DOCUMENTATION

ASSIGNMENT *1*

STUDENT NAME: Chipirliu Denis
GROUP: 30421

CONTENTS

1. Assignment Objective

This assignment focuses on developing a **JavaFX GUI application** that functions as a comprehensive **polynomial calculator**. Users will be able to:

- **Input polynomial expressions:** The calculator will handle polynomials with one variable and integer coefficients.
- **Select mathematical operations:** Users can choose from various operations like addition, subtraction, multiplication, division, differentiation, and integration.
- **View results:** The calculated outcome of the chosen operation will be displayed within the user-friendly graphical interface.

This project aims to enhance the user experience of performing polynomial calculations by providing a visual and interactive platform.

2. Problem Analysis, Modeling, Scenarios, Use Cases

2.1 Problem Analysis

Manual calculations involving polynomials can be tedious and error-prone, especially for complex expressions. This assignment aims to address this issue by developing a user-friendly tool to simplify polynomial manipulations.

2.2 Modeling

The calculator will be modeled as a JavaFX application with the following components:

User Interface (UI): *This graphical interface will allow users to enter polynomial expressions, select desired operations, and view the results. UI elements will include text fields for input, buttons for operation selection, and labels for displaying results.*
Polynomial Class: *This class will encapsulate the logic for representing and manipulating polynomials. It will utilize a Map data structure to store the monomials efficiently.*
Monomial Class: *This class will encapsulate the logic for representing and manipulating a single term of the polynomial.*

2.3 Scenarios

Here are some scenarios illustrating how users might interacr with the calculator:

Scenario 1: Adding Polynomials

- *User enters two polynomials expressions (e.g., "2x^2 + 3x - 1" and "x^2 - 2x + 5").*
- *User presses the "Add" operation button*
- *The calculator calculates the sum of the polynomials and displays the result (e.g., "3x^2 + x + 4").*

Scenario 2: Differentiating a Polynomial

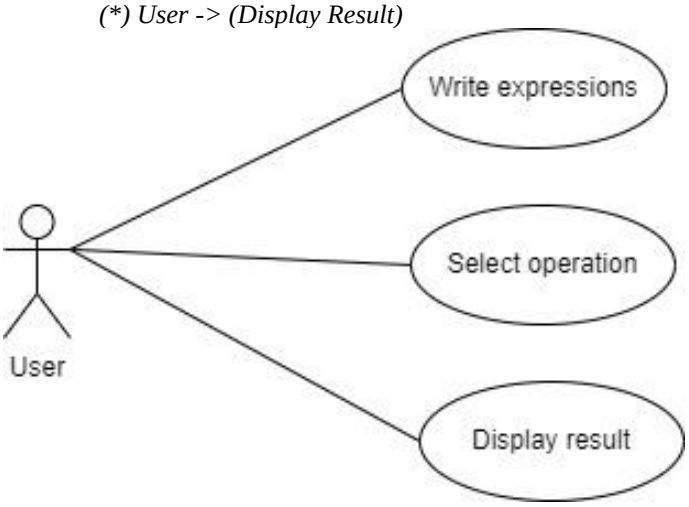
- *User enters a polynomial expression (e.g., "2x^3 - x^2 + 5").*
- *User selects the "Differentiate" operation button.*

- *The calculator computes the derivative of the polynomial and displays the result (e.g., "6x^2 - 2x").*

2.4Use Cases

Use case diagrams:

(*) User -> (Write Polynomial)
(*) User -> (Select Operation)
alt Operation
option Addition
option Subtraction
option Multiplication
option Division
option Differentiation
option Integration
end
(*) User -> (Display Result)



Use case description:

- ***Use Case: Add Polynomials:***

1. *User enters the first polynomial expression.*
2. *User enters the second polynomial expression.*
3. *User selects the "Add" operation button.*
4. *The system calculates the sum of the polynomials.*
5. *The system displays the resulting polynomial.*

- ***Use Case: Subtract Polynomials***

1. *User enters the first polynomial expression.*
2. *User enters the second polynomial expression.*
3. *User selects the "Subtract" operation button.*
4. *The system calculates the difference of the polynomials.*
5. *The system displays the resulting polynomial.*

- ***Use Case: Multiply Polynomials***

1. *User enters the first polynomial expression.*
2. *User enters the second polynomial expression.*
3. *User selects the "Multiply" operation button.*
4. *The system calculates the product of the polynomials using the appropriate multiplication algorithm.*
5. *The system displays the resulting polynomial.*

- ***Use Case: Divide Polynomials***

1. *User enters the first polynomial expression (dividend).*

2. *User enters the second polynomial expression (divisor).*
3. *User selects the "Divide" operation button.*
4. *The system checks for division by zero and handles it appropriately (e.g., displays an error message).*
5. *The system calculates the quotient and remainder of the polynomial division.*
6. *The system displays the quotient polynomial (or an error message if not possible).*

- ***Use Case: Differentiate a Polynomial***

1. *User enters a polynomial expression.*
2. *User selects the "Differentiate" operation button.*
3. *The system calculates the derivative of the polynomial using the power rule.*
4. *The system displays the resulting polynomial representing the derivative.*

- ***Use Case: Integrate a Polynomial***

1. *User enters a polynomial expression.*
2. *User selects the "Integrate" operation button.*
3. *The system calculates the indefinite integral of the polynomial using the reverse power rule.*
4. *The system displays the resulting indefinite integral expression.*

3. Design

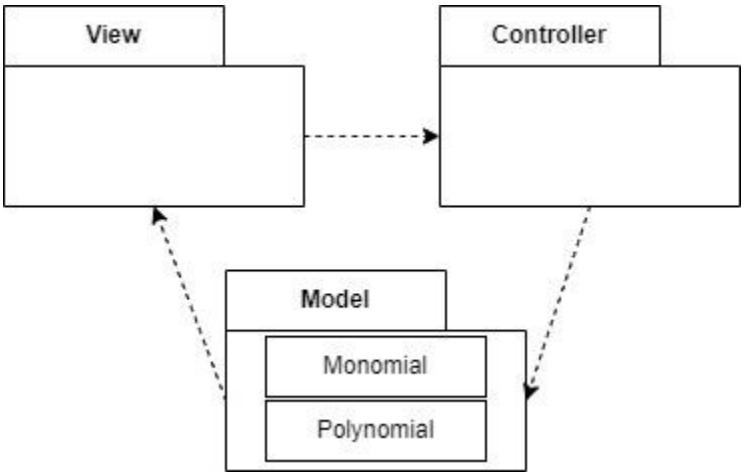
This section outlines the Object-Oriented Programming (OOP) design of the polynomial calculator application, including UML diagrams, data structures, and algorithms.

3.1. OOP Design

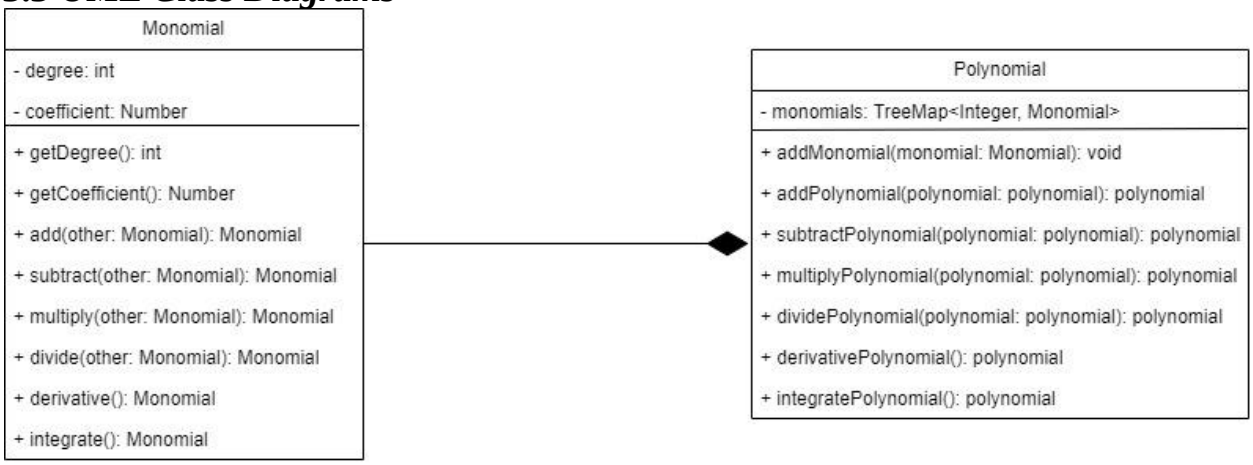
The application will utilize the following classes and their interactions:

***Monomial**: This class represents a single term in a polynomial. It will have attributes for coefficient (integer) and exponent (non-negative integer) and will manage the operations between monomials.*
***Polynomial**: This class will represent a polynomial expression. It will utilize a TreeMap<Integer, Monomial> to store the terms, where the key is the exponent and the value is the corresponding Monomial object. The Polynomial class will provide methods for adding, subtracting, multiplying, dividing (with error handling), differentiating, and integrating polynomials using the implemented Monomial operations for easier code handling and readability.*
***Controller**: This class will act as the controller for the JavaFX application. It will handle user interactions with the UI elements (text fields, buttons) and interact with the Polynomial class to perform the desired operations. It will update the UI elements to display the results.*

3.2 UML Package Diagram



3.3 UML Class Diagrams



4. Implementation

This section dives into the implementation details of the JavaFX polynomial calculator application. We'll explore the key functionalities of each class and the FXML UI design.

4.1 Monomial Class

Fields:

- *degree: Integer (non-negative) representing the exponent of the variable.*
- *coefficient: Number (handles both integers and floats) for the monomial's coefficient.*

Methods:

- *Constructor(int degree, Number coefficient): Initializes a monomial.*
- *Getters for degree and coefficient: Retrieve their values.*
- *Arithmetic operations (add, subtract, multiply, divide): Perform operations on monomials, returning new Monomial instances.*
- *Derivative and integration: Calculate the derivative and indefinite integral of the monomial, respectively.*

4.2 *Polynomial Class*

Fields:

- monomials: A TreeMap<Integer, Monomial> stores monomials sorted in descending order of degree, enabling efficient operations.

Methods:

- Constructors (empty and from string): Create polynomial objects.
- **Methods for various polynomial operations:**
  - addPolynomial, subtractPolynomial, multiplyPolynomial, dividePolynomial: Perform respective operations on polynomials.
  - derivativePolynomial, integratePolynomial: Calculate derivative and indefinite integral.
- **Utility methods:**
  - parsePolynomial(String): Parses a string representation into a polynomial object.
  - toString(): Converts the polynomial back to a string.

4.3 *Controller Class*

Fields:

- @FXML annotated fields representing GUI elements:
  - TextFields for polynomial input and result display
  - Buttons for performing operations

Methods:

- initialize(): - Called when the controller is loaded. - Sets event handlers for buttons to trigger corresponding methods.

- addButton(), subtractButton(), etc.: - Each handles a specific polynomial operation: - Retrieves input polynomials as text from TextFields - Converts text to Polynomial objects using the model classes - Performs the operation using those objects - Displays the result as text in the result TextField - Catches potential exceptions (e.g., invalid input, division by zero) and displays an error message

Key Points:

- The Monomial class handles basic monomial operations.
- The Polynomial class manages collections of monomials for complex polynomial manipulation.
- The controller class (not provided) handles user interactions and connects the model with the GUI.
- Error handling and validation aren't explicitly included in the provided code but are crucial for a robust application.

4.4 *GUI*

5. Results

JUnit is a popular unit testing framework for Java applications. It allows you to write test cases that verify the functionality of individual units of code (in this case, the Polynomial class and its methods).

Test Class Structure:

We'll create a JUnit test class called PolynomialTest that contains various test methods for different operations. Each test method will:

- *Set up the necessary data (e.g., create polynomial objects with specific coefficients and degrees).*
- *Call the method under test (e.g., addPolynomial, subtractPolynomial, etc.).*
- *Use JUnit assertions to verify the expected outcome.*

Testing Addition:

```
@ParameterizedTest
@CsvSource({"2x^2 + 3x - 1, x^3 + 2x - 5, 2x^3 + 2x^2 + 5x - 6",
"3x^2 + 2x - 1, 2x^2 + 3x - 1, 5x^2 + 5x - 2", "2x^2 + 3x - 1, -
2x^2 - 3x + 1, 0"})
void testAddPolynomial(String p1, String p2, String expected) {
    Polynomial polynomial1 = new Polynomial(p1);
    Polynomial polynomial2 = new Polynomial(p2);
    Polynomial result = polynomial1.addPolynomial(polynomial2);
    assert result.toString().equals(expected);
}
```

This test case takes two polynomials from the Csv Source and the expected result for their addition. It then calls the addPolynomial method and uses the assertEquals assertion to verify that the calculated result matches the expected polynomial.

Testing Other Operations:

Similar test cases can be written for subtraction (subtractPolynomial), multiplication (multiplyPolynomial), division (dividePolynomial), derivative(derivaticvePolynomial) and integrate(integratePolynomial). These tests would follow a similar structure, including setting up sample polynomials, defining expected results, and verifying the calculated output using JUnit assertions. Remember to test for edge cases and error conditions within each operation.

The results of the testing: out of 18 tests (3 for each operation) all were completed successfully.

6. Conclusions

This project has involved the development of a JavaFX polynomial calculator application. The application allows users to perform basic polynomial operations (addition, subtraction, multiplication, division) and calculate derivatives and integrals of polynomials. The implementation utilizes JUnit testing to ensure the correctness of the implemented functionalities.

Key Learnings:

- ***JavaFX:*** *The project provided hands-on experience with JavaFX, a framework for building graphical user interfaces. This involved understanding UI components like buttons, text fields, and proper event handling.*
- ***Polynomial Operations:*** *The implementation involved working with polynomial representations, implementing algorithms for various operations, and handling edge cases.*
- ***JUnit Testing:*** *JUnit tests were written to verify the correctness of different polynomial operations. This reinforced the importance of unit testing in software development.*

Future Developments:

- ***Advanced Operations:*** *The calculator could be extended to support more complex operations like finding roots of polynomials or symbolic differentiation/integration.*
- ***Error Handling:*** *More robust error handling can be implemented to handle invalid user input gracefully and display informative messages.*
- ***User Interface Enhancements:*** *The UI could be improved by adding features like graphical visualization of polynomials, user-defined variable support, or saving/loading polynomial expressions.*

Overall, this project provided valuable experience in various aspects of Java programming, including UI development, algorithmic problem-solving, and effective testing practices.

7. Bibliography

- JavaFX Documentation: https://docs.oracle.com/javafx/2/
- JUnit Documentation: https://junit.org/
- Introduction to Polynomials: https://www.khanacademy.org/math/algebra-home/alg-polynomials