POLYNOMIALS CALCULATOR

Cosmin Stretea

Group 30422

1. Objective of this project

The main objective of the project is to create a fully functional polynomial calculator. This consists of three secondary objectives which are the following: identify the polynomial model using OOP principles, create the main controller that will have the main operations of the calculator and implement the user interface of the application using external tools.

2. Problem analysis, modeling, scenarios, use cases

a. Problem analysis

To be able to design a solution for our application, we have to understand first what is a polynomial. It is an expression which consists of variables also called indeterminates and coefficients. For our problem we use only polynomials of one variable. The general form of a such polynomial is:

$$a_n x^n + a_{n-1} x^{n-1} + \dots + a_2 x^2 + a_1 x + a_0$$

It is represented as a sum of terms. Those terms, also called monomials, consist of coefficient (a_n), x is the indeterminate and the exponent n (x^n).

To assure a good functionality of the application we have to choose a suitable data structure for storing our polynomials. We can observe that the useful information that need to be stored about our polynomials is represented by coefficients and exponents of every monomial.

For instance, the polynomial $6x^3 + 2x^2 + 7x - 2$ will be converted to relevant data in a form like this:
[(6, 3), (2, 2), (7, 1), (-2, 0)]. Each pair is represented as (coefficient, exponent) of the term monomial.

In this case we will be able to implement and perform the operations of polynomials such as addition, subtraction, division, modulo, multiplication, differentiation and integration.

b. Problem modeling

At the start of the application the user is prompted with an interface. In that interface the user can insert one or two polynomials (specific cases when the operation differentiate or integrate is selected). By selecting on of the listed operations the user will be able to get the result.

The main features of the application are listed below:

- Addition
- Subtraction
- Division
- Modulo
- Multiplication
- Differentiation
- Integration

If the selected operations are differentiate or integrate then the user will be able to enter only one polynomial. The second input text field will be invalid and showed as grayed in that case.

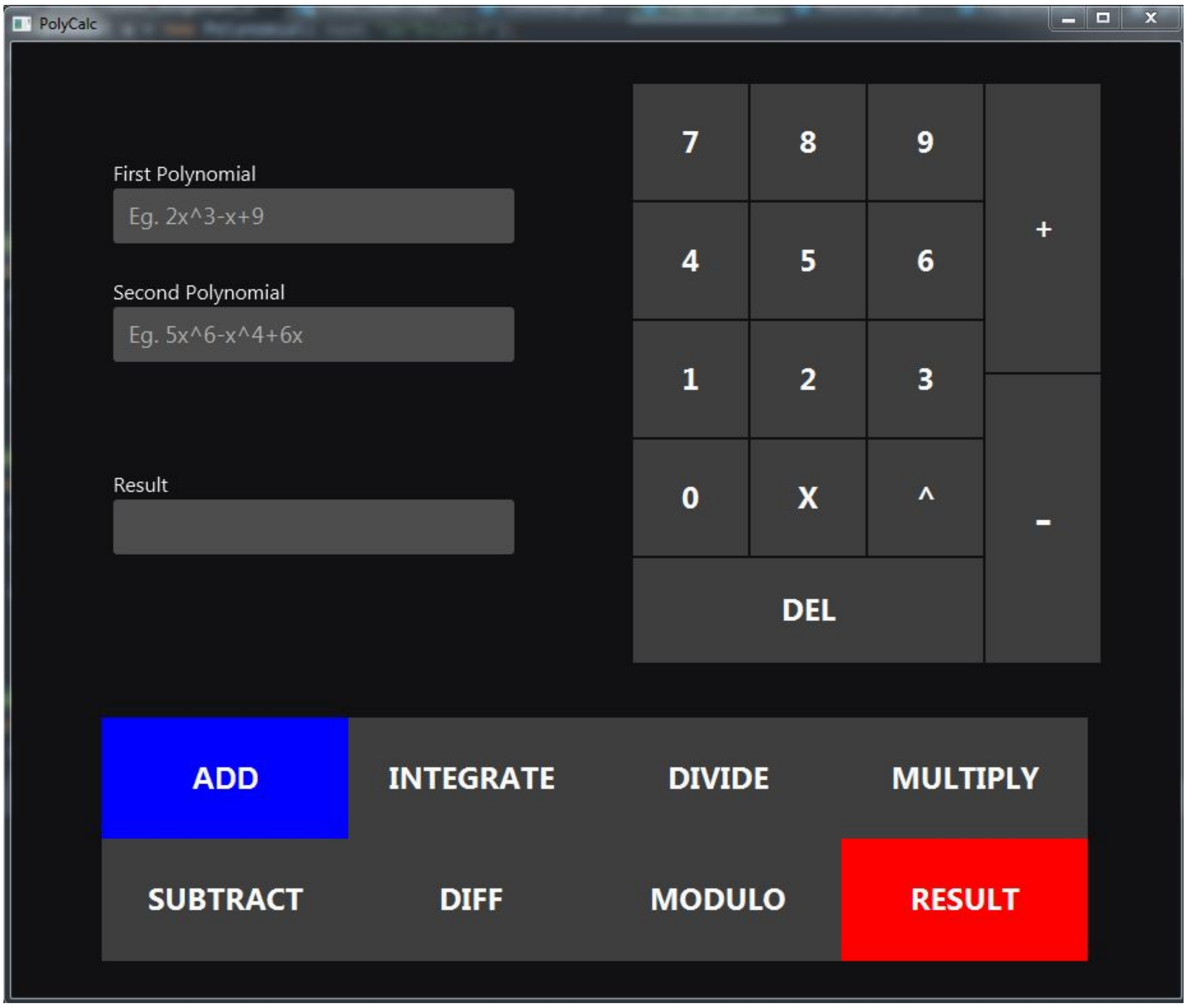
On the other hand, before printing the result of the selected operation on the entered polynomials, the application will prompt a message to notify the user if the input was invalid and it is unable to proceed with the computation.

In order to get the result, the user has to press the "result" button after he selected the operation and entered the polynomials. The outcome will be printed in the result field.

c. Scenarios and use cases

A use case is a methodology used in system analysis to identify, clarify and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. The method creates a document that describes all the steps taken by a user to complete an activity.

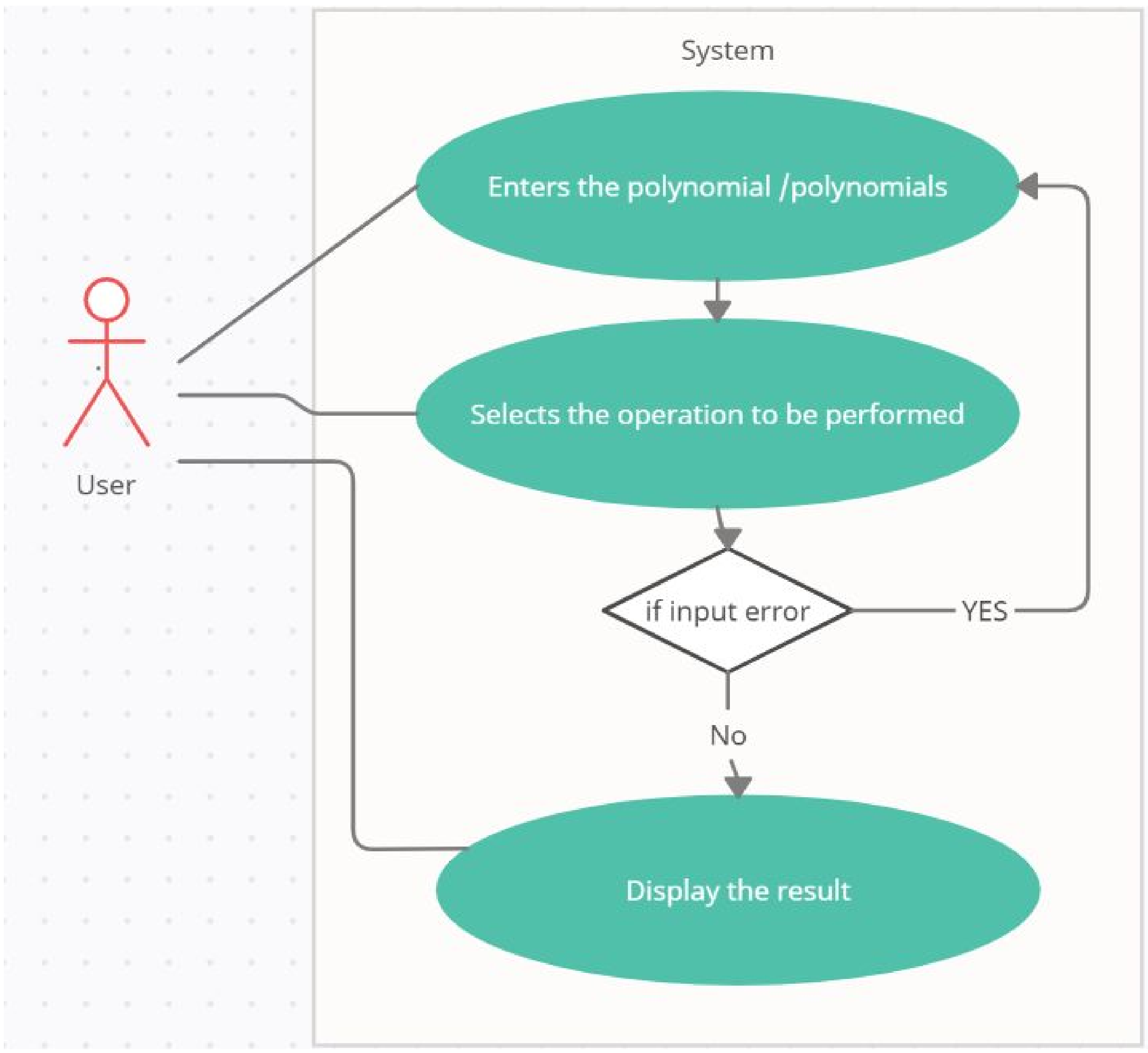
In our case this such activity is computing the outcome of two polynomials on which it is performed an operation listed above and display the result in a user friendly manner.



The user will be able to use the generated input buttons on the right side of the application to input the polynomial/polynomials for computation. Those polynomials will be stored internally until the associated text field will be modified. To enter the poynomials the user can also use the keyboard along the presented buttons, but to reduce the probability of getting input errors the application has this feature. Along the buttons to insert text there is also a button to delete the input field. If the user wants to delete only one character from the selected text field he has to use the backspace button from the keyboard due to the fact the generated button “del” from the interface deletes the whole text field input.

The bottom section is used to select the operation to be performed from which the result will be shown in the result field. It makes sure that the input is in a good format so we won’t generate any internal corruption errors. The selected operation is highlighted in a vibrant blue color. By switching between the operations the fields will be changed accordingly. That means in the case of integration or differentiation the second text field will be disabled and the operation will be applied only on the first text field. In that case the internal state of the second entered polynomial will not be changed until the text field is available and edited by the user. The button “result” will act on the current entered polynomials and perform the currently selected operation and print the outcome in the result text field. In case of error the result field won’t change and neither the internal state of the polynomials and a message will be displayed.

The left side of the interface is used to display all the information such as input, result and errors. The user will be able to see all the application’s changes displayed in that area and also the alert messages in case if an error occurs. The presented input text field also display, as a placeholder text, informations about the correct input of a polynomial. As the user enters any input those placeholders will disappear and the input will be shown. The result text field is is editable in such way the user can only copy the result from it. The error label is placed between the input fields and result field and is displayed only when there is an error.


System
Enters the polynomial /polynomials
Selects the operation to be performed
if input error
YES
No
Display the result
User

The use case presents the actor, which in our case is the user that interacts with the application. He can perform several actions on the two chosen polynomials (or the single one, by case), such as addition, subtraction, multiplication, division, modulo, integration and differentiation.

3. Design

● Class Diagram

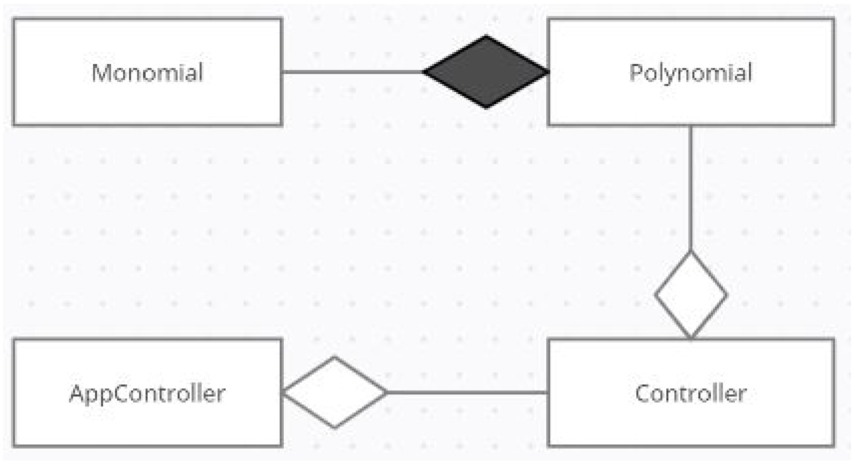
The Class Diagram showed below consists of four classes: Monomial, Polynomial, AppController and Controller.

- **Monomial Class**: it is used to store individual data about the terms of the polynomial and manage any operation at the lowest level of the polynomial.

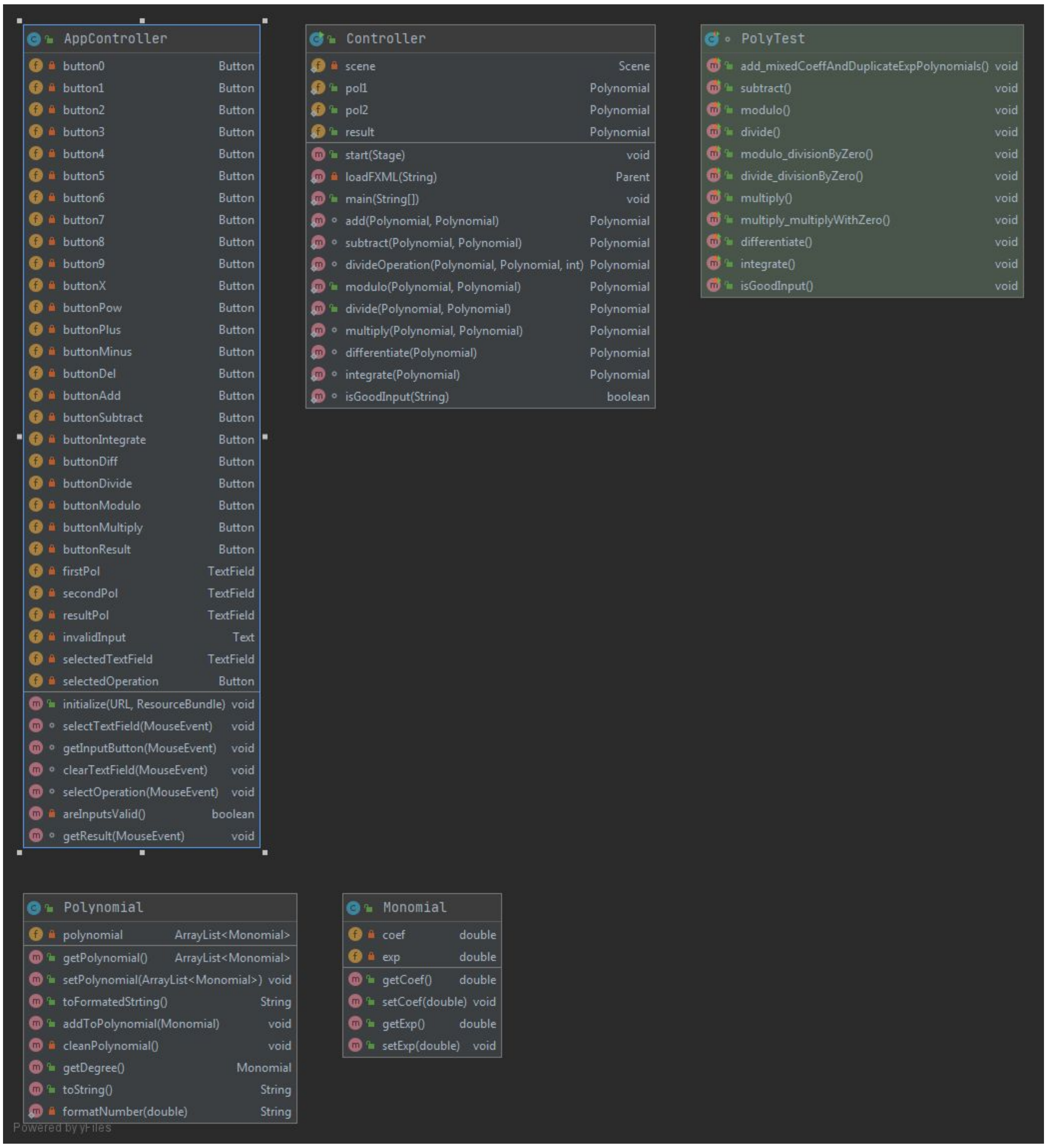
- **Polynomial Class**: the main class of the application which is used to store the polynomial as a list of Monomial Class' objects and manage the state of the polynomial.

- **Controller Class**: the class that manages the main operations of the application such as addition, subtraction, division, modulo, multiplication, integration and differentiation.

- **AppController Class**: it is the class that manages the all the interface states using the resources from the Controller Class.
- **PolyTest Class**: the class used to test the main operations on samples of polynomials.



The methods and the field of each class are presented in the image below:



AppController	
button0	Button
button1	Button
button2	Button
button3	Button
button4	Button
button5	Button
button6	Button
button7	Button
button8	Button
button9	Button
buttonX	Button
buttonPow	Button
buttonPlus	Button
buttonMinus	Button
buttonDel	Button
buttonAdd	Button
buttonSubtract	Button
buttonIntegrate	Button
buttonDiff	Button
buttonDivide	Button
buttonModulo	Button
buttonMultiply	Button
buttonResult	Button
firstPol	TextField
secondPol	TextField
resultPol	TextField
invalidInput	Text
selectedTextField	TextField
selectedOperation	Button
initialize(URL, ResourceBundle)	void
selectTextField(MouseEvent)	void
getInputButton(MouseEvent)	void
clearTextField(MouseEvent)	void
selectOperation(MouseEvent)	void
areInputsValid()	boolean
getResult(MouseEvent)	void

Controller	
scene	Scene
pol1	Polynomial
pol2	Polynomial
result	Polynomial
start(Stage)	void
loadFXML(String)	Parent
main(String[])	void
add(Polynomial, Polynomial)	Polynomial
subtract(Polynomial, Polynomial)	Polynomial
divideOperation(Polynomial, Polynomial, int)	Polynomial
modulo(Polynomial, Polynomial)	Polynomial
divide(Polynomial, Polynomial)	Polynomial
multiply(Polynomial, Polynomial)	Polynomial
differentiate(Polynomial)	Polynomial
integrate(Polynomial)	Polynomial
isGoodInput(String)	boolean

PolyTest	
add_mixedCoeffAndDuplicateExpPolynomials()	void
subtract()	void
modulo()	void
divide()	void
modulo_divisionByZero()	void
divide_divisionByZero()	void
multiply()	void
multiply_multiplyWithZero()	void
differentiate()	void
integrate()	void
isGoodInput()	void

Polynomial	
polynomial	ArrayList<Monomial>
getPolynomial()	ArrayList<Monomial>
setPolynomial(ArrayList<Monomial>)	void
toFormatedString()	String
addToPolynomial(Monomial)	void
cleanPolynomial()	void
getDegree()	Monomial
toString()	String
formatNumber(double)	String

Monomial	
coef	double
exp	double
getCoef()	double
setCoef(double)	void
getExp()	double
setExp(double)	void

● Data structure

The most used data structures in this application are the following: String, double and ArrayList. The String variables were used to convert data to a readable format so it can be later displayed in the application. The double variables were most used in the Monomial Class but most frequently accessed in the Controller Class in which the application stores the main functionality represented by the polynomials' operations.

The last most used data structure and the most important is ArrayList. The main usage of it was to store the terms of the polynomial, also known as monomial, in a way it can be very easy to manage when performing any operation on the current polynomial.

Regarding the implementation of the user interface, the most used data structures were: Button and TextField. The buttons were used to insert the input as polynomials and to trigger actions such as selecting the operation to be performed and eventually to get the result of the polynomials. The TextFields were used to display information to the application user such as the input and also the outcome of the operations in the specified fields.

● Class Design

A class can be described as a collection of objects (attributes), behavior (methods), or relationship with other objects and have common semantics. The principles of class design, however preliminary it may seem, are decisive of the foundation of an application. The main paradigm used is the so called "divide and conquer" which consists in splitting the task in many small tasks. In that way every Class has it's own responsability to do the task that is assigned to.

In a more top-view way this application is implemented using the MVC architecture. MVC architectural pattern follows an elementary idea, it must separate the responsibilities in any application on the following basis:

- **Model:** it handles data and business logic
- **View:** presents the data to the user whenever it is requested
- **Controller:** it manages the communication between Model and View through specific methods

● Algorithms

- <u>Addition</u>

The result of the addition is computed by adding the coefficients of the monomials that have the same exponent or adding to the result polynomial the ones that do not have any matching monomial with same exponent.

- Subtraction

The result of this operation is the same as the addition but it is obtained by subtracting the above specified monomials.

- Multiplication

The outcome is obtained by computing the product term-wise of the two given polynomials.

- Division

The division of the two given polynomials is performed using the long division algorithm.

- Differentiation

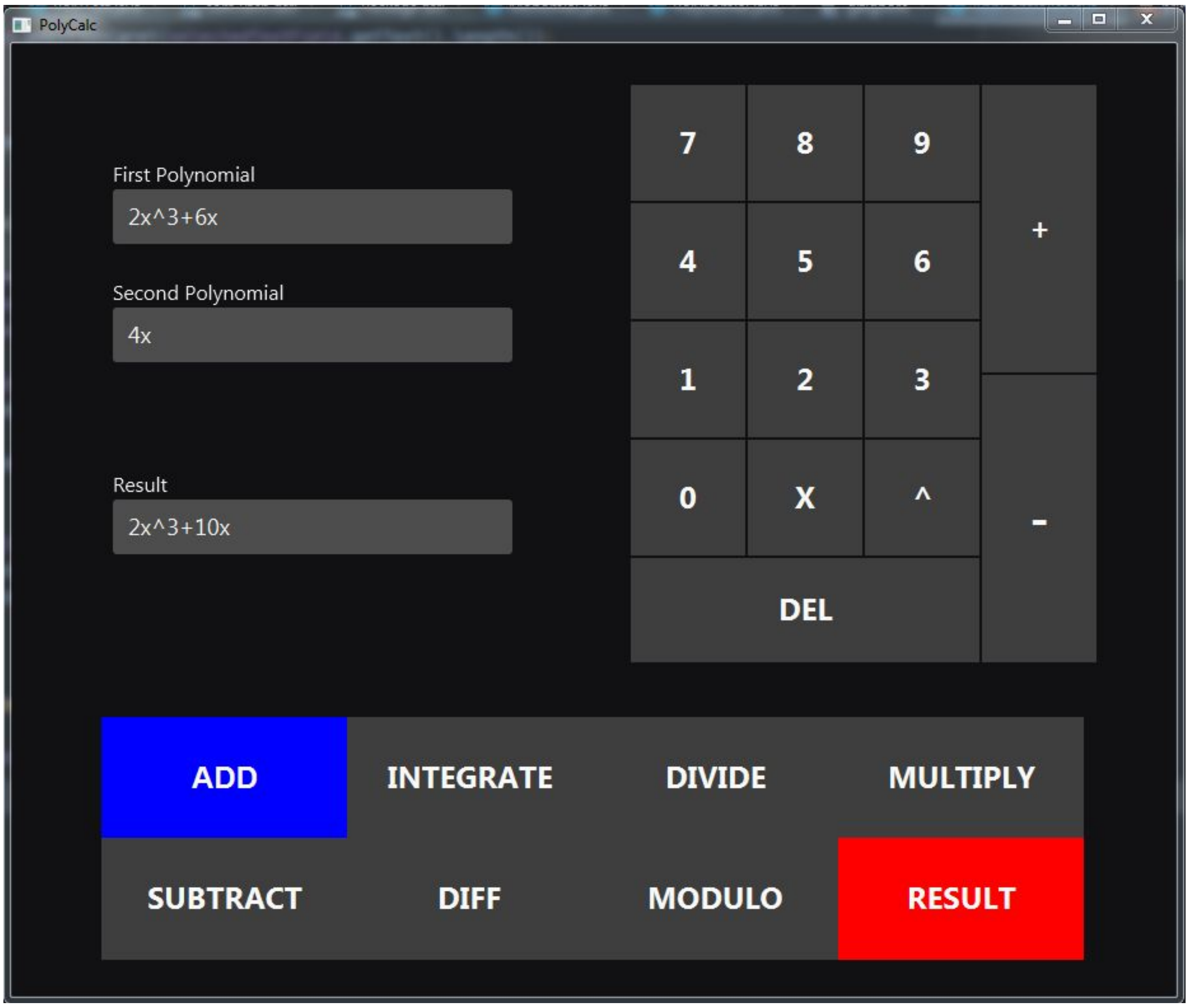
The algorithm consists of multiplying the coefficient of the monomial with the exponent and decrementing by one the exponent. In case the monomial has zero exponent then the term is removed.

- Integration

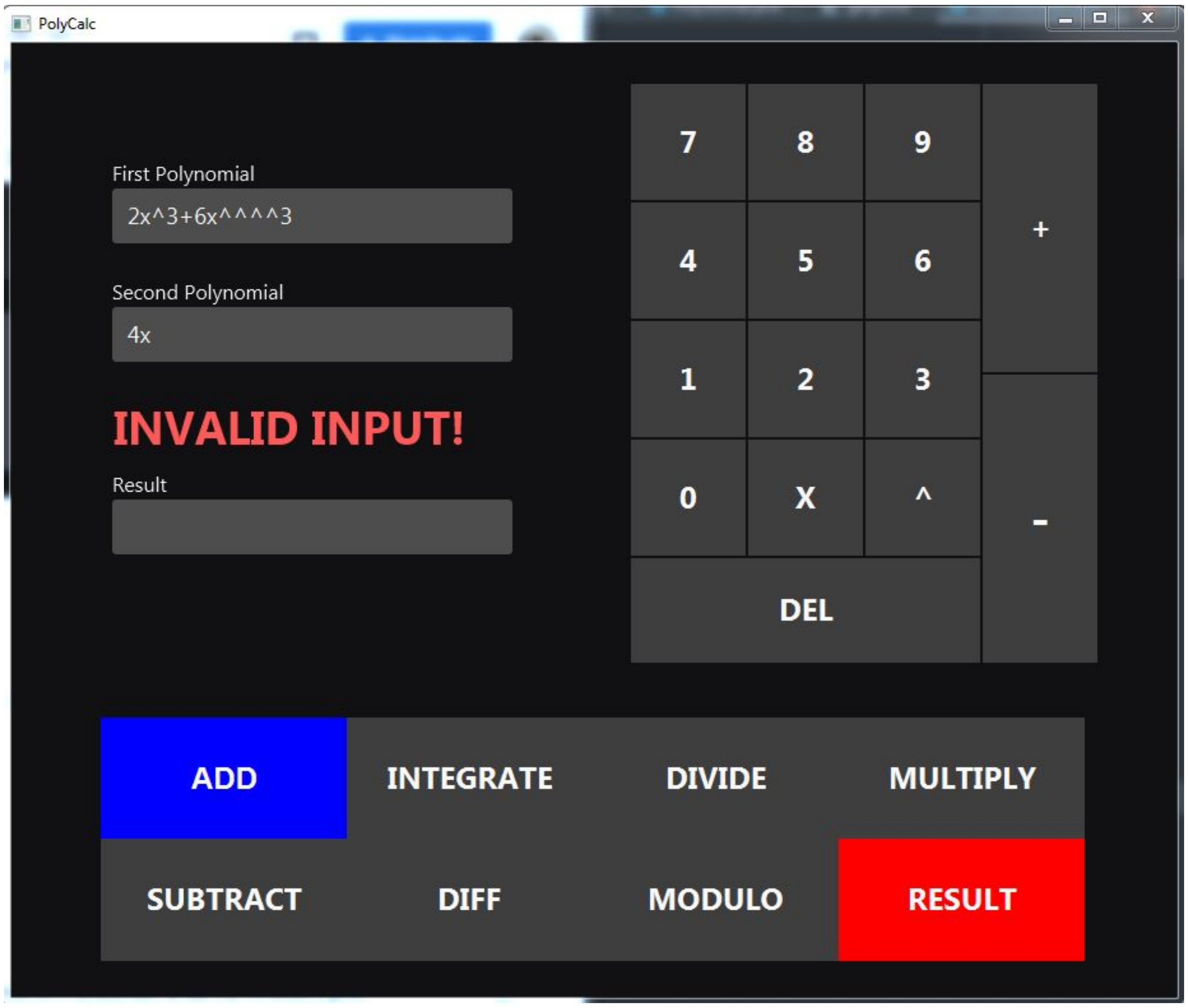
The integration algorithm is done by incrementing the exponent value by one and dividing the coefficient of the current monomial with our new exponent’s value.

- **User interface**

The user interface is the one showed below:



In case of an error at the input:



First Polynomial

2x^3+6x^^^^3

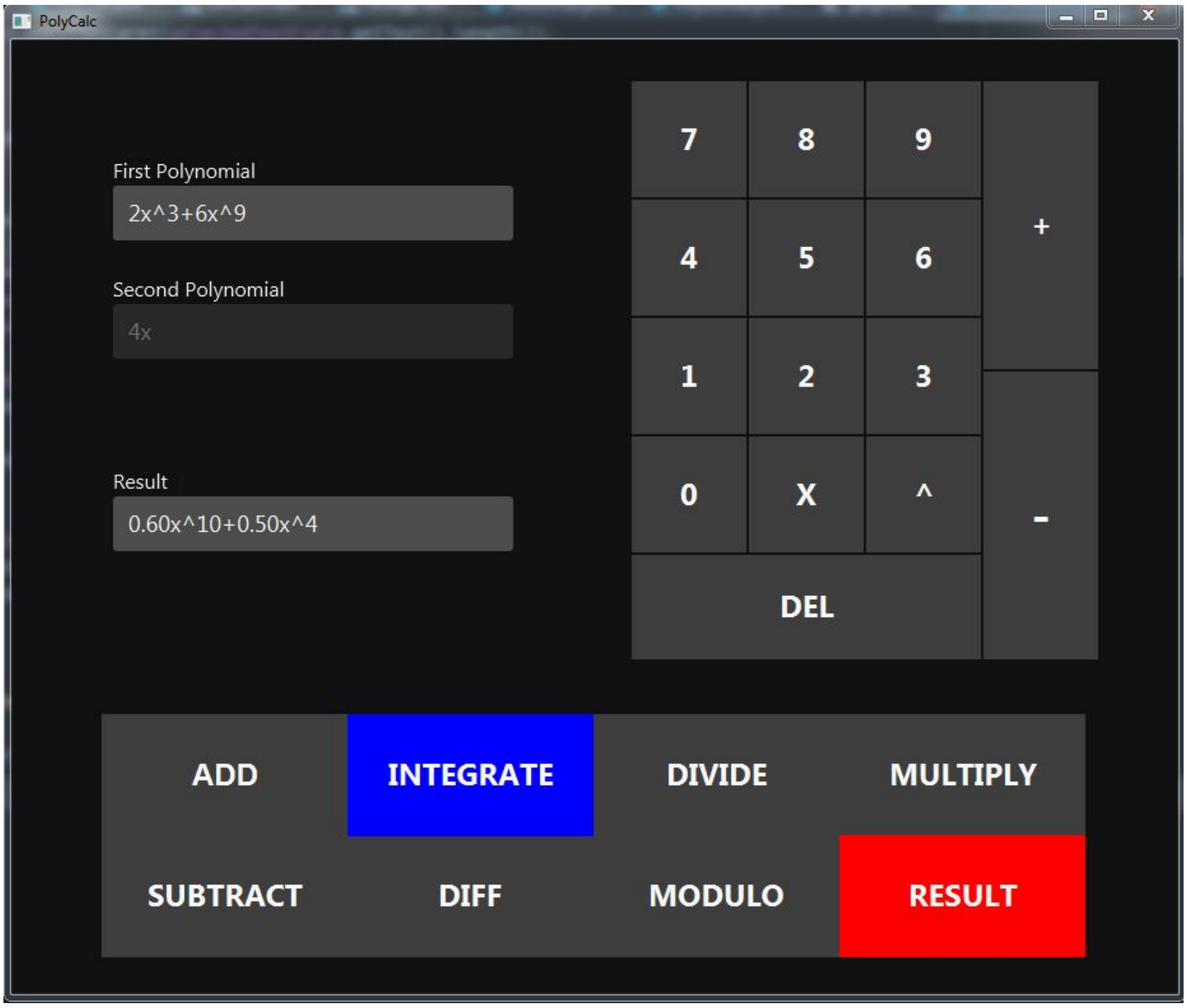
Second Polynomial

4x

INVALID INPUT!

Result

Special case of integration or differentiation:


PolyCalc

First Polynomial: 2x^3+6x^9

Second Polynomial: 4x

Result: 0.60x^10+0.50x^4

4. Implementation

The whole application is implemented on the paradigm of the MVC architecture. In that way we assure a good and clear usage of the classes to accomplish every task.

We cand start the detailed description of each class by the three main categories:

- **Model**
    - Monomial Class

The monomial class is used to manage the state and actions of an individual term of the polynomial to ensure the low level actions of an polynomial

The default *constructor* creates a monom with the default coefficient one and exponent zero. There is also implemented the *constructor* with parameters of coefficient and exponent.

The *fields* are represented by the private variables “coef” and “exp” which are declared as double type variables.

The main *methods* are the getters and setters for the private fields described above: getCoef(), setCoef(double coef), getExp(), setExp(double exp).

- Polynomial Class

The polynomial class is used to represent the main polynomial state and actions.

There are many *constructors* in this class. The default (empty) constructor creates an empty ArrayList, the constructor with a list of monomials creates an ArrayList with the given list. The polynomial can be created also only with one monomial which is given as a parameter and also from another polynomial which is also given as parameter. In the latter case the newly created polyomial will be a “hard copy” of the one given as parameter. The most important constructor is the one which takes as an input a string. It converts the string into a polynomial assuming the string represents a valid input string polynomial.

The only *field* used in this class is an ArrayList named “polynomial”.
The *methods* are the following: toString() which converts the polynomial into readable string format; cleanPolynomial() removes all monomials with exponent equal to zero; addToPolynomial() adds a monomial to a polynomial in such way it won’t be any duplicates of same exponent; getDegree() returns the monomial with the greatest exponent. There are also getters and setters for this class.

- **Controller**

  - Controller Class

This class manages the communication between the Model and the View of the application.

Due to the fact this class is the supposed Controller in our MVC architecture it has no *contructors.*

The *fields* are the “pol1” , “pol2” and “result” which represent the polynomials with which we are working in our application. The “scene” field is used to manage the interface of the application.

The main *methods* “add()”, “subtract()”, “divide()”, “modulo()”, “multiply()”, “integrate()” and “differentiate()” represent the main functionality of our application. Those methods are static ones that return the result of the two polynomials given as paramteres. The method “isGoodInput()” checks if the input string is a valid polynomial using regex.

- **View**

  - <u>AppController Class</u>

This class manages all the states of the user interface. It handles all the actions of the user on the displayed window.

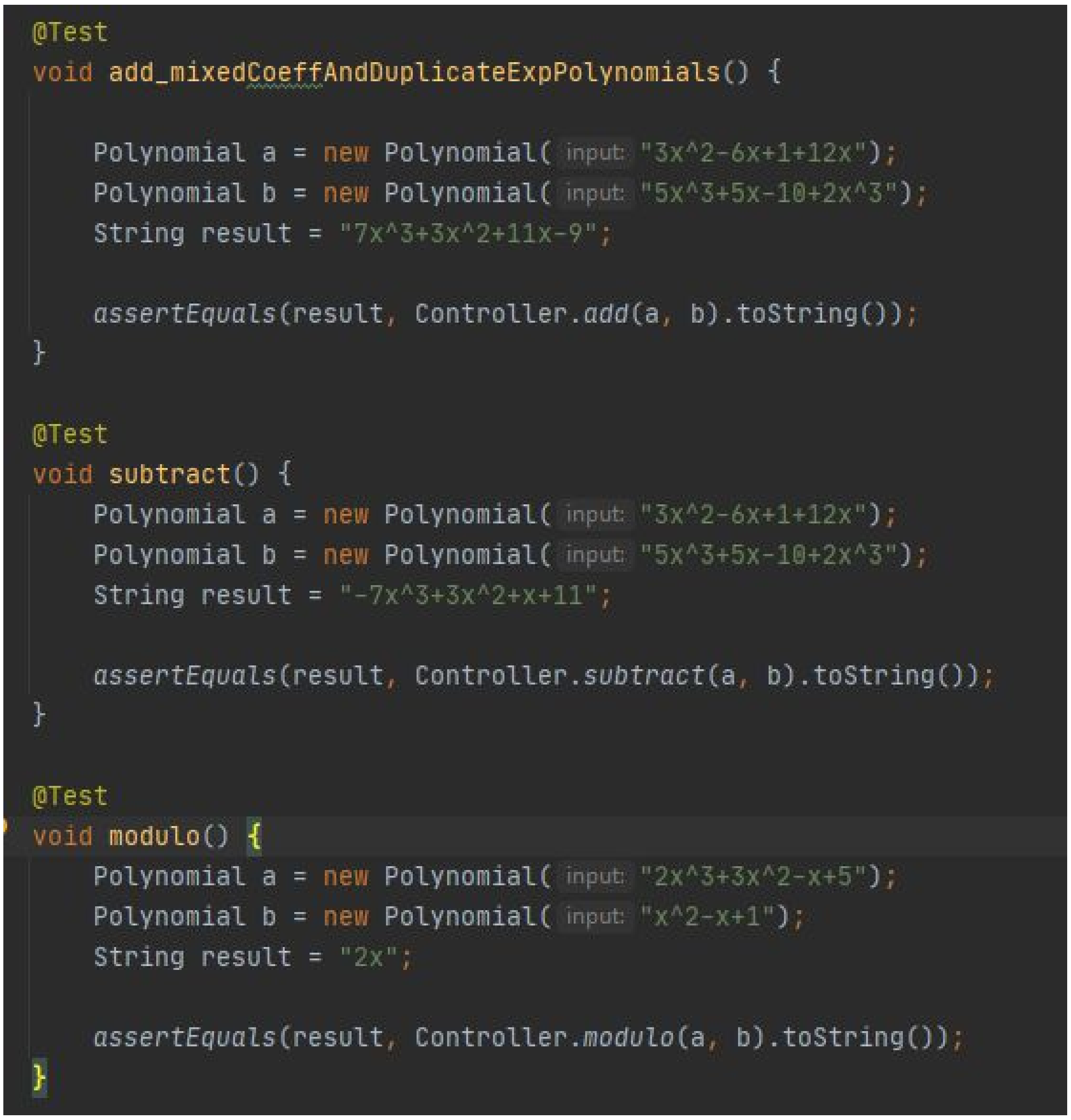
It has no *constructors* because it is a controller assigned to the primary fxml file which is used to give structure to the user interface.

The *fields* are represented by all the Buttons and TextFields used int the interface.

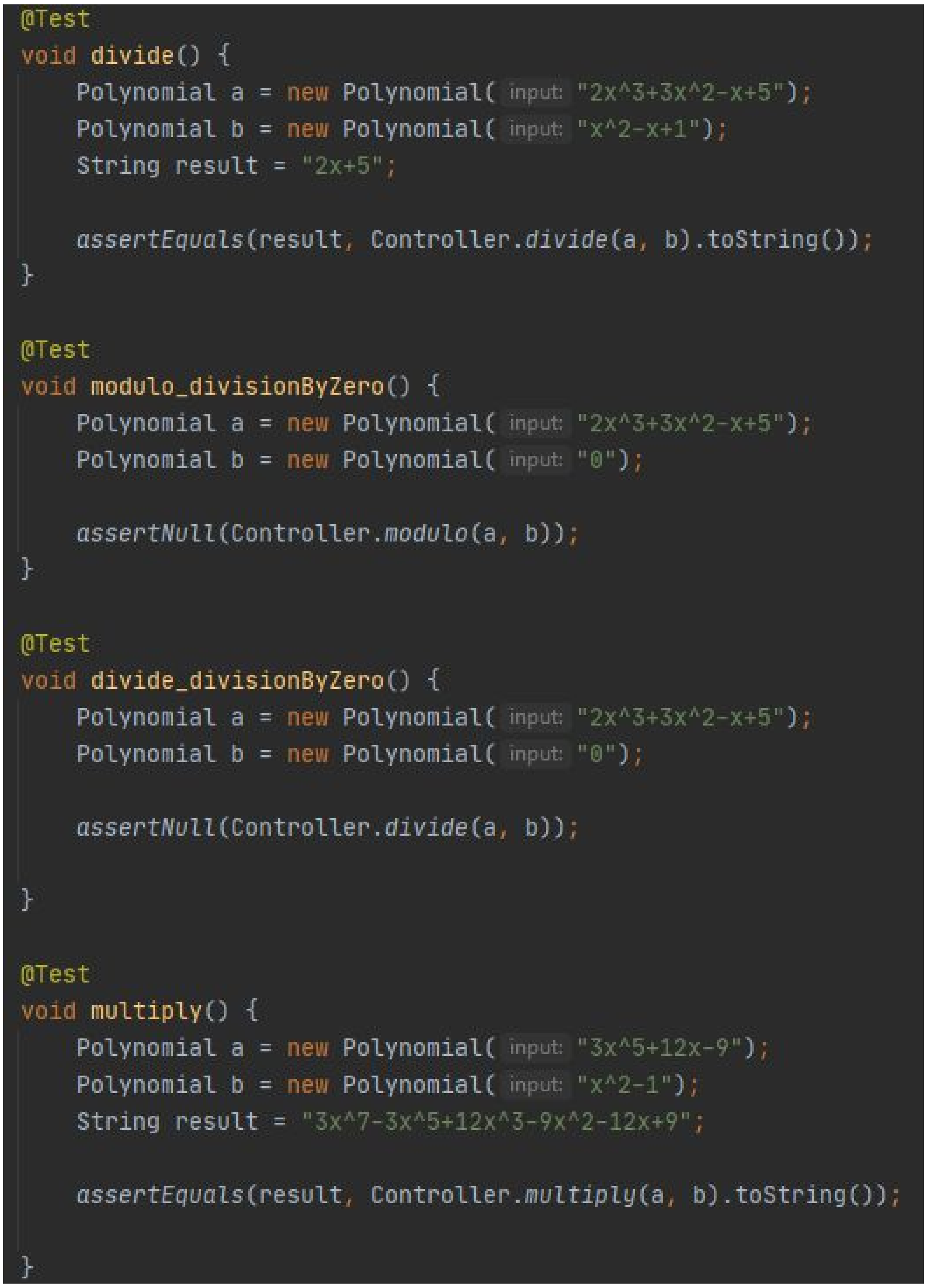
The *methods* are represented by the actions of the buttons. The most important methods are: “getReult()” which computes the result with all the current states; “areInputsValid()” checks the inputs of the text fields; “clearTextField()” deletes the current input in the selected text field.

5. Results

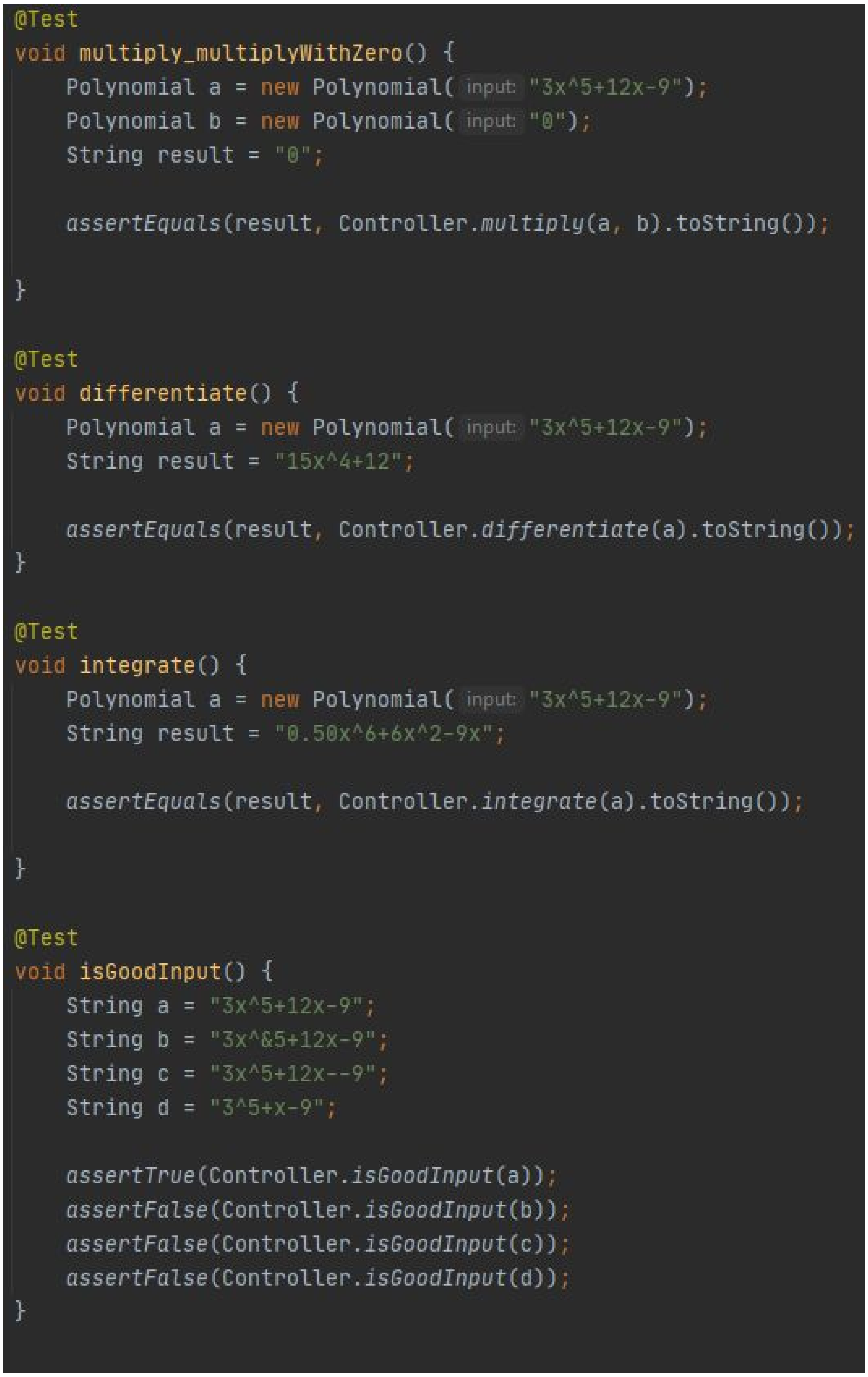
For the testing purposes we have used Junit. The testing was done on the main functionality of the application which consists of the methods: add, subtract, divide, modulo, multiply, integrate and differentiate.



```java
@Test
void add_mixedCoeffAndDuplicateExpPolynomials() {

    Polynomial a = new Polynomial( input: "3x^2-6x+1+12x");
    Polynomial b = new Polynomial( input: "5x^3+5x-10+2x^3");
    String result = "7x^3+3x^2+11x-9";

    assertEquals(result, Controller.add(a, b).toString());
}

@Test
void subtract() {
    Polynomial a = new Polynomial( input: "3x^2-6x+1+12x");
    Polynomial b = new Polynomial( input: "5x^3+5x-10+2x^3");
    String result = "-7x^3+3x^2+x+11";

    assertEquals(result, Controller.subtract(a, b).toString());
}

@Test
void modulo() {
    Polynomial a = new Polynomial( input: "2x^3+3x^2-x+5");
    Polynomial b = new Polynomial( input: "x^2-x+1");
    String result = "2x";

    assertEquals(result, Controller.modulo(a, b).toString());
}
```

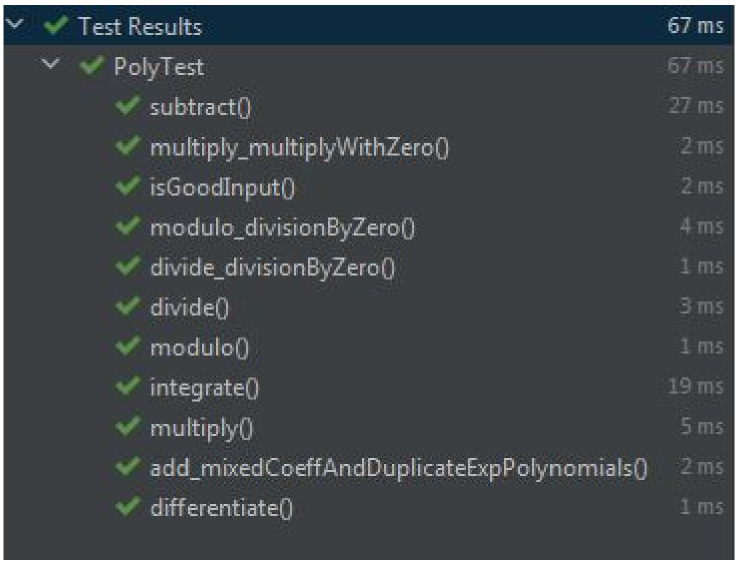


```
@Test
void divide() {
    Polynomial a = new Polynomial( input: "2x^3+3x^2-x+5");
    Polynomial b = new Polynomial( input: "x^2-x+1");
    String result = "2x+5";

    assertEquals(result, Controller.divide(a, b).toString());
}

@Test
void modulo_divisionByZero() {
    Polynomial a = new Polynomial( input: "2x^3+3x^2-x+5");
    Polynomial b = new Polynomial( input: "0");

    assertNull(Controller.modulo(a, b));
}

@Test
void divide_divisionByZero() {
    Polynomial a = new Polynomial( input: "2x^3+3x^2-x+5");
    Polynomial b = new Polynomial( input: "0");

    assertNull(Controller.divide(a, b));

}

@Test
void multiply() {
    Polynomial a = new Polynomial( input: "3x^5+12x-9");
    Polynomial b = new Polynomial( input: "x^2-1");
    String result = "3x^7-3x^5+12x^3-9x^2-12x+9";

    assertEquals(result, Controller.multiply(a, b).toString());

}
```



```java
@Test
void multiply_multiplyWithZero() {
    Polynomial a = new Polynomial( input: "3x^5+12x-9");
    Polynomial b = new Polynomial( input: "0");
    String result = "0";

    assertEquals(result, Controller.multiply(a, b).toString());

}

@Test
void differentiate() {
    Polynomial a = new Polynomial( input: "3x^5+12x-9");
    String result = "15x^4+12";

    assertEquals(result, Controller.differentiate(a).toString());
}

@Test
void integrate() {
    Polynomial a = new Polynomial( input: "3x^5+12x-9");
    String result = "0.50x^6+6x^2-9x";

    assertEquals(result, Controller.integrate(a).toString());

}

@Test
void isGoodInput() {
    String a = "3x^5+12x-9";
    String b = "3x^&5+12x-9";
    String c = "3x^5+12x--9";
    String d = "3^5+x-9";

    assertTrue(Controller.isGoodInput(a));
    assertFalse(Controller.isGoodInput(b));
    assertFalse(Controller.isGoodInput(c));
    assertFalse(Controller.isGoodInput(d));
}
```

The final result is that all the tests were successfully passed:



```
✓ Test Results                               67 ms
  ✓ PolyTest                                 67 ms
    ✓ subtract()                             27 ms
    ✓ multiply_multiplyWithZero()             2 ms
    ✓ isGoodInput()                           2 ms
    ✓ modulo_divisionByZero()                 4 ms
    ✓ divide_divisionByZero()                 1 ms
    ✓ divide()                                3 ms
    ✓ modulo()                                1 ms
    ✓ integrate()                            19 ms
    ✓ multiply()                              5 ms
    ✓ add_mixedCoeffAndDuplicateExpPolynomials() 2 ms
    ✓ differentiate()                         1 ms
```

6. Conclusions

This project was an interesting one even if it is considered to be a basic Java application. I am happy with the outcome of this assignment and thankful for this opportunnity. I learned how to apply all the theory the Java comes with and applied it, in a more practical manner, in the design of this application.

Further improvements:

- **Exponent display**: the display of the polynomials can be improved by switching from the current exponent representation to a more mathematical look
- **Error alerts**: the error messages can be repleced with more specific ones and can be placed in a window pop-out
- **Swap inputs**: another good feature to be implemented would be the ability to swap the input polynomials to get quick results
- **History**: it would be useful to have a history of the previous computed polynomials

7. Bibliography

https://stackoverflow.com/
https://searchsoftwarequality.techtarget.com/definition/use-case
https://en.wikipedia.org/
https://www.geeksforgeeks.org/