DOCUMENTATION

Polynomial Calculator: ***ASSIGNMENT_1***

STUDENT NAME: Crăciunaș Victor
GROUP: 30425

CONTENTS

1. Assignment Objective

Main objective:

Performing polynomial operations may be hard and time consuming, so as a solution for this problem, we are required to implement and design a polynomial calculator with a graphical user interface that allows users to insert polynomials, select a mathematical operation (ex: addition, subtraction, multiplication, division, derivative, integration) and display the results.

Sub-Objectives:

Sub-Objective	**Description**	**Section where it will be addressed**
• **Problem Analysis**	Analyze the problem to identify the requirements and the use cases for the polynomial calculator, focusing on the necessity of simplifying polynomial operations through a software solution.	Addressed in Section 2: Problem Analysis
• **System design**	Develop a comprehensive design for the polynomial calculator, which includes the overall system design, UML package and class diagrams, used data structures, defined interfaces and used algorithms	Addressed in Section 3: Design
• **Implement the polynomial calculator**	Implement the designed system, ensuring the creation of a user-friendly graphical interface and the correct functioning of all mathematical operations.	Addressed in Section 4: Implementation
• **Test the polynomial calculator**	To ensure the reliability and effectiveness in performing polynomial operations, we need to make multiple tests, which are made with JUNIT	Addressed in Section 5: Results

2. Problem Analysis, Modeling, Scenarios, Use Cases

In the problem analysis phase, I carefully examined the requirements and identified the functional and non-functional ones for the polynomial calculator, aiming to simplify polynomial operations by giving a software solution. This involved understanding the challenges people face when performing polynomial operations manually and recognizing how a dedicated calculator could solve these time consuming problems.

For functional requirements, the following functions describe what my polynomial calculator can do:

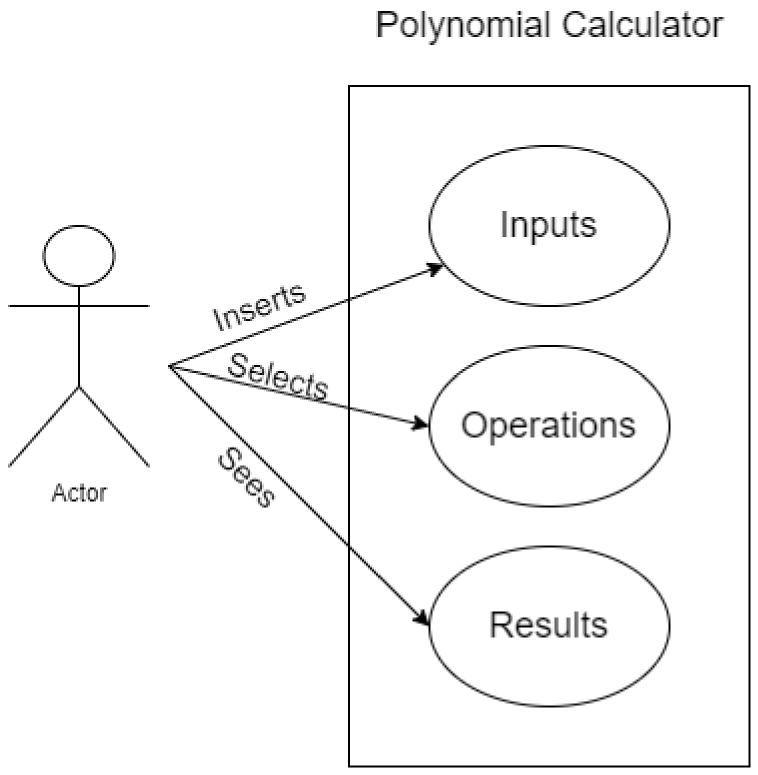
- Input Handling: The calculator accepts the input from users in a specific polynomial format, by handling various degrees of polynomials with integer coefficients;
- Input Validation: The app validate the input polynomials by ensuring they conform to the required format and provide feedback if the input is invalid;
- Operation Selection: Allows the users to select from a set of mathematical operations for polynomials, such as addition, subtraction, multiplication, division, derivation, and integration;
- Calculation Execution: Upon selection of an operation and input of the necessary polynomials, the calculator performs the calculation correctly;
- Result Display: The calculator displays the result of the computation in a readable polynomial format.
- Error Handling: The calculator rigorously handles the errors that my occur during computation and informs the user;

And as for the non-functional functions:

- Usability: The calculator has an intuitive interface that is easy to use, with clear instructions and feedback;
- Performance: Calculations are done fast, ensuring that users do not experience significant delays

- Reliability: The calculator performs accurately, returning the correct results every time without fail.

For the use cases, I developed clear descriptions to illustrate how users would interact with my app:



Use case: Performing polynomial operation

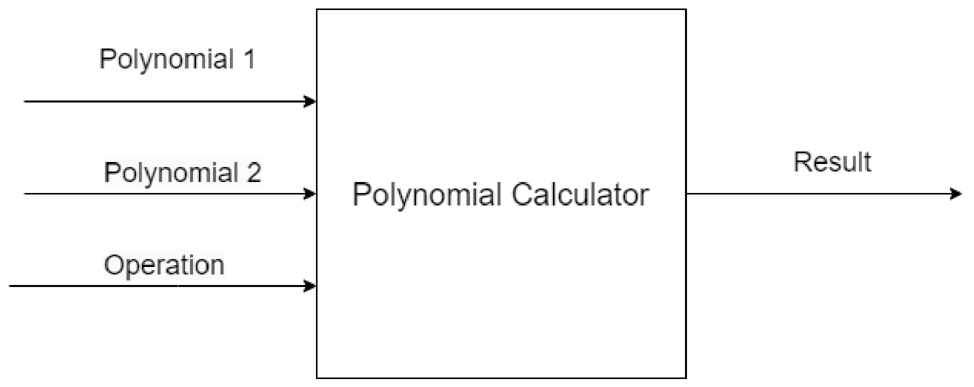
- Primary Actor: User;
- Main Success Scenario:
    1. The user launches the polynomial calculator.
    2. The user inputs two polynomials calculator.
    3. The user selects the operation that he wants the calculator to perform.

4. The system processes the operation and displays the result to the user.

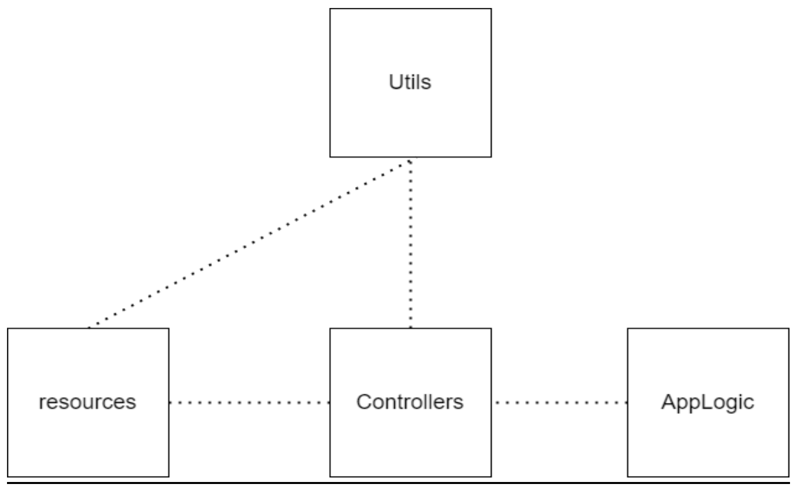
- Alternative Scenarios:
    1. The user inserts incorrect polynomials.
    2. The scenario returns to step 2 and inputs different polynomials.

3. Design

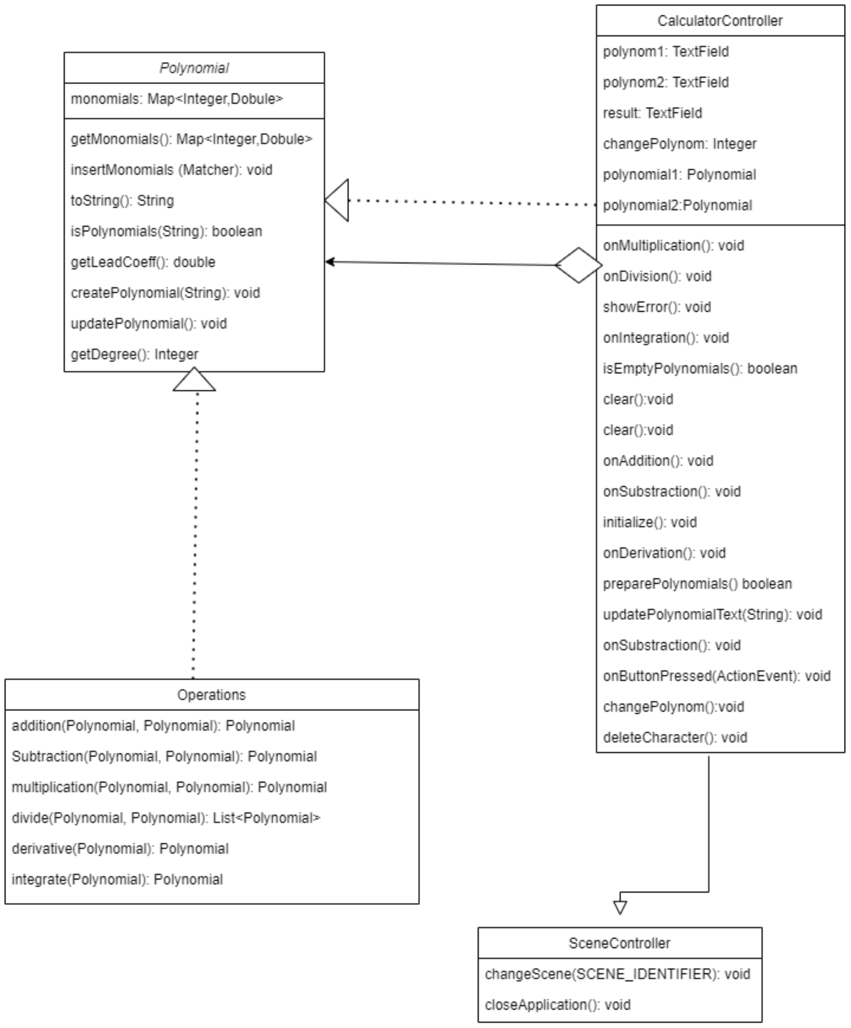
Overall design



Package Diagram



- Utils: The package where the classes that launch the app are;
- Resources: The package where we make the design of the app;
- Controllers: The package where the controllers of the views are(the place where GUI classes are);
- AppLogic: The package where the classes that contains the app's logic are;


Polynomial
monomials: Map<Integer,Dobule>
getMonomials(): Map<Integer,Dobule>
insertMonomials (Matcher): void
toString(): String
isPolynomials(String): boolean
getLeadCoeff(): double
createPolynomial(String): void
updatePolynomial(): void
getDegree(): Integer
CalculatorController
polynom1: TextField
polynom2: TextField
result: TextField
changePolynom: Integer
polynomial1: Polynomial
polynomial2:Polynomial
onMultiplication(): void
onDivision(): void
showError(): void
onIntegration(): void
isEmptyPolynomials(): boolean
clear():void
clear():void
onAddition(): void
onSubstraction(): void
initialize(): void
onDerivation(): void
preparePolynomials() boolean
updatePolynomialText(String): void
onSubstraction(): void
onButtonPressed(ActionEvent): void
changePolynom():void
deleteCharacter(): void
Operations
addition(Polynomial, Polynomial): Polynomial
Subtraction(Polynomial, Polynomial): Polynomial
multiplication(Polynomial, Polynomial): Polynomial
divide(Polynomial, Polynomial): List<Polynomial>
derivative(Polynomial): Polynomial
integrate(Polynomial): Polynomial
SceneController
changeScene(SCENE_IDENTIFIER): void
closeApplication(): void

Data structures used:

- <u>HasMap with Map as interface</u>: I use this data structure in order to save the polynomial. The key will represent the power and the value will represent the coefficient.
- <u>TreeMap</u>: I use TreeMap in order to print the polynomial in the inverse order.
- <u>ArrayList:</u> I use ArrayList in order to save the divide operation result(remainder and the quotient)

4. Implementation

Classes:

- ### Polynomial:

This class will represent the class where we are going to build the polynomial. We have as a field a Map with the form: Map<Integer,double> which is named monomials. In this field we are going to save our polynomial by grouping the monomials together. Each entry is going to be a monomial having as key the power of it and the value as the coefficient.

As methods we have:

-isPolynomial: Where we check if the input from the user is of a polynomial form using regex;

-createPolynomial: In this method we are going to create the polynomial by dividing the polynomial string into groups, each group being a monomial match from the regex;

-toString: Here we are going to print the polynomial by using a TreeMap in order to print it in a correct polynomial form.

- ### Operations:

This class is a static class which has all the operations that the app is going to execute. Each operation is going to be a static method which is called in the CalculatorController.

-Addition: We are going to initialize a new polynomial which we name it result, and will give it the value of the first polynomial. Then we are going to go through the second polynomial and if we see a power from the second polynomial in the result, then we are going to add the coefficients, else we are going to put the monomial with the new power found in the result.

-Subtraction: We are going to initialize a new polynomial which we name it result, and will give it the value of the first polynomial. Then we are going to go through the second polynomial and if we see a power from the second polynomial in the result, then we are going to subtract the coefficients, else we are going to put the monomial with the new power found in the result, but with coefficient negated.

-Multiplication: Initializes a new Polynomial, named result. The method iterates over all monomials of polynomial1 and polynomial2, multiplying their coefficients and adding their powers to generate new monomials. If the resulting power already exists in result, it updates the coefficient by adding the new value; otherwise, it adds the new monomial to result. The result is a Polynomial representing the product of the two input Polynomials.

-Derivative: Produces a new Polynomial that represents the derivative of the input polynomial. It iterates over each monomial, multiplies the coefficient by its power, and then decrements the power by one. If the new power is non-negative, the method records the new monomial in result. The outcome is the derivative of the input Polynomial.

-Integrate: Forms a new Polynomial representing the indefinite integral of the input polynomial. For each monomial, it divides the coefficient by the incremented power and then increments the power. These adjusted monomials are placed into result.

-Divide: Represents Polynomial division, returning a list containing the quotient and the remainder in the form of Polynomials. It repeatedly subtracts the product of the divisor Polynomial and a term from the dividend Polynomial until the remainder's degree is less than that of the divisor. The quotient Polynomial is built up during this process, and both the quotient and the remainder are returned as a list.

- CalculatorController:

  This class represents the GUI class in which we are going to control the interface and retrieve the inputs from the user, made the expected operation and display it.

  Fields:

  -polynom1: is a TextField where the user is going to insert its first polynomial;

  -polynom2: is a TextField where the user is going to insert its second polynomial;

  -result: is a TextField where we are going to display the result;

  -polynomial1: is the first polynomial object that we are going to use for the operations;

-polynomial1: is the second polynomial object that we are going to use for the operations;

Methods:

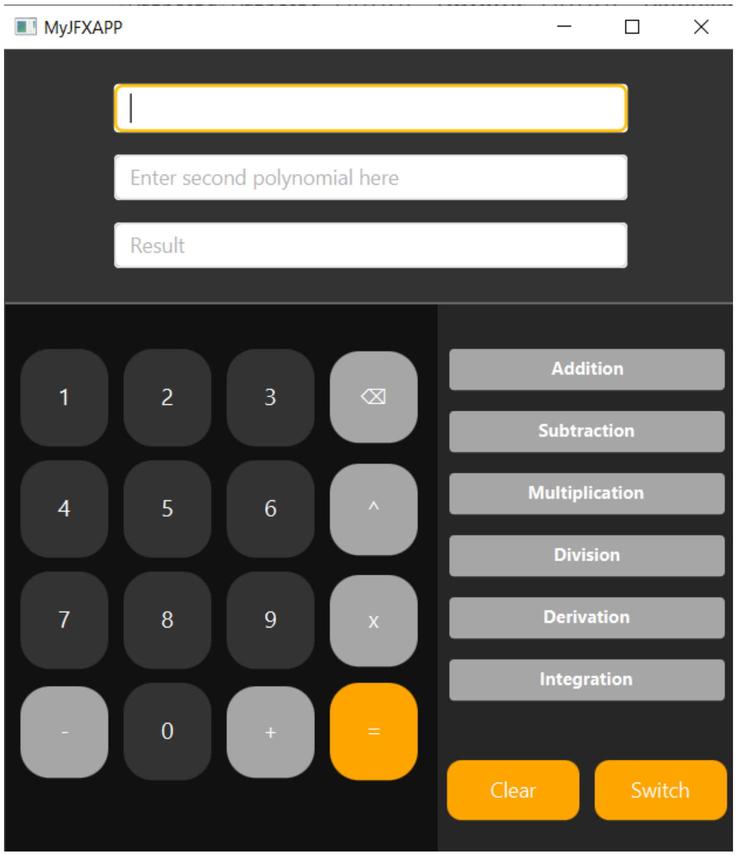
-onButtonPressed, updatePolynomialText: are the methods in which we construct the polynomial by allowing the user to press different buttons (0-9,+,-,x,^);

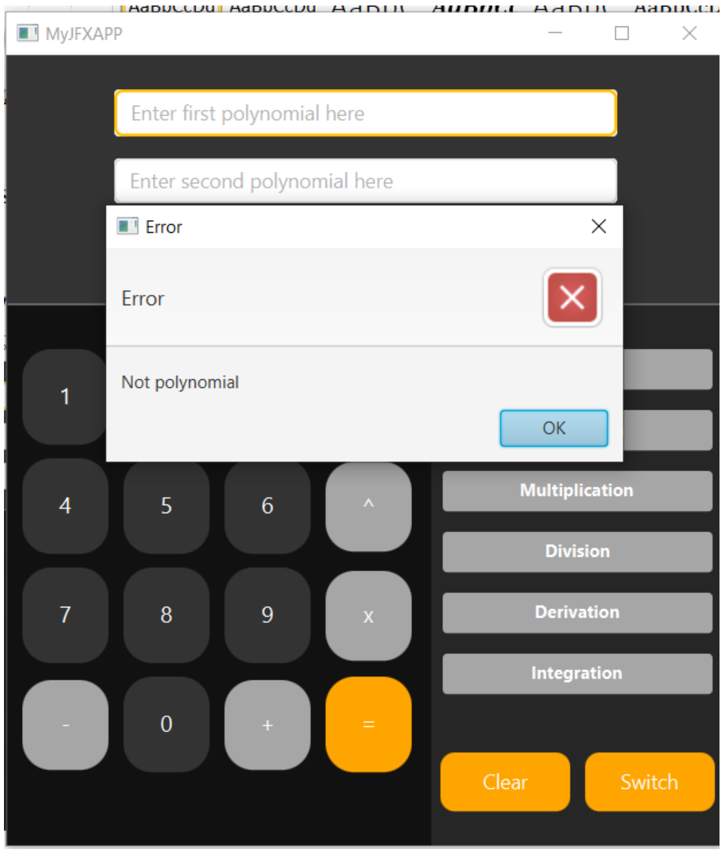
-onAddition, onDivision, onDerivation, onSubtraction, onIntegration, onMultiplication: with these methods, we know what operation to display. The user is going to press the button with what operation he wants the calculator to perform.

-clear: When the user is going to press the clear the button, all the inputs and the results fields are going to be cleared.

-showError: With this method we display errors if the inputs aren't of a polynomial format.

-changePolynom: When the user presses the switch button, we change the textField in which the user is going to insert its polynomial.





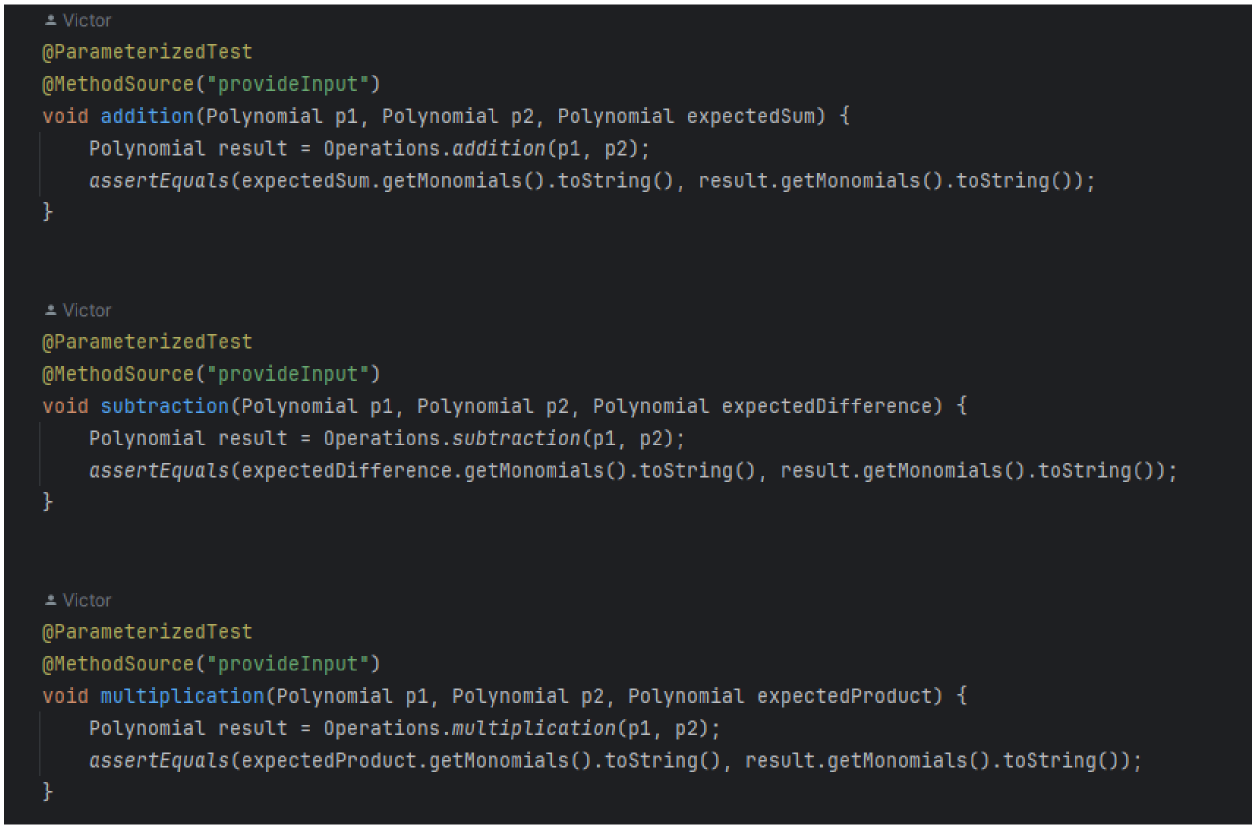
Error

Error

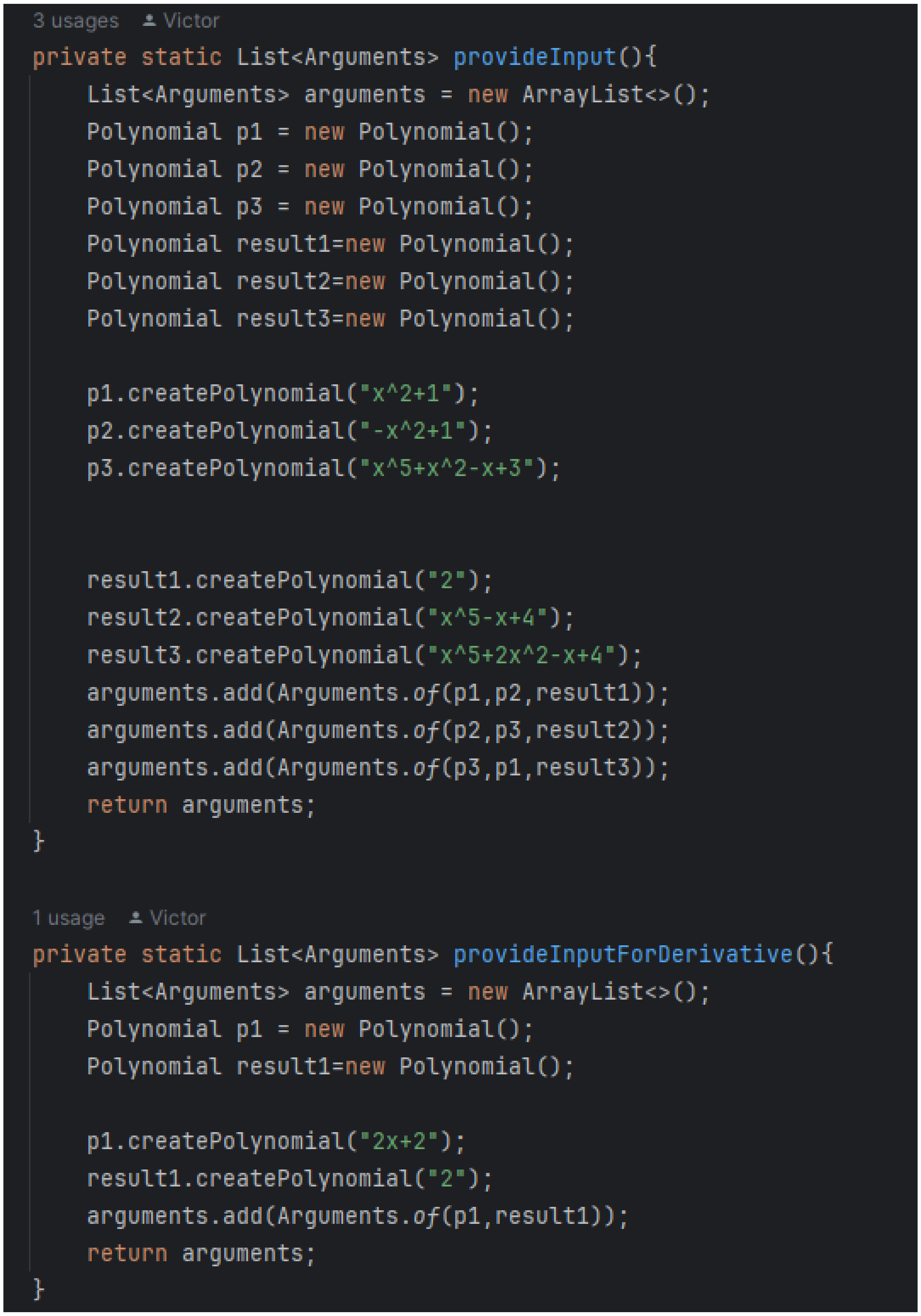
Not polynomial

5. Results

We have OperationsTest class where we use JUNIT in order to test our operation methods.

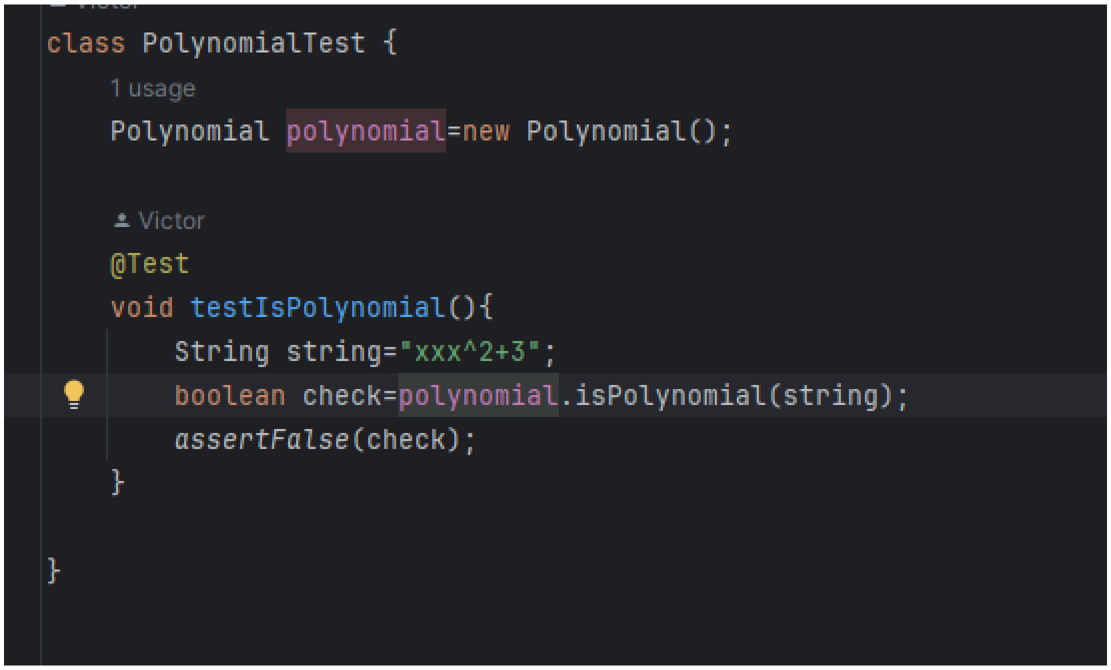


```
Victor
@ParameterizedTest
@MethodSource("provideInput")
void addition(Polynomial p1, Polynomial p2, Polynomial expectedSum) {
    Polynomial result = Operations.addition(p1, p2);
    assertEquals(expectedSum.getMonomials().toString(), result.getMonomials().toString());
}


Victor
@ParameterizedTest
@MethodSource("provideInput")
void subtraction(Polynomial p1, Polynomial p2, Polynomial expectedDifference) {
    Polynomial result = Operations.subtraction(p1, p2);
    assertEquals(expectedDifference.getMonomials().toString(), result.getMonomials().toString());
}


Victor
@ParameterizedTest
@MethodSource("provideInput")
void multiplication(Polynomial p1, Polynomial p2, Polynomial expectedProduct) {
    Polynomial result = Operations.multiplication(p1, p2);
    assertEquals(expectedProduct.getMonomials().toString(), result.getMonomials().toString());
}
```



```
3 usages  Victor
private static List<Arguments> provideInput(){
    List<Arguments> arguments = new ArrayList<>();
    Polynomial p1 = new Polynomial();
    Polynomial p2 = new Polynomial();
    Polynomial p3 = new Polynomial();
    Polynomial result1=new Polynomial();
    Polynomial result2=new Polynomial();
    Polynomial result3=new Polynomial();

    p1.createPolynomial("x^2+1");
    p2.createPolynomial("-x^2+1");
    p3.createPolynomial("x^5+x^2-x+3");


    result1.createPolynomial("2");
    result2.createPolynomial("x^5-x+4");
    result3.createPolynomial("x^5+2x^2-x+4");
    arguments.add(Arguments.of(p1,p2,result1));
    arguments.add(Arguments.of(p2,p3,result2));
    arguments.add(Arguments.of(p3,p1,result3));
    return arguments;
}

1 usage  Victor
private static List<Arguments> provideInputForDerivative(){
    List<Arguments> arguments = new ArrayList<>();
    Polynomial p1 = new Polynomial();
    Polynomial result1=new Polynomial();

    p1.createPolynomial("2x+2");
    result1.createPolynomial("2");
    arguments.add(Arguments.of(p1,result1));
    return arguments;
}
```

PolynomialTest is a class where we test isPolynomial, which checks if the input is Polynomial



```
class PolynomialTest {
    1 usage
    Polynomial polynomial=new Polynomial();

    Victor
    @Test
    void testIsPolynomial(){
        String string="xxx^2+3";
        boolean check=polynomial.isPolynomial(string);
        assertFalse(check);
    }

}
```

6. Conclusions

In conclusion, this polynomial calculator is a very useful tool that is going to make the math calculations a lot faster. I've learned a lot during the creation process, for example: how to make the design better and more intuitive, how to make the code more efficient and more clearly and the most important part that I've learned and the most difficult part of the project was the implementation of the regex.

7. Bibliography

1. https://dsrl.eu/courses/pt/materials/PT_2024_A1_S1.pdf
2. https://dsrl.eu/courses/pt/materials/PT_2024_A1_S2.pdf
3. https://dsrl.eu/courses/pt/materials/PT_2024_A1_S3.pdf
4. https://regex101.com/