DOCUMENTATION

ASSIGNMENT 1

STUDENT NAME: OLTEAN ROBERT-VIRGILIU

GROUP: 30422

# CONTENTS

[1. Assignment Objective 3](#_Toc128043139)

[2. Problem Analysis, Modeling, Scenarios, Use Cases 4](#_Toc128043140)

[3. Design 6](#_Toc128043141)

[4. Implementation 9](#_Toc128043142)

[5. Results 11](#_Toc128043143)

[6. Conclusions 12](#_Toc128043144)

[7. Bibliography 12](#_Toc128043145)

# Assignment Objective

**Main objective:**

Develop a polynomial calculator application using Java for back-end and Java Swing for graphical user interface.

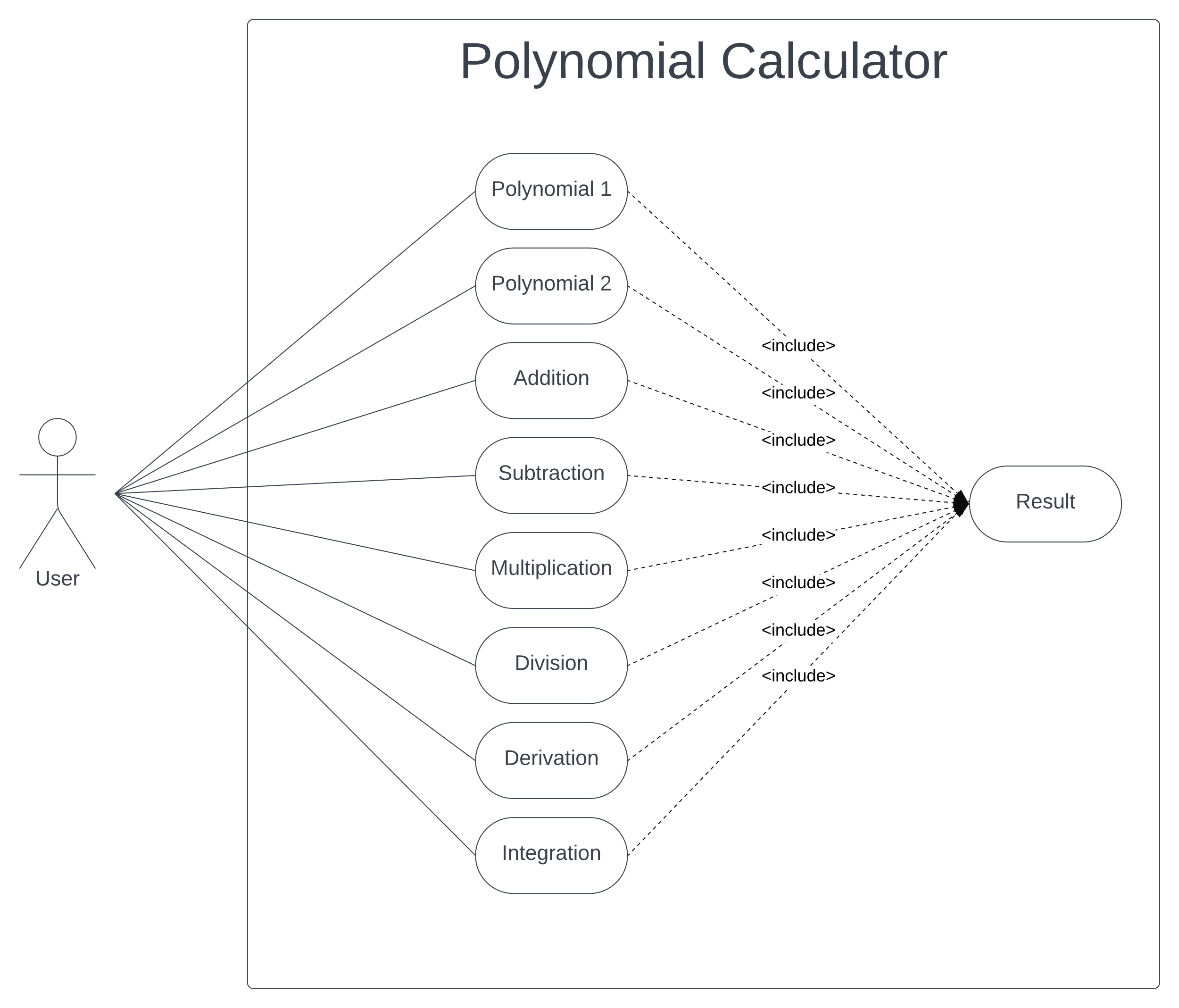
**Sub objectives:**

The following list outlines all the steps necessary to achieve developing the application, where each step is explained briefly. This provides a clear roadmap for understanding the goals for the assignment.

* **Design GUI Layout**
  + Creating an intuitive graphical user interface for the user (Java Swing).
* **Input validation**
  + Implementation of validation checks to ensure that user’s inputs are valid polynomials using Java Regular Expressions (RegEx).
* **Implementation of polynomial operations**
  + development of algorithms for polynomial addition, subtraction, multiplication, division, derivative and integrate.
* **Error handling**
  + Implementation of error handling mechanics to ensure valid inputs/output and operations.
* **Testing and debugging**
  + Conducting extensive testing to ensure a good and stable functioning of the calculator and debugging any issues encountered.
* **Documentation**
  + Creating comprehensive documentation covering all aspects of the project.

# Problem Analysis, Modeling, Scenarios, Use Cases

**Use cases**

**

**Use case diagram description**:

* Perform Polynomial Operation
  + This use case represents the core functionality of the calculator, including addition, subtraction, multiplication, division, derivation, and integration.
* Input Polynomial
  + Users can input two polynomials into the calculator to perform the operations mentioned above.
* View Result
  + After performing a polynomial operation, the calculator will show the user the result.

**Use case:** add polynomials

**Primary Actor:** user

**Main success scenario:**

1. the user must enter into the box-fields two polynomials written in this syntax “…±ax^3±bx^2±cx^1+dx^0” where “±” can be either “+” or “-” sign and “a,b,c,d,…” are the integer coefficients.

*E.g.*

*If the user wants to enter the polynomial:*

*Then the user must enter this in the text box:*

*1x^2-6x^1+9x^0.*

1. the user selects the ”Addition” operation from the drop-down menu,
2. the user clicks on the “calculate” button,
3. the polynomial calculator performs the addition of the two polynomials and displays the result.

**Alternative sequence:** incorrect polynomials

* the user inserts incorrect polynomials (e.g. not following the intended syntax),
* a pop-up message box will appear informing the user about the error,

the scenario returns to step 1/

**Functional requirements:**

* + - The application must take input two polynomials from the user, let the user choose from the programmed operations: addition, subtraction, multiplication, division, derivation, integration
    - **Polynomial** **addition** involves combining two or more polynomials to form a single polynomial.

**Non-functional requirements:**

* the polynomial calculator should be intuitive and easy to use by the user.
* the output should be clearly displayed and easily understandable.

**Use case:** subtract polynomials

**Primary Actor:** user

**Main success scenario:**

1. the user must enter into the box-fields two polynomials written in this syntax “…±ax^3±bx^2±cx^1+dx^0” where “±” can be either “+” or “-” sign and “a,b,c,d,…” are the integer coefficients.
2. the user selects the ”Subtraction” operation from the drop-down menu,
3. the user clicks on the “calculate” button,
4. the polynomial calculator performs the subtraction of the two polynomials and displays the result.

**Alternative sequence:** incorrect polynomials

* the user inserts incorrect polynomials (e.g. not following the intended syntax),
* a pop-up message box will appear informing the user about the error,

the scenario returns to step 1

**Functional requirements:**

* + - The application must take input two polynomials from the user, let the user choose from the programmed operations: addition, subtraction, multiplication, division, derivation, integration.
    - **Polynomial** **subtraction** is like addition but involves subtracting one polynomial from another.

**Non-functional requirements:**

* the polynomial calculator should be intuitive and easy to use by the user.
* the output should be clearly displayed and easily understandable.

**Use case:** multiply polynomials

**Primary Actor:** user

**Main success scenario:**

1. the user must enter into the box-fields two polynomials written in this syntax “…±ax^3±bx^2±cx^1+dx^0” where “±” can be either “+” or “-” sign and “a,b,c,d,…” are the integer coefficients.
2. the user selects the ”Multiplication” operation from the drop-down menu,
3. the user clicks on the “calculate” button,
4. the polynomial calculator performs the multiplication of the two polynomials and displays the result.

**Alternative sequence:** incorrect polynomials

* the user inserts incorrect polynomials (e.g. not following the intended syntax),
* a pop-up message box will appear informing the user about the error,

the scenario returns to step 1

**Functional requirements:**

* + - The application must take input two polynomials from the user, let the user choose from the programmed operations: addition, subtraction, multiplication, division, derivation, integration.
    - **Polynomial** **multiplication** entails multiplying each term of one polynomial by every term of another polynomial and then combining like terms.

**Non-functional requirements:**

* the polynomial calculator should be intuitive and easy to use by the user.
* the output should be clearly displayed and easily understandable.

**Use case:** divide polynomials

**Primary Actor:** user

**Main success scenario:**

1. the user must enter into the box-fields two polynomials written in this syntax “…±ax^3±bx^2±cx^1+dx^0” where “±” can be either “+” or “-” sign and “a,b,c,d,…” are the integer coefficients.
2. the user selects the ”Division” operation from the drop-down menu,
3. the user clicks on the “calculate” button,
4. the polynomial calculator performs the division of the two polynomials and displays the result.

**Alternative sequence:** incorrect polynomials

* the user inserts incorrect polynomials (e.g. not following the intended syntax),
* a pop-up message box will appear informing the user about the error,

the scenario returns to step 1

**Functional requirements:**

* + - The application must take input two polynomials from the user, let the user choose from the programmed operations: addition, subtraction, multiplication, division, derivation, integration.
    - **Polynomial division** involves dividing one polynomial by another to obtain a quotient and possibly a remainder.

**Non-functional requirements:**

* the polynomial calculator should be intuitive and easy to use by the user.
* the output should be clearly displayed and easily understandable.

**Use case:** derivate polynomial

**Primary Actor:** user

**Main success scenario:**

1. the user must enter into the box-fields two polynomials written in this syntax “…±ax^3±bx^2±cx^1+dx^0” where “±” can be either “+” or “-” sign and “a,b,c,d,…” are the integer coefficients.
2. t the user selects the ”Derivate” operation from the drop-down menu,
3. the user clicks on the “calculate” button,
4. the polynomial calculator performs the derivate of the first polynomial and displays the result.

**Alternative sequence:** incorrect polynomials

* the user inserts incorrect polynomials (e.g. not following the intended syntax),
* a pop-up message box will appear informing the user about the error,

the scenario returns to step 1

**Functional requirements:**

* + - The application must take input two polynomials from the user, let the user choose from the programmed operations: addition, subtraction, multiplication, division, derivation, integration.
    - **Polynomial** **derivation** involves finding the derivative of a polynomial function with respect to its variable.

**Non-functional requirements:**

* the polynomial calculator should be intuitive and easy to use by the user.
* the output should be clearly displayed and easily understandable.

**Use case:** integrate polynomial

**Primary Actor:** user

**Main success scenario:**

1. the user must enter into the box-fields two polynomials written in this syntax “…±ax^3±bx^2±cx^1+dx^0” where “±” can be either “+” or “-” sign and “a,b,c,d,…” are the integer coefficients.
2. the user selects the ”Integrate” operation from the drop-down menu,
3. the user clicks on the “calculate” button,
4. the polynomial calculator performs the integration of the first polynomial and displays the result.

**Alternative sequence:** incorrect polynomials

* the user inserts incorrect polynomials (e.g. not following the intended syntax),
* a pop-up message box will appear informing the user about the error,
* the scenario returns to step 1.

**Functional requirements:**

* + - The application must take input two polynomials from the user, let the user choose from the programmed operations: addition, subtraction, multiplication, division, derivation, integration.
    - **Polynomial** **integration** involves finding the indefinite or definite integral of a polynomial function with respect to its variable.

**Non-functional requirements:**

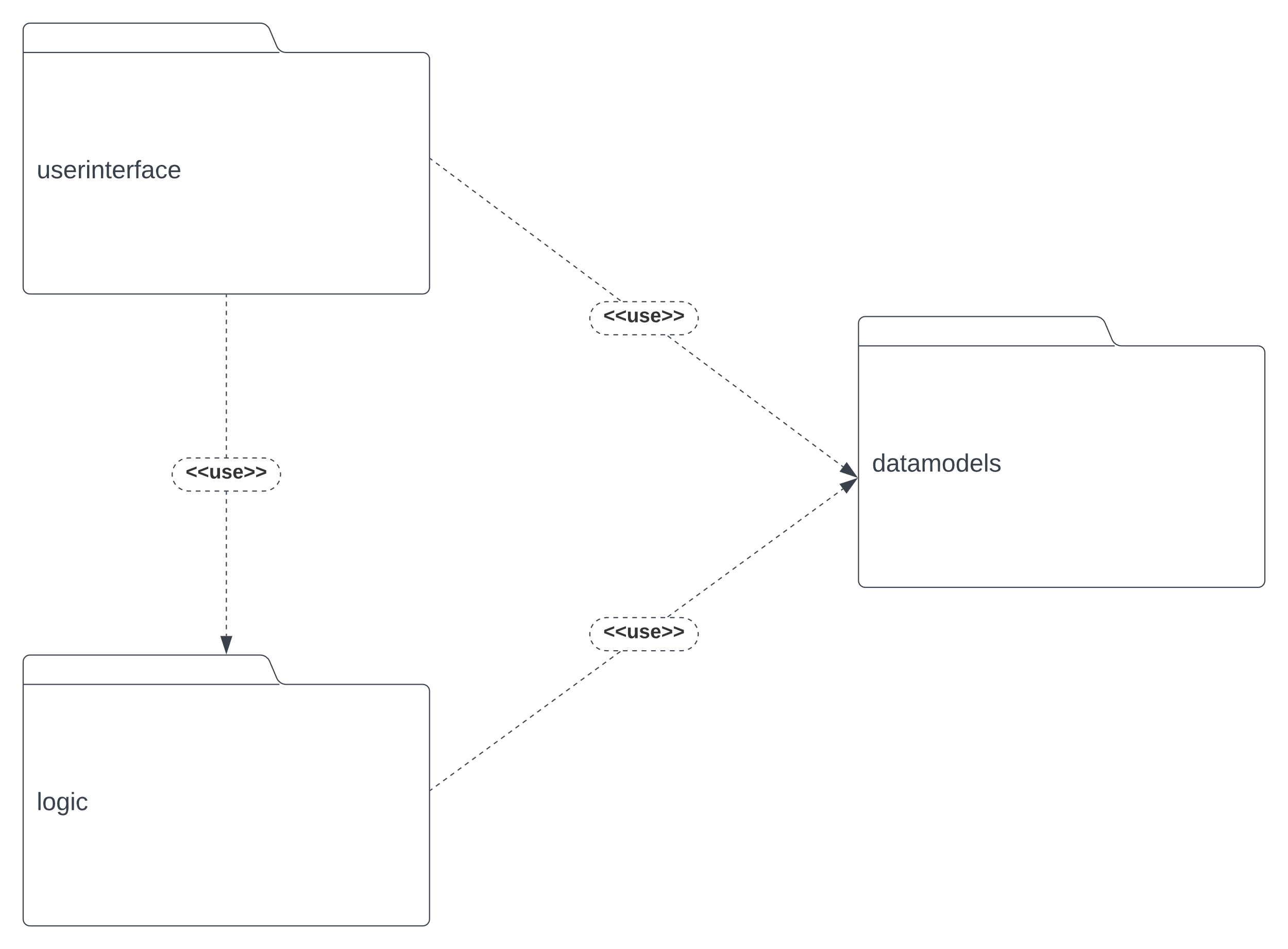
* the polynomial calculator should be intuitive and easy to use by the user.
* the output should be clearly displayed and easily understandable.

# Design

**Object-Oriented Programming**

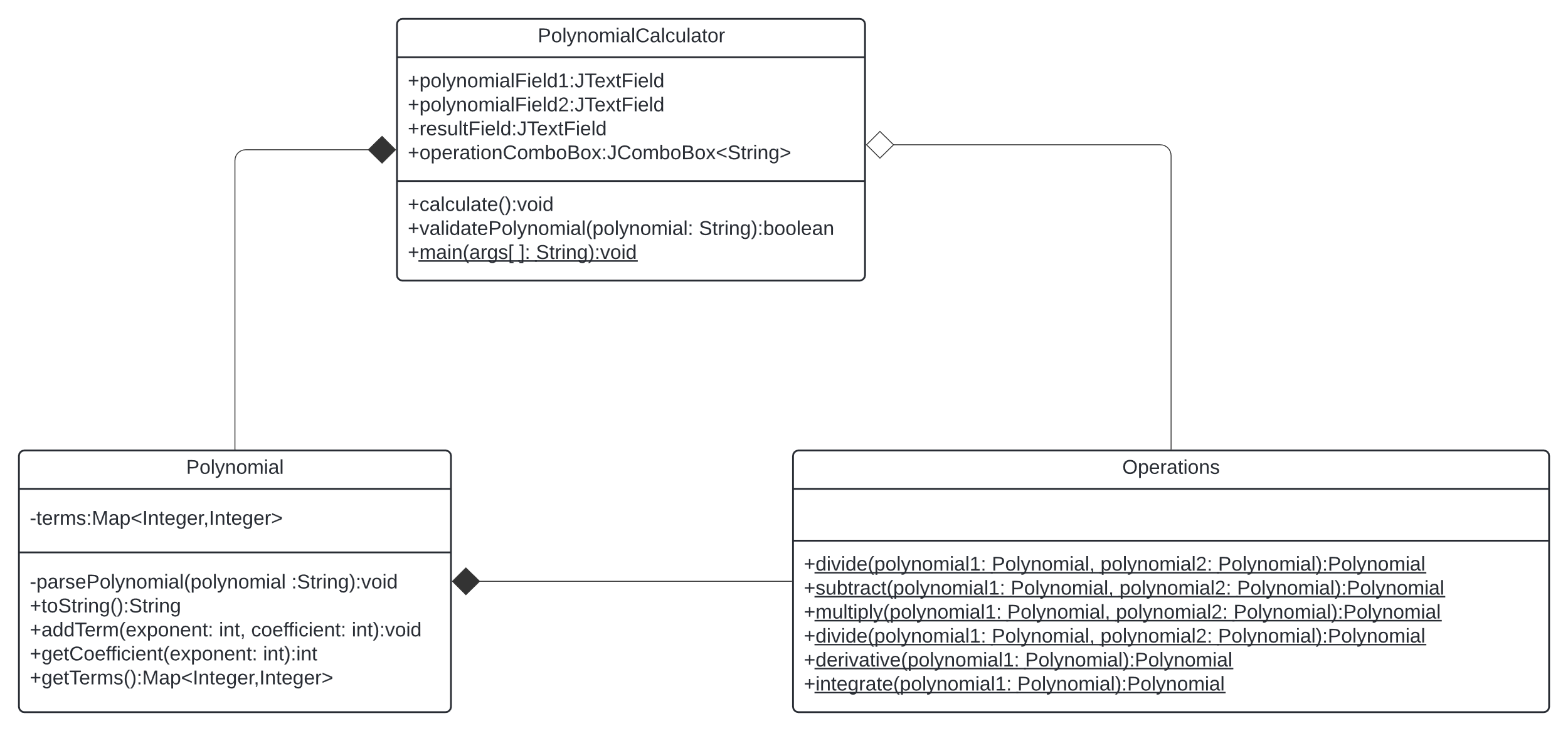
The application’s functionality is organized into several classes, each responsible for specific tasks, using OOP principles such as encapsulation and polymorphism.

* Encapsulation
  + Encapsulation is achieved by defining classes with private fields and providing public methods to interact with these. For example, in the “**Polynomial**” class, “**terms**” field is private and methods like “**addTerm**”, “**toString**” are provided to access/manipulate polynomial terms.
* Polymorphism
  + Polymorphism is demonstrated through method overriding. For example, “**toString**” method in “**Polynomials**” class is overridden to provide a custom string representation of the polynomials.

**UML Package Diagram:**

* **userinterface** package: contains the classes implementing the graphical user interface
* **logic** package: contains the classes implementing the mathematical or operational functionality
* **datamodels** package: contains the classes modeling the application data

**UML Classes Diagram:**



*The UML Class Diagram is a division of UML Package Diagram.*

**Data Structures used:**

* HashMap
  + Works on the principle of hashing
    - **Hashing** = assigning a unique code for any variable/object after applying any formula/algorithm on its properties
    - The **Hash** function should return the same hash code each and every time when the function is applied on same or equal objects => two equal objects must produce the same hash code
  + Handling collisions
    - Each bucket in Java contains a **LinkedList**
    - The Java implementation of **Hashtable** solves collisions by chaining.
  + The “**Polynomial**” class utilizes a **HashMap** to store polynomial terms, where the keys are the exponents, and the values represent the coefficients.
  + This data structure stores the terms of the polynomial in an efficient way and allows quick manipulation of data during the operations.

**Algorithm used:**

* Addition sums up the coefficients for same exponents. It checks each term from both polynomials. Adding two polynomials has time complexity O (n + m). Here, n and m are term counts in the polynomials.
* Subtraction compares terms across polynomials. Like addition, it iterates through terms subtracting coefficients with matching exponents. Its time complexity is O (n + m) too, with n and m being term counts.
* The multiplication algorithm takes every term from polynomial one and multiplies it by every term from polynomial two. This results in a new polynomial, representing their product. Its time complexity is O(n \* m), n and m being the number of terms of the two polynomials.
* Division is more complex. It involves a technique called polynomial long division. Its time complexity can vary depending on the implementation used.
* The derivative method calculates the derivative of a polynomial. It multiplies each term's coefficient by its exponent and reduces the exponent by one. The derivative method has a time complexity of O(n), where n represents the number of terms in the polynomial.
* The integration method computes the integral of a polynomial. The algorithm divides each term's coefficient by the exponent and raises the exponent by one. It also adds “ +C” at the final output because of the indefinite integral. The integration approach also has a time complexity of O(n).

# Implementation

**Class Description:**

* **PolynomialCalculator** Class:
* Fields:

+polynomialField1:JTextField

+polynomialField2:JTextField

+resultField:JTextField

+operationComboBox:JComboBox<String>

* part of the **userinterface** package
* the Front-end has been designed using **Java Swing** has as input two text fields for inputting the polynomials , one drop-down menu for selecting the operation wanted and a single button “**Calculate**”
* has the elements for the Graphical User Interface and regular expression pattern matching (**RegEx**)
* beside the front-end methods, the “validatePolynomial” method is one of the most important because it verifies that the polynomial typed in by the user are in typed in the correct syntax. It uses a **RegEx** for verifying the syntax:

^([-+]?\d\*x(\^\d+)?([-+](?!$)|$))+

* **Polynomial** Class
* Fields:

-terms:Map<Integer,Integer>

* Its main use is to manipulate and store the data got from the “**PolynomialCalculator**” using a **HashMap** and the “**parsePolynomial**” method.
* The “**Polynomial**” method uses a HashMap to store the data in an efficient manner, the keys are the exponents and the values are the coefficients.
* The “**parsePolyonimal**” method is also very important because it parses a polynomial string and splits the polynomials into monomials using a **RegEx**: \s\*(?=[+-]) and also splits the monomials to get the exponent so it can populate the polynomial from the string got from the user’s input.
* **Operations** Class
* This class contains all the methods for the operations: addition, subtraction, multiplication, division, derivation and integration.
* **GUI – Graphical User Interface**

All the GUI elements are stored in “**PolynomialCalculator**” class which is also the main controller class. The GUI features a simple, intuitive and easy to use design. The user will be prompted to enter one polynomial in each text box, an additional text is displayed on the bottom-left part to inform the user of how the polynomial should be entered. If the polynomial syntax is not respected then a pop-up error dialog will show informing the user to correct his input. After inputting the polynomials the user must select from the combo-box the preferred operation and then press the “**Calculate**” button to show the result.

**A screenshot of a computer

Description automatically generatedMain screen of the GUI:**

**Error handling – invalid input form the user:**

A screenshot of a computer

Description automatically generated

**A screenshot of a computer

Description automatically generatedDrop-Down menu for choosing the action:**

# Results

The testing of the application was made using a JUnit testing class named ”**PolynomialCalculatorTest**”. The testing was conducted with the help of the trusted online calculator **Symbolab**. Two polynomials were chosen then using Symbolab, we got the correct result for each individual operation type then compared the result from our polynomial calculator using **“assertArrayEquals(expectedChars, actualChars);“**a function that compares the two result strings and if they are the same the test will be passed.

**Testing using the Intellij IDEA built-in tester:**

A screenshot of a computer

Description automatically generated

**Testing using Maven test:**

A screen shot of a computer

Description automatically generated

# Conclusions

During the time developing this application, I have deepened my understanding of polynomial characteristics and **OOP** programming. Overall, the project was a success despite having some difficulties understanding some paradigms and algorithms but in the end I managed to overcome them.

As for future developments, a nicer UI using more sophisticated design tools like **JavaFX** would make an addition to the project. Also, a way of not being restricted to only one correct syntax for the polynomial would also be beneficial.

# Bibliography

1. *Java Regular Expressions -* [*www.w3schools.com/java/java\_regex.asp*](https://www.w3schools.com/java/java_regex.asp)
2. *Java HashMap -* [*www.w3schools.com/java/java\_hashmap.asp*](https://www.w3schools.com/java/java_hashmap.asp)
3. *Symbolab -* [*https://www.symbolab.com*](https://www.symbolab.com)
4. *Creating UML Diagrams -* [*https://www.lucidchart.com*](https://www.lucidchart.com)