***Programming Techniques***

***Assignment 1***

***Polynomial Calculator***

Student : Scutea Emilia

Group: 30423

Laboratory Assistant: Antal Marcel

Teacher: Ioan Salomie

Technical University of Cluj Napoca

2021

Contents

[**1.** **Objectives** 3](#_Toc66980857)

[**2.** **Problem Analysis, Modelling, Scenarios, Use Cases** 4](#_Toc66980858)

[**I.** **Problem Analysis** 4](#_Toc66980859)

[**II.** **Modelling** 4](#_Toc66980860)

[**III.** **Scenarios and Use Cases** 4](#_Toc66980861)

[**3.** **Design** 8](#_Toc66980862)

[**I.** **Class Design and UML Diagram** 8](#_Toc66980863)

[**II.** **Packages and Relationships** 9](#_Toc66980864)

[**III.** **Algorithms and Data Structures** 10](#_Toc66980865)

[**IV.** **Graphical User Interface** 10](#_Toc66980866)

[**4.** **Implementation** 12](#_Toc66980867)

[**I.** **Class Polynomial** 12](#_Toc66980868)

[**II.** **Class MathematicalOperations** 12](#_Toc66980869)

[**III.** **Class Alert** 12](#_Toc66980870)

[**IV.** **Controller** 13](#_Toc66980871)

[**V.** **Main** 14](#_Toc66980872)

[**5.** **Results** 14](#_Toc66980873)

[**6.** **Conclusions** 15](#_Toc66980874)

[**7.** **Bibliography** 15](#_Toc66980875)

# **Objectives**

The main purpose of this project is to design a calculator which performs operations on polynomials. The polynomials used should have integer coefficients and only one variable.

In order to implement a functional application, we make use of the following mathematical operations:

1. Addition of two polynomials
2. Subtraction of two polynomials
3. Multiplication of two polynomials
4. Division of two polynomials
5. Derivation of a polynomial
6. Integration of a polynomial

For a user-friendly approach we implement a graphical user interface in order to perform the above mentioned operations.

# **Problem Analysis, Modelling, Scenarios, Use Cases**

## **Problem Analysis**

A system which performs mathematical operations on polynomials can be used in any domain mainly because of the great importance of mathematics in any specific field of work. Performing mathematical operations on polynomials is commonly used by highschool and univeristy students, which study mathematics during the years of school, but also by workers in any domain that requires mathematical computations.

The application provides a user-friendly interface where the user can easily introduce the input and select the desired operation because of the buttons and textfields displayed on the application. There are also given some instructions on how to use the polynomial calculator and how to introduce the polynomial, such that the usage of the calculator gets easy with no errors encountered.

## **Modelling**

A polynomial consists of a list of monomials. A monomial is an object which has two main attributes: the coefficient of the variable and the power of the variable. Performing operations on polynomials means performing operations on each monomial from the list of monomials and more specifically, the operations are performed on each coefficient and power attributes of each monomial from the list.

## **Scenarios and Use Cases**

In order to perform operations, the user must introduce two polynomials in the displayed text fields. The format of a polynomials is the following: coefficient **x** ^ exponent. The coefficient can be an integer or a double value, while the exponent must be an integer nonnegative value.

An example of a correct input of the polynomials is:

1. 2x+1
2. 2+7x+3x^2
3. 4x^5+7x^2+6

Note that there must be no space between any of the introduced values or variables.

The introduced variable must always be **x**.

If an input does not have succesive powers of the variable, like in the third example where the forth, third and first power of the variable are missing, there is no need to introduce them with 0 as a coefficient.

Another remark ist that even if the result will always be displayed in increasing order of the power of x, the polynomials can be introduced both in decreasing or increasing order of the variable’s power.

The use case diagram of this application is the following:

Diagram

Description automatically generated

*Use case: Addition*

Performed by: user

Best case scenario:

1. User introduces the two polynomials in the provided text fields
2. User presses the button **+**
3. The program reads and validates the two introduced polynomials
4. The program performs the addition and computes the result
5. The result is displayed by the program

*Use case: Subtraction*

Performed by: user

Best case scenario:

1. User introduces the two polynomials in the provided text fields
2. User presses the button -
3. The program reads and validates the two introduced polynomials
4. The program performs the subtraction and computes the result
5. The result is displayed by the program

*Use case: Multiplication*

Performed by: user

Best case scenario:

1. User introduces the two polynomials in the provided text fields
2. User presses the button **\***
3. The program reads and validates the two introduced polynomials
4. The program performs the multiplication and computes the result
5. The result is displayed by the program

*Use case: Division*

Performed by: user

Best case scenario:

1. User introduces the two polynomials in the provided text fields
2. User presses the button **/**
3. The program reads and validates the two introduced polynomials
4. The program performs the division and computes the result
5. The result is displayed by the program

*Use case: Integration*

Performed by: user

Best case scenario:

1. User introduces one or two polynomials in the provided text fields
2. User chooses on which polynomial to perform the operation by pressing the radio button provided next to the text field
3. User presses the button **/**
4. The program reads and validates the introduced polynomial
5. The program performs the integration and computes the result
6. The result is displayed by the program

*Use case: Derivation*

Performed by: user

Best case scenario:

1. User introduces one or two polynomials in the provided text fields
2. User chooses on which polynomial to perform the operation by pressing the radio button provided next to the text field
3. User presses the button **()’**
4. The program reads and validates the introduced polynomial
5. The program performs the derivation and computes the result
6. The result is displayed by the program

# **Design**

## **Class Design and UML Diagram**

The design of this project is based in the Model-View-Controller design, thus there are 3 main packages in which we are interested in.

Diagram

Description automatically generated

The class *Polynomial* defines the data structure used in the entire implementation: the definition of the polynomial. A polynomial is represented by a hashmap which contains an integer as a key and a double as a value.

The key represents the power of the monomial, while the value represents the coefficient of the monomial. Each polynomial contains a list of monomials, each monomial being in fact an entry of the defined hashmap. To sum up, the hashmap represents several entries of monomials.

The class *MathematicalOperations* implements the mathematical operations defined in 1. Objectives: addition, subtraction, multiplication, division, derivation, integration. Each operation is defined as a public method having 1 ore more parameters of the type *Polynomial* and the return type *Polynomial*.

The class *Controller* controlls the execution of the mathematical operations selected in the user interface. The methodes defined in this class are used to validate the inputs, to delete the inputs and to perform mathematical operations on the inputs: addition, subtraction, multiplication, division, integration and derivation.

The class *Alert* defines several methods used to generates alerts. Alerts are used to inform the user on input mistakes (invalid format of the input) or to provide guidance to perform the integration and derivation in case if the user did not execute all required steps in order for the program to successfully perform the chosen operation.

The class *Main* launches a new window, starts the application and creates a new instance of the class Controller, which will execute the operations chosen by the user.

## **Packages and Relationships**

The application contains 3 packages:

1. Logic

This package contains the class *MathematicalOperations* which includes methods performing operations on the polynomials.

1. Model

This package contains the class *Polynomial*, which defines the data structure used to

store polynomials.

1. GUI

As the name suggests, this package is used for classes defining the graphical user interface. The classes included in this package are class Controller and class Alert.

The relationship between class Controller and class MathematicalOperations, as the ones between class Controller and class Polynomial and between class Controller and class Alert is a „uses a” relationship. This dependency relationship occurs because the specified classes are local variables in methods, parameters and fields in the class Controller.

## **Algorithms and Data Structures**

In this implementation all algorithms used to implement the mathematical operations are trivial and are mainly used since highschool.

The addition and the subtraction are the simplest algorithms defined in this project. The algorithms are based just on adding/subtracting the coefficients of each monomial of the two polynomials.

The multiplication algorithm is based on multiplying each monomial of the first polynomial with each monomial of the second polynomial, i.e. multiplying the coefficients and adding the powers. Another remark concerning this algorithm ist that the obtained monomials which have the same power must be added in order to get a correct and valid result.

The division algorithm is created using the long division method. The first requirement is that the maximum degree of the first polynomial must be greater or equal than the maximum degree of the second polynomial. If this condition is met, then polynomials can be divided using the specified method. The division process stops when the condition is no longer met and if there are remained monomials, then they represent the remainder of the divison.

The derivation performs the simple algorithm of multiplying each coefficient with the power of the variable and then decreasing the value of the power with 1.

The integration process performs the inverse of the derivation algorithm. The power of each monomial is increased by 1, while the coefficient is divided by the new value of the power.

The data structures used in the implementation of this six algorithms are:

* HashMap<Integer,Double> which is the data structue used to define a polynomial
* ArrayList<Integer> used in the division algorithm to get the powers of each monomial in ascending order

## **Graphical User Interface**

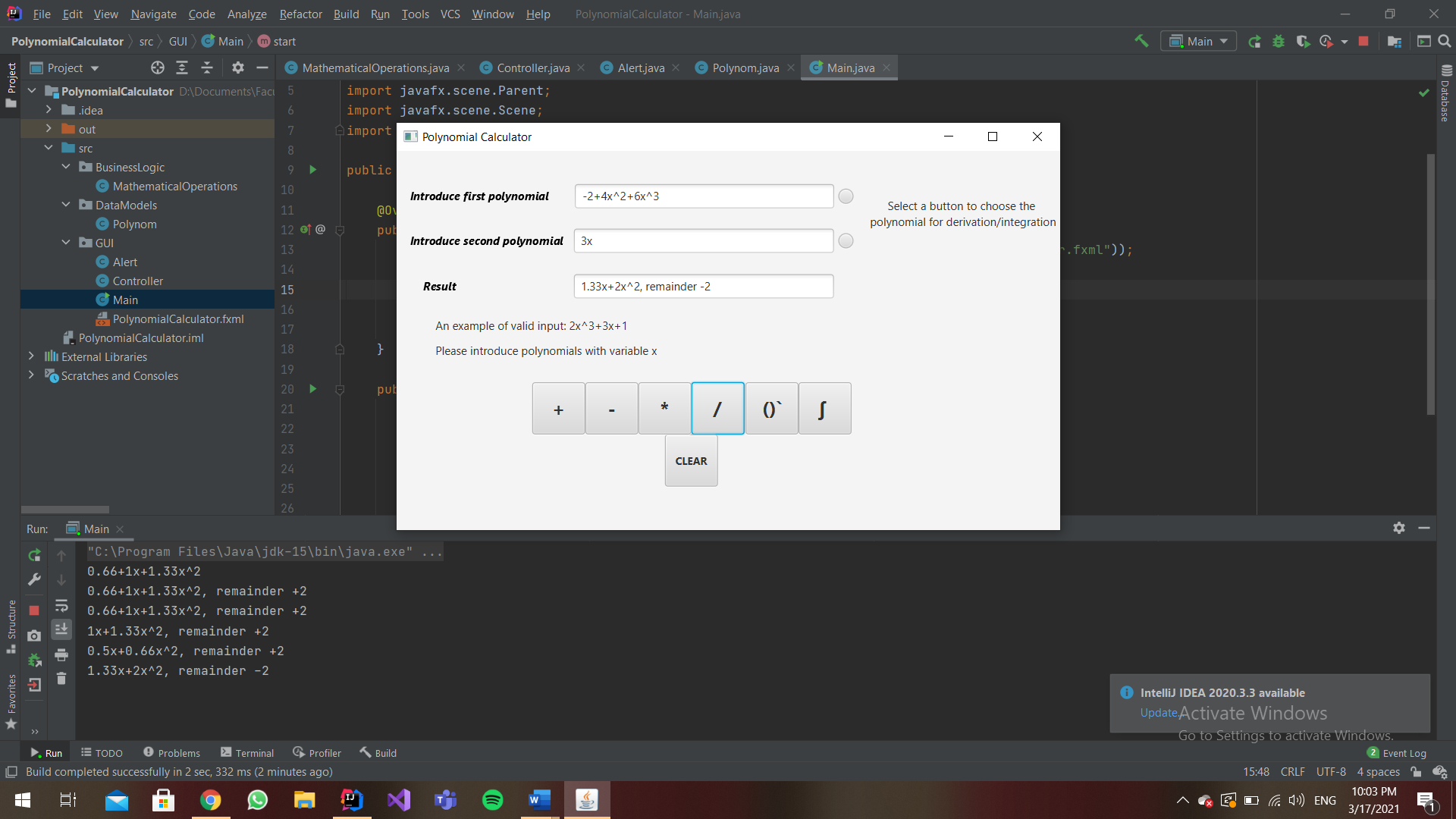
The graphical user interface is a simple interface with buttons and text fields. The buttons are used to select with mathematical operation to perform and they have labels which describe de operation: + for addition, - for subtraction, \* for multiplication, / for division, ()’ for derivation and ∫ for integration. There is also a button used to clear the input i.e. t odelete the already introduced polynomials.

The text fields are used to introduce the two polynomials and also to display the result. The text field used for displaying the result is not editable.

There are also some labels provided to guide the user in introducing the correct format of the polynomials and also to provide some information about the division and the integration.

If the introduced polynomials do not have the correct format, an alert will appear on the screen, specifying the mistake in the input.

The radiobuttons next to the input text fields are provided in order for the user to select on which polynomial wants to perform derivation or integration, as written in the label next to them.



# **Implementation**

## **Class Polynomial**

In the class *Polynomial* is defined the data structue used to store the polynomials: HashMap <Integer, Double>. The polynomial contains a list of monomials thus each entry of the hashmap represents a monomial. The key is of Integer type and represents the power of the monomial while the value is of Double type and represents the coefficient of the monomial.

The defined methods in this class are:

* setPolynomial (Polynomial polynomial): sets the polynomial
* getPolynomial (): returns/gets the polynomial
* add (double coefficient, int power): creates a new entry in the hashmap of the polynomial having the key *power* and the value *coefficient*

## **Class MathematicalOperations**

Class MathematicalOperations contains all methods used to execute the operations of the calculator. There are only methods defined in this class:

* addition (Polynomial a, Polynom b): returns the sum of the two polynomials
* subtraction (Polynom a, Polynom b): returns the difference of the two polynomials
* multiplication (Polynom a, Polynom b): returns the product of the two polynomials
* division (Polynom a, Polynom b): returns the quotient and the remainder of the division of the two polynomials
* derivation (Polynom a): returns the derivative of the polynom
* integration (Polynom a): returns the integral of the polynom

## **Class Alert**

Class Alert provides the user with several alerts which will pop up on the screen once the user has introduced an invalid input or has not followed all steps in order to perform division/integration.

## **Controller**

The controller class defines methods which are set on action when buttons in the user interface are pressed. The defined attributes are:

* RadioButton buttonP1 🡪 defines the radiobutton next to the first input text field
* RadioButton buttonP2 🡪 defines the radiobutton next to the second input text field
* TextField polynom1txt 🡪 defines the text field used to introduce the first polynom
* TextField polynom2txt 🡪 defines the text field used to introduce the second polynom
* String text1 🡪 used in the validation of the first input
* String text2 🡪 used in the validation of the second input
* Polynom polynom1 🡪 the first polynom
* Polynom polynom2 🡪 the second polynom
* Alert alert 🡪 an instance of class Alert in order to display alerts, when needed

The methods defined in the class Controller:

* validateInput (TextField polynomtxt, Polynom polynom, String text): returns true if the input is valid, otherwise returns false
* validateInput1 (): calls the method validateInput with parameters reffered to the first polynom
* validateInput2 (): calls the method validateInput with parameters reffered to the second polynom
* displayResult (double coefficient, char sign, int power, String resultString): returns a string representing a monomial in the result of an operation, with a double coefficient
* displayResult (int coefficient, char sign, int power, String resultString): returns a string representing a monomial in the result of an operation, with an integer coefficient
* computeResult (Polynom polynom, String resultString): returns a string representing the result of an operation, calls the method displayResult for each monomial
* addInputs (): performs the addition of the polynomials, if the inputs are valid and displays the result on the screen by calling the method computeResult
* subtractInputs (): performs the subtraction of the polynomials, if the inputs are valid and displays the result on the screen by calling the method computeResult
* multiplyInputs (): performs the multiplication of the polynomials, if the inputs are valid and displays the result on the screen by calling the method computeResult
* divideInputs (): performs the division of the polynomials, if the inputs are valid and displays the result on the screen by calling the method computeResult
* deriveInput (Polynom polynom): performs the derivative of a polynom
* derive (): checks if one of the radiobuttons is pressed and if so, calls the method deriveInput with the associated polynomial
* integrateInput (Polynom polynom): performs the integral of a polynom
* integrate (): checks if one of the radiobuttons is pressed and if so, calls the method integrateInput with the associated polynomial
* clear (): clears the two polynomials, the result and the associated text fields

## **Main**

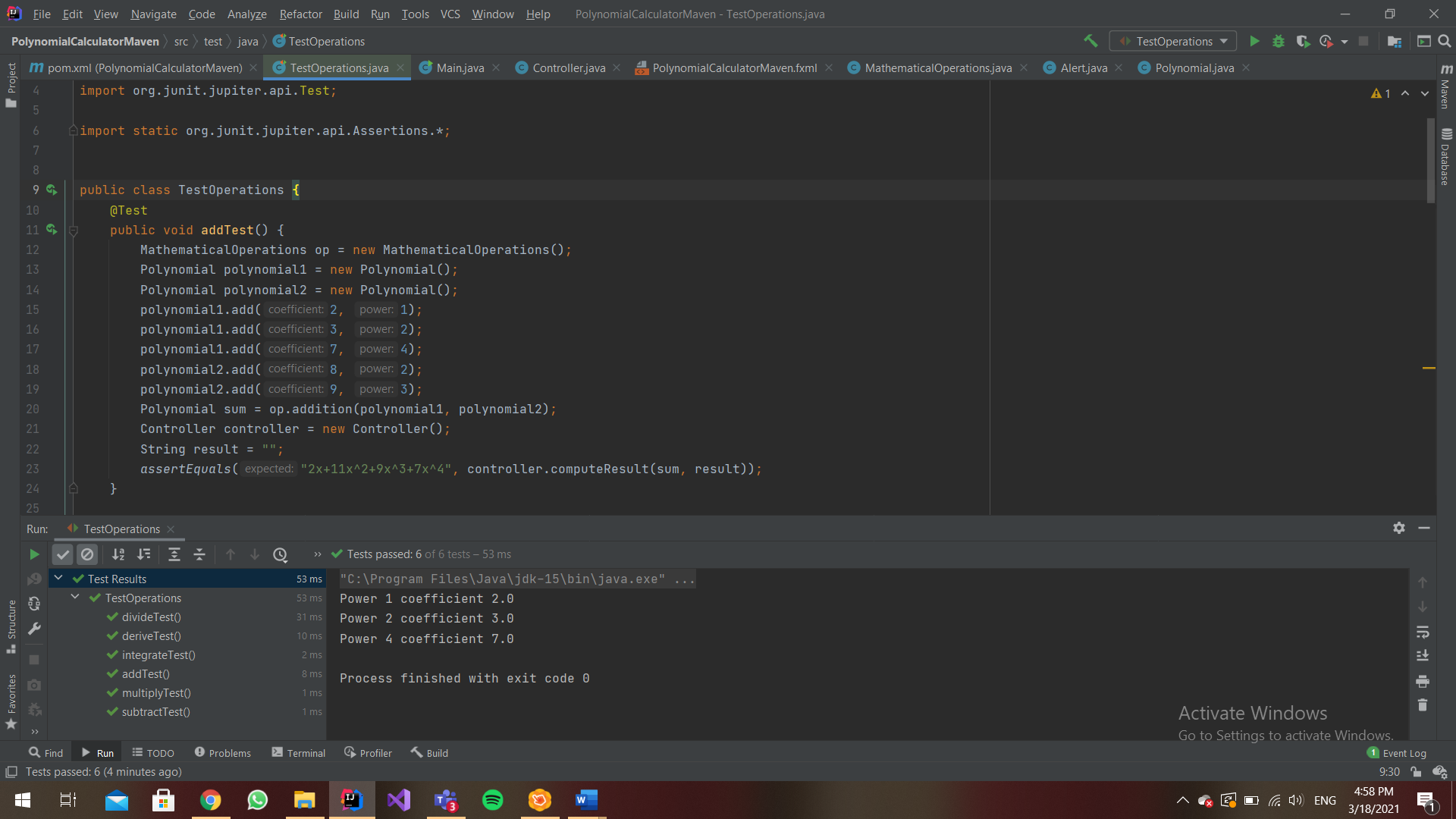
In the Main class, the method start creates a new stage which will be displayed on the screen, while in the main method the stage is launched and a new instance of the controller is created.

# **Results**

To perform the mathematical operations, the user has to introduce two polynomials in the providade text fields. In order to execute the operations, the input must have a valid format. An example of va valid input would be:

* 2x^3+4x^4
* 15x+12
* 10+4x^3+7x^6

The testing of the calculator was performed by JUnit using assertions: assertEqual():



# **Conclusions**

I can affirm that during the development of this assignment I have got more familiar with some aspects about how classes should be implemented and organized in an application.

Some future development ideas could be:

* A more interactive user interface, perhaps with animations
* More advacned mathematical operations: polynomials with two variables
* Polynomials with negative powers of the variable

# **Bibliography**

<https://en.wikipedia.org/wiki/Polynomial_long_division>

<https://openjfx.io/>

<https://mvnrepository.com/>