

**Fundamental Programming Techniques**

**Assignment 1: Polynomial Calculator**

**Student:** Diana-Maria Jurcan

**Group:** 30422

**Laboratory professor:** Marcel Antal

**Course professor:** Cristina Bianca Pop

# CUPRINS

1. Assignment objective……………………………………….3
2. Problem analysis, modelling, scenarios, usage cases……….3
3. Design……………………………………………………….5
4. Implementation…………………………………………….10
5. Results……………………………………………....……...13
6. Conclusions………………………………………………...14
7. Bibliography……………………………………………......14
8. **Assignment objective**

The objective of the current laboratory topic was to propose, design and implement a system of processing polynomials of a single variable, with integer coefficients. In other words, it is a mini application that can process operations with a maximum of two polynomials, operations such as: addition, subtraction, derivation, and integration.

This calculation system has as input data of two polynomials, and as output data we have the result generated after performing one of the operations. In addition, the application also has a graphical user-friendly interface, made in order to facilitate the use, and to make the use of the application as clear as possible.

1. **Analysis**

A polynomial is an expression, consisting of several monomials, which, in turn, are constructed with the help of coefficients and exponents. There are several methods by which a computer of polynomials can be made. The way in which the information is entered (coefficients and exponents), the way in which they are subsequently placed, the final appearance, and the appearance of the graphical interface remain only with the attitude of the programmer.

A polynomial P in an indeterminate X is formally defined as:

𝑃(𝑋) = 𝑎𝑛 ∗ 𝑋𝑛 + 𝑎𝑛−1 ∗ 𝑋𝑛−1 + ⋯ + 𝑎1 ∗ 𝑋 + 𝑎0

where: c1, c2, …, cn

represent the polynomial’s coefficients

n represents the polynomial degree

A monomial is a special type of polynomial with only one term.

Consider another polynomial Q in the indeterminate X which is formally defined as:

𝑄(𝑋) = 𝑏𝑛 ∗ 𝑋𝑛 + 𝑏𝑛−1 ∗ 𝑋𝑛−1 + ⋯ + 𝑏1 ∗ 𝑋 + 𝑏0

**Project modeling**

The complexity of the project can be done in dozens of implementation ways, each with its own efficiency. The first implementation that everyone would think of would be to use vectors to store the coefficients and degree of each polynomial. Then add, subtract, integrate and derive the coefficients and the degree displayed in a third vector from where the String to be displayed will be processed. This implementation would have some problems, especially related to efficiency.

Another way would be to work directly on Strings. We should separate and convert each degree and coefficient into the correct type and process them, after which we should save them again in a third String.

I tried to implement it in the following way: I took over each String separately, I separated by "token" functions according to the sign and then saved them in different Strings. Each is taken and processed separately, the coefficients and degrees of each will be extracted, then they will be saved in a list of monomials (polynomial consisting of a single term, e.g., 4x, 3x^2). The list will contain both the coefficient and the degree and string itself.

After processing these monomials and storing them in a list of monomials, all the functions corresponding to operations in the "Polynomial" class can be applied. Both polynomials will be used for addition, subtraction, derivation, and integration.

After entering the result in a new monomial, it will be returned to the form of a String and finally displayed in the graphical interface.

For the program to function properly, the string entered in the graphical interface must follow the “coefficient ^ degree” pattern, without which an error message will be displayed. If the polynomial is entered correctly, the options are clear: addition, subtraction, derivation, and integration, each with its own fields for displaying results.

**Use cases**

This project solves the operations on polynomials, whatever the introduced polynomial, the result being correct every time. This project can be easily used as an application for both teachers and students.

At the same time, considering that the graphical interface is present, and it is not necessary to understand the code, it can be used by anyone, without presenting difficulties. The task is simple, consisting in entering any two polynomials and displaying the result after selecting the desired option, through the 4 buttons drawn with the icon suggestive of the operations, the result being displayed in each text field dedicated to the operations performed.

1. **Design**

* **Use case diagram**

Diagram

Description automatically generated

The user will firstly enter a polynomial in the text field of their choice between the first two or will enter two polynomials in the first two text fields. After that, the user will press the button of the corresponding operation of whose result it wants. After pressing the button, the result text field should be filled with a string with the given result. To empty a text field, the user can press the empty button corresponding to the text field that needs to be emptied. Finally, for exiting the application, there is an exit button present in the down right corner.

* **UML Diagram**

Unified Modeling Language or UML for short is a standard language for describing models and software specifications. UML was originally developed to represent the complexity of object-oriented programs, the foundation of which is the structuring of programs into classes, and their instances (also called objects). However, due to its efficiency and clarity in the representation of abstract elements, UML is used beyond the IT domain. On the next page you can see the UML diagrams generated with IntelliJ.

A picture containing graphical user interface

Description automatically generated

**A picture containing graphical user interface

Description automatically generated**

**A picture containing graphical user interface

Description automatically generated**

In the UML diagram above are presented all the classes, with the attributes and the relations between them, the main classes are:

**MVC type project:**

**Model:**

The “Polynomial” class where the classes “getMonomials” which converts a string to a list of monomials and “toString” which transforms a polynomial in a string for it to be displayed.

The “Operations” class which contains the addition, subtraction, derivation, and integration functions.

The “InputMismatchException” class which throws an exception every time the input received in the text fields is wrongly typed.

The “Monomial” class which contains the getters and setters that are used to access the degree and coefficient of a monomial.

**View:**

Class "PolyCalculatorView", where the main is also present, consists of JPanel, JTextField and JButton, with the help of which the user can call each function in the program and display their results. At each button, the two polynomials are pressed as Strings, the operation is performed, and the result is returned as String.

**Controller:**

Here, the class “Controller” is implemented. It contains the creation of the following buttons: addition, subtraction, derivation, integration buttons; the empty field buttons for each text field present and the exit button.

**Junit Test:**

The “AppTest” class is a Test class implemented with Junit that checks each operation to see if it is correct or not, using a few examples for testing. If any operation is correct, it will display the error message.

* **Package Diagram**

The packages described above are represented below. All of this is part of the “main” package.

Diagram

Description automatically generated

Another package, that contains the “AppTest” for Junit Testing, is “Test”.

Diagram

Description automatically generated

1. **Implementation**

**Data structures**

The following data structures were chosen:

* Monomial - the class that contains 2 parameters of type “double” and “int” (coefficient, degree) corresponding to the values of each monomial.
* Polynomial - the class that contains an ArrayList of monomials, playing the role as close as possible to the abstraction of the polynomial.

**Algorithms**

In the following, all the algorithms used for operations on polynomials will be presented, namely:

1. Addition of polynomials

This algorithm is implemented in the "Operations" class and works on a relatively simple principle. The 2 polynomials are stored in 2 separate lists and their assembly is done in a loop using foreach, thus, the assembly is done term by term checking at each step the equal exponents.

This method will create a new polynomial with the monomial list parameter and will eventually return it. A boolean variable will be used to add the monomial which appears only once during the iteration of the two polynomials.

1. Subtraction of polynomials

The major difference between the addition and subtraction method is that instead of adding, we will subtract at each step, and through a forum, we will go through the list term by term, and we will address each coefficient in the second polynomial a “ - “ in order to reduce them. During the boolean variable procedure used above for addition, we multiply the monomials which appear once only in the second polynomial with -1 to convert the “+” sign to a “-“.

1. Derivation of a polynomial

This algorithm is implemented in the "Operations" class, along with the methods responsible for addition, subtraction operations. The " derivation" method that implements it receives this time the list of monomials on which it works. The polynomial is stored in a list where the effective derivation will be made. Of the five methods, this is the easiest to implement since only the coefficient and the exponent must be modified for each monomial. Specifically, derivation must be applied, and we only need one if and another implemented in a loop throughout the list, namely:

After this, a list of new coefficients and exponents resulting from each Monomial will be created and it will be concentrated thus creating a new polynomial having as parameter the list of monomials, it will create it and finally return.

1. Integration of polynomials

This last algorithm present in the "Operations" class together with the above works on the same principle as the derivation having only some changed conditions, besides the fact that the result will have to be with real coefficients.

The "integration" method that implements it receives this time the list of monomials on which it works. The polynomial is stored in a list where the effective integration will take place.

Specifically, integration must be applied and we only need one if and another implemented in a loop throughout the list.

1. “getMonomials” method

This method implements in the “Polynomial” class will get a string as an input from the text fields and will replace all “-“ signs with “+-“. After that, using a regex, it will split the string into a string array which will contain each monomial. Going through every monomial, the degree and coefficient is determined through various conditions. The method will return nothing when the coefficient is zero, “x” when the coefficient and degree are equal to 1, a constant when the coefficient is not 0 and the degree is 0. The rest are added to the list of monomials as normal.

1. “toString” method

This method goes through every case from below, but the reverse is done. Thus, the function will convert a polynomial to a string for it to be displayed in the result text field after the operation buttons are pressed.

**Graphical User Interface**

The Graphical User Interface is a user interface based on a display system that uses graphics. The graphical interface is called the one-screen graphical display system, which is functionally located between the user and electronic devices. We use a User-Friendly graphical interface to be able to use this computer of polynomials and non-specialized people.

Graphical user interface

Description automatically generated

The familiar was used to create the interface of the current application Java Swing. The GUI is built of simple and meaningful elements that induce the utility and method of using the application.

Thus, in the composition of the GUI we have:

* 3 text fields corresponding to first and second polynomial and the result of the operations;
* 4 buttons that perform the operation specific to their name;
* 3 buttons that perform the empty instruction for their corresponding text field;
* 1 exit button.

1. **Results**

Using JUnit Testing, it is clear to see that the methods implemented are correct.

**Addition & Subtraction:**

Graphical user interface, application, Word

Description automatically generated Graphical user interface, application, Word

Description automatically generated

**Derivation & Integration:**

**Graphical user interface, application, Word

Description automatically generated Graphical user interface, application, Word

Description automatically generated**

1. **Conclusions**

In my opinion, this application can be used by anyone who will need to calculate an operation on polynomials efficiently and quickly.

Further developments could be implemented: multiplication and division buttons, and a remainder text field for the division operation. A sorting function should be implemented for the polynomials to be displayed in descending order according to the degree of the monomial.

Also, at the interface level, for the user to have the pleasure to interact with it, I will introduce new aspects, diagrams, drawings, buttons, maybe something dynamic to help the user in this way.

Being the first project of this size, with all the complex classes, with diagrams, with tests, with the interface, I can honestly say that it helped me in developing my programming skills both at the level of thinking and at the level of data structuring.

1. **Bibliography**

* <https://stackoverflow.com/>
* <https://dsrl.eu/courses/pt/materials/A1_Support_Presentation.pdf>
* <https://docs.oracle.com/javase/tutorial/uiswing/>
* <https://www.vogella.com/tutorials/JUnit/article.html>
* <https://www.baeldung.com/junit-5>
* <https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html>
* <http://tutorials.jenkov.com/java-regex/pattern.html>
* <https://www.geeksforgeeks.org/>
* <https://en.wikipedia.org/wiki/Use_case>
* <https://www.vogella.com/tutorials/JavaRegularExpressions/article.html>
* <https://www.tutorialspoint.com/java/java_regular_expressions.htm>
* <https://google.github.io/styleguide/javaguide.html>
* <https://www.lucidchart.com/>
* Courses, laboratories, homework, and OOP project from year 2, semester 1.