DOCUMENTATION

ASSIGNMENT 1

STUDENT NAME: Dincă Diana
GROUP: 30223

CONTENTS

1. Assignment Objective

The main objective of the assignment is to design and implement a polynomial calculator capable of performing various mathematical operations on one or two polynomials, including addition, subtraction, division, multiplication, differentiation and integration.

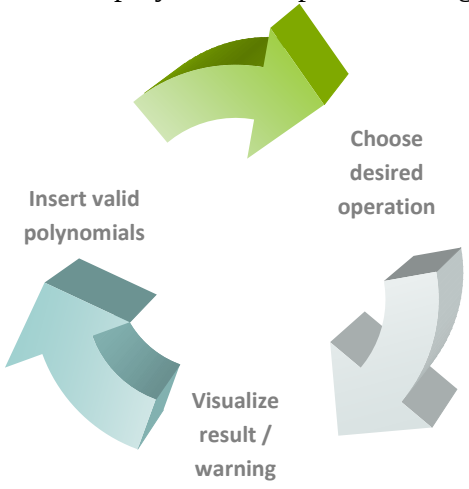
To develop this project, I pursued the following sub-objectives:

- Analyze the problem and identify requirements:
  The calculator allows users to insert one or two polynomials and to select the desired mathematical operation they want to be performed.
  The user expects a correct mathematical answer in return.
  For the result to be valid, the input must be valid. (detailed at point "2. Problem Analysis, Modeling, Scenarios, Use Cases")
- Design the polynomial calculator:
  The calculator design must be intuitive and well thought. (detailed at point "3. Design")
- Implement the polynomial calculator:
  The development of the code must be well-organized into packages and classes. (detailed at point "4. Implementation")
- Test the polynomial calculator:
  For proper functionality, every function of the calculator must be verified with examples. (detailed at point "5. Results")

2. Problem Analysis, Modeling, Scenarios, Use Cases

This project relies on two main classes: Polynomial and Monomial. The Polynomial class represents a polynomial expression, consisting of one or more monomials. Each Monomial object contains a coefficient and an exponent. The Polynomial and Monomial classes work together to enable users to input, manipulate, and perform operations on polynomial expressions efficiently.

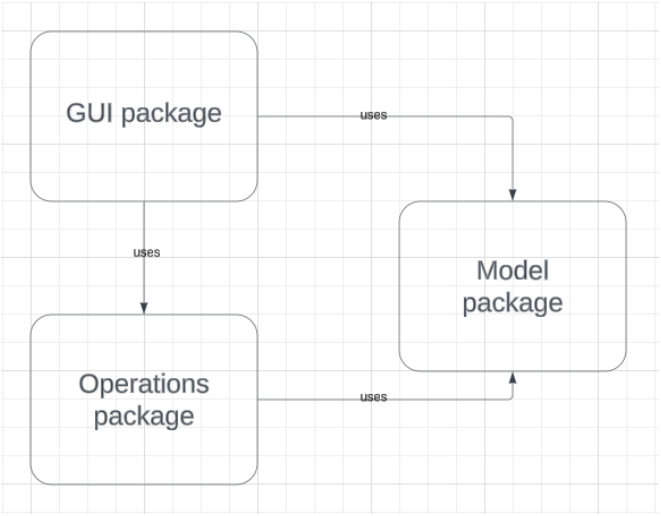
Different situations might occur in our daily lives, where we require quick answers to polynomial operations. A student might need to verify their math assignment, an engineer could be designing a mechanical system that involves polynomial modeling, or a researcher may need to involve polynomial expressions to gain a better understanding of their data.



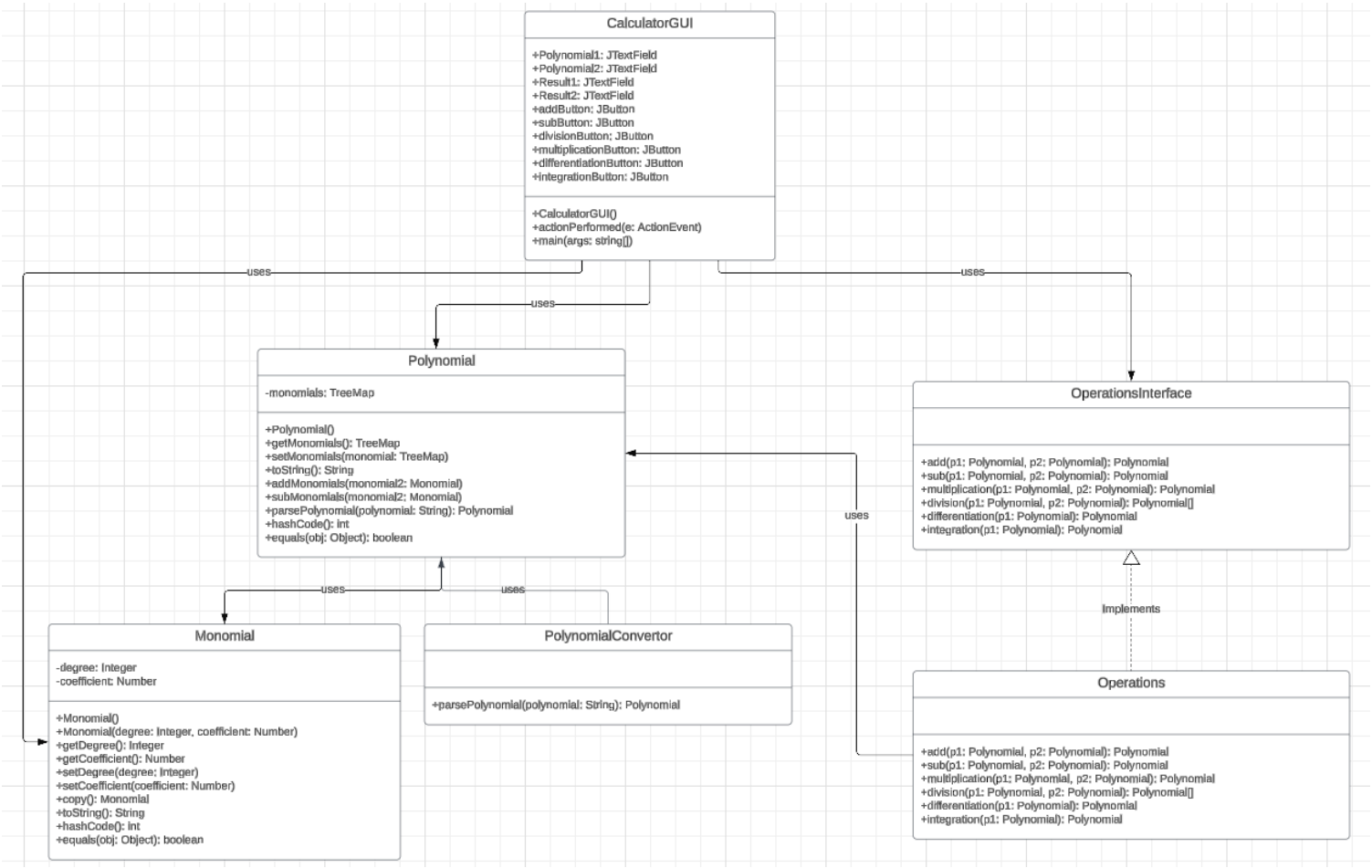
In order for the Polynomial Calculator to work and solve the operation, the user is required to introduce two valid polynomial expressions, for addition, subtraction, division, and multiplication, and only one valid polynomial expression for differentiation and integration, leaving the other field empty. In response, the user will receive a correct polynomial expression for every operation he desires, except in the case of division, where the calculator will return in the result in two fields, the Quotient and the Remainder.

3. Design

Package Diagram:



Classes Diagram:



The application's implementation is organized into four distinct packages:

- *gui package*: this package houses the *CalculatorGui* class, which manages the graphical user interface for the application;
- *model package*: within this package, you'll find the *Monomial* and *Polynomial* classes, essential for representing and manipulating polynomial data;
- *operations package*: this package includes the *OperationInterface*, *Operations* and *PolynomialConvertor* classes, providing the necessary methods for executing polynomial operations and the conversion from a string to a polynomial using pattern matching;
- *testing package*: this package contains the *OperationsTest* and *ParsingTest* classes, designed to conduct JUnit tests to verify the calculator's functionality.

For an improved functionality, I used a *TreeMap<Integer, Monomial>* structure to store the monomials of the polynomial. These monomials, containing only a coefficient and a degree, are sorted in ascending order in the TreeMap, based on their exponents, facilitating printing from the greatest exponent to the smallest.

4. Implementation

Monomial Class- stores the exponent and the coefficient related to the monomial of the polynomial.

Polynomial Class- uses Monomial Class to store a *TreeMap* where the degree is the key and the monomial is the value.

- *toString* method helps printing the polynomial
- *addMonomials* method adds the current monomial with the one sent as a parameter
- *subMonomials* method subtracts the monomial sent as a parameter from the current monomial

PolynomialConvertor Class- contains the *parsePolynomial* method, which converts the string given as a parameter (extracted from the interface) into a polynomial with monomials.

OperationsInterface Class- defines all six of the operations needed for the calculator.

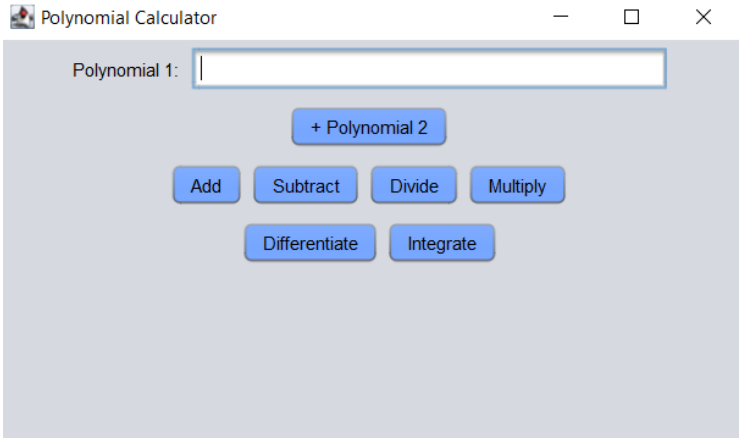
Operations Class- implements OperationsInterface and overrides the methods with the appropriate code to ensure correct outcomes for each operation.

- *add* method is given two polynomials and it calls the *addMonomials* method form the Polynomial Class to return the sum of the polynomials
- *sub* method takes two polynomials as input and invokes the *subMonomials* method from the Polynomial Class to obtain the subtraction of the polynomials
- *multiplication* method takes two polynomials as input and returns their product
- *division* method is given two polynomials and returns their division
- *differentiation* method takes only one polynomial as input and returns its derivative
- *integration* method accepts a single polynomial as input and computes its integral

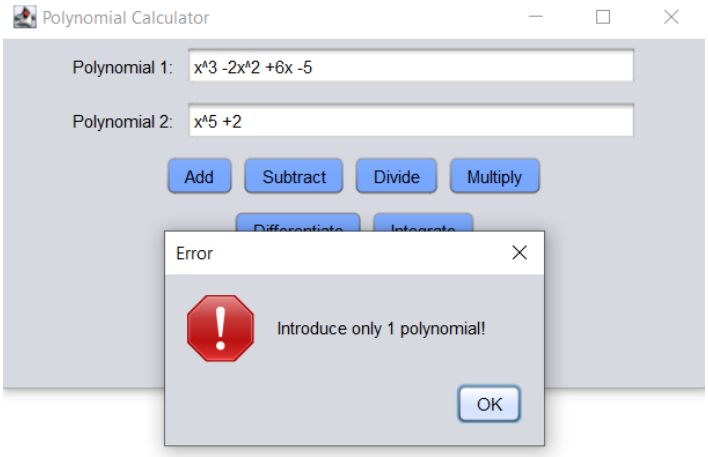
CalculatorGUI Class- represents the graphical user interface that helps the user to interact with the polynomial calculator.

- The interface features six intuitively named buttons, enabling users to obtain the desired operation's result with just a click.

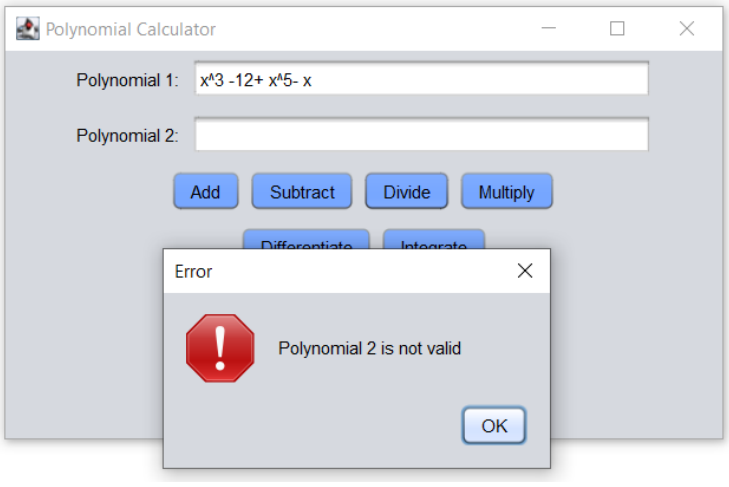
- It contains two text fields, one for each polynomial, but only one is visible in the beginning. The visibility of the second text field can be toggled by clicking the "Polynomial2" button, enabling users to input a second polynomial expression.



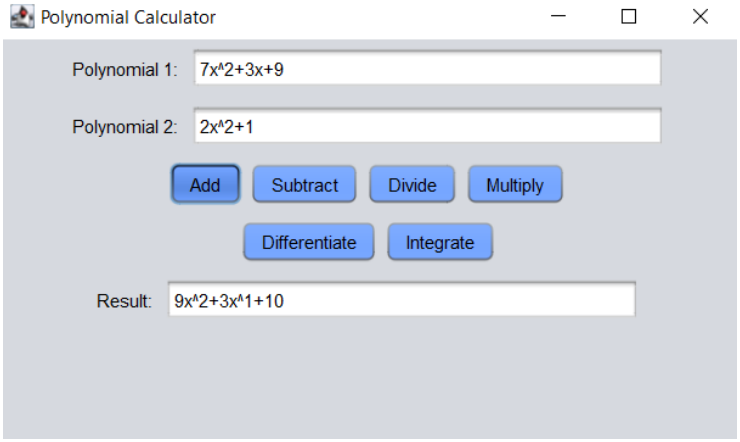
- If the user intends to perform integration or differentiation, they can input the polynomial expression in either the Polynomial1 or Polynomial2 text field. However, attempting to input expressions in both fields will prompt a "Introduce only 1 polynomial!" warning.



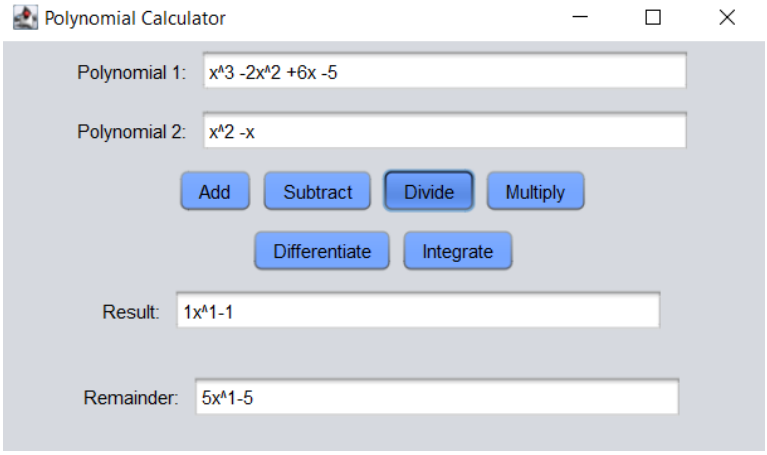
- For the other four operations, both text fields must contain valid polynomial expressions, otherwise, a "Polynomial 1 is not valid" or "Polynomial 2 is not valid" warning will be displayed.



- The interface also includes one output area (showing the result of the addition, subtraction, multiplication, differentiation and integration) visible only after the user clicks a button and the polynomial expressions are validated.



- Only if the division button is pressed, two output areas will be presented: one for the Quotient and one for the Remainder.



5. Results

I have conducted two tests for each operation and two tests for parsing the polynomial. All 14 tests have passed successfully, indicating that the functionality is working as expected.

6. Conclusions

The Polynomial Calculator aims to address the challenges users encounter when dealing with polynomial equations. It provides a user-friendly interface for performing addition, subtraction, multiplication, division, differentiation, and integration operations.

Besides gaining proficiency in polynomial operations, working with the TreeMap structure helped me organize and manage data efficiently. Moreover, the project helped me understand better OOP concepts and application of design patterns, especially in the context of creating graphical user interfaces.

In the future, additional features could be incorporated, such as root finding and equation solving or the possibility of a more flexible input by expanding the options to allow operations on multiple polynomials.

7. Bibliography

Swing: https://docs.oracle.com/javase/tutorial/uiswing/index.html

Junit: https://www.baeldung.com/junit-5