Technical University of Cluj-Napoca

Programming Techniques

Assignment 1

Polynomial Calculator

Teacher: prof. Ioan Salomie

Teacher Assistant: Dr. Viorica Chifu

Student: Rusu Horia Gabriel

Group : 30421/2

1.Objectives

Main objective: the main objective is to design and implement a polynomial calculator with a dedicated graphical interface trough which the user can insert polynomials, select the mathematical operation to be performed and view the results.

Sub-objectives:

* Analyze the problem and identify requirements
* Design the polynomial calculator
* Implement the polynomial calculator
* Test the polynomial calculator

2.Problem analysis,modeling,scenarios,use cases

a. Problem analysis

In mathematics, a polynomial is an expression consisting of variables and coefficients, that involves the operations of addition, subtraction, multiplication and non-negative exponantion of variables. In the case of this specific requirement, the polynomial will have only one variable, which means the general term of a polynomial of this case is axn, where x represents the variable, a represents the coefficient and n represents a non-negative integer. An example of such a term would be 4x3, here obviously 4 is the coefficient, 3 is the power, which also represents the degree of the polynomial, if there is no term with a greater power. This particular case of a polynomial is named of a monomial, which means a polynomial with one single term, with one coefficient and one power. Actually, a polynomial is a collecion of multiple monomial linked together by plus signs in order to create the full function. An example for this would be x2+x+1, a polynomial made out of three monomial, the first one being x2, which has the coefficent 1 and the power as 2, the second one is x, with coefficient 1 and power 1 and 1 being the coefficent and in this case 0 being the power of x, as to represent the one. The operations which can be implemented using the polynomials will be addition, subtraction, division, multiplication, integration and derivation. At the end of all this, the resulting polynomial will have the same form as the original polynomials, with coefficient and power.

b. Modelling of the problem

The user will be able to use the implemented interface in order to write in the required slots the proper polynomials, after which the proper operation can be implemented out of a list of specific operations, which are:

* Addition of two polynomials
* Subtraction of polynomials
* Multiplication of polynomials
* Integration of a polynomial
* Derivation of a polynomial

The result of the chosen operation will be displayed in the proper slot, and will have the same syntax form as the one required in order to write the polynomial/polynomials in the proper slots. If the proper syntax is not used, a message will appear indicating this situation.

c. Scenarios and Use Cases

Polynomial Calculator

User

Integrate Polynomialsa

Derivate Polynomials

Multiply Polynomials

Subtract Polynomials

Add Polynomials

User Case: add polynomials

Primary Actor: user

Main Success Scenario:

1. The user inserts the polynomials in the graphical user interface
2. The user selects the „addition” operation
3. The user clicks on the addition button
4. The polynomial calculator performs the addition of the two polynomials and display the result

Alternative Sequence: Incorrect polynomials

* The user inserts a polynomial using the wrong syntax
* A message shows up on screen, the scenario returns to step 1

User Case: subtract polynomials

Primary Actor: user

Main Success Scenario:

1. The user inserts the polynomials in the graphical user interface
2. The user selects the „subtraction” operation
3. The user clicks on the subtraction button
4. The polynomial calculator performs the subtraction of the two polynomials and display the result

Alternative Sequence: Incorrect polynomials

* The user inserts a polynomial using the wrong syntax
* A message shows up on screen, the scenario returns to step 1

User Case: derivate polynomial

Primary Actor: user

Main Success Scenario:

1. The user inserts the polynomials in the graphical user interface
2. The user clicks on the derivation button to perform the operation
3. The polynomial calculator performs the derivation of the polynomial and display the result

Alternative Sequence: Incorrect polynomials

* The user inserts a polynomial using the wrong syntax
* A message shows up on screen, the scenario returns to step 1

User Case: integral polynomial

Primary Actor: user

Main Success Scenario:

1. The user inserts the polynomial in the graphical user interface
2. The user clicks on the integration button to perform the operation
3. The polynomial calculator performs the integration of the polynomial and display the result

Alternative Sequence: Incorrect polynomials

* The user inserts a polynomial using the wrong syntax
* A message shows up on screen, the scenario returns to step 1

User Case: multiply polynomial

Primary Actor: user

Main Success Scenario:

1. The user inserts a polynomial in the graphical user interface
2. The user clicks on the multiplication button to perform the operation
3. The polynomial calculator performs the multiplication of the polynomial and display the result

Alternative Sequence: Incorrect polynomials

* The user inserts a polynomial using the wrong syntax
* A message shows up on screen, the scenario returns to step 1

Functional Requirements:

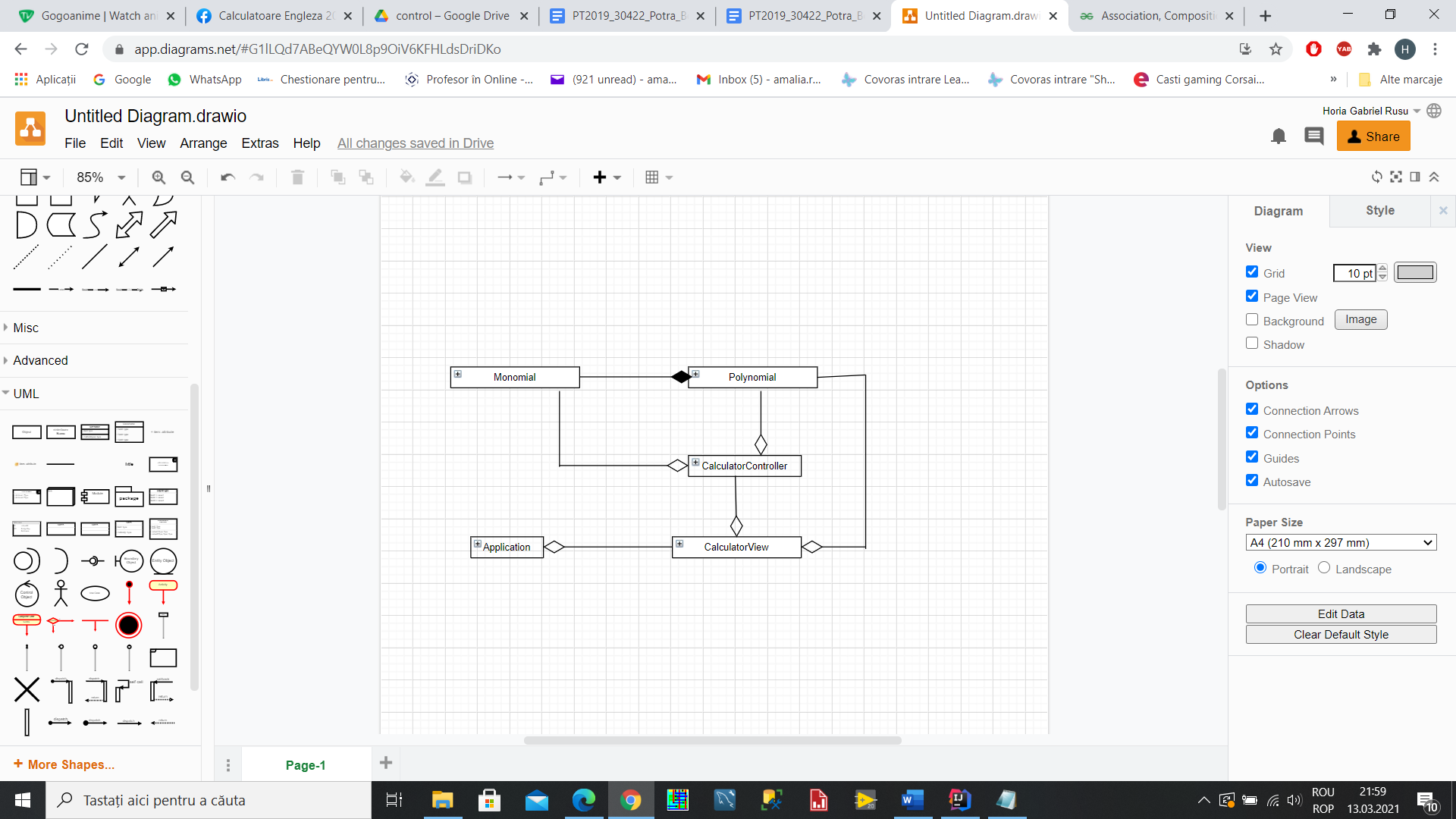
* The polynomial calculator should allow users to insert polynomials
* The polynomial calculator should allow users to select the mathematical operation
* The polynomial calculator should add two polynomials
* The polynomial calculator should subtract two polynomials
* The polynomial calculator should multiply two polynomials
* The polynomial calculator should integrate a polynomial
* The polynomial calculator should derivate a polynomial

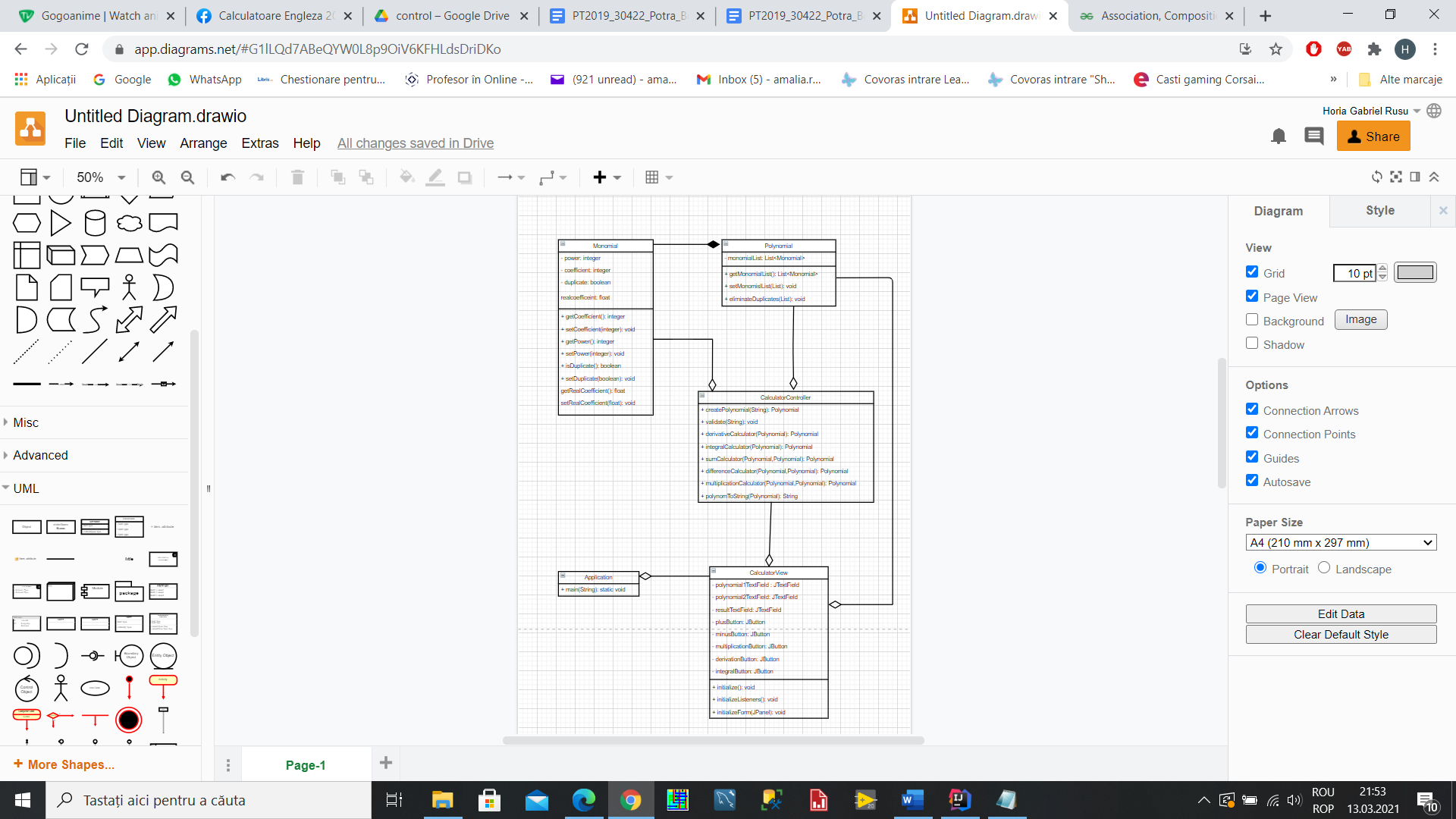
Non-Functional Requirements:

* The polynomial calculator should be intuitive and easy to use by the user
* The polynomial calculator should display the result in a clear and easy to understand way

3 Design

1. Class Diagram





1. Data Structures

The data structures used are both simple of a primitive type such as integers or doubles for the coefficient and power, as well as the more complex ones, specifically the use of ArrayList in order to create the polynomial by adding the various monomials in the list.

ArrayLists are more useful than simple arrays because of the ease with which the new objects can be added. Also they are more efficient when it comes to the memory managmnet.

1. Implementation

The program is built on the Model-View-Controller architectural pattern, which means than other than the main application which launches the entire program, it also contains three packages called Model, View and Controller. The Model package contains the classes for the main structures used. In the case of this problem we have the Monomial class and the Polynomial class made out of multiple monomials. The View class initiates the GUI, the user interface trough which the data is put in by the user, the one or two polynomials and the result is displayed on the screen. The Controller class takes the polynomials and makes various operations with them and returning them to the View in order to be displayed. Besides this we also use a package for Exceptions where we have tj=he exception for when the program does not have as input the proper format, case in which the program will not compute and show you a message instead.

Class Design: The class design has a divide and conquer method, through which the problem is divided into multiple smaller problems in order to be able to solve more simple, well-known problems instead of one big problem which seems too complicated to solve. The division of the program into classes is one part in the divide and conquer method and due to the Model-View-Controller pattern used, we divide the problem into the before mentioned multiple packages. Thus the program consists in total of five parts, which are:

1. The Model – contains the logic of the application:
2. The Monomial Class

A monomial is the most basic form of a polynomial, so it is obvious to use them to create the Polynomial.

This class has five variabels, the first two and most important ones are the power and coefficient variables, both integer types are the ones that have the most important role because this two are responsible for creating the polynomial. One more variable is of boolean type which is duplicate, which is used in the operations of addition, subtraction and multiplication, where we need to find the monomials with the same power and add them together. Finally, we have the variable realCoefficient, which is used in the integral in case the coefficent ends up being of type double.

Methods:

* public int getCoefficient() : returns the coefficent of the momonial
* public void setCoefficient(int coefficient) : sets the coefficent of the monomial
* public int getPower() : returns the power of the monomial
* public void setPower(int power) : sets the power of the monomial
* public boolean isDuplicate() : returns yes if the monomial is a duplicate
* public void setDuplicate(boolean duplicate) : sets a monomial as duplicate in the context of a polynomial
* public float getRealCoefficient() : returns the real coefficent of the momonial
* public void setRealCoefficient(float realcoefficient) : sets the real coefficent of the monomial

1. The Polynomial Class

The class with one variable which is List<Monomial> monomialList, which is a list of all the monomials that compose the polynomial.

Methods:

* public List<Monomial> getMonomialList() : returns the polynomials monomial list
* public void setMonomialList(List<Monomial> monomialList) : sets the monomial list as this polynomials monomial list
* public void eliminateDuplicates(List<Monomial> monomialList) : is used when after the addition, multiplication or subtraction the resulting polynomial has multiple elements with the same power, case in which the method adds them

2.) The View – contains the graphical user interface:

1. CalculatorView Class:

The class extends JFrame and has multiple private variables. This are the test fields for the two polynomials called polynomial1TextField and polynomial2TextField. Other than that there is the resultTextField for the result of the operation. The other variables are JButtons, which are plus-, minus-, multiplication-, integral- and derivationButton. They are used in order to initiate the calculation that is about to be provided.

Methods:

* public void initialize() : is used to initialize the JPanel for the interface
* public void initializeListeners() : is used to initiate the response to the pressing of one of the buttons to make a calculation
* private void initializeForm(JPanel panel) : adds the buttons and the text fields to the panel

3.) The Controller: creates relations between the View and the Model

1. CalculatorController Class:

This class does not contain any variables, only methods for the operations that need to be implemented.

Methods:

* public Polynomial createPolynomial(String poly1) : is used to take the strings of the polynomials read from the text fields and transforms them into polynomials, by taking each monomial and adding them to a list
* private void validate(String poly): is used to verify if the polynomial from the textfield has the proper format as required. If not here will be an exception thrown
* public Polynomial derivativeCalculator(Polynomial poly1) : is used to calculate the derivative of the polynomial read from the text field
* public Polynomial integralCalculator(Polynomial poly1) : is used to calculate the integral of the polynomial read from the text field
* public Polynomial sumCalculator(Polynomial poly1, Polynomial poly2) : is used to calculate the sum of the two polynomials read from the text field
* public Polynomial differenceCalculator(Polynomial poly1, Polynomial poly2): is used to calculate the difference of the two polynomials read from the text field
* public Polynomial multiplicationCalculator(Polynomial poly1, Polynomial poly2) : is used to calculate the multiplication of the two polynomials read from the text field
* public String polynomToString(Polynomial result) : is used to take the polynom resulted from the operation chosen and turn it to a string to be shown in the result text field.

4.) The Exceptions

1. UnproperFormatException:

Contains no variables, methods or constructors, but extends Exception an is thrown when the validate method finds than the introduced polynomial does not have the expected format.

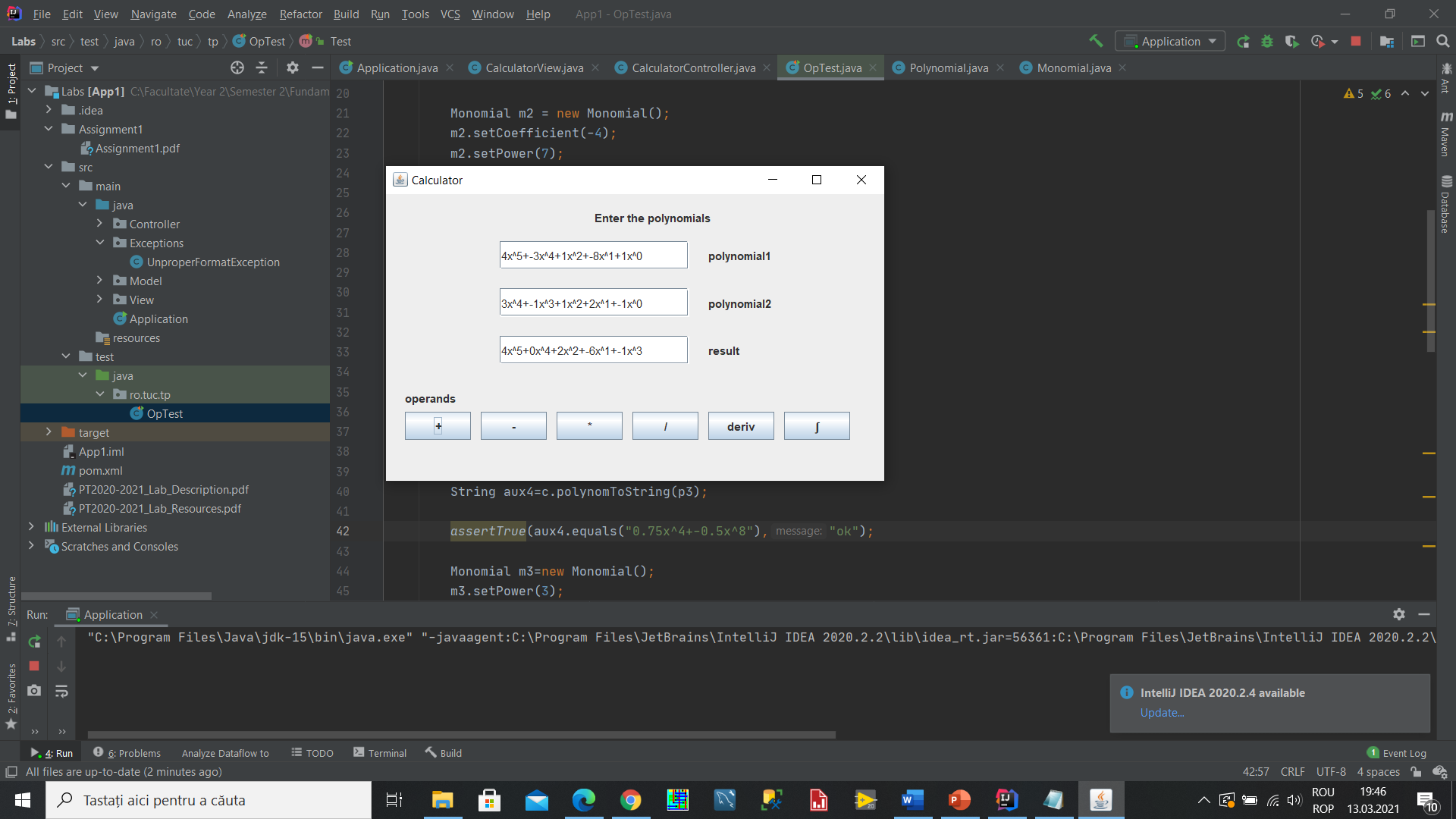
5.) Application

The application only has the main method which initializes the calculators view so that the computation can be made.

Algorithms:

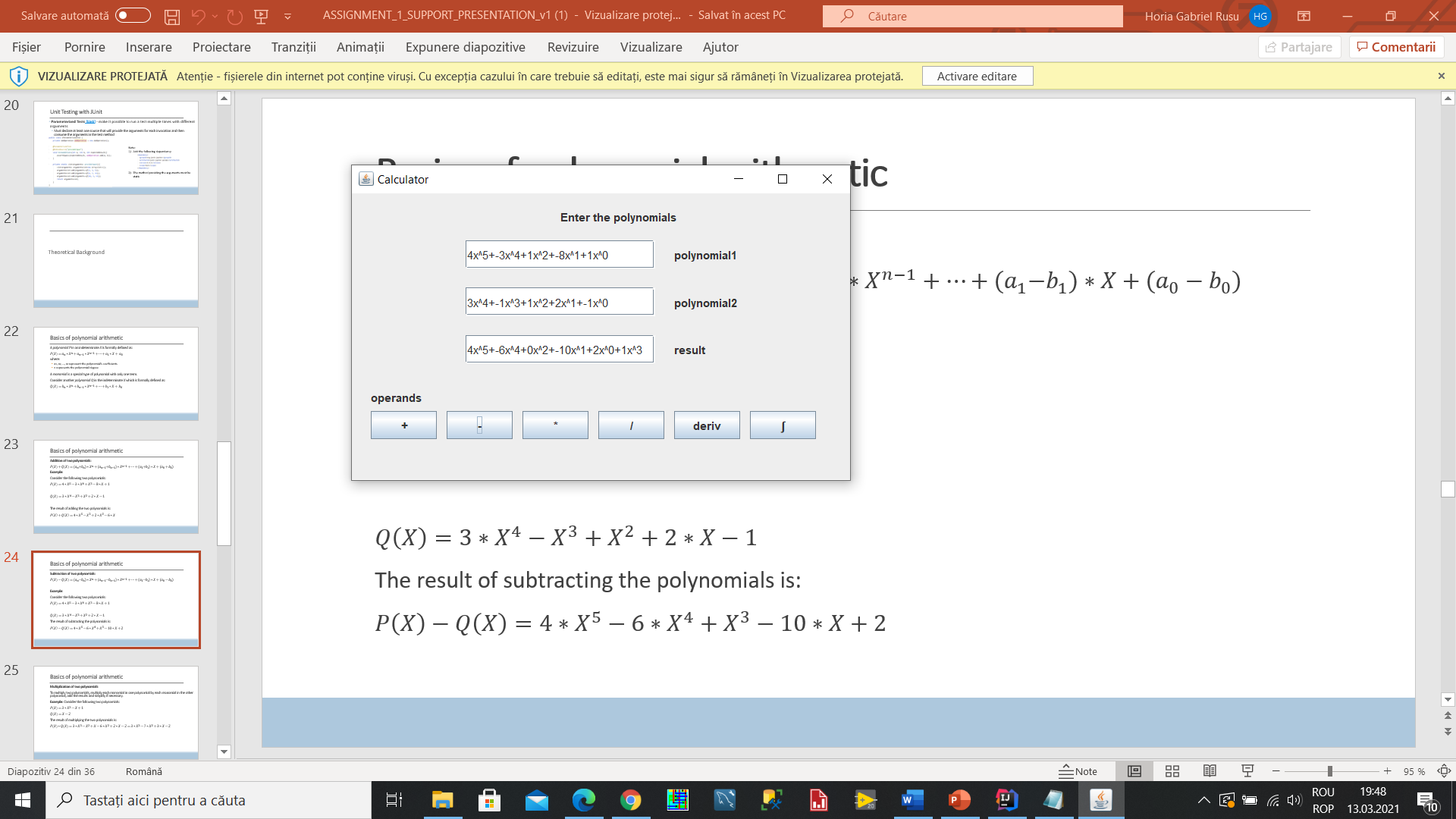
1. Addition:

The algorithm for addition is made by taking all the monomials from the two lists of monomials of the polynomials and adding them to the list of the resulting polynomial. After this the eliminate duplicates method is called, which looks in the list and adds together monomials with the same power and eliminates one of them after the addition. The result is then transformed in a string and displayed.



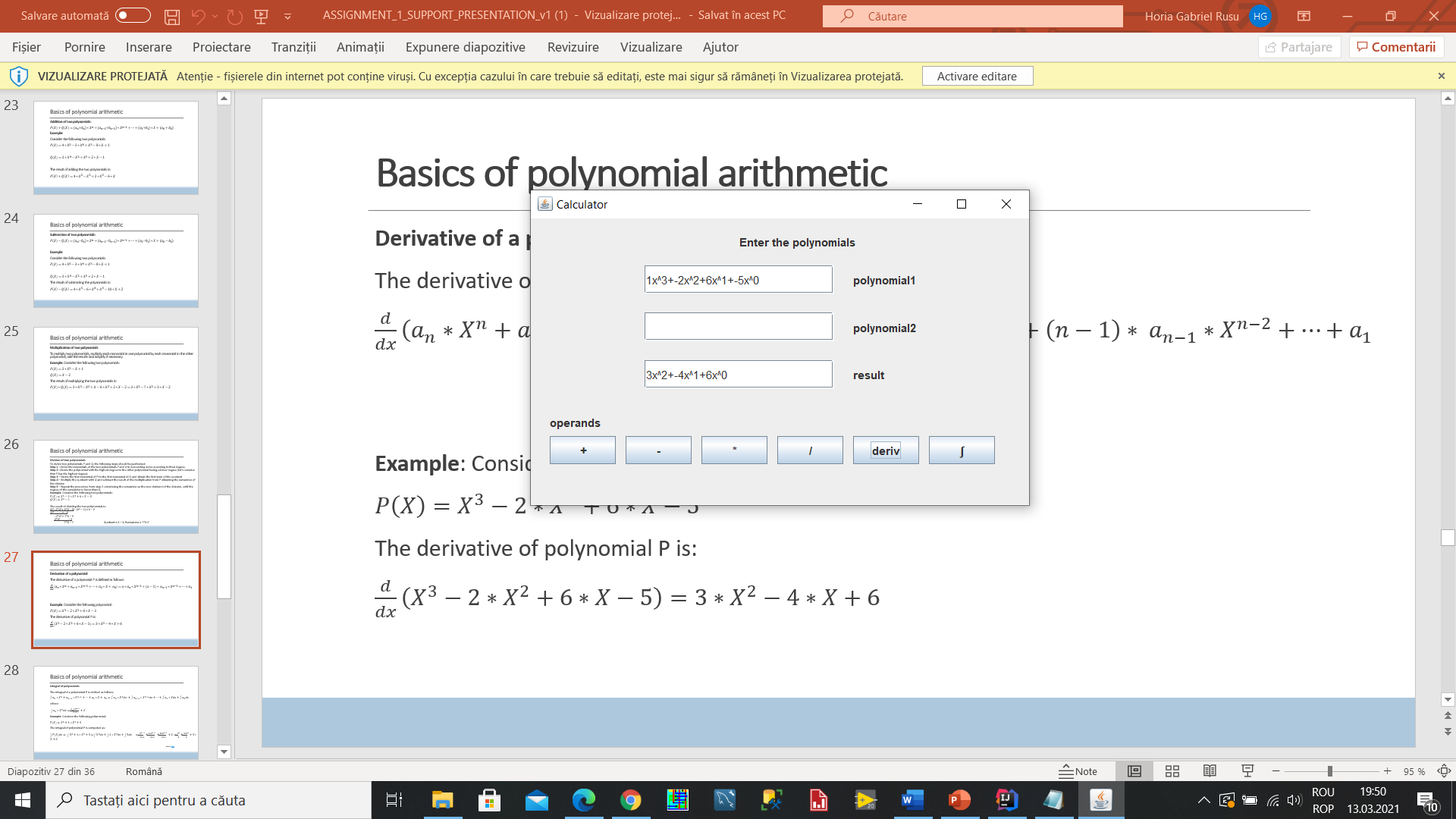
1. Subtraction:

The algorithm for subtraction is made by taking all the monomials from the two lists of monomials of the polynomials and adding them to the list of the resulting polynomial, where the second polynomial has its coefficients negated. After this the eliminate duplicates method is called, which looks in the list and adds together monomials with the same power and eliminates one of them after the addition. The result is then transformed in a string and displayed.



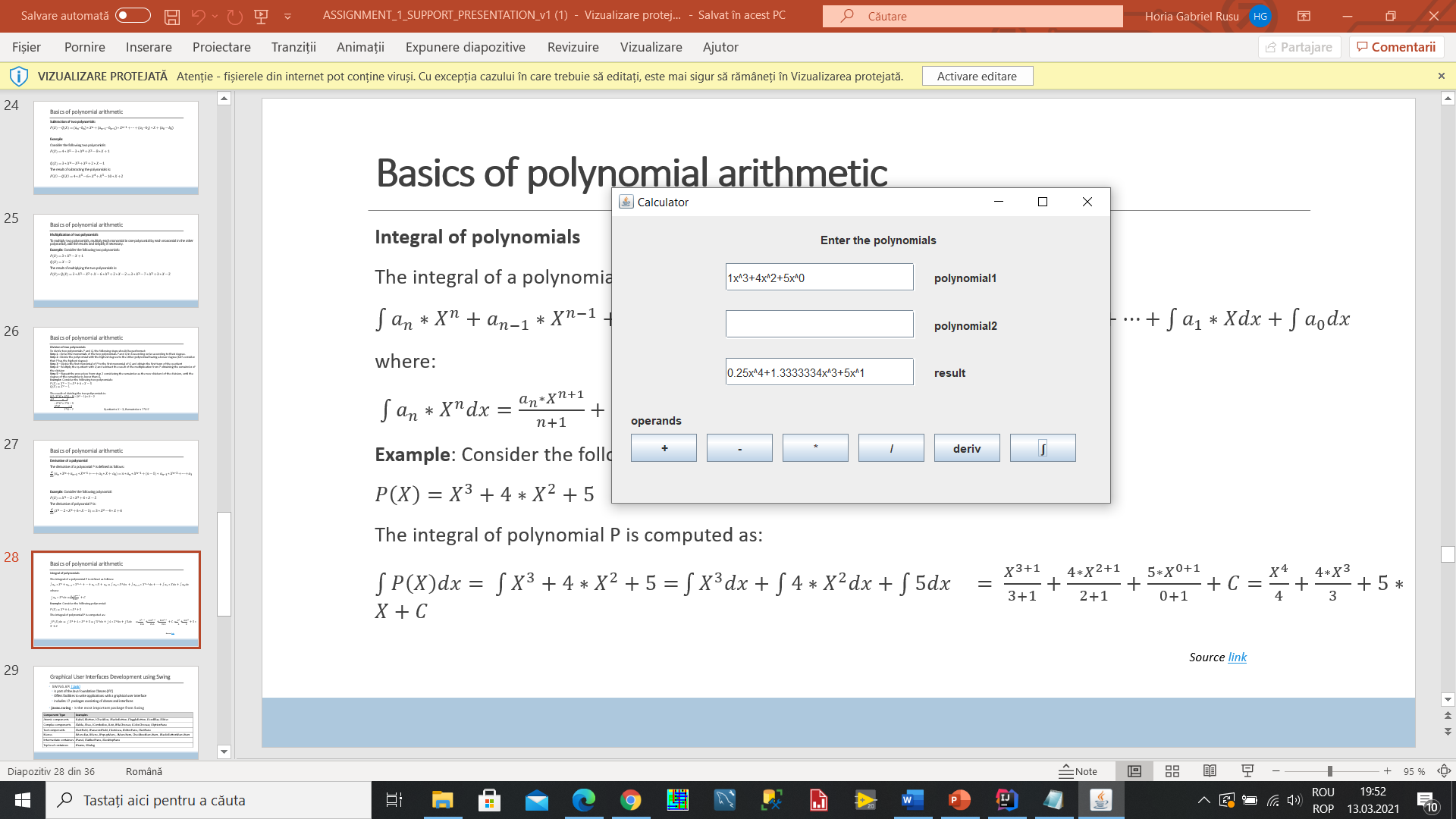
1. Derivation:

The algorithm for derivation is made by taking the original monomials from the original polynomial and setting up new monomials for the result with the coefficient equal with the original coefficient plus the power and the power is decremented. The result is transformed into a string and then displayed.



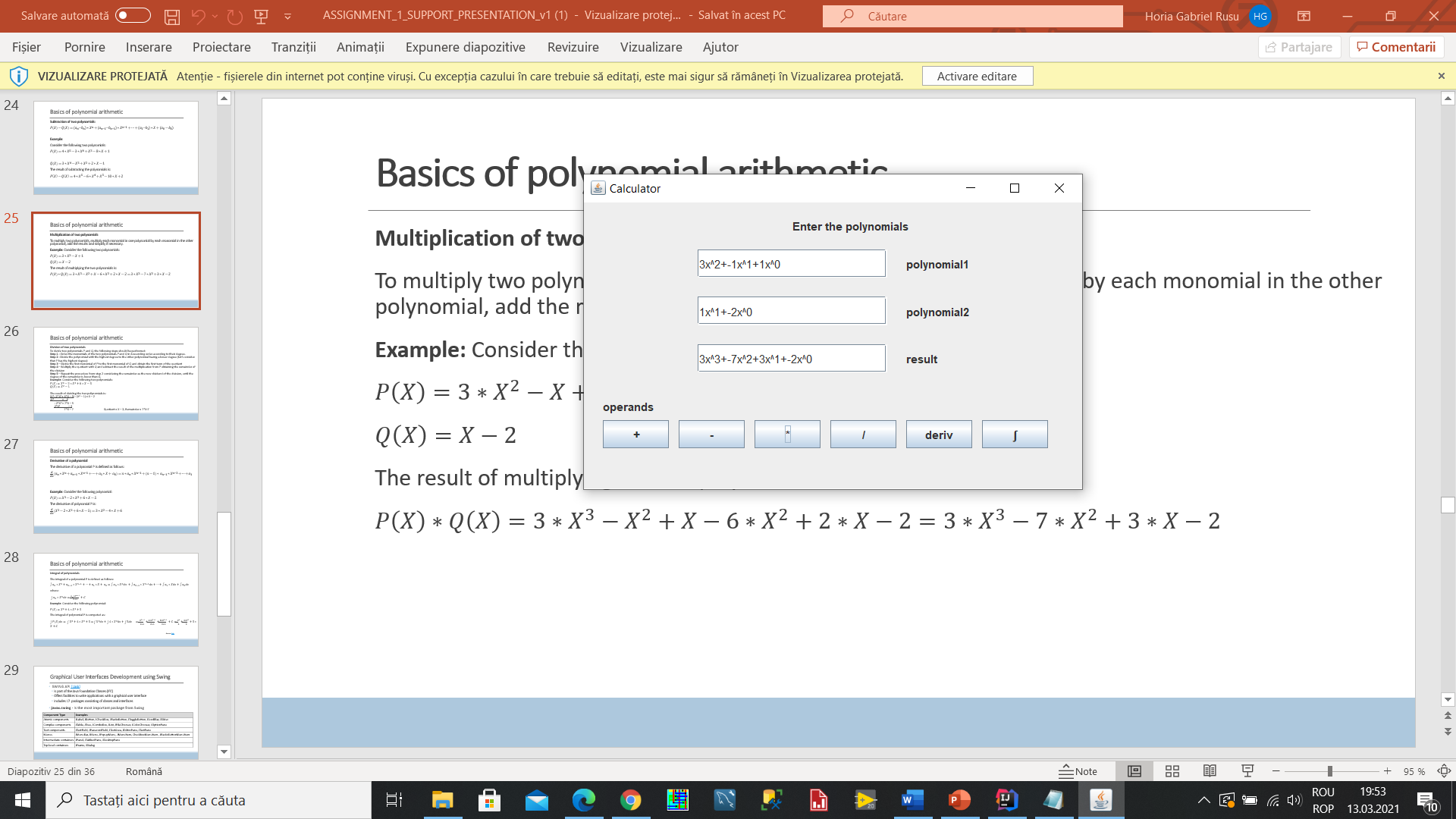
1. Integration:

The algorithm for integration is made by taking the original monomials from the original polynomial and setting up new monomials for the result with the coefficient equal with the original coefficient divided by the power and the power is incremented. The result is transformed into a string and then displayed.

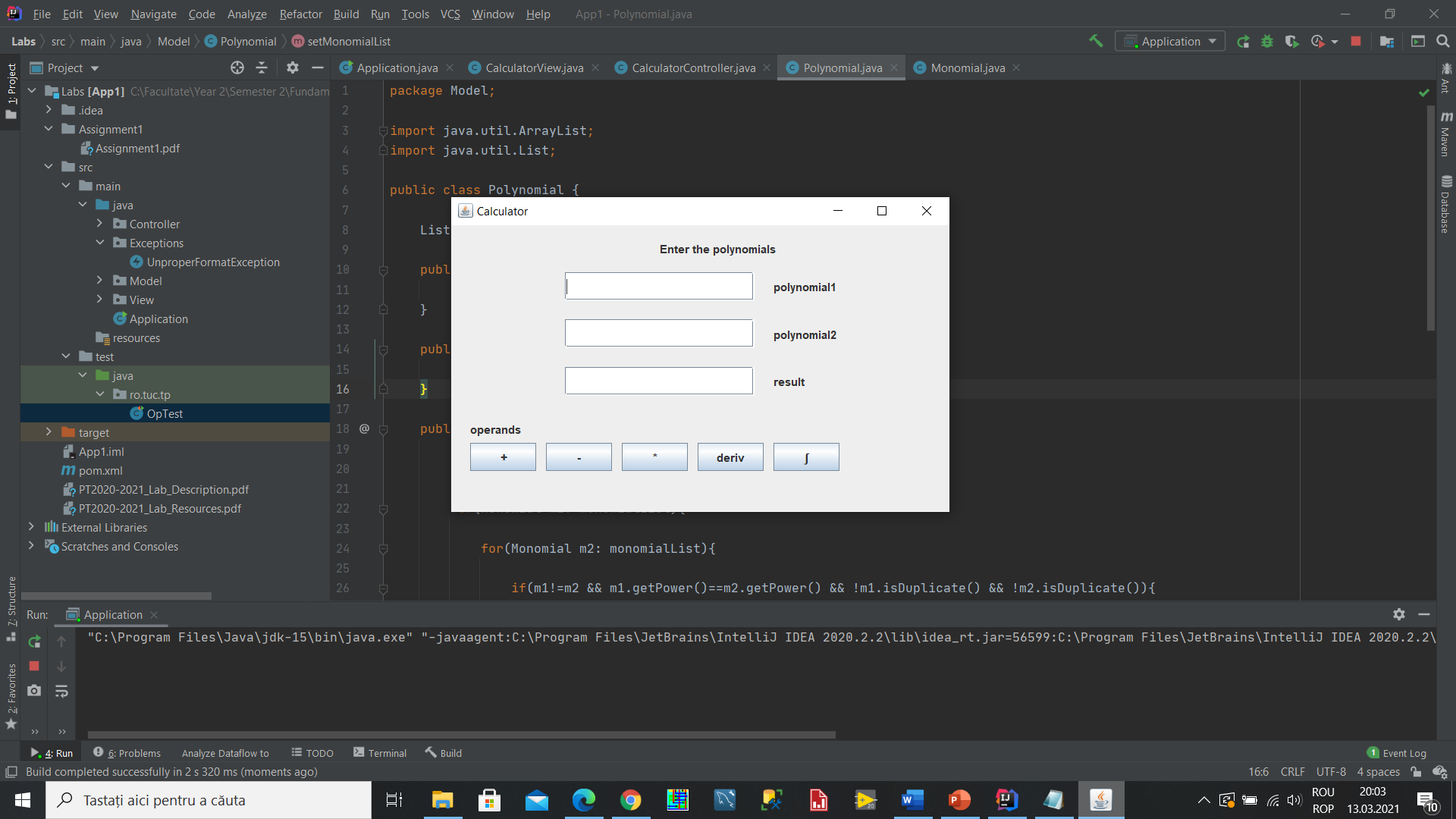


1. Multiplication:

The algorithm for multiplication is made by taking the original monomial list and setting up the new monomial list where each element has the coefficent equal with the multiplication of two other coefficients and the sum of the powers. Then the duplicate elements are removed and the final result is transformed into string form and then displayed.



Graphical User Interface

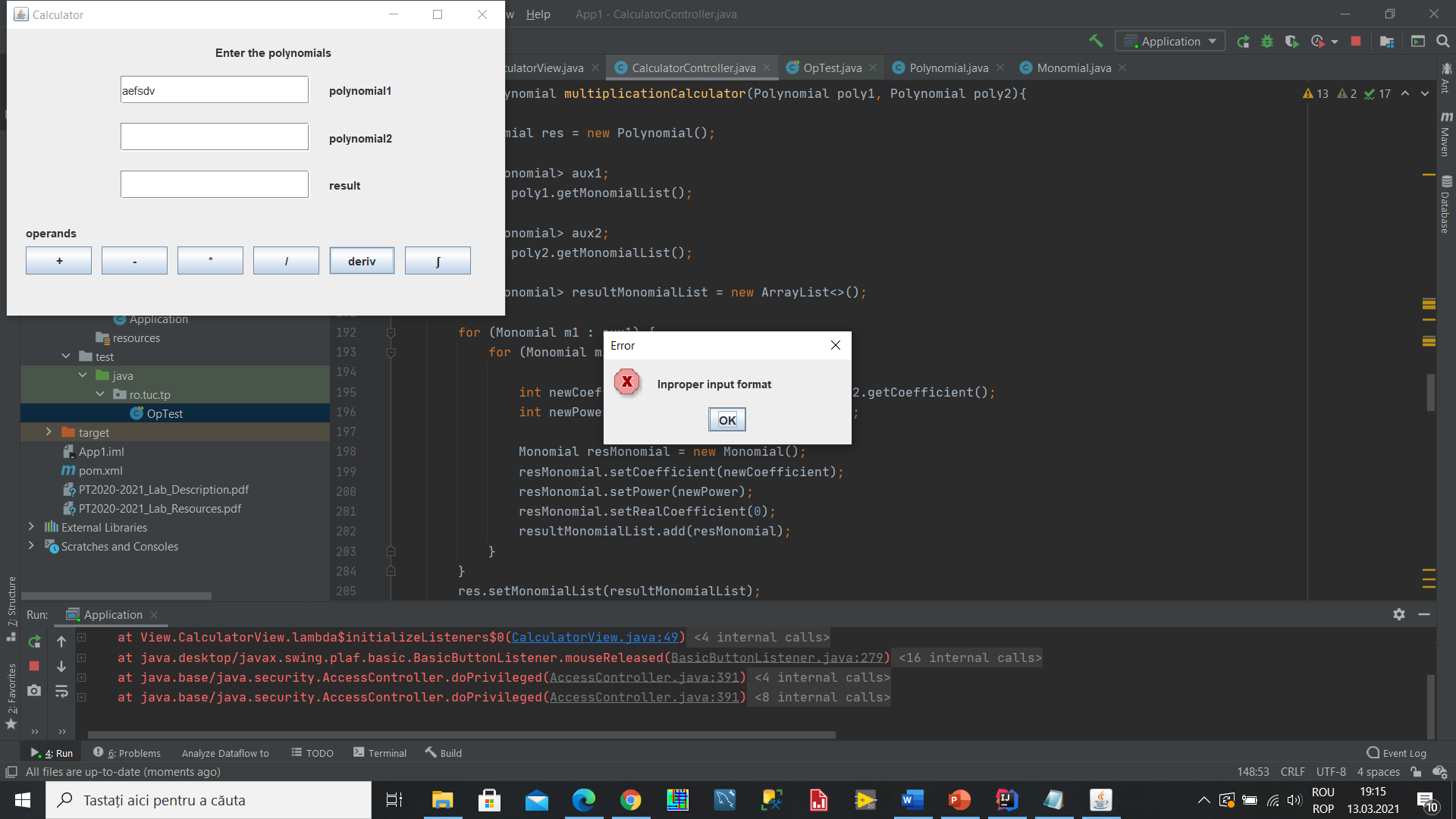


The interface has three text fields for polynomials. The first two are made for inputing the polynomials made for calculation, meanwhile the third one is used for the result that needs to be displayed. The operands are shown below and each one of them has a dedicated button that when pressed makes a different computation and displayes the reseult.

The condition is that the polynomial have a specific format, which is :

Coeffx^power+Coeffx^power+...

Any other form of input will be rejected.



5.) Results

In this section the result of the testing will be presented in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| What I test | Input Data | Expected Output | Actual Output |
| Addition | 3x^3+-4x^7; -5x^3+15x^2 | -2x^3+-4x^7+15x^2 | -2x^3+-4x^7+15x^2 |
| Subtraction | 3x^3+-4x^7; -5x^3+15x^2 | 8x^3+-4x^7+-15x^2 | 8x^3+-4x^7+-15x^2 |
| Multiplication | 3x^3+-4x^7; -5x^3+15x^2 | -15x^6+45x^5+20x^10+-60x^9 | -15x^6+45x^5+20x^10+-60x^9 |
| Derivation | 3x^3+-4x^7 | 9x^2+-28x^6 | 9x^2+-28x^6 |
| Integration | 3x^3+-4x^7 | 0.75x^4+-0.5x^8 | 0.75x^4+-0.5x^8 |

The testing has been done by inputing the above written data and calling the various methods and then comparing the results with the expected output and seeing if they match, which they do.

6.) Conclusions

The project was useful for remembering some of the basic concepts of object oriented programming as well as learning some new concepts or using some already known concepts in practic. The use of the MVC pattern, the use of the regular expressions and the use of foreach instead of the standard for function are some of the things that I learned while working on this project. The modelling of the problem is a very important part of the programming and having a clear idea of what the project should look like in terms of classes and how they all relate to each other is extremely important and when done before the actual implementation starts saves a lot of times and helps a lot with having a clear picture of the project.

Some future improvements that can be made are the implementation of the division function, which was unsuccesfull in this regard due to some problems when it comes to the repeated division and the ordering of the monomials according to the power in the monomial list. This could be solved with some more time. Also the disply of the result could be better.

Other than the required operations which were not implemented, other possible operations could be finding the roots of a polynomial, computing the square of a polynomial and computing the value of the polynomial in a certain point.

7.) Bibliography

<https://www.geeksforgeeks.org/list-interface-java-examples/>

<https://www.geeksforgeeks.org/for-each-loop-in-java/>

<https://www.vogella.com/tutorials/JavaRegularExpressions/article.html>

ASSIGNMENT\_1\_SUPPORT\_PRESENTATION\_v1

Object-Oriented Programming - Lecture Slides of prof. Marius JOLDOS