

# DOCUMENTATION

## ASSIGNMENT 1



STUDENT NAME: URDEA MARA-CRISTINA  
GROUP: 30422

# CONTENTS

<b>1.</b>	<b>Assignment Objective .....</b>	<b>3</b>
<b>2.</b>	<b>Problem Analysis, Modeling, Scenarios, Use Cases .....</b>	<b>3</b>
<b>3.</b>	<b>Design .....</b>	<b>4</b>
<b>4.</b>	<b>Implementation .....</b>	<b>7</b>
<b>5.</b>	<b>Results .....</b>	<b>8</b>
<b>6.</b>	<b>Conclusions .....</b>	<b>9</b>
<b>7.</b>	<b>Bibliography .....</b>	<b>9</b>

## 1. Assignment Objective

Design and implement a system for polynomial calculator.

To achieve the objective it is needed to have a dedicated graphical interface through which the user can insert polynomials, select the mathematical operation to be performed and view the result, also it is needed the implementation of the class polynomial and the implementation of the mathematical operations performed on the polynomials.

NOTE: Consider the polynomials of one variable and integer coefficients.

## 2. Problem Analysis, Modeling, Scenarios, Use Cases

### Problem Analysis

A polynomial is a mathematical expression that involves constants and variables combined through addition, multiplication, and non-negative integer exponentiation. When it comes to polynomials that depend on a single variable, they can always be expressed in the form of

$$\sum_{i=0}^n (a_i x^i) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n$$

where  $a_0, a_1, a_2, \dots, a_n$  are constants and  $x$  is the variable. The variable  $x$  is considered indeterminate, which means it doesn't have a specific value, but it can be substituted with any value.

Polynomials are made up of multiple monomials. Each monomial contains the variable "x," a coefficient, and an exponent that represents the power to which the variable is raised. The exponent is always a positive integer, and the coefficient can be any real number. Nevertheless, the polynomial calculator will only use rational numbers as coefficients.


### Modelling the problem

To utilize the functions of the calculator, the user must input two polynomials correctly into the interface and select the desired operation to be performed. The available operations include: adding, subtracting, multiplying, dividing two polynomials, differentiating a polynomial (the first polynomial) and integrating a polynomial (the first polynomial). The interface will display the outcome of the selected operation.

### Use cases and scenarios

A use case is a method employed in system analysis to identify, clarify, and organize system requirements. It comprises a set of potential sequences of interactions between users and systems in a particular environment, all aimed at achieving a specific goal.

As use cases are closely tied to the actions taken by users, the interface design prioritizes user-friendliness. The following design is the result of this approach.

 Polynomial Calculator

—

□

×

Polynomial 1:

Polynomial 2:

Result:

Sum

Subtraction

Multiplication

Division

Derivative

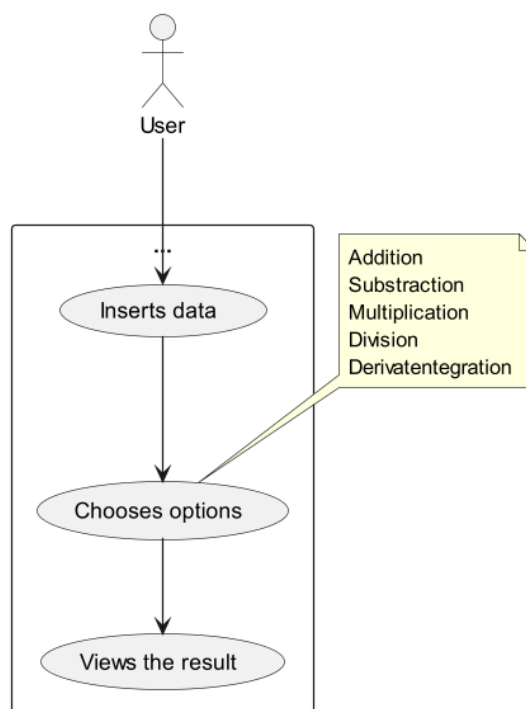
Integration

To use the calculator, the user must input two polynomials in the corresponding text fields and select the desired operation to perform. It is crucial to write the polynomials correctly for the calculator to work properly. When adding, subtracting, multiplying, and dividing, the result follows the order of the polynomials. However, for integration and differentiation, only the first polynomial will be used to perform the operation.

To write the polynomials correctly, use the following format: [coefficient]\*x^[power] [+/-] [coefficient]\*x^[power], and so on.

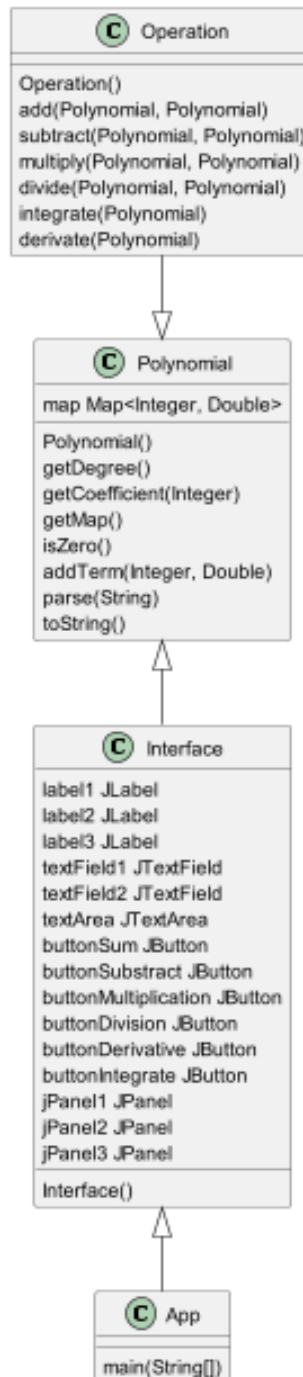
### 3. Design

#### Use cases diagram



Use cases are a technique used in system analysis and software engineering to identify, clarify, and categorize system requirements. They provide a way to model the behavior of a system by defining its interactions with external users. Use cases are written in natural language and describe the sequence of steps that a user takes to achieve a specific goal using the system. They help in understanding the system's functionality. Use cases also serve as a foundation for testing and validating the system's functionality.

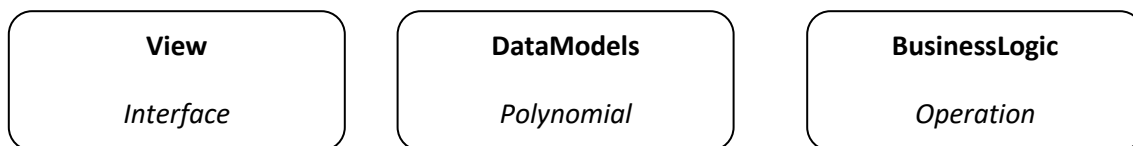
### Class diagram



The UML, Unified Modelling Language, diagram presented above is a visual representation used to illustrate the software system. It includes the Java classes used to implement the project and their relationships and dependencies.

### Packages

The purpose of Java packages is to group related classes together for easy organization. In Object-Oriented Programming (OOP), the Model-View-Controller (MVC) pattern is a common approach to designing projects for optimal efficiency. This pattern is widely used in most of the programming languages. In my project, a modified version of the MVC pattern was utilized, with the model and view components remaining the same, but the controller section being altered into a package named BusinessLogic for a better organization.



The project is made up of the DataModels, View, and BusinessLogic as its core parts. The logical organization of the project and the high-level classes connected to it are under the control of the DataModels. All user interactions, including entering polynomials, choosing operations, and viewing results, are handled by the View. Finally, all the algorithms used to carry out the polynomial operations and determine the results are included in the BusinessLogic package.

- The App is not technically a package, but it can be thought of as one. The main component of the application is located here and serves as the starting point.
- The Polynomial class, which models the entire problem, can be found in the Model package.
- The View package contains the Interface, which includes an interactive class where users can enter polynomials and view results.
- The BusinessLogic package only has one class, which includes all of the methods required to perform arithmetic operations on polynomials.

### Data structures

Both primitive and complex data structures are used in this project. Integer, Double, and String make up the primitive data structures, whereas the Map interface is the complex one. The Map interface is useful for creating a "table" out of large amounts of values. The HashMap implementation of the Map interface, which has a search complexity close to  $O(1)$  and is ideal for retrieving values from the data structure, was used in this project. The polynomials used are kept in a hash map.

## 4. Implementation

### Classes:

#### 1. Polynomial

It has an empty constructor that initializes a HashMap called "map" to store the degree-coefficient pairs of the polynomial. The class has several methods, including:

- addTerm(): this method takes a degree and coefficient and adds it to the map if the coefficient is non-zero or removes it if it is zero.
- getDegree(): This method returns the highest degree of the polynomial.
- getCoefficient(): This method returns the coefficient of a given monomial.
- toString(): This method returns a string representation of the polynomial in a user-friendly format.
- isZero(): This method checks whether the polynomial is identically zero.
- parse(): This method takes a string representation of a polynomial and converts it into a Polynomial object by parsing its terms and adding them to the map using the addTerm() method described before.

#### 2. Operation

Here is a brief description of each method included:

- add(Polynomial a, Polynomial b): This method takes two Polynomial objects as input and returns their sum.
- subtract(Polynomial a, Polynomial b): This method takes two Polynomial objects as input and returns their difference.
- multiply(Polynomial a, Polynomial b): This method takes two Polynomial objects as input and returns their product.
- derivative(Polynomial a): This method takes a Polynomial object as input and returns its derivative.
- integrate(Polynomial a): This method takes a Polynomial object as input and returns its integral.
- divide(Polynomial a, Polynomial b): This method takes two Polynomial objects as input and returns their quotient and remainder as an ArrayList of Polynomial objects.

#### 3. Interface

The class extends JFrame and provides a graphical user interface (GUI) for polynomial operations.

The components of the GUI include:

- Three JLabels: label1, label2, and label3
- Two JTextFields: textField1 and textField2
- One JTextArea: textArea
- Six JButtons: buttonSum, buttonSubtract, buttonMultiplication, buttonDivision, buttonDerivative, and buttonIntegrate

The class constructor initializes the components, sets their fonts, colors, and backgrounds, creates three JPanels to organize the components, and sets the layout of the JFrame to BorderLayout.

The class also defines six ActionListeners for the JButtons to perform the polynomial operations: addition, subtraction, multiplication, division, derivative, and integration. These listeners retrieve the polynomial expressions from the JTextFields, parse them into polynomial objects, perform the requested operation using the Operation class defined in the package "org.example.BusinessLogic", and display the result in the JTextArea.

#### 4. App

In this class, the only field that can be found is the main.

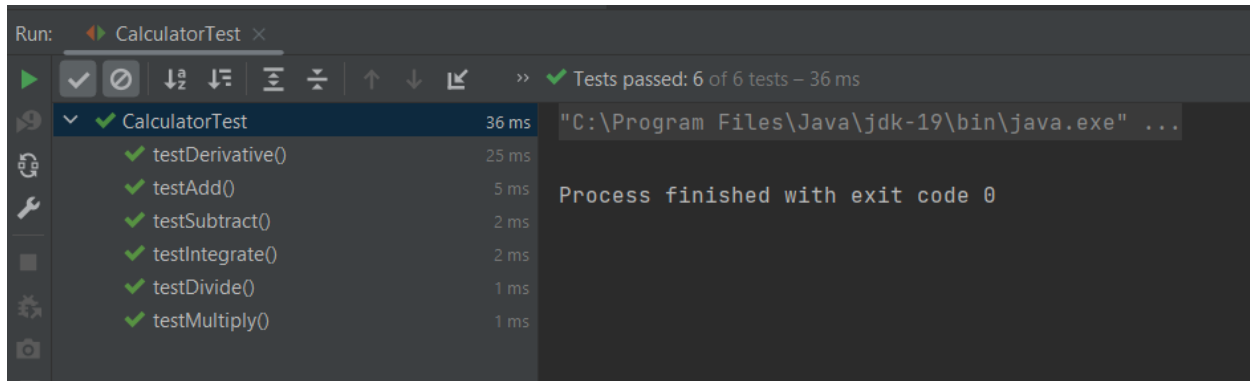
#### Algorithms

1. Addition - add(Polynomial a, Polynomial b): This algorithm adds two polynomials a and b together. It creates an empty result polynomial, and then loops over the terms in a and b. For each term, it adds the term's degree and coefficient to the result polynomial. It then returns the result polynomial.
2. Subtraction - subtract(Polynomial a, Polynomial b): This algorithm subtracts b from a. It follows a similar process to the add() algorithm, except that it subtracts the coefficients of b's terms from the result polynomial instead of adding them.
3. Multiplication - multiply(Polynomial a, Polynomial b): This algorithm multiplies a and b. It creates an empty result polynomial, and then loops over the terms in a and b. For each pair of terms, it multiplies their degrees and coefficients together, and adds the resulting term to the result polynomial.
4. Differentiation - derivative(Polynomial a): This algorithm calculates the derivative of a. It creates an empty result polynomial, and then loops over the terms in a. For each term, it calculates the derivative (which involves multiplying the term's coefficient by its degree, and subtracting 1 from the degree), and adds the resulting term to the result polynomial.
5. Integration - integrate(Polynomial a): This algorithm calculates the indefinite integral of a. It creates an empty result polynomial, and then loops over the terms in a. For each term, it calculates the integral (which involves dividing the term's coefficient by its degree plus 1, and adding 1 to the degree), and adds the resulting term to the result polynomial. It then returns the result polynomial.
6. Division - divide(Polynomial a, Polynomial b): This algorithm divides a by b, returning both the quotient and the remainder as separate polynomials. It initializes result and remainder polynomials to be copies of a, and then repeatedly divides the highest-degree term of remainder by the highest-degree term of b. It adds the resulting term to result, and subtracts b times the resulting term from remainder. It repeats this process until the degree of remainder is less than the degree of b. Finally, it returns the result and remainder polynomials as an ArrayList.

## 5. Results

Junit was used to carry out the tests, which covered both the Model and the BusinessLogic thoroughly.





## 6. Conclusions

I learned a lot throughout the project, from brushing up on basic ideas to implementing more complex algorithms that are needed to effectively complete the assignment requirements. The project was a good challenge with a few confusing parts, but definitely enjoyable. There is plenty of room for future development, including the addition of features that were not included in the requirements. For instance, we could add nicer GUI, add new operations, or even let users work with more than two polynomials.

## 7. Bibliography

<https://regexr.com/>  
<https://docs.oracle.com/javase/tutorial/uiswing/index.html>  
<https://www.baeldung.com/junit-5>  
<https://en.wikipedia.org/wiki/Polynomial>  
<https://dsrl.eu/courses/pt/>  
<https://www.geeksforgeeks.org/>