**DOCUMENTATION**

**ASSIGNMENT** *1*

STUDENT NAME: Jaden Horst

GROUP: 30422

# **CONTENTS**

[1.](#_3znysh7) Assignment Objective 3

[2.](#_2et92p0) Problem Analysis, Modeling, Scenarios, Use Cases 3

[3.](#_tyjcwt) Design 3

[4.](#_3dy6vkm) Implementation 3

[5.](#_1t3h5sf) Results 3

[6.](#_4d34og8) Conclusions 3

[7.](#_2s8eyo1) Bibliography 3

# Assignment Objective - Polynomial Calculator With A Gui

*Create a User Interface: The most important step in this project was to create a user interface that allows the user to input polynomials in a user-friendly way. The user interface was created using Java Swing library. I left this step for last and opted to use command line to get the logic of my program working.*

*Parse Input: After the user inputs the polynomials, the program needs to parse the input and store it in an appropriate data structure. This will be done using Java string manipulation with classes such as BufferedReader, InputStreamReader, and String’s own replaceAll Function. Instead of using REGEX for the entire parsing I decided to use a Char[] array and make my own parser*

*Implement Polynomial Addition: The program needs to be able to add polynomials. For this I would add the coefficients of the second polynomial to the respective Monomial of the same degree.*

*Implement Polynomial Subtraction: One method I tried was adding the inverted coefficient of the second polynomial to the first. This worked at first but I changed my implementation to manually subtract the coefficients.*

*Implement Polynomial Multiplication: Multiplication would be pretty simple as we can just iterate through all the Monomials. We multiply the coefficients and add the degrees. Finally we just add our new Monomials into a new Polynomial class.*

*Display Results: After performing the polynomial operations, the program needs to display the results in a user-friendly way with our Swing Gui.*

# Problem Analysis, Modeling, Scenarios, Use Cases

*For the Java Polynomial Calculator with a Swing GUI, it is important to identify the functional requirements and create use cases to understand the system's behavior and its interaction with users. In this project, I will create a use case diagram and use case descriptions for the functional requirements of the system.*

*Functional Requirements:*

*User Interface: The calculator should have a user-friendly interface that allows the user to input and display polynomials.*

*Input Polynomials: The user should be able to input polynomials in a standard mathematical format.*

*Display Polynomials: The calculator should display the input and result polynomials in a standard mathematical format.*

*Perform Polynomial Operations: The calculator should be able to perform polynomial operations such as addition, subtraction, multiplication, and evaluation.*

*Error Handling: The calculator should handle errors and provide appropriate error messages to the user.*

*Use Case Descriptions:*

*User Interface:*

*The user opens the calculator.*

*The system displays a user-friendly interface for the user to input and display polynomials.*

*Input Polynomials:*

*The user enters a polynomial in a standard mathematical format.*

*view.getFirstNumberTextField().getText().toCharArray();*

*The system verifies the input and stores it in an appropriate data structure.*

*Display Polynomials:*

*The system displays the input and result polynomials in a standard mathematical format.*

*Perform Polynomial Operations:*

*The user selects an operation to perform on the polynomials (addition, subtraction, multiplication, evaluation).*

*The system performs the selected operation on the polynomials and displays the result.*

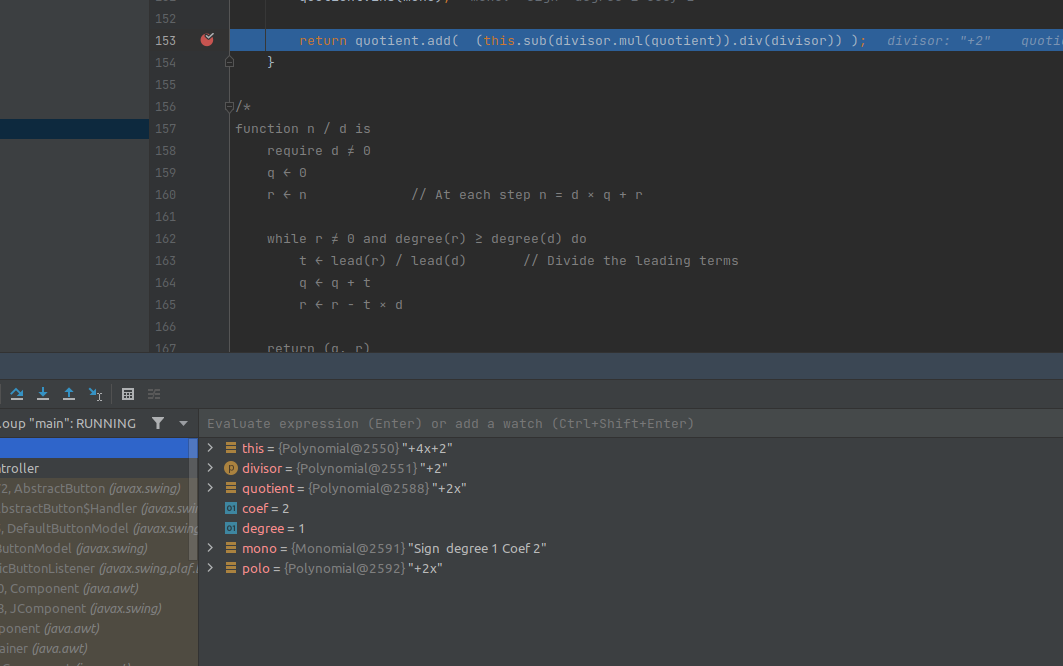
*Error Handling:*

*If the user enters an invalid polynomial or tries to do division by 0 then an Error will be thrown.*

*The User can select from the four operations appropriately.*

*By creating a use case diagram and use case descriptions for the functional requirements, we can better understand the system's behavior and its interaction with users for the Java Polynomial Calculator with a Swing GUI.*

*I also learned to use breakpoints which will allow me to debug functions much quicker. The attached photo below shows how I could step through the Division function to see the values of each Polynomial.*

**

# Design

*In the development of any software application, the object-oriented programming (OOP) design is an essential aspect of the process. For the Java Polynomial Calculator with a Swing GUI, it is important to define the OOP design and create UML package and class diagrams that represent the structure of the program. We want to do this before we start coding so we can understand exactly what we need to do. This helps save us time from spending most of our time fixing bugs.*

*OOP Design: The OOP design of the Java Polynomial Calculator with a Swing GUI will consist of the following classes:*

*Polynomial: Represents a polynomial and its associated operations (addition, subtraction, multiplication, evaluation).*

*UserInterface: Handles the user input and display of the polynomials.*

*Calculator: Acts as the controller for the application, managing the interaction between the Polynomial and UserInterface classes.*

*Defined Interfaces: The Java Polynomial Calculator with a Swing GUI will use the following interface:*

*Controller: Defines the methods for polynomial operations (addition, subtraction, multiplication, evaluation) that will be implemented by the Polynomial class.*

*By presenting the OOP design of the application, UML package and class diagrams, used data structures, defined interfaces, and used algorithms, we can better understand the structure and functionality of the Java Polynomial Calculator with a Swing GUI.*

# Implementation

Polynomial Class: The Polynomial class represents a polynomial and its associated operations. It has the following fields:

monos: A TreeMap that stores the coefficients of the polynomial with degree as their keys.

degree: An integer that represents the degree of the polynomial.

leadingCoefficient : The coefficient of the highest degree monomial inside the polynomial

The important methods of the Polynomial class are:

add(Polynomial poly2): Adds the given polynomial to the current polynomial.

sub(Polynomial poly2): Subtracts the given polynomial from the current polynomial.

mul(Polynomial poly2): Multiplies the current polynomial by the given polynomial.

mul(Polynomial poly2): Divides recursively the current polynomial by the given polynomial.

UserInterface Class: The UserInterface class handles the user input and display of the polynomials. It has the following fields:

Gui: A JTextArea that displays the polynomials and their results.

Input: A JTextField that allows the user to enter polynomials and their operations.

The important methods of the View class are prepareResults(), prepareGUI and they include our JPanel functionality.

Controller Class: The Calculator class acts as the controller for the application, managing the interaction between the Polynomial and UserInterface classes. It has most of our logic which sends the parsed Polynomials to the GUI.

Graphical User Interface Implementation: The implementation of the graphical user interface involves creating a JFrame and adding the necessary components (JTextField, JTextArea, JButtons) to it. The input JTextField allows the user to enter polynomials and their operations. The display JTextArea shows the entered polynomials, their results, and any error messages. The JButtons allow the user to perform polynomial operations (addition, subtraction, multiplication).

The implementation also involves adding ActionListeners to the JButtons, which call the appropriate methods of the Calculator class to perform the desired operation. When the user enters a polynomial in the input JTextField I essentially send the input to the same parser that I used for command line processing.

For this we used a Model, View, Controller design which basically broke our project down into Logic, GUI, and Buttons.

# Results

The testing scenarios for the Polynomial Calculator can be broken down into several categories:

Input validation: The application should be able to handle various types of input from the user, including invalid input such as non-numerical characters, empty input fields, and incorrectly formatted polynomials.

Polynomial operations: The application should be able to perform polynomial operations (addition, subtraction, multiplication) accurately and efficiently for different degrees of polynomials. For this we use a Treemap which is an extended HashMap in the Java Collections library. This would allow us to store many different degrees of polynomials without worrying about having to store 0 for every degree which a Monomial is absent.

GUI functionality: The graphical user interface should be simple and resemble something similar to what we are used to in our every day operating system.

# Conclusions

*In conclusion, the Java Polynomial Calculator with a Swing GUI project was an interesting and challenging assignment. Through the course of the project, I was able to learn a lot about object-oriented programming, data structures, and algorithms, as well as graphical user interface design and implementation.*

*One of the most important things I learned from this assignment is the importance of careful planning and design in software development. By taking the time to create detailed UML diagrams and design the overall architecture of the application, I was able to create a robust and efficient program that was easy to modify and extend.*

*In terms of future developments, there are several possible avenues to explore. One potential direction would be to add support for more advanced polynomial operations, such as differentiation and integration. I would also like to add a nicer GUI when I have more experience with Java Swing*

*Overall, I feel that this project has helped me to develop valuable skills and knowledge that will be useful in my future studies and career.*

# Bibliography

Cay S. Horstmann, Java Concepts: Compatible with Java 5, 6 and 7, Publisher: Wiley, ISBN: 978-0470509470, Published: 2011.

Joshua Bloch, Effective Java (2nd Edition), Publisher: Addison-Wesley Professional, ISBN: 978-0321356680, Published: 2008.

Herbert Schildt, Java: A Beginner's Guide (6th Edition), Publisher: McGraw-Hill Education, ISBN: 978-0071809252, Published: 2014.

Robert Lafore, Data Structures and Algorithms in Java (2nd Edition), Publisher: Sams, ISBN: 978-0672324536, Published: 2002.

Sedgewick, Robert and Wayne, Kevin, Algorithms, Fourth Edition, Publisher: Addison-Wesley Professional, ISBN: 978-0321573513, Published: 2011.

Oracle's official Java documentation, which includes detailed explanations of the Java class libraries and APIs. https://docs.oracle.com