DOCUMENTATION

Assignment\_1

***POLYNOMIAL CALCULATOR***

Vidrean Marius-Iuliu

30422

Table of contents

1. Assignment Objective
2. Problem analysis, modelling, use cases
3. Design
4. Implementation
5. Results
6. Conclusions
7. Bibliography

**1.ASSIGNMENT OBJECTIVE**

The goal of this assignment is to familiarize ourselves with operations on polynomials, and it is also a way to remind us of Java programming. Each operation (addition, subtraction, multiplication, division, derivation and integration) had to be implemented in Java code, exemplified and tested.

**2. PROBLEM ANALYSIS, MODELLING, USE CASES.**

A polynomial is an expression that can be built from constants and symbols called indeterminates or variables by means of addition, multiplication and exponentiation to a non-negative integer power.

The topic of polynomial operations is a complex process that requires attention at every step, both in the implementation of each operation and in testing as many examples as possible in order to verify them.

Such an approach is necessary due to the fact that each operation has its exceptions and must be treated individually in order for the program to work in this impeccable condition.

In order to implement the problem we need to see what data we need, how we want to receive input data, how to save polynomials, how to create a structure inside the software with which we can easily maneuver them for required operations, how to perform operations for use functions that are as easy as possible and use less storage space. We also have to consider creating an interface that is as practical and easy to use as possible, in order for anybody to be able to use it.

The polynomial consists of a list of certain terms, also called monomials, for example 3x^2  is a monomial. The coefficient is 3, the indeterminate is x and the degree is 2. Forming a sum of several terms produces a polynomial, like the following one: 3x^2-5x+4, which has three terms with different exponents:  the first is degree two, the second is degree one, and the third is degree zero.

Functional applications:

- The polynomial computer should allow users to insert polynomials

- The polynomial computer should allow users to select the mathematical operation

- The polynomial computer should allow users to add two polynomials

- The polynomial computer should allow users to subtract two polynomials

- The polynomial computer should allow users to multiply two polynomials

- The polynomial computer should allow users to divide two polynomials

- The polynomial computer should allow users to derive a polynomial

- The polynomial computer should allow users to integrate a polynomial

- The polynomial calculator should allow users to display the result

Non-functional applications:

- The polynomial computer should be intuitive and easy to use for the user

- The polynomial calculator should also be able to be used by non-professionals

- The polynomial calculator should also be a clean work environment

- A distinction must be made between control and the graphical interface of an activity

- The polynomial calculator should be clear: The function of each element of the user interface should be clear to the person using it.

- The polynomial calculator should be consistent: Articles should be consistent across the product so that users can recognize patterns.

- The polynomial computer should be effective: Minimum user input is required to get the desired result.

The use case presents the actor, which in our case is the user that interacts with the application. She/ He can perform several actions on the two chosen polynomials, such as addition, subtraction, multiplication, division, integration and differentiation.

Diagram

Description automatically generated

Use cases:

Considering the fact that 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 5 buttons drawn with the icon suggestive of the operations, the result being displayed in each text field dedicated to the operations performed. The use cases are strongly connected with the user steps. This is the reason why I tried to design my interface in a very friendly mode and the result is the following:

s m n o r l m n t r s f h r u j m l o r q a e y u t m n y v g f d r u k m u n b t v c d f t

Graphical user interface, application

Description automatically generated

**3.DESIGN**

Graphical user interface, text, application

Description automatically generatedA m h b c f r y o p r w a r l m n t r s m l u r f r c v b n q w r t m n j t s c r t o m l a

The general assembly of the project consists of:

- 2 inputs representing the two polynomials to be processed

- Operation selected by the user

- The polynomial calculator itself

- The result that consists of a third polynomial

THE 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.

Graphical user interface

Description automatically generated

The UML diagram above shows all the classes, with their attributes and relationships.

**4. IMPLEMENTATION:**

The MVC pattern has been heralded by many developers as a useful pattern for the reuse of object code and a pattern that allows them to significantly reduce the time it takes to develop applications with user interfaces.

The model-view-controller pattern proposes three main components or objects to be used in software development:

- *Model*, which represents the underlying, logical structure of data in a software application and the high-level class associated with it. This object model does not contain any information about the user interface.

- *View*, which is a collection of classes representing the elements in the user interface (all of the things the user can see and respond to on the screen, such as buttons, display boxes, and so forth)

- *Controller,* which represents the classes connecting the model and the view, and is used to communicate between classes in the model and view.

I have a MVC implementation design:

MODEL :

“Operation” class where all operations on polynomials are stored: addition, subtraction, multiplication, derivation and integration.

“Polinom” class where I have a list of Monom type. A “toString” method which helps me to convert a list of monomials into a string and display it in a text field. Here I also have a method named “parsing” which converts the input that I get from the user, into a polynomial that I can process. I also have a “sortPolinom” method, which as its name says, basically sorts a polynom.

The "Monom" class is the class where each monomial is stored after the split, here are stored the coefficients, exponent.

VIEW:

The “MainView” class is where I define all the buttons and also make all the getters for the buttons, textfields etc.

CONTROLLER:

The Basic Operation and Advanced Operation classes have a close connection because both include the call of the functions that perform operations on polynomials depending on the command received from the outside (pressing the button corresponding to the operation). The Advanced Operation class includes performing the integration and derivation of a polynomial as well as creating it as an output string. In the Elementary Operation class the other operations are performed and the direct call link of the control functions is performed and their result is displayed in the application GUI.

JOINT TEST:

The "OperationTest" 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.

DATA STRUCTURES USED:

The most common structures used in this project are the ArrayLists, used for monomials "Monom" in order to have real coefficients, it is also necessary to enter and remove from the list throughout the program coefficients, strings and exponents. Lists give us a much more efficient and complex data handling capability, especially in relation to array structures. It is no longer necessary to count, search for a certain element using an index, but through the lists we will access at each step the monomial objects "Monom" which is much more efficient and useful.

RELATIONS:

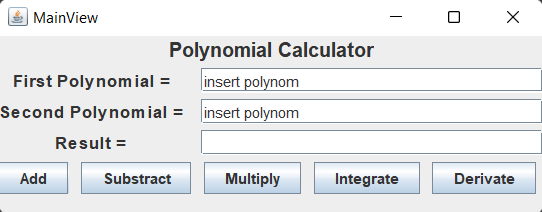
The classes work as a whole, so if one is missing the program will not work properly, it is not an independent project, but dependent, dependent on the implementation of each class and the relationships between them. Relationships can be easily seen in the UML diagram, both 1: n and 1: 1 relationships.

INTERFACE:

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

To make the graphical interface of this Polynomial Calculator I used Swing. And its builder, through which we had the opportunity to create a custom window.

The "MainView" class creates the graphical interface used throughout the project. Determine the size of the window, the title and other aspects related to it. The user interface is as simple as possible, being an application for calculating operations on polynomials, we introduced only the strictly necessary, namely: the 5 buttons for operations, in addition to each button we have a JField where the result of each will be displayed pressed operations, and 2 JText with "First Polynomial" and "Second Polynomial" respectively the 2 JField where the polynomials will be introduced.

A m n s f t r g n b h r y g h t r f b n v c f r e d s g b j k m u y b g t y h n j b f t y v

**5. RESULTS:**

Unit testing is the testing of a small logic or code to verify that the output of the code is as expected when specific data is entered and / or certain conditions are met.

JUnit is an open-source framework that is used to write and run unit tests in the Java programming language. It is one of the best known unit testing framework.

JUnit The "OperationTest" test class includes tests to verify the validity of each operation applied to the two polynomials.

The result is shown below, where we see that the two strings match. Therefore, the JUnit test is passed.

T f m k g y m n o m n u n l m n g n m n m n m n m n m n m n m n m n m n m n

**Graphical user interface, text, application

Description automatically generated with medium confidence**

**6.CONCLUSIONS:**

In conclusion, I believe that this project has deepened my knowledge of the Java language, the implementation of OOP paradigms and the creation of a program with a proper graphical interface.

As further developments we can mention:

- New operations can be added such as finding the roots of a polynomial.

- New operations can be added such as higher order derivatives

- New operations can be added such as the application of the Laplace transform

- Calculate the value of the polynomial at a certain point

- Graphs corresponding to the two polynomials

**7.BIBLIOGRAPHY:**

-YouTube: https://www.youtube.com/

-StackOverflow: https://www.stackoverflow.com/