**DOCUMENTATION**

**ASSIGNMENT *1***

STUDENT NAME: Hoza Violeta Maria

GROUP: 30424

**CONTENTS**

[1. Assignment Objective 3](#_Toc128043139)

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

[3. Design 6](#_Toc128043141)

[4. Implementation 7](#_Toc128043142)

[5. Results 9](#_Toc128043143)

[6. Conclusions 10](#_Toc128043144)

[7. Bibliography 11](#_Toc128043145)

1. **Assignment Objective**
2. ***Main Objective***: Propose, design, and implement a system for polynomial computation, considering polynomials of a single parameter and integer coefficients.
3. ***Sub-objectives:***

* Analyzing the given problem and finding the requirements for solving it – Section 2
* The design and the implementation of the polynomial calculator – Sections 3 and 4:
* design a graphical user interface (GUI) for users to input polynomials and perform operations visually.
* integrate the polynomial operations logic with the graphical user interface to perform calculations based on user input.
* Testing the calculator: testing scenarios are presented to validate the functionality of the application – Section 4

1. **Problem Analysis, Modeling, Scenarios, Use Cases**
2. ***Functional requirements***

* Allow users to input two polynomials.
* Perform arithmetic operations such as addition, subtraction, multiplication, division, differentiation, and integration.
* Display the result of the operations.
* Be able to show error message without exiting the application for: invalid input format, division by zero.

1. ***Non-functional requirements:***

The polynomial calculator should be intuitive and easy to use by the user. The polynomial calculator should have a well-structured graphical interface and be as visually appealing as possible.

1. ***Use cases:***

The use case presents the actor, which in our case is the user that interacts with the application. He can perform several actions on the two chosen polynomials, such as addition, subtraction, multiplication, division, integration, and differentiation.

A diagram of a system

Description automatically generated

The use cases are strongly connected with the user steps. The user will introduce the two polynomials in the corresponding TextFields, and then he will fetch them so that the application will generate internally those two polynomials. If he wants to perform a certain operation, the user will have just to press the corresponding button, and the result will be written in the result TextField. The user must pay attention to following the format of the polynomial, which appears in the GUI. If the inputs contain invalid characters (the only valid characters are digits, ‘x’, ‘+’, ‘-‘, ‘^’ and spaces), an error message dialog will be displayed. When the user wants to perform a subtraction, the second polynomial will be subtracted from the first polynomial. In the case of differentiation and integration, if the user provides only one input, then that input will be used for the computation, otherwise if both inputs are provided, the first polynomial will be the used one. In the case of division, regardless of the order in which the user provides the inputs, the polynomial of the greatest order will be considered as the dividend. If the user gives one of the inputs 0, an error message dialog will be displayed, not allowing the division by zero.

*A screenshot of a calculator

Description automatically generated* A screenshot of a computer

Description automatically generated

a) Addition of two polynomials:

Main actor: user

Success scenarios: User inputs the two polynomials in the graphical interface and selects the "Add" operation. The polynomial calculator performs the addition and displays the result.

Alternative sequence: incorrectly entered polynomials. User inputs polynomials in another form (e.g., separates the coefficient of x with the "\*" sign). The scenario returns to the step where the user needs to input the polynomials again.

b) Subtraction of two polynomials: same as for addition, but select “Subtract”

c) Multiplication of two polynomials: same as for addition, but select “Multiply”

d) Division of two polynomials: same as for addition, but select “Divide”

e) Derivative of a polynomial:

Main actor: user

Success scenarios: User inputs the one or two polynomials in the graphical interface. The polynomial calculator performs the derivative of the entered polynomials if only one polynomial is given, otherwise it derivates the first polynomial.

Alternative sequence: incorrectly entered polynomials. The scenario returns to the step where the user needs to input the polynomials again.

f) Integration of a polynomial: same as for derivation, but select “Derivate”.

1. **Design**

The Object-Oriented Programming (OOP) design of the polynomial calculator application involves several key aspects, including class design, inheritance, encapsulation, and polymorphism.

For representing a polynomial, I have used the TreeMap data structure, storing key-value pairs in descending order, where the key represents the power of a monomial and the value its coefficient.

Class Diagram:

A diagram of a software application

Description automatically generated

The application is divided into several packages, each containing related classes. These packages include org.example.application, org.example.GUI, org.example.logic, and org.example.models. Each package contains classes responsible for specific functionalities, such as the main application logic (Operations), GUI components (UserInterface), and polynomial representation (Polynomial).

A diagram of a logic

Description automatically generated

1. **Implementation**

* App Class: serves as the entry point of the application. It sets up the main application window and initializes the graphical user interface for the polynomial calculator application
* Polynomial Class: represents a mathematical polynomial and provides methods for creating, manipulating, and converting polynomials.

- Instance Variables: polynomial - This variable is a Map<Integer, Double> representing the polynomial. It maps each power (exponent) of the polynomial to its corresponding coefficient.

- Constructor: Polynomial(): Initializes the polynomial as an empty TreeMap with reverse order for keys. The TreeMap implementation is used with reverse order for keys to ensure that terms are ordered from highest to lowest power.

- Methods: setPolynomial(Map<Integer, Double> polynomial): Sets the polynomial to the specified map.

getPolynomial(): Returns the polynomial map.

maxPower(): Returns the order of the polynomial.

addMonom(int power, double coeff): Adds a monomial (term) to the polynomial with the specified power and coefficient. If a term with the same power already exists, the coefficients are added or subtracted accordingly.

polynomialToString(): Converts the polynomial to a string representation. This method iterates over the polynomial terms and constructs a string representing the polynomial expression.

createPolynomialFromString(String input): Static method that creates a polynomial from a string representation. It parses the input string, identifies each term using a regular expression, extracts the coefficient and power, and adds the term to the polynomial.

* Operations Class:

This class contains static methods for performing various operations on polynomials, such as addition, subtraction, multiplication, division, computing derivatives, and integrals-.

Methods:

add(Polynomial p1, Polynomial p2): Computes the sum of two polynomials by adding corresponding terms. It iterates over the entries of both polynomials, adding each term to the sum polynomial using the addMonom method from the Polynomial class.

subtract(Polynomial p1, Polynomial p2): Computes the difference of two polynomials by subtracting corresponding terms. It iterates over the entries of both polynomials, adding terms from p1 and subtracting terms from p2 from the result polynomial.

multiply(Polynomial p1, Polynomial p2): Computes the product of two polynomials using the distributive property of multiplication. It iterates over the terms of both polynomials, multiplying each pair of terms and adding them to the product polynomial.

divide(Polynomial p1, Polynomial p2): Computes the division of two polynomials using polynomial long division. It iteratively determines the quotient and remainder by dividing the leading terms of the dividend by the leading terms of the divisor until the degree of the remainder is less than the degree of the divisor.

derivative(Polynomial polynomial): Computes the derivative of a polynomial by applying the power rule for differentiation. It iterates over the terms of the polynomial, multiplying each term's coefficient by its power and reducing the power by one.

integrate(Polynomial polynomial): Computes the indefinite integral of a polynomial by applying the power rule for integration. It iterates over the terms of the polynomial, dividing each term's coefficient by the increased power and increasing the power by one.

* UserInterface Class:

This class implements the graphical user interface (GUI) for the polynomial calculator. It contains components like text fields for inputting polynomials, buttons for performing operations, and a text field for displaying results. It also handles user interactions and invokes appropriate methods from the Polynomial and Operations classes to perform calculations.

The class extends JFrame, making it a top-level container for the GUI. It implements the ActionListener interface to handle user actions, such as button clicks. The constructor initializes the GUI by invoking the prepareGui() method, which sets up the window size, default close operation, content pane, and visibility. The prepareGui() method configures the layout, fonts, colors, and borders of the GUI components. The actionPerformed() method handles user actions, such as button clicks. It retrieves input from text fields, validates the input, performs polynomial operations using the Operations class, and updates the result text field accordingly. The method also displays error messages using JOptionPane if input validation fails. The validateInput() method checks whether the polynomial input provided by the user contains valid characters. It returns 1 if the input is valid and 0 otherwise. There are helper methods (customizeButton(), customizeLabel(), customizeTextField()) to customize the appearance (font, color) of buttons, labels, and text fields. The showErrorDialog() method displays an error message dialog using JOptionPane with custom font and color settings.

1. **Results**

The system was tested for various scenarios, including addition, subtraction, multiplication, division, differentiation, and integration. Each test case involved providing input data, executing operations, comparing the expected and actual results, and determining the pass or fail status. JUnit tests were conducted to ensure the correctness of the implemented methods.

The results of the JUnit tests:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| What I tested | Inputs | Actual output | Expected output | Pass / Fail |
| Addition | 2x^5+2x^3+2x^2+2; x^3+3x | 2.0x^5+2.0x^3+2.0x^2+2.0 | 2.0x^5+2.0x^3+2.0x^2+2.0 | Pass |
| Subtraction | 9x^2-3x+2; x^3+3x-7 | -x^3+9.0x^2-6.0x+9.0 | -x^3+9.0x^2-6.0x+9.0 | Pass |
| Multiplication | x+1; x-1 | x^2-1.0 | x^2-1.0 | Pass |
| Division | 3x^4+2x; x | Quotient:3.0x^3+2.0 Remainder: 0 | Quotient:3.0x^3+2.0 Remainder: 0 | Pass |
| Differentiation | 3x^2-2x+2 | x^3-x^2+2.0x | x^3-x^2+2.0x | Pass |
| Integration | 3x^2-2x+2 | 6.0x-2.0 | 6.0x-2.0 | Pass |
| Addition | 2x^3-3x+2; x^3+3x | 2.0x^2+2.0 | 3.0x^3+2.0 | Fail |
| Subtraction | 2x^2-3x; x^3+3x | -x^3+2.0x^2-6.0x+2.0 | -x^3+2.0x^2-6.0x | Fail |
| Multiplication | x+1; x-1 | x^2-2.0x+1.0 | x^2-1.0 | Fail |
| Division | x^3-2x^2+x+8; x-1 | Quotient: x^3 Remainder: 0 | Quotient:x^2-x Remainder: 8.0 | Fail |
| Differentiation | 3x^2+2 | x^3-x^2+2.0x | x^3+2.0x | Fail |
| Integration | 3x^2-2x+2 | 6.0x | 6.0x-2.0 | Fail |

**A computer screen shot of a program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

1. **Conclusions**

This project was a great way to remember what I learned about OOP in my first semester and pick up some new stuff too. At first, it was a bit tough, but I found it helpful. I have learned to design the graphical user interface by code, using Java Swing, which was a little bit harder because last semester I used SceneBuilder and JavaFX.

* ***Future developments***:

While currently focused on fundamental polynomial operations, this application holds potential for expansion through additional functionalities, including:

1. Calculating the square of the polynomial, facilitating multiplication with itself, potentially via a separate button.
2. Determining the value of the polynomial at a specific point.
3. Identifying the roots of the polynomial, particularly applicable when the polynomial degree is less than five.
4. Visualizing the polynomial graphically, enabling users to plot its curve for enhanced analysis and comprehension.
5. **Bibliography**
6. *What are Java classes? -* [*www.tutorialspoint.com*](http://www.tutorialspoint.com)
7. *Object-Oriented Programming - Lecture Slides of prof. Marius Joldoș*
8. *Programming Techniques - Lectures of prof. Cristina Pop*
9. *app.smartdraw.com – for UML package and class diagrams*
10. *app.creately.com – for the use case diagram*
11. [*www.stackoverflow.com*](http://www.stackoverflow.com) *– for the regular expression*
12. *A guide to JUnit5 -* [*www.baeldung.com/junit-5*](http://www.baeldung.com/junit-5)
13. [*https://docs.oracle.com/javase/tutorial/uiswing/index.html*](https://docs.oracle.com/javase/tutorial/uiswing/index.html)
14. *Regular expressions in Java - https://www.geeksforgeeks.org/regular-expressions-in-java/*