

# DOCUMENTATION

## ASSIGNMENT *1*

GIUROIU TUDOR  
GROUP 30424

# CONTENTS

|  |    |
|--|----|
| 1. Assignment Objective .....                            | 3  |
| 2. Problem Analysis, Modeling, Scenarios, Use Cases..... | 3  |
| 3. Design .....  | 5  |
| 4. Implementation .....                                  | 7  |
| 5. Results.....  | 10 |
| 6. Conclusions.....                                      | 10 |
| 7. Bibliography .....                                    | 10 |

## 1. Assignment Objective

The main objective of the assignment is to design and develop a polynomial calculator including a graphical user interface, through which the user can interact with the program by inserting polynomials, selecting the required operation and observing the resulted output.

Sub-objectives:

| Sub-objective | Description   | Section        |
|---------------|---|----------------|
| 1             | Defining the polynomial class                                 | Implementation |
| 2             | Implementing the polynomial Operations class                  | Implementation |
| 3             | Creating the GUI  | GUI            |
| 4             | Integrating the GUI interface with the back-end functionality | GUI            |
| 5             | Test polynomial calculator application                        | Testing        |

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

### Problem Analysis

A polynomial is a mathematical expression consisting of one or more terms, each of which involves a variable raised to a non-negative integer power and multiplied by a coefficient. Each of these terms are called monomials, which are composed of a real number (the coefficient) and a non-negative integer power of  $x$ . Because polynomials are formed by a multitude of monomials, each of which consists of a real number and a positive integer pair, they can be represented in an alternative way:

$$\{(a_0, 0), (a_1, 1), (a_2, 2), \dots, (a_n, n)\},$$

where each pair represents a monomial,  $a_n$  being the coefficient and  $n$  being the power of the variable  $x$ .

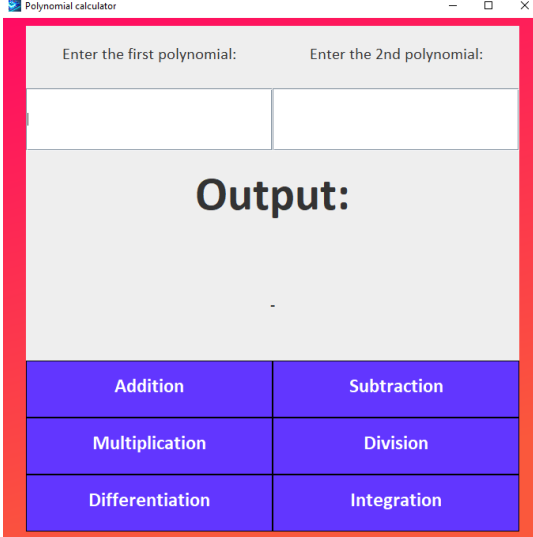
For example, the polynomial  $5.2x^2 + x - 49$  can be represented in the following way:  $\{(-49, 0), (1, 1), (5.2, 2)\}$ . This way of representing the polynomials will prove to be practical for implementing the polynomials in a computer program and for performing the needed operations.

## Modelling

The user will be able to insert via keyboard one or two polynomials and select an operation from the following: addition, subtraction, multiplication, division, differentiation, and integration. After pressing the button of the desired operation, the program will display the resulted polynomial on the screen or an error message if the input provided by the user was incorrect or the operands are not valid.

## Scenarios

After introducing the polynomials in the text boxes, the user can select the operation that he/she wants to be performed by pressing the button corresponding to the desired operation. Immediately after, under the “Output: “ message the result should be displayed.



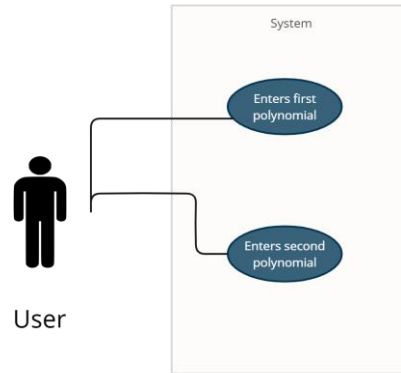
| Enter the first polynomial: |             | Enter the 2nd polynomial: |  |
|-----------------------------|-------------|---------------------------|--|
|                             |             |                           |  |
| <b>Output:</b>              |             |                           |  |
|                             |             |                           |  |
| Addition                    | Subtraction |                           |  |
| Multiplication              | Division    |                           |  |
| Differentiation             | Integration |                           |  |

## Use cases

The functional requirements of the application are the following:

1. The system should allow the user to enter two polynomials: the user enters two polynomials into the calculator by typing in the classical way of representing a polynomial, signaling the powers of x via the character '^'.

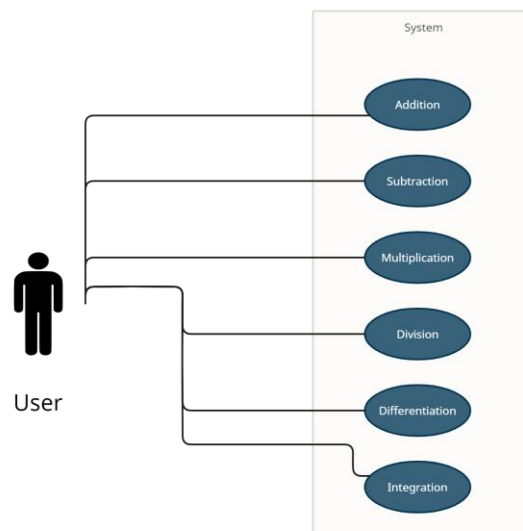
- The user inputs the first polynomial.
- The user inputs the second polynomial.
- The user verifies that the input is correct.



2. The system should allow the user to perform polynomial addition, subtraction, multiplication, differentiation, and integration operations: The user performs arithmetic operations on the two or only on the first polynomials entered in the previous use case.

- The user selects the desired arithmetic operation from the GUI.
- The system performs the selected operation on the two input polynomials.
- The result is displayed on the GUI.

3. The system should provide an easy-to-use graphical user interface (GUI).



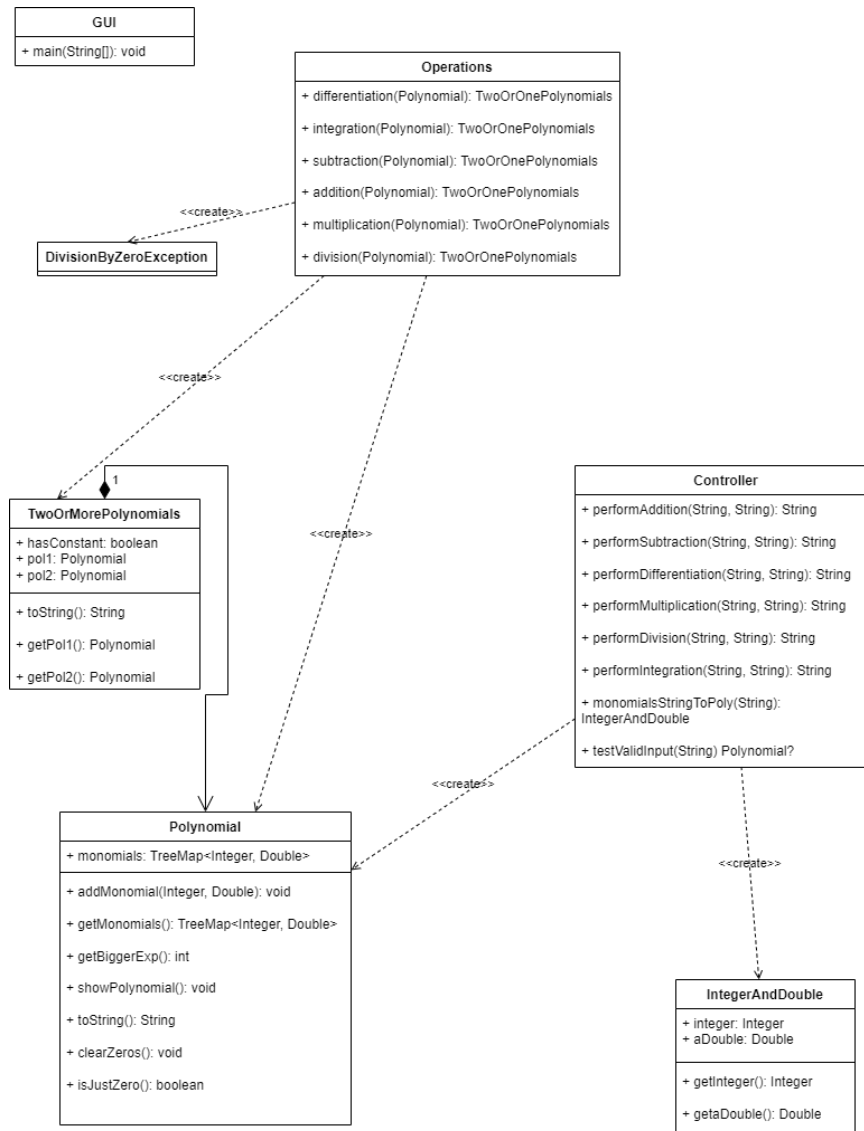
### 3. Design

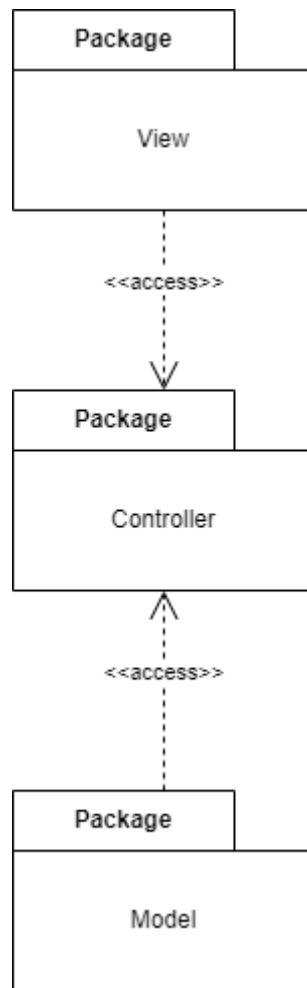
The Object-Oriented Programming approach to the polynomial calculator involves defining classes and objects that represent polynomials and their various operations. This approach allows for a more organized and modular design, making it easier to modify and maintain the code over time.

To begin with, we would define a Polynomial class that helps us manage the individual polynomials. Each polynomial is composed of numerous monomials, which will be represented through the TreeMap data structure. This way we simplify the storing of a polynomial, avoiding the creation of another class, Monomials.

For the operations that can be performed on polynomials, the class Operations is responsible. Here, using abstraction, the developers can work with the static methods in performing operations on the polynomials, not needing to know the way these operations work. Abstraction is also used in the methods of the Controller class, allowing for the GUI to not focus on the implementation of the operations and String to Polynomial conversions, instead of just using the simple method calling.

A set of methods are provided for each class in order to perform some functionalities specific to that class. For each class a constructor is provided, in addition to getters, which are used to protect the fields of an object and allow just the necessary level of access.





## 4. Implementation

The Polynomial class is a model class that represents a mathematical polynomial. It utilizes a TreeMap data structure to store the polynomial's monomials, where the keys represent the exponents and the values represent the coefficients. The class provides various methods for manipulating the polynomial, such as adding monomials, clearing zero monomials, getting the polynomial's highest exponent, and converting the polynomial to a string representation.

It also provides a basic implementation of a polynomial that can be used in a polynomial calculator application. Its use of a TreeMap ensures that the monomials are stored in order by their exponents, which is a key property of polynomials. It was also chosen for the TreeMap to not include the exponents for which the coefficient is 0, because doing this would load the memory with redundant data.

The class Operations provides six static methods: addition, subtraction, multiplication, division, differentiation, and integration, of which the first four take two Polynomial objects as arguments and the last two just one Polynomial object and return a TwoOrOnePolynomials object.

The addition method computes the sum of two polynomials by iterating over their terms, adding corresponding terms with the same degree, and creating a new polynomial with the resulting coefficients. The bigger polynomial is determined based on their highest degree.

The subtraction method works similarly, but subtracts corresponding terms instead of adding them. It also handles the case where the polynomials have different degrees.

The multiplication method computes the product of two polynomials by iterating over all possible combinations of terms, computing the product of their coefficients and summing the products with the same degree. It returns a new polynomial with the resulting coefficients.

The division method implements polynomial long division, where the dividend is divided by the divider until the degree of the dividend is less than the degree of the divider. The method starts by checking which polynomial has the highest degree and initializing the quotient and remainder to 0. It then iteratively computes the quotient of the highest degree terms and subtracts the product of the quotient and the divider from the dividend, until the degree of the dividend is less than the degree of the divider. The method returns a TwoOrOnePolynomials object that contains both the quotient and remainder.

The differentiation method performs differentiation of just one Polynomial. For each monomial, the method checks if its degree is not equal to zero. If the degree is not zero, the method computes the derivative of the monomial by multiplying its coefficient with its degree and subtracting one from its degree. The result is then added as a new monomial to the "result" polynomial.

The integration method functions similarly, but instead of computing the derivative of the monomials, it computes the primitive.

The class IntegerAndDouble serves as a helper class, which stores both an Integer object and a Double one. It is designed to store the data needed for a monomial and provides just the getters for the two fields of the class, the Integer and the Double.

The Controller class provides methods for each operation to be performed and serves as a connection between the GUI and the implementation. Using this class, the developers of the GUI do not need to be concerned about the implementation of the polynomials operations. One key method contained in this class is the "testValidInput" method, which takes a String as an argument and provides null if the entered string is not written in the right form for a polynomial, of the inputted polynomial, in the other case. Also, the method "monomialsStringToPoly" provides a quick way of converting a monomial passed in as a String to a pair of two values, Integer and Double. These last two methods make use of the regular expressions to filter the information that comes in.



The class `TwoOrOnePolynomials` has three private instance variables: `hasConstant` (a boolean), `pol1` (a `Polynomial` object), and `pol2` (another `Polynomial` object). The class has three getter methods: `getPol1()`, `getPol2()`, and `toString`. The `toString()` method returns a `String` representation of the object. This method contains conditional statements that check whether the `pol2` instance variable is null or not. If `pol2` is null, it concatenates the `String` representation of `pol1` to the output `String`. If `hasConstant` is true, it also adds the `String` `+ constant` to the output (used in representing the output of the integration operation). If `pol2` is not null, it concatenates a formatted `String` that includes the `String` representation of `pol1` and `pol2` (used in representing the output of the division operation).

The `DivisionByZeroException` class is a custom exception class that extends the `Exception` class. This class is used to handle situations where there is an attempt to divide a polynomial by zero, which is an illegal operation in mathematics.

The class `GUI` represents the Graphical User Interface of a polynomial calculator application. The GUI is implemented using `Swing`, a GUI toolkit for Java.

The GUI consists of a main frame with a title `"Polynomial calculator"`, and a custom icon. It has a main panel, with a gradient background, which contains three sub-panels: an upper panel, a middle panel, and a bottom panel.

The upper panel contains two labels, `"Enter the first polynomial"` and `"Enter the second polynomial"`, and two text fields for the user to input polynomials.

The middle panel contains a label `"Output"` and another label to display the result of the operation performed on the two input polynomials.

The bottom panel contains six buttons for performing various polynomial operations such as addition, subtraction, multiplication, division, differentiation, and integration. Each button has an `ActionListener` that calls a corresponding method in the `Controller` class to perform the operation and update the result displayed in the middle panel.

The GUI also contains two custom classes: `JPanelGradient`, which extends `JPanel` and creates a gradient background for the main panel, and `MyJButton`, which extends `JButton` and creates a custom button with specific color, font, and border

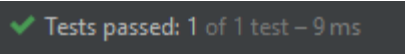
.

## 5. Results

The testing scenarios are generated via Junit. The OperationsTest test class. The Operations class contains static methods for performing mathematical operations on polynomials, including addition, subtraction, multiplication, division, integration, and differentiation.

The test methods in this class test the correctness of each operation by creating two or more polynomial objects, performing the operation on them using the corresponding method from the Operations class, and comparing the resulting polynomial object with the expected result.

The results after running a method that incorporates all the tests:



## 6. Conclusions

In conclusion, the polynomial calculator was successfully designed and implemented, fulfilling the objectives set out in the assignment. The Object-Oriented Programming approach made the code more organized, modular, and easy to maintain, while the use of the TreeMap data structure ensured that the monomials were stored in order by their exponents, which is a key property of polynomials. The GUI interface provided a user-friendly experience, and the integration with the back-end functionality was successful. Overall, the polynomial calculator is a reliable and accurate tool for performing polynomial operations and can be used in a variety of applications.

I especially learned to Connecting the GUI with the backend involving integrating the user interface with the functionality of the polynomial calculator. This step was crucial because it enables the user to interact with the program and perform the desired operations. The process of connecting the GUI with the backend involved defining event handlers that respond to user actions, such as button clicks or input changes, which was a really new thing to learn. Learning to connect the GUI with the backend requires a good understanding of both the GUI and the backend code. The project also required a lot of debugging, which was a nice experience, for now, because the scale of the program.

## 7. Bibliography

<https://dsrl.eu/courses/pt/>