POLYNOMIALS CALCULATOR

Horvath Ariana-Cristine

Table of Contents

1. Assignment objective

Main 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. Consider the polynomials of one variable and integer coefficients.

Sub-objectives

- Analyze the problem and identify requirements – how the application should work, what it should do and what are the use cases (described in chapter 2. Analysis)
- Design the polynomial calculator – which architectural pattern should be used, how the packages, classes and methods are divided (described in chapter 3. Design)
- Implement the polynomial calculator – the Java code written for the implementation (described in chapter 4. Implementation)
- Test the polynomial calculator – testing the application with JUnit to check if it works properly (described in chapter 5.Testing)

2. Analysis

2.1 Theoretical background

A polynomial P in an indeterminate X is formally defined as:
$P(X) = a_n * X^n + a_{n-1} * X^{n-1} + \cdots + a_1 * X + a_0$
where:

- $a_1, a_2, \ldots, a_n$ represent the polynomial's coefficients
- n represents the polynomial degree

A monomial is a special type of polynomial with only one term.

Addition of two polynomials:

$P(X) + Q(X) = (a_n + b_n) * X^n + (a_{n-1} + b_{n-1}) * X^{n-1} + \cdots + (a_1 + b_1) * X + (a_0 + b_0)$

Subtraction of two polynomials:

$P(X) - Q(X) = (a_n - b_n) * X^n + (a_{n-1} - b_{n-1}) * X^{n-1} + \cdots + (a_1 - b_1) * X + (a_0 - b_0)$

Multiplication of two polynomials

To multiply two polynomials, multiply each monomial in one polynomial by each monomial in the other polynomial, add the results and simplify if necessary.

Division of two polynomials

To divide two polynomials *P* and *Q*, the following steps should be performed:

Step 1 - Order the monomials of the two polynomials *P* and *Q* in descending order according to their degree.

Step 2 - Divide the polynomial with the highest degree to the other polynomial having a lower degree (let's consider that *P* has the highest degree)

Step 3 – Divide the first monomial of *P* to the first monomial of *Q* and obtain the first term of the quotient

Step 4 – Multiply the quotient with *Q* and subtract the result of the multiplication from *P* obtaining the remainder of the division

Step 5 – Repeat the procedure from step 2 considering the remainder as the new dividend of the division, until the degree of the remainder is lower than *Q*.

Derivative of a polynomial

$$\frac{d}{dx}\left(a_n * X^n + a_{n-1} * X^{n-1} + \cdots + a_1 * X + a_0\right) = n * a_n * X^{n-1} + (n-1) * a_{n-1} * X^{n-2} + \cdots + a_1$$

Integral of a polynomial

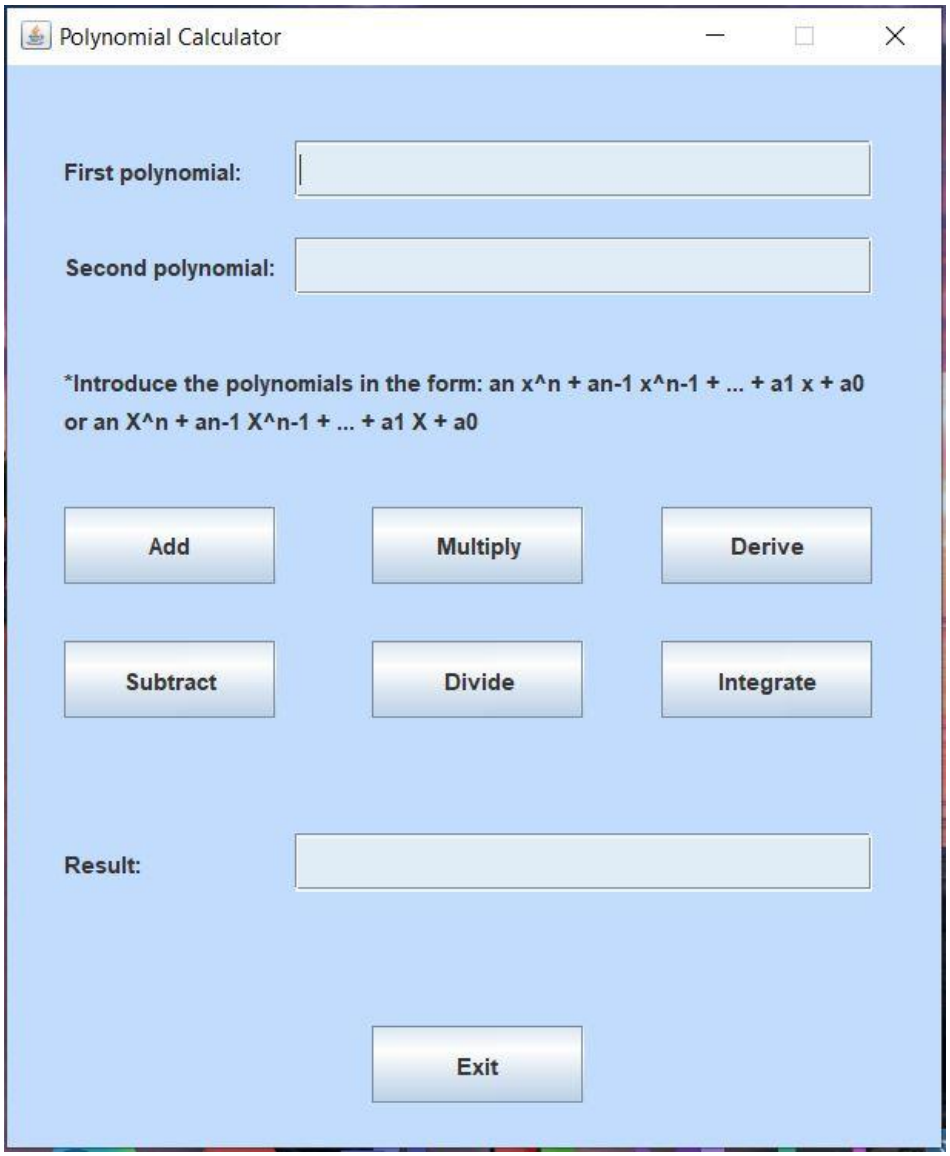
$$\int P(X) = \int a_n * X^n dx + \int a_{n-1} * X^{n-1} dx + \cdots + \int a_1 * X dx + \int a_0 dx$$

where:

$$\int a_n * X^n dx = \frac{a_n * X^{n+1}}{n+1} + C$$

2.2 Use-cases and scenarios

The Graphical User Interface of the Polynomial Calculator looks like the following image:



The user introduces one or two polynomials (depending on the operation) in the text fields in the form:

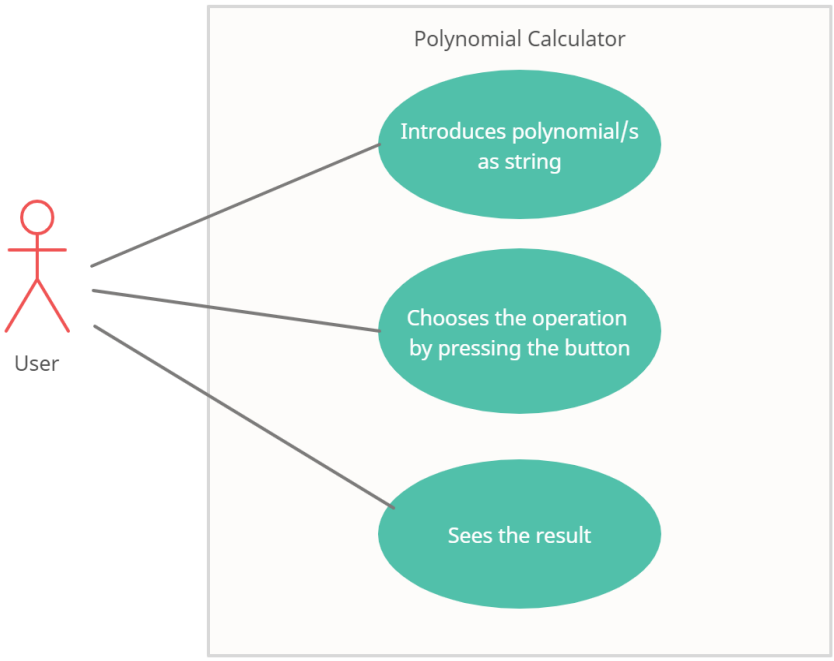
$a_nx\text{^}n + a_{n-1}x\text{^}n\text{-}1 + \ldots + a_1x + a_0$ or

$a_nX\text{^}n + a_{n-1}X\text{^}n\text{-}1 + \ldots + a_1X + a_0$

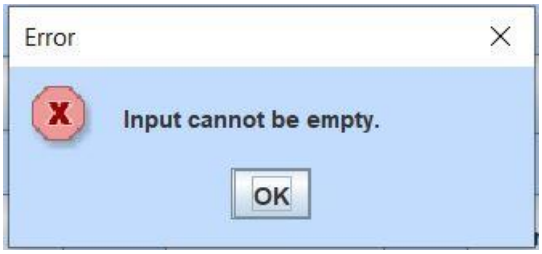
(the input strings can have any space, anywhere, for readability purposes).

Then, he/she chooses the operation to be performed by pressing the corresponding button and sees the result appear in the 'Result' text field.

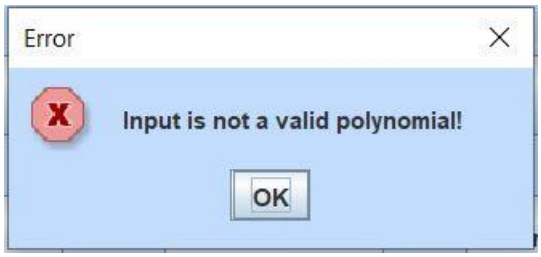
For the addition, subtraction, multiplication and division operations, both text fields have to contain the polynomials and for the derivative and integration operations, only the first text field (First polynomial) is considered.



If the user presses a button without introducing anything, an error message will be displayed:

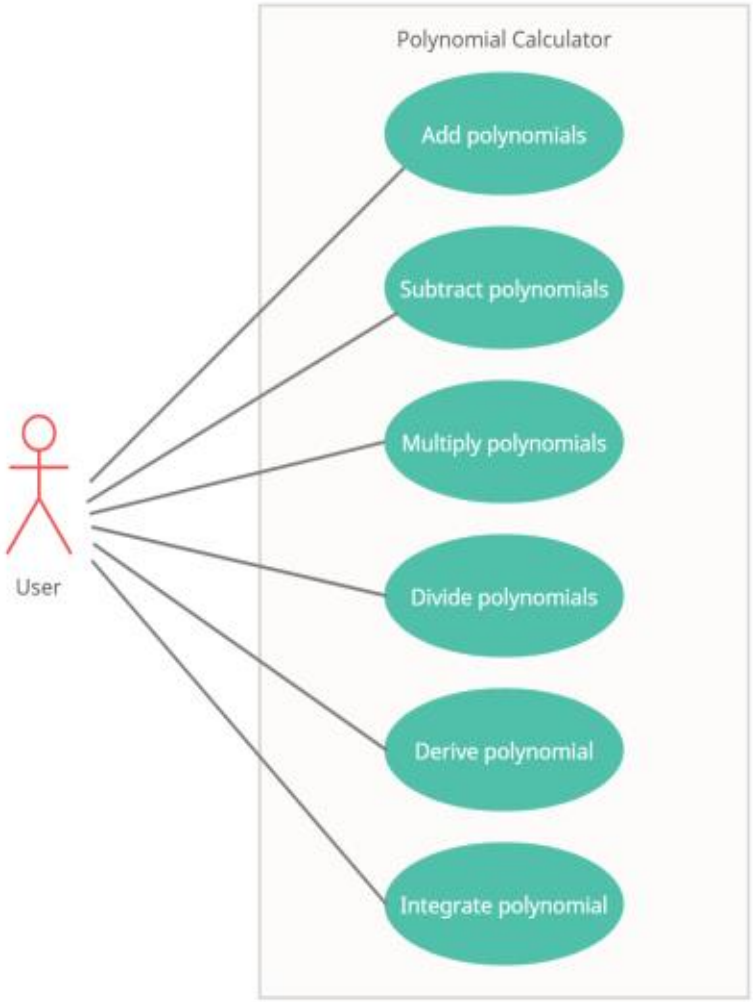


If the user introduces illegal characters (any character other than 'x', 'X', '^', '+', '-', digits and spaces is considered to be illegal) or the polynomial is not in the required form (example: -3x^^4+2x; -3x^4+2x-; -3x^4+2x^-1) an error message will be displayed:



If the 'Exit' button is pressed, the application closes.

Next, the use-cases scenarios for each operation will be presented.



- **Use Case**: Add polynomials

Primary Actor: user

Main Success Scenario:

1. The user inserts the two polynomials as strings in the graphical user interface, in both text fields, in the correct form;

2. The user selects the addition operation, by pressing button “Add”;

3. The polynomial calculator performs the addition of the two polynomials and displays the result in the “Result” text field, as string, in the form of a polynomial.

Alternative Sequence: Incorrect polynomials

- The user inserts incorrect polynomials or it leaves empty one of the text fields or both;

- A pop-up with an error message is displayed;

- The user presses the “Ok” button or “x”(Close);

- The scenario returns to step 1.

- **Use Case**: Subtract polynomials

Primary Actor: user

Main Success Scenario:

1. The user inserts the two polynomials as strings in the graphical user interface, in both text fields, in the correct form;

2. The user selects the subtraction operation, by pressing button “Subtract”;

3. The polynomial calculator performs the subtraction of the two polynomials and displays the result in the “Result” text field, as string, in the form of a polynomial.

Alternative Sequence: Incorrect polynomials

- The user inserts incorrect polynomials or it leaves empty one of the text fields or both;

- A pop-up with an error message is displayed;

- The user presses the “Ok” button or “x”(Close);

-The scenario returns to step 1.

- **Use Case**: Multiply polynomials

Primary Actor: user

Main Success Scenario:

1.The user inserts the two polynomials as strings in the graphical user interface, in both text fields, in the correct form;

-The user selects the multiplication operation, by pressing button “Multiply”;

-The polynomial calculator performs the multiplication of the two polynomials and displays the result in the “Result” text field, as string, in the form of a polynomial.

Alternative Sequence: Incorrect polynomials;

- The user inserts incorrect polynomials or it leaves empty one of the text fields or both;

- A pop-up with an error message is displayed;

- The user presses the “Ok” button or “x”(Close);

-The scenario returns to step 1.

- **Use Case**: Divide polynomials

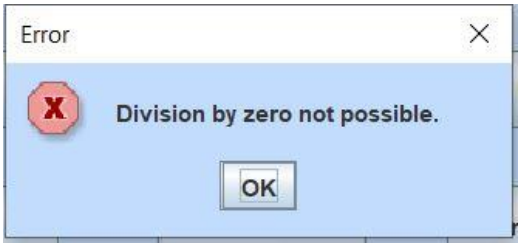
Primary Actor: user

Main Success Scenario:

1.The user inserts the two polynomials as strings in the graphical user interface, in both text fields, in the correct form. The first polynomial is the dividend and the second one is the divisor;

-The user selects the division operation, by pressing button “Divide”;

-The polynomial calculator performs the division of the two polynomials and displays the result in the “Result” text field, as string, in the following form: “Quotient = (polynomial) Remainder = (polynomial)“. If the coefficients are real, they are displayed with 2 decimals.



Alternative Sequence: Incorrect polynomials

- The user inserts incorrect polynomials, it leaves empty one of the text fields or both or introduces “0” in the second text field;

- A pop-up with an error message is displayed;

- The user presses the “Ok” button or “x”(Close);

-The scenario returns to step 1.

- **Use Case**: Derive polynomial

Primary Actor: user

Main Success Scenario:

1.The user inserts one polynomial as string in the graphical user interface, in the first text field, in the correct form;

-The user selects the derivative operation, by pressing button “Derive”;

-The polynomial calculator performs the derivative of the polynomial and displays the result in the “Result” text field, as string, in the form of a polynomial.

Alternative Sequence: Incorrect polynomial;

- The user inserts incorrect polynomial or it leaves empty the first text field;

- A pop-up with an error message is displayed;

- The user presses the “Ok” button or “x”(Close);

-The scenario returns to step 1.

- **Use Case**: Integrate polynomial

Primary Actor: user

Main Success Scenario:

1.The user inserts one polynomial as string in the graphical user interface, in the first text field, in the correct form;

-The user selects the integration operation, by pressing button “Integrate”;

-The polynomial calculator performs the integration of the polynomial and displays the result in the “Result” text field, as string, in the form of a polynomial. If the coefficients are real, they are displayed with 2 decimals.

Alternative Sequence: Incorrect polynomial;

- The user inserts incorrect polynomial or it leaves empty the first text field;

- A pop-up with an error message is displayed;

- The user presses the “Ok” button or “x”(Close);

-The scenario returns to step 1.

2.3 Requirements

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 display the result of the 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 divide two polynomials
- The polynomial calculator should derive one polynomial
- The polynomial calculator should integrate one polynomial
- The polynomial calculator should display the result polynomial/s for the user

Non-Functional requirements

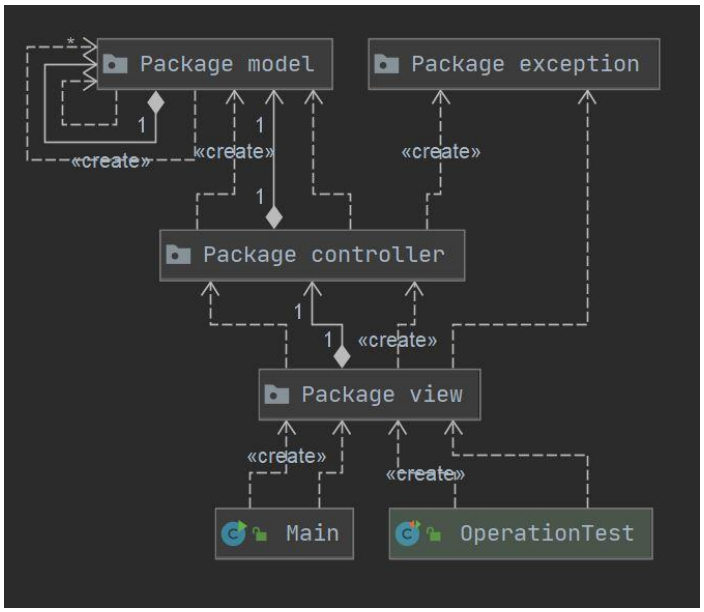
- The polynomial calculator should be intuitive and easy to use by the user
- The polynomial calculator should not allow the user to introduce invalid input
- The polynomial calculator should warn the user if the input is not valid, or empty, or the operation is not possible (ex: division by zero).

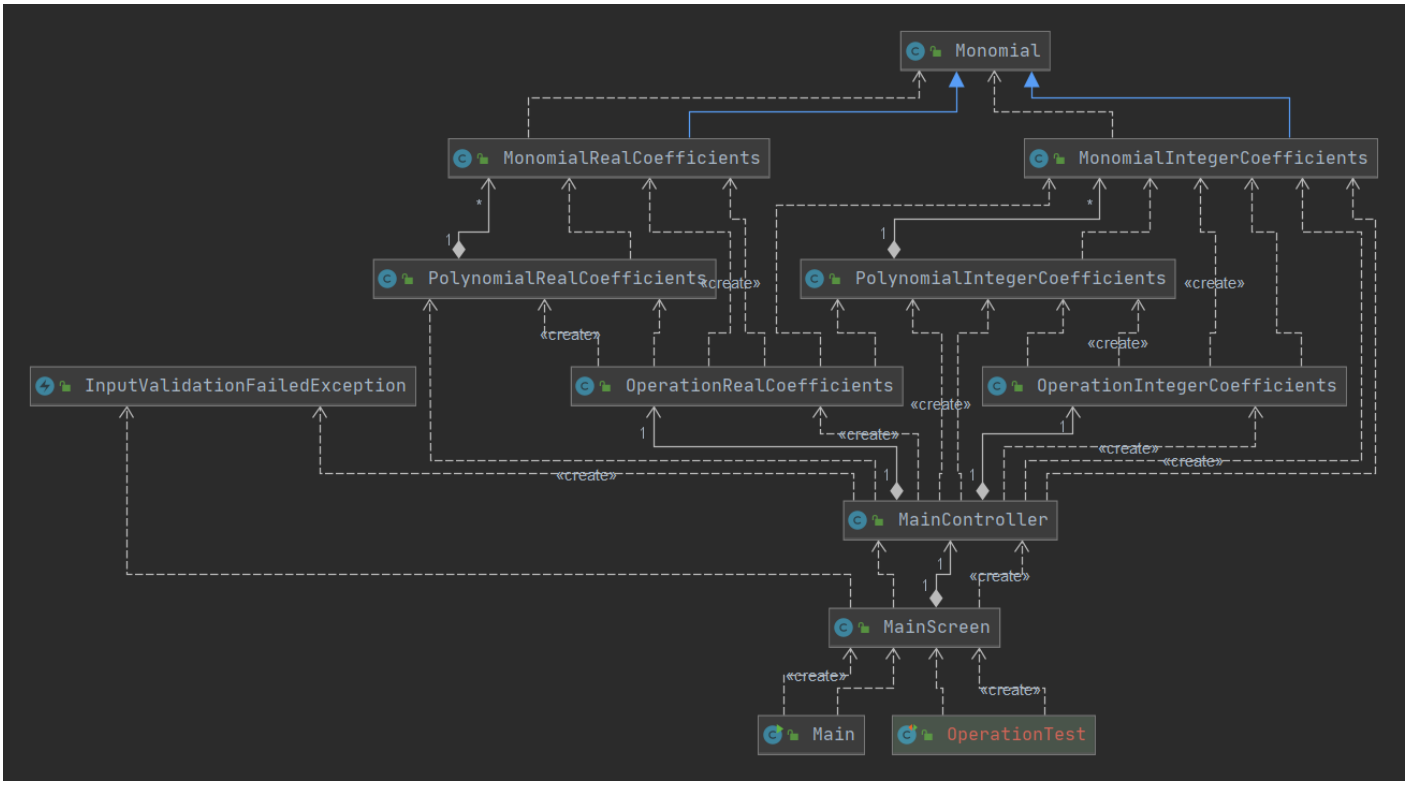
3. Design

In the design of the Polynomial Calculator, the MVC (Model View Controller) architectural pattern was used, which is an interactive systems pattern.

- The Model components encapsulate core data and functionality and are independent of the user interface.
- The View components receive input (which is given to the controller) and display data to the user, through the user interface.
- The Controller components make the relationship between the model and the view components, they validate and parse the input, call the methods from the model and give the output back to view components

3.1 UML Diagrams

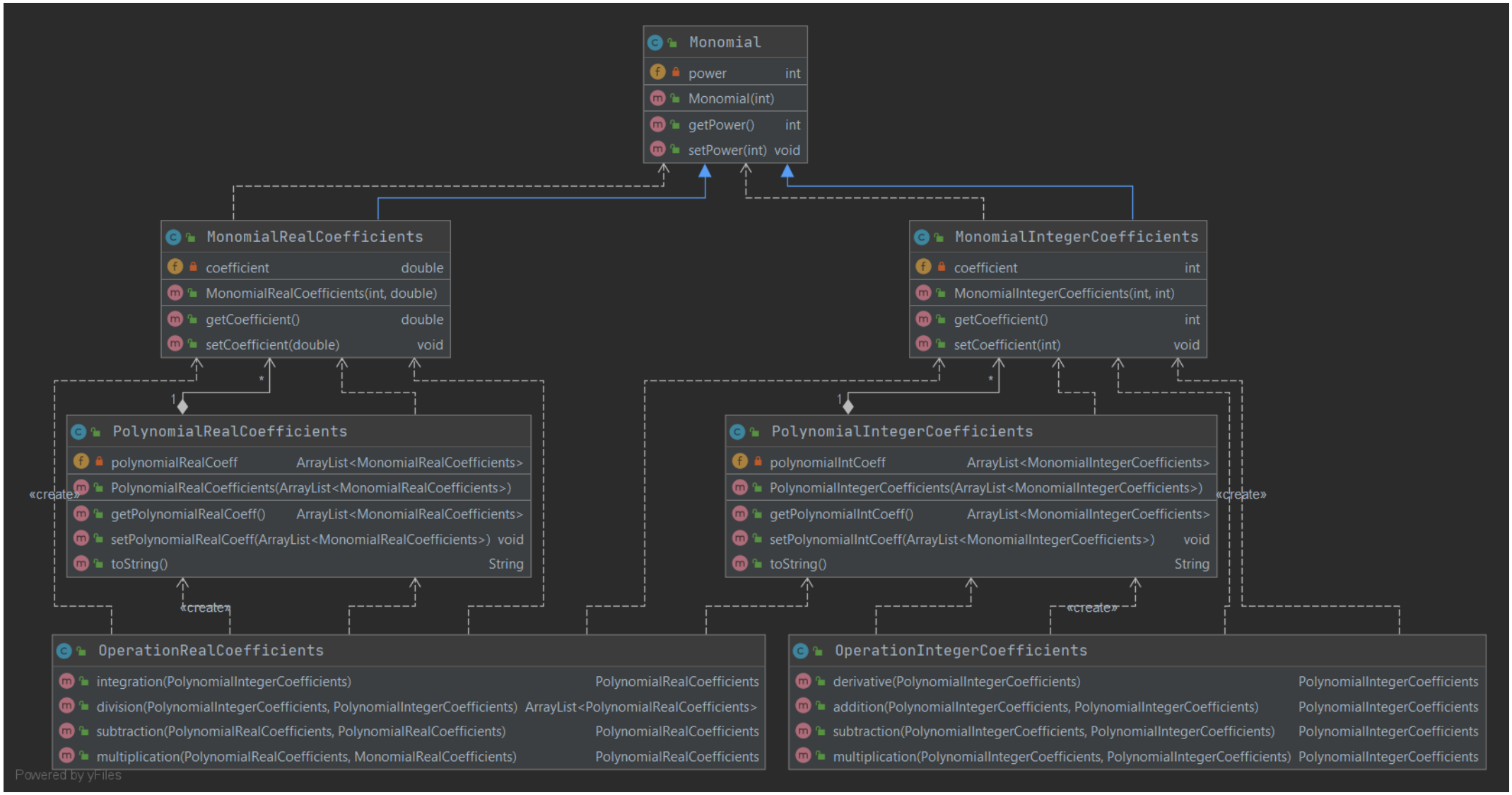




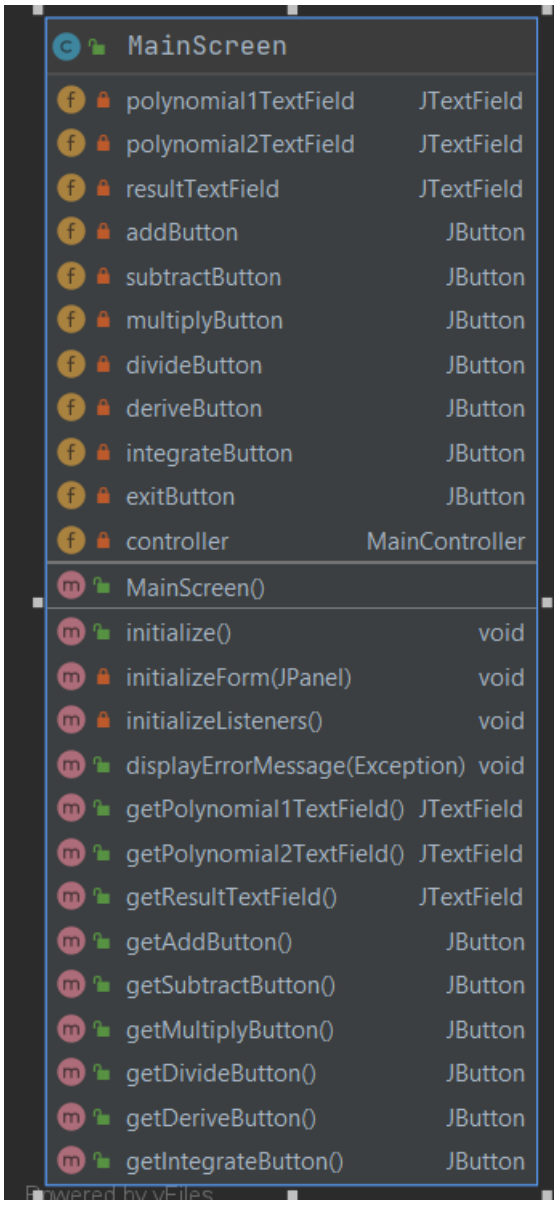
3.2 Packages and classes

-All the classes contain getters, setters and constructors, where needed.

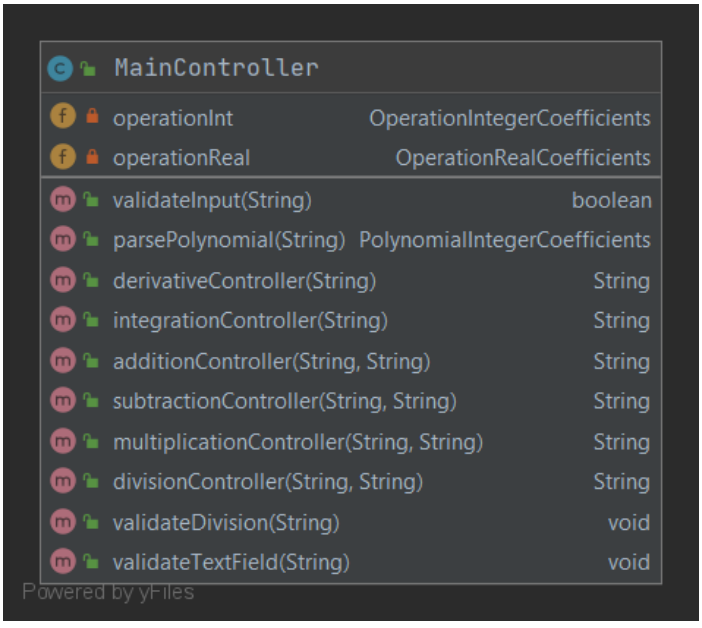
- **Package *model***
  - *Monomial*
  - *MonomialIntegerCoefficients* (extends *Monomial*)
  - *MonomialRealCoefficients* (extends *Monomial*))
  - *PolynomialIntegerCoefficients* – contains a list of *MonomialIntegerCoefficients*
  - *PolynomialRealCoefficients* –contains a list of *MonomialRealCoefficients*
  - *OperationIntegerCoefficients* –contains methods for operations on integer coefficients (addition, subtraction, multiplication, derivative)
  - *OperationRealCoefficients* - contains methods for operations on real coefficients (integration, division – for the division usage there are a subtraction and multiplication method adapted to real coefficients)


Monomial
power int
Monomial(int)
getPower() int
setPower(int) void
MonomialRealCoefficients
coefficient double
MonomialRealCoefficients(int, double)
getCoefficient() double
setCoefficient(double) void
MonomialIntegerCoefficients
coefficient int
MonomialIntegerCoefficients(int, int)
getCoefficient() int
setCoefficient(int) void
PolynomialRealCoefficients
polynomialRealCoeff ArrayList<MonomialRealCoefficients>
PolynomialRealCoefficients(ArrayList<MonomialRealCoefficients>)
getPolynomialRealCoeff() ArrayList<MonomialRealCoefficients>
setPolynomialRealCoeff(ArrayList<MonomialRealCoefficients>) void
toString() String
PolynomialIntegerCoefficients
polynomialIntCoeff ArrayList<MonomialIntegerCoefficients>
PolynomialIntegerCoefficients(ArrayList<MonomialIntegerCoefficients>)
getPolynomialIntCoeff() ArrayList<MonomialIntegerCoefficients>
setPolynomialIntCoeff(ArrayList<MonomialIntegerCoefficients>) void
toString() String
«create»
«create»
«create»
«create»
OperationRealCoefficients
integration(PolynomialIntegerCoefficients) PolynomialRealCoefficients
division(PolynomialIntegerCoefficients, PolynomialIntegerCoefficients) ArrayList<PolynomialRealCoefficients>
subtraction(PolynomialRealCoefficients, PolynomialRealCoefficients) PolynomialRealCoefficients
multiplication(PolynomialRealCoefficients, MonomialRealCoefficients) PolynomialRealCoefficients
OperationIntegerCoefficients
derivative(PolynomialIntegerCoefficients) PolynomialIntegerCoefficients
addition(PolynomialIntegerCoefficients, PolynomialIntegerCoefficients) PolynomialIntegerCoefficients
subtraction(PolynomialIntegerCoefficients, PolynomialIntegerCoefficients) PolynomialIntegerCoefficients
multiplication(PolynomialIntegerCoefficients, PolynomialIntegerCoefficients) PolynomialIntegerCoefficients
Powered by yFiles

- **Package *view***
  - *MainScreen* - the GUI class of the Polynomial Calculator having a single frame
    - the input is introduced as string in the text fields, one button for operation is pressed and the result is displayed in the result text field, or error messages are displayed.



- **Package *controller***
  - *MainController*- controls the relationship between the model classes and the view
    - it validates input strings, parse the input into Polynomial objects, calls the functions for operations and returns the result as string



- **Package *exception***
  - *InputValidationFailedException*
    - extends *RuntimeException*

- ***Main* class** – initializes the GUI

- ***OperationTest* class –** used for testing with JUnit

4. Implementation

- ***PolynomialIntegerCoefficients / PolynomialRealCoefficients***

  - public String **toString**()
    - used to print/display a polynomial as a string
    -if considers a lot of particular cases, apart from the general case: +/-a X^n
    -for the first term, is the coefficient is positive, we don't have to print a '+'
    -if the coefficient is 1/-1, we don't print the '1', only the sign
    -if the power is 1 we print only 'X' instead of 'X^1'
    -if the power is zero we print only the coefficient
    -for the real coefficients we display 2 decimals, if they are integer, we display the number (not with .00)

- ***OperationIntegerCoefficients***

  - public PolynomialIntegerCoefficients **addition**(PolynomialIntegerCoefficients polynomial1, PolynomialIntegerCoefficients polynomial2)
    - adds 2 polynomials and returns a polynomial as result
    - takes the first polynomial and checks each monomial with the monomials a of second polynomials; if they have the same power the coefficients are added (and the result is added in the polynomialSum) and the monomial from second polynomial is removed
    - if we still have monomials in the second polynomials it means that there are terms not added in the sum because they don't have the same power as a monomial from first polynomial, so they are added as they are
    - if there are monomials in the sum with coefficient 0 they are removed

  - public PolynomialIntegerCoefficients **subtraction**(PolynomialIntegerCoefficients polynomial1, PolynomialIntegerCoefficients polynomial2)
    - subtracts 2 polynomials and returns a polynomial as result
    - the behaviour is exactly the same as the addition function, only that the coefficients are subtracted

  - public PolynomialIntegerCoefficients **derivative**(PolynomialIntegerCoefficients polynomial)
    - derives a polynomial and returns the polynomial result
    - takes all the monomials from polynomial and multiply the coefficient with the power, then decrease the power

  - public PolynomialIntegerCoefficients **multiplication**(PolynomialIntegerCoefficients polynomial1, PolynomialIntegerCoefficients polynomial2)
    - multiplies 2 polynomials and returns a polynomial as result
    - each monomial of the first polynomials is multiplied with each monomial of second polynomial, i.e. coefficients are multiplied and powers are added
    - after all were multiplied, there are monomials with the same power, so their coefficients are added together and only one monomial will remain;
    - this is done with a map having the key the power of the monomials and the values are the sum of coefficients

- ***OperationRealCoefficients***

  - public PolynomialRealCoefficients **integration**(PolynomialIntegerCoefficients polynomial)
    - integrates a polynomial and returns the polynomial result with real coefficients
    - the coefficients of all monomials are divided by the (power + 1) and the powers are increased by 1

  - public ArrayList<PolynomialRealCoefficients> **division**(PolynomialIntegerCoefficients polynomial1, PolynomialIntegerCoefficients polynomial2)

- divides polynomial1 by polynomial2 and returns a list of 2 polynomials, the quotient and the remainder
- if the degree of the first polynomial is smaller than the degree of the second polynomial, the quotient is 0 and the remainder is the first polynomial, like the division of a number by a greater number
- divide the first monomial of remainder (the new dividend at each step) to the first monomial of second polynomial (divisor) and obtain the first term of the quotient
- multiply the quotient with second polynomial and subtract the result of the multiplication from first polynomial obtaining the remainder of the division
- repeat the procedure from two steps above considering the remainder as the new dividend of the division, until the degree of the remainder is lower than second polynomial.

- ***MainController***

  - public boolean **validateInput**(String polynomialString)
    - validates the input string to be in the polynomial form, without illegal characters, negative powers, real coefficients etc.
    - eliminates all spaces introduced for readability
    - uses regex Pattern and Matcher to see if polynomial has the desired form

  - public PolynomialIntegerCoefficients **parsePolynomial**(String polynomialString)
    - transforms the input string into the corresponding PolynomialIntegerCoefficients object
    - splits the polynomial into monomials by the sign +/-, with the sign taken into the monomial from the right
    - takes all the monomial strings to convert them into MonomialIntegerCoefficients objects
    - considers the particular cases of coefficient 1, power 1, power 0
    - sorts the monomials by power, to be in the normal form of a polynomial

  - public String **additionController**(String polynomialString1, String polynomialString2) {
    if (validateInput(polynomialString1) && validateInput(polynomialString2)) {
    PolynomialIntegerCoefficients polynomial1 = parsePolynomial(polynomialString1);
    PolynomialIntegerCoefficients polynomial2 = parsePolynomial(polynomialString2);
    PolynomialIntegerCoefficients resultPolynomial = operationInt.addition(polynomial1, polynomial2);
    return resultPolynomial.toString();
    }
    else return null;
    }
    -controls the addition operation, by validating inputs, parsing inputs, calling the method from model and returns the result as a string

  - the other "controller" methods for the other operations are like the additionController method

- ***MainScreen***

  - public void **initialize**()
    - initializes the aspect of the frame: title, dimension, background color, close operation, layout etc.

  - private void **initializeForm**(JPanel panel)
    - adds the elements to the frame - buttons, text fields and labels.

  - private void **initializeListeners**()
    - all the buttons for operations call the validateTextField method from controller, so that if the text field is empty an error message pop-up appears
    - after that it is called the method corresponding to the operation from controller
    - if everything is ok the result is added to the resultTextField
    -also here are caught the exceptions threw by methods validateTextField and operation controllers

-example :

```
addButton.addActionListener(e-> {
    try {
        controller.validateTextField(polynomial1TextField.getText());
        controller.validateTextField(polynomial2TextField.getText());
        String result = controller.additionController(polynomial1TextField.getText(),
                                                      polynomial2TextField.getText());
        resultTextField.setText(result);
    } catch (InputValidationFailedException exception) {
        displayErrorMessage(exception);
    }
});
```

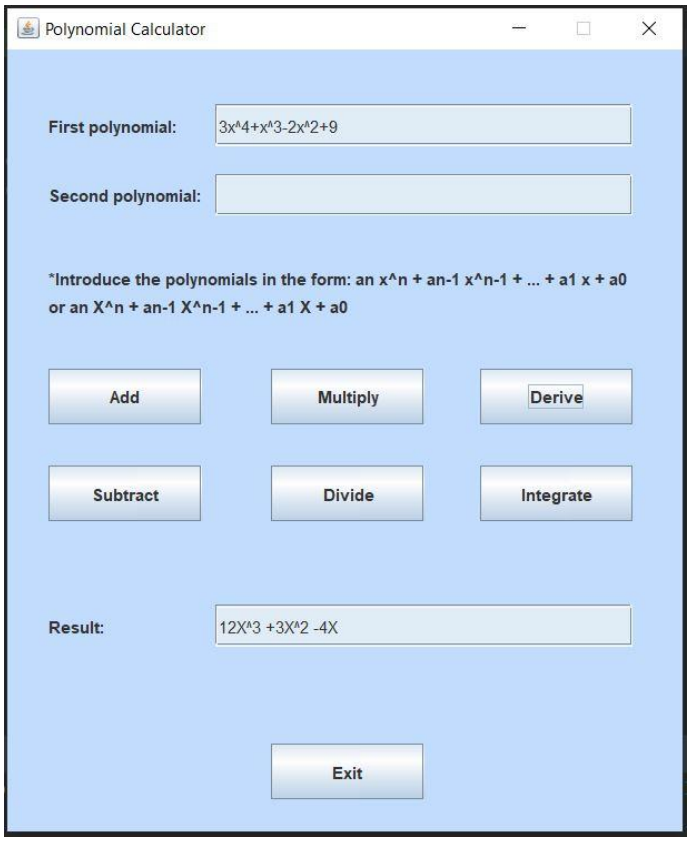
5. Results

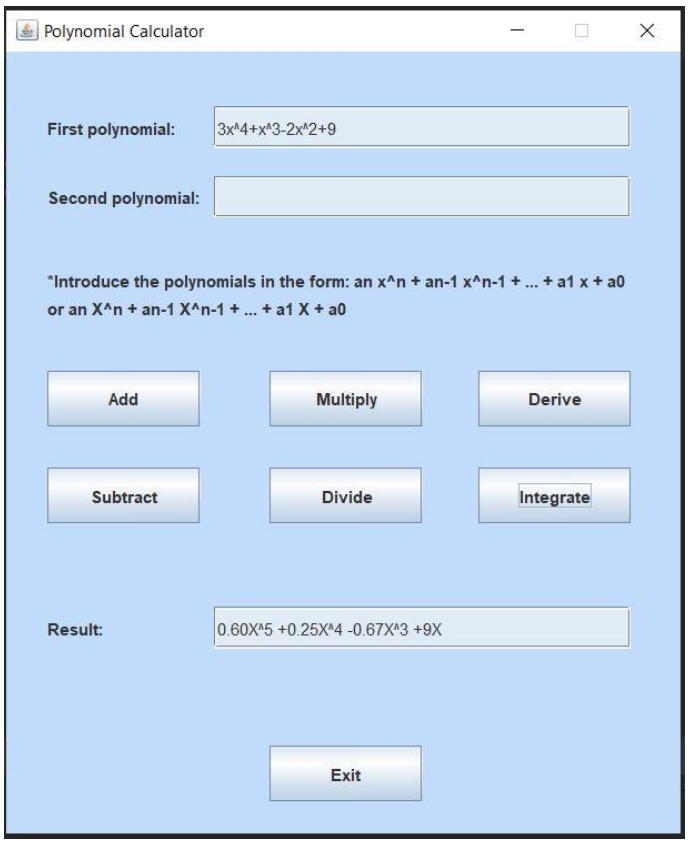
The testing is done using JUnit, in the OperationTest class.

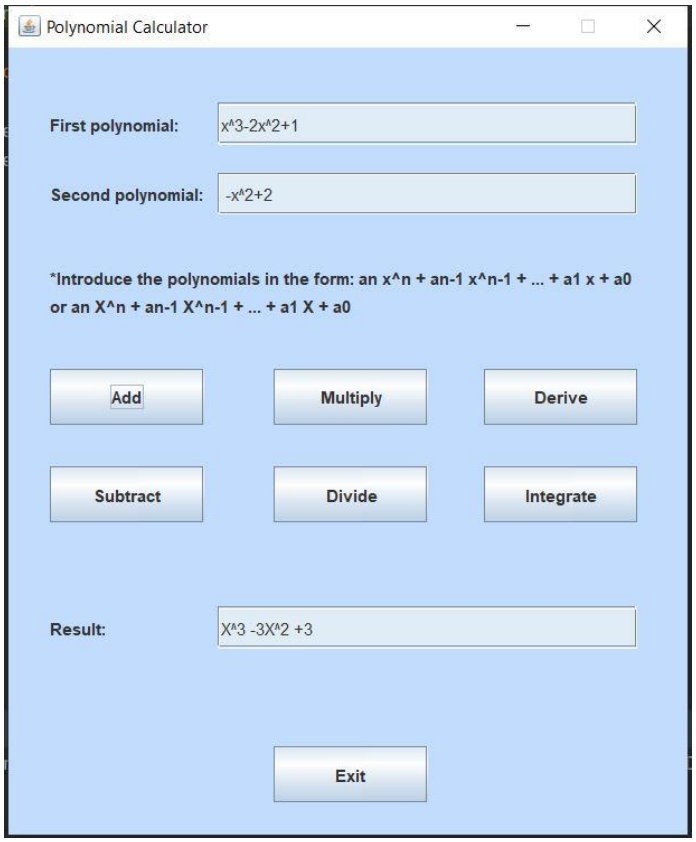
Example:

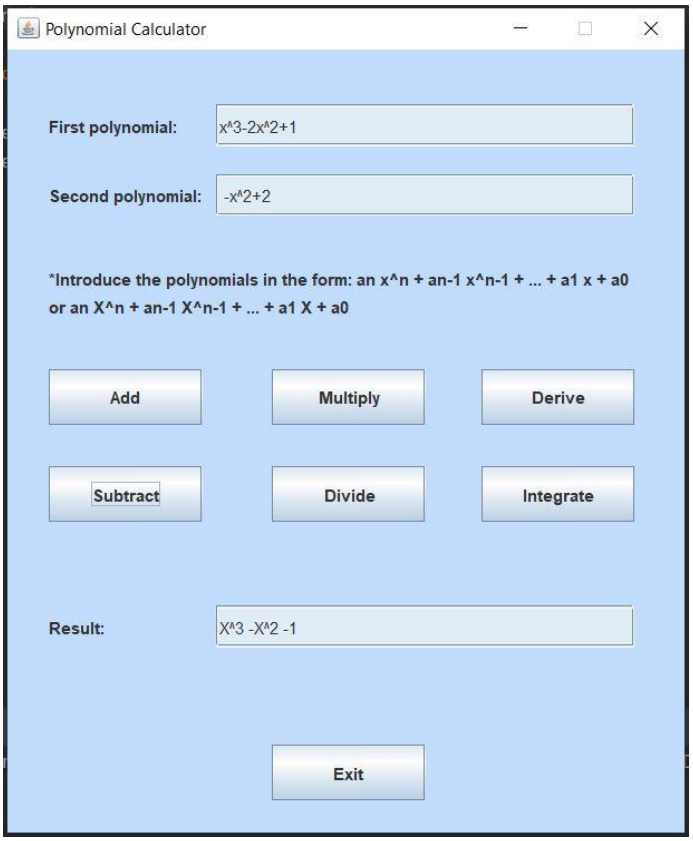
```
@Test
public void additionTest() {
    MainScreen GUI = new MainScreen();
    GUI.getPolynomial1TextField().setText("3x^4+x^3-2x^2+9");
    GUI.getPolynomial2TextField().setText("-x^3+7x^2+4x-1");
    GUI.getAddButton().doClick();
    assertEquals("3X^4 +5X^2 +4X +8",
GUI.getResultTextField().getText());
}
```

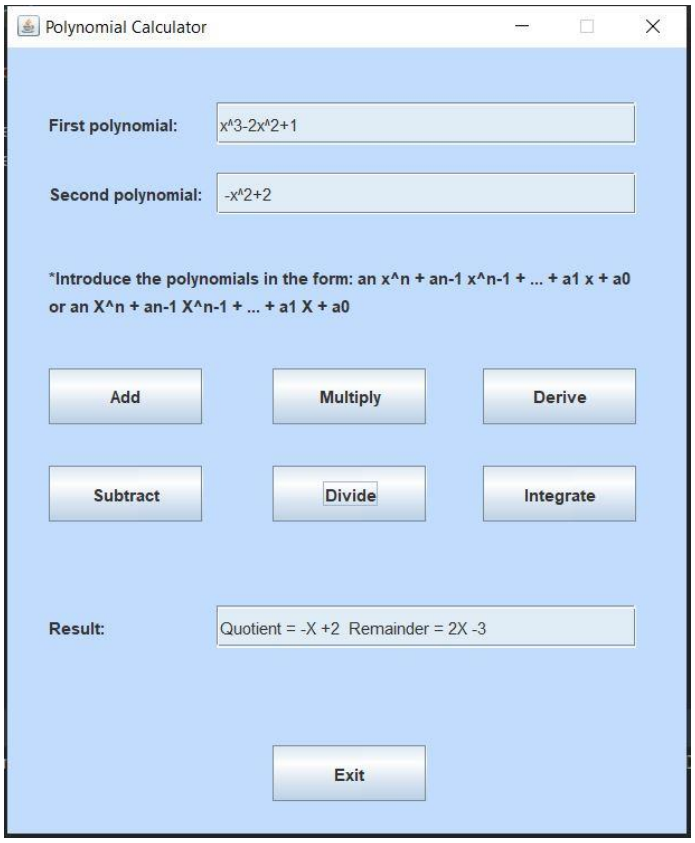
Tested operation	Input Data	Expected Output	Actual Output	Pass / Fail
Derivative	3x^4+x^3-2x^2+9	12X^3 +3X^2 -4X	12X^3 +3X^2 -4X	Pass
Integration	3x^4+x^3-2x^2+9	0.60X^5 +0.25X^4 -0.67X^3 +9X	0.60X^5 +0.25X^4 -0.67X^3 +9X	Pass
Addition	3x^4+x^3-2x^2+9 -x^3+7x^2+4x-1	3X^4 +5X^2 +4X +8	3X^4 +5X^2 +4X +8	Pass
Addition	x^3-7x^2-4x+1 -x^3+7x^2+4x-1	0	0	Pass
Subtraction	3x^4+x^3-2x^2+9 x^3+7x^2+4x-1	3X^4 -9X^2 -4X +10	3X^4 -9X^2 -4X +10	Pass
Multiplication	x^3-2x^2+1 -x^2+2	-X^5 +2X^4 +2X^3 -5X^2 +2	-X^5 +2X^4 +2X^3 -5X^2 +2	Pass
Division	x^3-2x^2+6x-5 x^2-1	Quotient = X -2 Remainder = 7X -7	Quotient = X -2 Remainder = 7X -7	Pass
Division	x^2-1 x^3-2x^2+6x-5	Quotient = 0 Remainder = X^2 -1	Quotient = 0 Remainder = X^2 -1	Pass
Division	x^2-4x+4 x-2	Quotient = X -2 Remainder = 0	Quotient = X -2 Remainder = 0	Pass

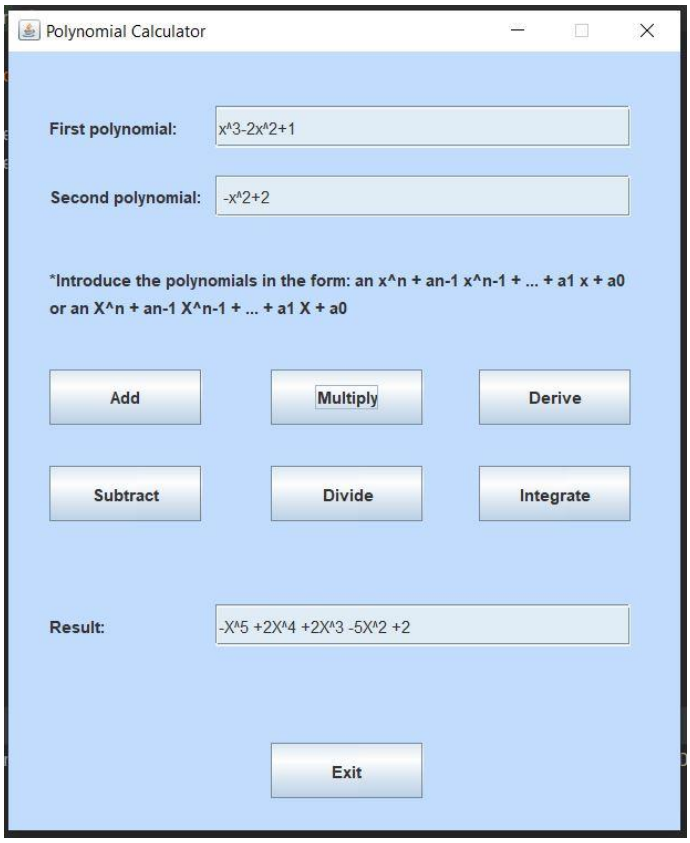

Polynomial Calculator
First polynomial:
3x^4+x^3-2x^2+9
Second polynomial:
*Introduce the polynomials in the form: an x^n + an-1 x^n-1 + ... + a1 x + a0
or an X^n + an-1 X^n-1 + ... + a1 X + a0
Add
Multiply
Derive
Subtract
Divide
Integrate
Result:
12X^3 +3X^2 -4X
Exit


Polynomial Calculator
First polynomial:
3x^4+x^3-2x^2+9
Second polynomial:
*Introduce the polynomials in the form: an x^n + an-1 x^n-1 + ... + a1 x + a0
or an X^n + an-1 X^n-1 + ... + a1 X + a0
Add
Multiply
Derive
Subtract
Divide
Integrate
Result:
0.60X^5 +0.25X^4 -0.67X^3 +9X
Exit


Polynomial Calculator
First polynomial:
x^3-2x^2+1
Second polynomial:
-x^2+2
*Introduce the polynomials in the form: an x^n + an-1 x^n-1 + ... + a1 x + a0
or an X^n + an-1 X^n-1 + ... + a1 X + a0
Add
Multiply
Derive
Subtract
Divide
Integrate
Result:
X^3 -3X^2 +3
Exit


Polynomial Calculator
First polynomial:
x^3-2x^2+1
Second polynomial:
-x^2+2
*Introduce the polynomials in the form: an x^n + an-1 x^n-1 + ... + a1 x + a0
or an X^n + an-1 X^n-1 + ... + a1 X + a0
Add
Multiply
Derive
Subtract
Divide
Integrate
Result:
X^3 -X^2 -1
Exit





6. Conclusions

I consider that the user interface of the Polynomial Calculator is intuitive and easy to use and that the application does everything that is required and it can be useful for operations on polynomials.

The implementation had a lot of particular cases to consider, mostly at parsing and printing the polynomials, also dealing with both integer and real coefficients was a bit difficult and redundant.

What I learned from this assignment was a deepening of the OOP concepts learned the last semester and working with ArrayLists, GUIs, Maven etc. but also learning new things, such as using regular expressions and writing tests with JUnit.

Future improvements of the application could be: computing the value of a polynomial for a given x, considering polynomials on more than one variable, computing the second, third power of a polynomial, solving the equation P(X) = 0 or plotting the graph of the polynomial.

7. Bibliography

- Fundamental Programming Techniques – Lecture Slides, Assignment_1_Support_Presentation
- https://app.creately.com/diagram/3AhVmbmf1bN/edit (for the Use-Case Diagram)
- Microsoft Word - MVCDemo.htm (tcu.edu)
- https://www.jetbrains.com/help/idea/class-diagram.html
- https://www.java2novice.com/junit-examples/assert-equals/#:~:text=for%20writing%20tests.-,Assert.,then%20this%20method%20returns%20equal.