DOCUMENTATION

ASSIGNMENT *ASSIGNMENT\_NUMBER*

STUDENT NAME: Morandau Felix

GROUP: 30425

# CONTENTS

[1. Assignment Objective 3](#_Toc128043139)

[2. Problem Analysis, Modeling, Scenarios, Use Cases 3](#_Toc128043140)

[3. Design 3](#_Toc128043141)

[4. Implementation 3](#_Toc128043142)

[5. Results 3](#_Toc128043143)

[6. Conclusions 3](#_Toc128043144)

[7. Bibliography 3](#_Toc128043145)

# Assignment Objective

The objective of the assignment is to design and implement a polynomial calculator in Java, that handles basic arithmetic operations:

* Addition
* Subtraction
* Multiplication
* Division
* Differentiation
* Integration

The goal is to achieve this polynomial calculator using OOP principles for an efficient and clean design.

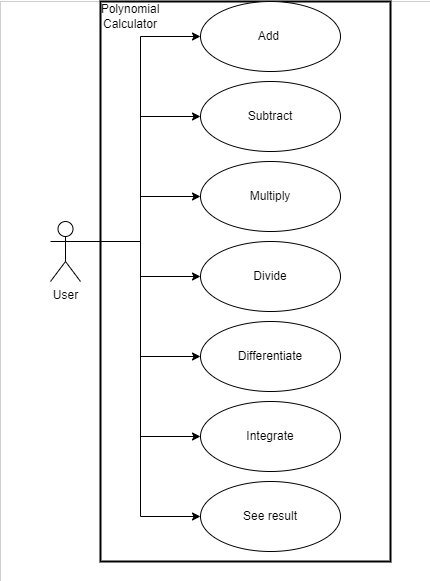
Sub-objectives

* Analyze the problem and identify requirements.
  + Stating the functional requirements.
  + Create the use cases diagram.
  + Create the use cases description.
  + Detail the possible scenarios.
  + These will be presented in section 2.
* Design the polynomial calculator.
  + Create the overall system diagram.
  + Create the package diagram.
  + Create the class diagram.
  + These will be presented in section 3.
* Implement the polynomial calculator.
  + Present each class with its important methods and fields.
  + These will be presented in section 4.
* Test the polynomial calculator.
  + Present the results from the tests made for each class in the controller.
  + Check the coverage.
  + These will be presented in section 5.

# Problem Analysis, Modeling, Scenarios, Use Cases

Functional requirements:

* The app should allow the user to insert two polynomials.
* The app should allow the user to choose the operation to be performed.
* The app should warn the user if the inserted polynomials are in the wrong format.
* The app should add two polynomials.
* The app should subtract the second polynomial from the first.
* The app should multiply two polynomials.
* The app should divide the first polynomial with the second.
* The app should differentiate the first polynomial.
* The app should integrate the first polynomial.



Use cases:

1. Use Case: Addition of Two Polynomials

Precondition: User has launched the polynomial calculator application.

1. Input Polynomial 1: User enters the first polynomial in the designated input field (e.g., "2x^3 + 4x + 5").
2. Input Polynomial 2: User enters the second polynomial in the next input field (e.g., "3x^3 - 2x + 3").
3. Initiate Addition: User clicks on the "Add" button to perform the addition of the two polynomials.
4. Validate Inputs: Application checks the validity of the entered polynomials.
   * 1. If Valid: Proceeds to step 5.
     2. If Invalid: Displays an error message and prompts the user to enter valid polynomials.
5. Perform Addition: The application adds the corresponding coefficients of the polynomials.
6. Display Result: The sum of the polynomials is displayed in the output area (e.g., "5x^3 + 2x + 8").
7. End: The operation concludes, and the user may perform another operation.
8. Use Case: Subtraction of Two Polynomials

Precondition: User has accessed the polynomial calculator application.

1. Input Polynomial 1: User inputs the minuend polynomial (e.g., "5x^2 + 3x - 7").
2. Input Polynomial 2: User inputs the subtrahend polynomial (e.g., "2x^2 - x + 4").
3. Initiate Subtraction: User selects the "Subtract" button to calculate the difference.
4. Validate Inputs: The application verifies the polynomials' formats.
   1. If Valid: Continues to the next step.
   2. If Invalid: An error message appears, asking for correct inputs.
5. Compute Difference: The calculator subtracts the coefficients of like terms.
6. Result Display: Shows the result of the subtraction (e.g., "3x^2 + 4x - 11").
7. End: The user can now either exit or perform a new operation.
8. Use Case: Multiplication of Two Polynomials

Precondition: The polynomial calculator application is ready for use.

1. Input Polynomial 1: The user enters the first polynomial (e.g., "x^2 + 2").
2. Input Polynomial 2: The user inputs the second polynomial (e.g., "3x - 4").
3. Start Multiplication: User clicks the "Multiply" button.
4. Input Validation: The application checks for correct polynomial formatting.
   1. Valid Inputs: Moves to computation.
   2. Invalid Inputs: Displays an error and requests valid polynomials.
5. Execute Multiplication: Performs polynomial multiplication.
6. Output Result: Displays the product (e.g., "3x^3 - 4x^2 + 6x - 8").
7. End: User may choose another operation or close the application.
8. Use Case: Division of Two Polynomials

Precondition: Polynomial calculator is open and operational.

1. Input Dividend Polynomial: User provides the dividend polynomial in the input field (e.g., "x^3 - 6x^2 + 11x - 6").
2. Input Divisor Polynomial: User enters the divisor polynomial (e.g., "x - 2").
3. Initiate Division: User selects the "Divide" button.
4. Validate Polynomials: Checks if both polynomials are valid.
   1. Valid: Proceeds with division.
   2. Invalid: Error displayed, user asked to correct inputs.
5. Perform Division: Calculator computes the quotient and remainder.
6. Display Outcome: Results (quotient and remainder) are shown (e.g., "Quotient: x^2 - 4x + 8, Remainder: -16").
7. End: Users can continue with new operations or exit.
8. Use Case: Differentiation of a Polynomial

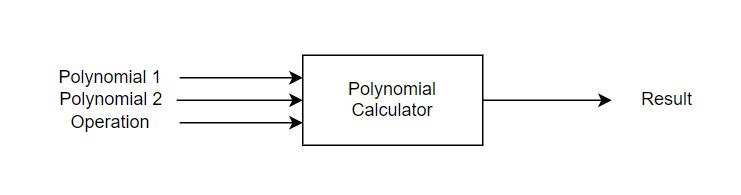
Precondition: The user has initiated the polynomial calculator.

1. Input Polynomial: The user enters a polynomial to differentiate (e.g., "3x^2 + 5x + 2").
2. Start Differentiation: User clicks on the "Differentiate" button.
3. Validate Input: Application checks the polynomial's format.
   1. Valid: Moves forward to differentiation.
   2. Invalid: An error message is shown; the user must re-enter a valid polynomial.
4. Compute Derivative: Calculates the derivative of the polynomial.
5. Result Display: Shows the differentiated polynomial (e.g., "6x + 5").
6. End: The operation concludes, allowing for further actions by the user.
7. Use Case: Integration of a Polynomial

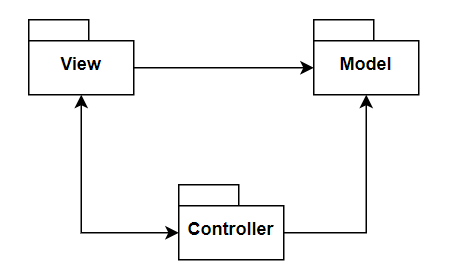
Precondition: Polynomial calculator application is active.

1. Input Polynomial: User inputs a polynomial for integration (e.g., "2x^2 + 6x + 9").
2. Initiate Integration: User presses the "Integrate" button.
3. Input Check: Validates the format of the entered polynomial.
   1. Valid: Continues to integration.
   2. Invalid: Displays an error, prompts for a correct polynomial.
4. Perform Integration: The application integrates the polynomial.
5. Show Result: The integrated polynomial is displayed with a constant of integration (e.g., "2/3x^3 + 3x^2 + 9x + C").
6. End: Users may choose to execute another operation or close the application.

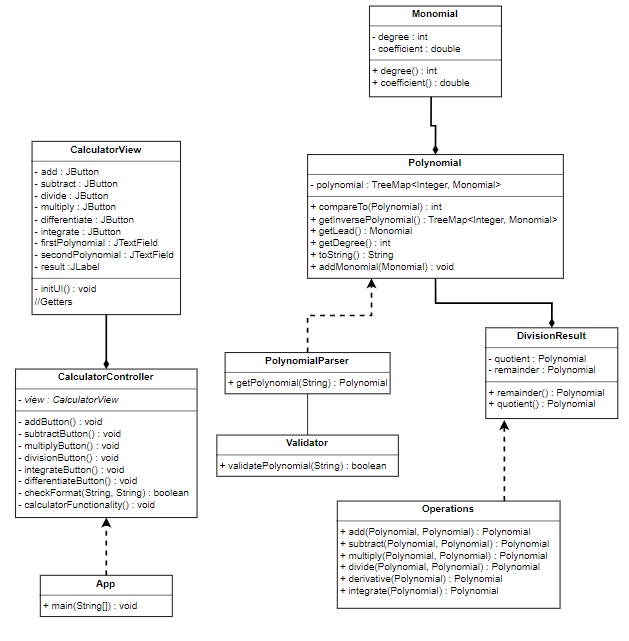
# Design



* *Blackbox of the Polynomial Calculator*



* *Package diagram*

**

* *Class diagram*

# Implementation

**Package: model**

* Monomial class

Represents a monomial with degree and coefficient as fields. It's defined as a record due to its simplicity and immutability.

Fields:

* degree (int)
* coefficient (double)
* Polynomial class

Represents a polynomial using a TreeMap<Integer, Monomial> to map degrees to monomials, facilitating sorting by degree.

Methods:

* addMonomial(int degree, double coefficient): Adds a monomial to the polynomial. If a monomial of the same degree exists, it adjusts the coefficient. Return type: void.
* toString(): Overridden to convert the polynomial into a string representation. Return type: String.
* getDegree(): Retrieves the highest degree of the polynomial. Return type: int.
* getLead(): Gets the leading monomial of the polynomial. Return type: Monomial.
* DivisionResult class

A record that holds the result of a polynomial division, including both quotient and remainder.

Fields:

* quotient (Polynomial)
* remainder (Polynomial)

**Package: controller**

* Operations class

Contains static methods for polynomial operations.

Methods:

* add(Polynomial a, Polynomial b): Addition. Return type: Polynomial.
* subtract(Polynomial a, Polynomial b): Subtraction. Return type: Polynomial.
* multiply(Polynomial a, Polynomial b): Multiplication. Return type: Polynomial.
* divide(Polynomial a, Polynomial b): Division. Return type: DivisionResult.
* differentiate(Polynomial a): Differentiation. Return type: Polynomial.
* integrate(Polynomial a): Integration. Return type: Polynomial.
* Validator class

Provides validation functionality.

Methods:

* isValidPolynomialString(String polynomialString): Checks if the input string is a valid polynomial representation using regex. Return type: boolean.
* PolynomialParser class

Parses a string representation of a polynomial.

Methods:

* parseStringToPolynomial(String polynomialString): Converts a string to a Polynomial object. Return type: Polynomial.
* CalculatorController class

Manages the logic behind the UI, coordinating interactions between the UI and the model. Handles actions triggered by UI elements such as buttons and text fields.

**Package: view**

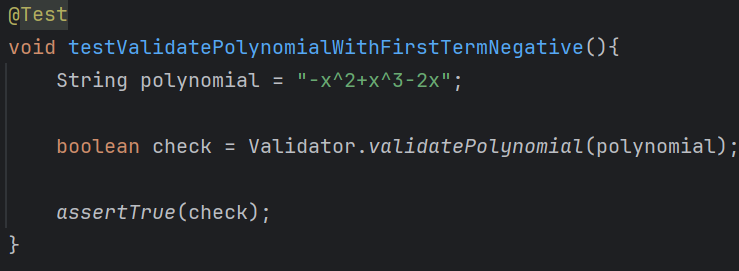
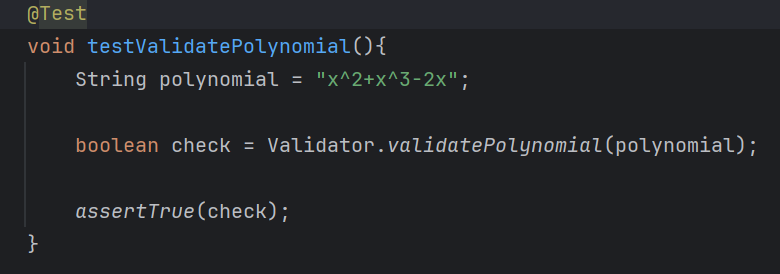
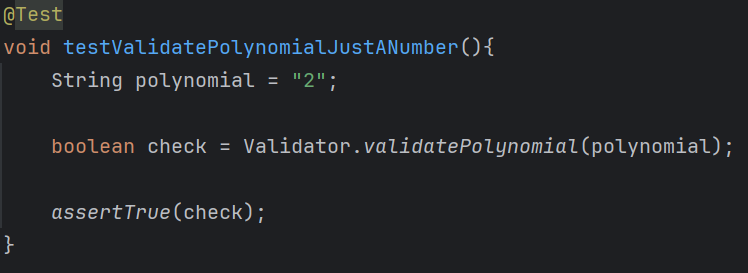
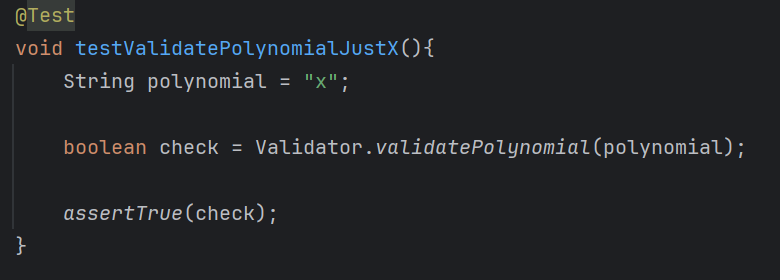
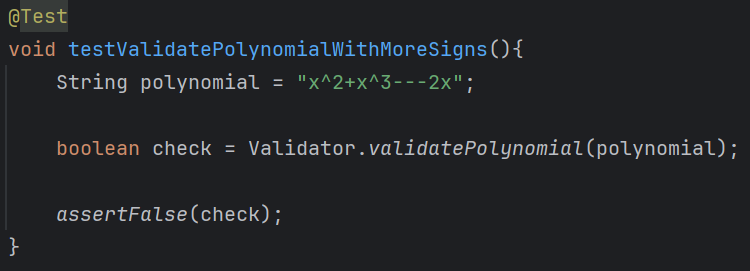
* CalculatorView class
* Constructs the UI for the polynomial calculator.
* The UI is divided into two main panels:
* Top Panel: Contains text fields for inputting polynomials.
* Bottom Panel: Contains buttons for performing operations (add, subtract, multiply, divide, differentiate, integrate).

# Results

The tests were made using Junit 5 and there are tests for three classes from the controller package:

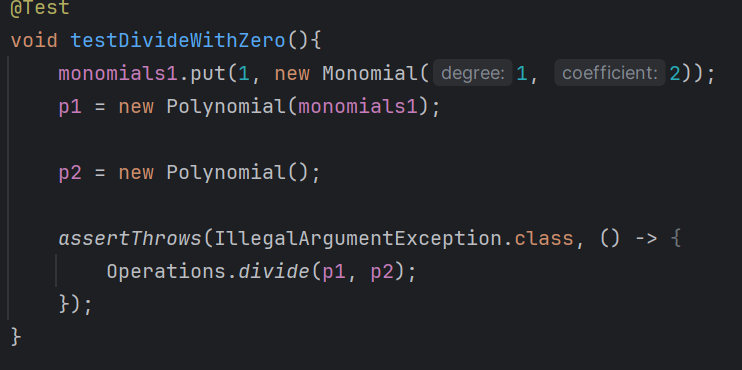
* ValidatorTest class:

This class tests all the possibilities of a polynomial being introduced. There are tests when the input is with spaces, without spaces, with signs, without signs, correct polynomials, and incorrect polynomials. Here are some of the tests implemented:



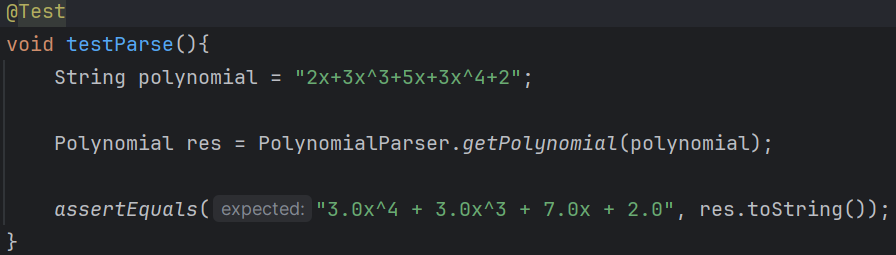
* OperationsTest class:

This class tests if the operations return the expected output. There are tests for all operations. Here are some of them:



* PolynomialParserTest class:

This class tests if the String is parsed correctly into a polynomial.



# Conclusions

In conclusion, the development of a polynomial calculator implemented in Java has been an enriching and challenging experience that significantly contributed to my growth as a programmer. This project not only provided me with a hands-on opportunity to apply theoretical concepts learned in class but also helped in honing my problem-solving skills, deepening my understanding of object-oriented programming, and enhancing my proficiency in Java. Throughout this project, I encountered various challenges, such as designing an efficient algorithm to handle polynomial operations and implementing a user-friendly interface. Overcoming these hurdles required thorough research, persistent effort, and creative thinking, all of which have greatly improved my abilities in coding, debugging, and designing software solutions.

# Bibliography

1. <https://dsrl.eu/courses/pt/materials/PT_2024_A1_S1.pdf>
2. <https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html>
3. <https://www.baeldung.com/junit-assert-exception>