the program two polynomials and the user outputs the result of the selected operation

1. User inserts the first polynomial in the f(x) field
2. User inserts the second polynomial in the g(x) field
3. User selects the required operation
4. User presses the “Calculate” button

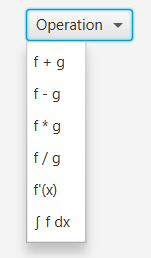
* Termination outcome: the correct result is displayed under the “Result:” field

Illustration of the steps a user has to make in order to correctly use the application:

The user interacts with the program through a dedicated UI found in the screenshot below. It is very simple to use and it indicates the user what every field represents.



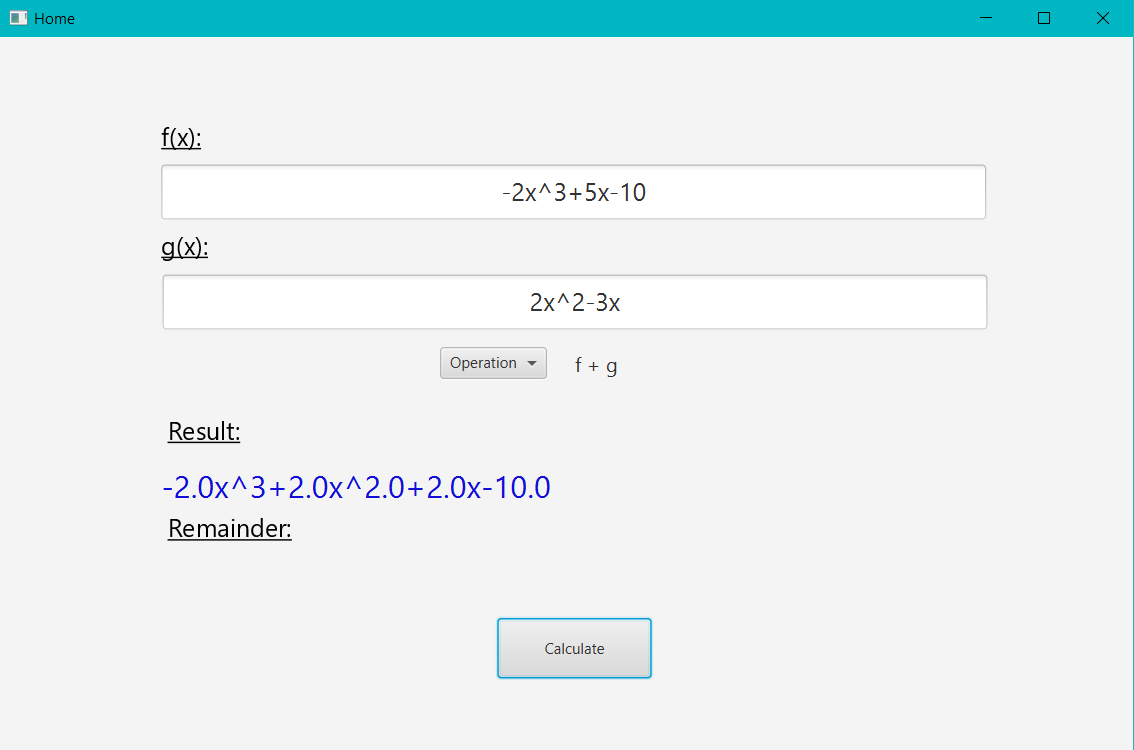
The first step is to insert the polynomials in the fields below “f(x):” and, respectively, “g(x):”. Then, the user needs to select the wanted operation using the menu button below.



In order not to create confusion by forgetting which operation was selected, it will be displayed next to the menu button.

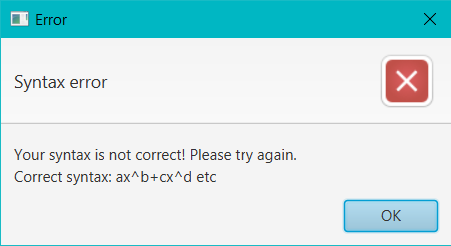


After inserting the polynomials and selecting the operation, the only step left is to press the “Calculate” button. The result will be displayed beneath “Result:”. If the performed operation is division, then the remainder will be displayed under “Remainder:”.

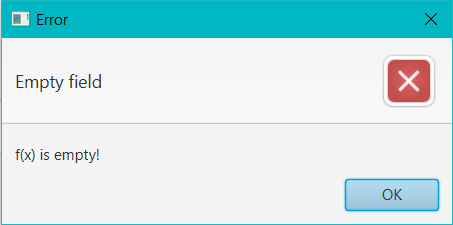
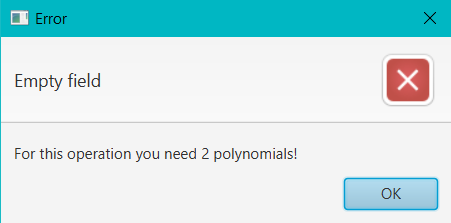


Cases when the program cannot function properly (error cases):

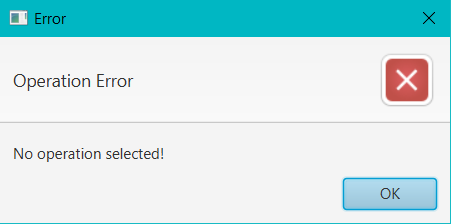
If the user does not respect the syntax for inserting polynomials the following error will pop up:



If an operation is selected and the “Calculate” button is pressed without inputting the required number of polynomials the following errors will occur: on the left hand side is the error which appears if sum, difference, multiplication or division is selected; the error on the right hand side is displayed if derivation or integration is selected.



The last possible error is occurred when no operation is selected and “Calculate” is pressed.

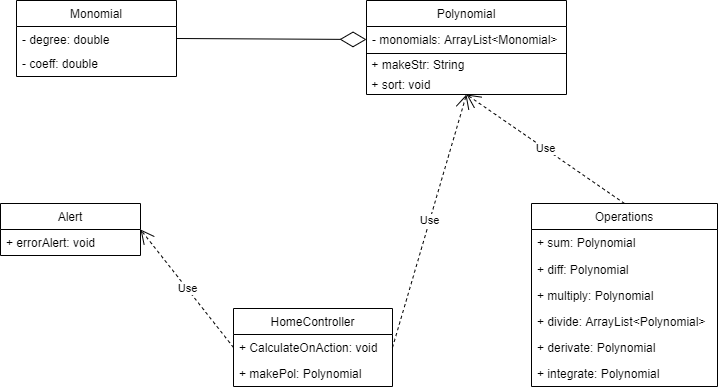


1. **Design**
   1. **Data Structures**

In order to implement the required problem, I have chosen to use two data structures: monomial, which has two double attributes, degree and coefficient (coeff); the polynomial is represented by an ArrayList of Monomials.

I have chosen to use ArrayList instead of a simple Array because it is more efficient for memory management and also, it does not have a fixed number of elements. An alternative data structure would be HashMap, but I found it easier and more convenient to implement the operations using ArrayLists.

* 1. **UML Diagram (Class Diagram)**

****

1. **Implementation**
   1. **Class Description**

I based my class structure on the Model – View – Controller methodology. It is a pattern for modularizing the classes based on what each of them do.

1. Model includes the classes which mostly deal with the back end of the project, the algorithms and the operations which allow the program to function. It is the part which the user does not see at all when using the app.
2. View represents the classes that form the user interface, the part with which the user interacts.
3. Finally, controller is the collection of classes which make the connection between back end and front end, model and view. For example, when a button is pressed, the controller is responsible for calling the appropriate methods and algorithms that apply to that action.

* Monomial Class

The monomial class contains two double fields, degree and coeff (coefficient). A Polynomial contains multiple monomials and this class deals with each and every one of them. A monomial is of the form ax^b, where a represents the coefficient and b the degree.

Methods:

* public Monomial(): the default constructor, used to create an empty Monomial object.
* public Monomial(double coeff, double degree): the constructor which overloads the default one. It initializez coeff and degree with values given in the call of the constructor.
* Setters and Getters which are needed to access private attributes.
* Polynomial Class

As previously said, a polynomial is a collection of monomials, so the only attribute Polynomial has is an Array List of instances of Monomial.

Methods:

* public Polynomial(): the default constructor which creates new instances of Polynomial
* Getter and Setter for the Array List
* public String makeStr(): method which goes through all the monomials of each object and creates a String which is to be printed. It also acts as a “pretty print”.
* public void sort(): this method sorts the monomials in the ArrayList in decreasing order of the degrees.
* Operations Class

This class contains all the mathematical operations that are to be applied on the input.

Methods:

* public Polynomial sum(Polynomial p1, Polynomial p2): this method returns the sum between the polynomials p1 and p2.
* public Polynomial diff (Polynomial p1, Polynomial p2): this method returns the difference between the polynomials p1 and p2 (p1 – p2).
* public Polynomial multiply(Polynomial p1, Polynomial p2): this method multiplies the two polynomials, p1 and p2, and returns the result.
* public ArrayList<Polynomial> divide(Polynomial p1, Polynomial p2): this method applies the long division algorithm on the polynomials (p1 / p2) and returns a list containing the quotient and the remainder of the operation.
* public Polynomial derivate(Polynomial p): this method derivates the input polynomial, p.
* public Polynomial integrate(Polynomial p): this method integrates the input polynomial, p.
* Alert Class

A class used to display the errors illustrated previously.

Methods:

* public void errorAlert(String header, String context): this method prints an error message on the screen and displays the parameter messages.
* HomeController Class

This class is used to control the interactions between the user and the user interface.

Methods:

* public void item1onAction (ActionEvent e): this method tells the program that the first menu item was selected (sum).
* public void item2onAction (ActionEvent e): this method tells the program that the second menu item was selected (difference).
* public void item3onAction (ActionEvent e): this method tells the program that the third menu item was selected (multuplication).
* public void item4onAction (ActionEvent e): this method tells the program that the fourth menu item was selected (division).
* public void item5onAction (ActionEvent e): this method tells the program that the fifth menu item was selected (derivation).
* public void item6onAction (ActionEvent e): this method tells the program that the sixth menu item was selected (integration).
* public Polynomial makePol(String[] str): this method receives as parameter an array of Strings that contains the monomials in String form, resulted from splitting the input String.
* Public void calculateOnAction(ActionEvent e): as it’s name suggests, this method calls all the methods needed to display the result of the selected operations.
* Main Class

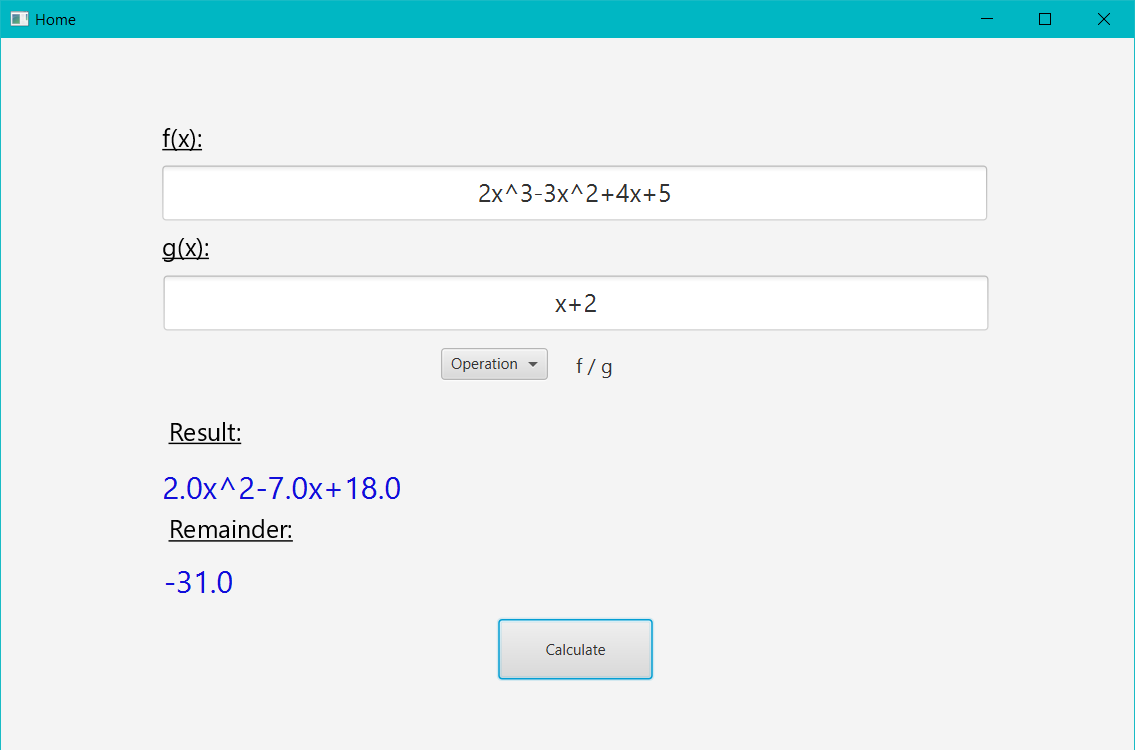
The main class extend the Application class and is used to start and shape the User Interface.

* 1. **Algorithms**

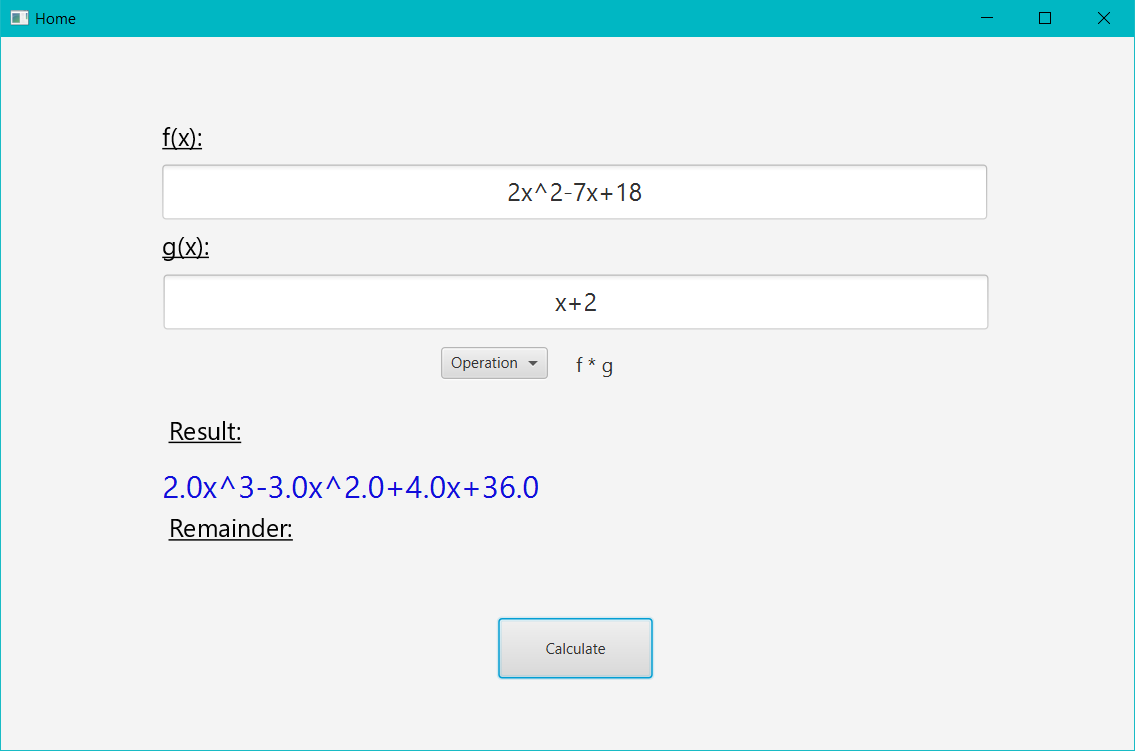
In this section I will briefly describe the algorithms I used to implement each operation.

1. Sum and difference: for these operations I used the same algorithm, with small differences. It uses a loop that goes through all the monomial. At each iteration, two monomials are compared, one from each polynomial. If their degrees are equal, a new monomial is added to the result with the same degree and the sum/difference of the two monomials’ coefficients. In the case the degrees are not equal, the one with the greater power is added to the result. After the loop is completed, the program checks if all of the monomials from the operands were checked and, if not, the remaining ones are added to the result.
2. Multiplication: to implement this operation I used the most efficient algorithm I could come up with to deal with the problem of creating multiple monomials of the same degree. Firstly, I create an array of doubles of the size degree(p1) + degree(p2) + 1, initialized with 0 on every position. To multiply the polynomials, I use a for in for to iterate through the lists of monomials and add to the value of the array, on the position equal to the sum of the two monomials’ powers, the multiplication of their coefficients. Afterwards, the program goes through the array and, if the value is not 0, creates a new monomial which is added to the result.
3. Division: this operation works with multiple polynomials. At first, a new one is created, which will be the quotient and p1 (the first operand) will be the remainder. A while loop iterates for as long as p1’s degree is greater than p2’s. In the loop, a new polynomial is created with the coefficient equal to p1’s first monomial’s coefficient / p2’s first monomial’s coefficient and the degree equal to degree (p1) – degree (p2). Then, the multiplication between the new monomial and p2 will be subtracted from p1.
4. Derivation and integration: no special algorithm was used to implement these operations. Only mathematical calculations on the monomials were used to create new ones.
5. **Testing**

The most important part of the project is the testing. If the results are not correct, then the program is not doing it’s job properly. In order to test whether the app is working well, we can just run the program, follow the steps presented in section 2, and compare the output to the computations made on paper. One such case can be observed in section 2, page 4. As we can see, the output is correct so the addition operation is working correctly. Let’s take another case, dividing the following polynomials: f(x) = 2x^3-3x^2+4x+5, g(x) = x+2. Below you will find a screenshot of the UI after inputting f, g, selecting division and pressing “Calculate”.



If we multiply the result with g(x) and subtract the remainder, we will see that the result will be equal to f(x). Below is illustrated this multiplication via my application.



As you can see, if we subtract 31 from the result above, it will be equal to f(x) from the division.

Another way of testing the app is to use JUnit testing. This can be seen in the OperationsTesting class. A test has been made for each operation with the same polynomials, f(x) = 2x^4-x^3-10x, g(x) = x^2+2x+5.

1. Sum

Expected: “2.0x^4-x^3+5.0x^2-8.0x+5”

Real: “2.0x^4-x^3+5.0x^2-8.0x+5”

1. Difference

Expected: “2.0x^4-x^3+3.0x^2-12.0x+5”

Real: “2.0x^4-x^3+3.0x^2-12.0x+5”

1. Multiplication

Expected: “2.0x^6+3.0x^5+12.0x^4-7x^3-50x”

Real: “2.0x^6+3.0x^5+12.0x^4-7x^3-50x”

1. Division

Expected: “2.0x^2-5.0x+4.0 7.0x-20.0”

Real: “2.0x^2-5.0x+4.0 7.0x-20.0”

1. Derivation

Expected: “8.0x^3-3.0x^2+8.0x-10.0”

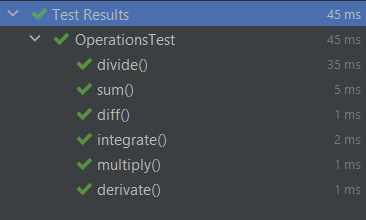
Real: “8.0x^3-3.0x^2+8.0x-10.0”

1. Integration

Expected: “0.4x^5-0.25x^4+1.3x^3-5.0x^2”

Real: “0.4x^5-0.25x^4+1.3x^3-5.0x^2”

As we can see, all of the tests are completed successfully.



1. **Conclusions and Further Improvements**

To conclude, this assignment was a very good exercise in writing java and object oriented programming code. Now, I would consider myself more experienced in the techniques of programming applications.

One of the most important things I learned was how to make Maven projects, which is very useful for importing libraries and convenient. Also, before this project I did not have the knowledge on how to work with, or use regular expressions for splitting and validating strings. The other important lesson I learned was how to manage my time, as I started working a bit too late and found myself really stressed with time. So, this will not happen anymore in the future.

The biggest problem I encountered was working with the inputs as strings, as it was pretty hard for me to figure out how to work with regex, but in the end I was able to succeed.

As for further improvements, I think it would be useful to make the output look better by not printing .0 for every integer. Another one would be improving the way I organize my code in order to be easier to read and understand.

1. **Bibliography**

* wikipedia.org
* [www.stackoverflow.com](http://www.stackoverflow.com)
* geeksforgeeks.org
* usability.gov