**DOCUMENTATION**

**HOMEWORK 1**

STUDENT NAME: Sandu Bogdan-Emilian

GROUP: 30423

**TABLE OF CONTENTS**

1. Homework objective……………………………………………..
2. Problem analysis, modeling, scenarios, utilization cases........…..
3. Design……………………………………………………………
4. Implementation…………………………………………………..
5. Results……………………………………………………………
6. Conclusions………………………………………………………
7. Bibliography……………………………………………………..
8. **Homework objective**

The main objective of this homework is designing and implementing a functional polynomial calculator in Java using OOP concepts. To accomplish this some secondary objectives had to be achieved first.

Secondary objectives:

* Designing and implementing a functional GUI
* Implementing polynomial operations (addition, subtraction, division, multiplication, derivation, integration)

1. **Problem analysis, modeling, scenarios, utilization cases**

The application approaches different operations that can be performed on polynomials. It must correctly process the input polynomials represented as string into data models contained in the application, it must perform correctly the following operations: add, subtract, multiply, divide, derivate and integrate, it must correctly format the internal data into readable string to the user and it must respond with adequate error messages.

Diagram

Description automatically generated

**Use Case: add polynomials**

**Primary Actor: User**

**Success Scenario Steps:**

1. The user introduces the polynomials
2. The user presses the “Add(+)” button
3. The calculator check if the polynomials are valid
4. The calculator performs the addition
5. The result is displayed on the screen
6. The User receives the result on the screen

**Alternative Sequences:**

If the polynomials are not valid, an adequate message is returned to the user.

**Use Case: subtract polynomials**

**Primary Actor: User**

**Success Scenario Steps:**

1. The user introduces the polynomials
2. The user presses the “Subtract(-)” button
3. The calculator check if the polynomials are valid
4. The calculator performs the subtraction
5. The result is displayed on the screen
6. The User receives the result on the screen

**Alternative Sequences:**

If the polynomials are not valid, an adequate message is returned to the user.

**Use Case: multiply polynomials**

**Primary Actor: User**

**Success Scenario Steps:**

1. The user introduces the polynomials
2. The user presses the “Multiply(\*)” button
3. The calculator check if the polynomials are valid
4. The calculator performs the multiplication
5. The terms of the polynomial are grouped
6. The result is displayed on the screen
7. The User receives the result on the screen

**Alternative Sequences:**

If the polynomials are not valid, an adequate message is returned to the user.

**Use Case: divide polynomials**

**Primary Actor: User**

**Success Scenario Steps:**

1. The user introduces the polynomials
2. The user presses the “Divide(/)” button
3. The calculator check if the polynomials are valid
4. The calculator checks if the first polynomial degree is bigger or equal to the second
5. The calculator performs the division
6. The result is displayed on the screen
7. The User receives the result on the screen

**Alternative Sequences:**

If the polynomials are not valid or one is 0, an adequate message is returned to the user.

**Use Case: derive polynomial**

**Primary Actor: User**

**Success Scenario Steps:**

1. The user introduces the polynomial in the first field
2. The user presses the “Derive first polynomial” button
3. The calculator check if the polynomial is valid
4. The calculator performs the derivation
5. The result is displayed on the screen
6. The User receives the result on the screen

**Alternative Sequences:**

If the polynomial is not valid, an adequate message is returned to the user.

**Use Case: integrate polynomial**

**Primary Actor: User**

**Success Scenario Steps:**

1. The user introduces the polynomial in the first field
2. The user presses the “Integrate first polynomial” button
3. The calculator check if the polynomial is valid
4. The calculator performs the integration
5. The result is displayed on the screen
6. The User receives the result on the screen

**Alternative Sequences:**

If the polynomial is not valid, an adequate message is returned to the user.

1. **Design**

The design was realized with OOP concepts in mind. In order to input correct polynomials, each term MUST have a sign.

The project contains the following packages:

* gui: contains classes related to the graphical user interface and the controller
* ControllerCalc
* ViewCalc
* logic: contains a class which contains operations related to polynomials: add, subtract, multiply, divide, integrate, derivate. For the division of two polynomials the algorithm Long Division was used

Long Division: is an algorithm for dividing a polynomial by another polynomial of the same or lower degree, a generalized version of the familiar arithmetic technique called long division. It can be easily done by hand, because it separates an otherwise complex division problem into smaller ones.

1. Divide the first term of the dividend by the highest term of the divisor
2. Multiply the divisor by the result just obtained
3. Subtract the product just obtained from the appropriate terms of the original dividend
4. Repeat the previous three steps considering the remainder as the new dividend of the division, until the degree of the remainder is lower than the dividend

* Operations
* model: contains classes related to the data used to model the problem
* Monomial
* Polynomial
* utils: contains a class which contains static methods used for data formatting, data processing, different small operations performed on the data and some other useful methods, it also contains a checked exception class
* PolyUtil
* PolyFormatException

1. **Implementation**

All the classes were created with the concept of data encapsulation in mind, therefore all the classes attributes are private and getters and setters are used.

* Monomial: is a class used for modeling the data used in the application. It implements the Comparable interface. It has the following attributes: coefficient (type Number) and power (type int).
* Monomial(Number coefficient, int power, char sign)

Is a constructer used to create a new Monomial during the processing of the input string representing a polynomial. It receives a coefficient (which is always a positive integer), a power and the sign of the monomial in order to give the correct sign to the coefficient.

* equals

It is an overridden method used in order to correctly create a Set of Monomials used in other methods (HashSet was used as a Set).

* hashCode

It is an overridden method used in order to correctly create a set of Monomials used in other methods (HashSet was used as a Set).

* compareTo

It is an overridden method in order to correctly sort the list of Monomials each time an operation which might alter their correct order is used.

* Polynomial: is a class used for modeling the data used in the application. It has only one attribute, an ArrayList of the Monomials it contains.
* addMonomial

It is a method used to add a Monomial to the list of Monomials

* addMonomials

It is a method used to add multiple Monomials to the list of Monomials

* removeMonomial

It is a method used to remove a Monomial from the list of Monomials

* removeMonomials

It is a method used to remove multiple Monomials from the list of Monomials

* Polynomial(String inputPoly) throws PolyFormatException

It creates a new Polynomial from the input string. It uses two static methods from PolyUtil class, polyExtractor and polyCombine. After the polynomial is created it is sorted in a reverse order.

* equals

It is an overridden method used in order to correctly sort the list of monomials

* PolyFormatException: is an exception class which is thrown when the format of the input string representing a polynomial is incorrect.
* PolyUtil: is a utility class which contains multiple static methods. It has no attributes.
* polyExtractor

It is a method used to correctly extract a list of monomials representing the input string. If the input string is not a correct polynomial an exception is thrown. It makes use of regex expressions in order to match and extract monomials. Two regex expressions are used, the first one is for finding monomials in different forms from the input string and the second is used for extracting the coefficient, power and sign of the monomial in order to create a new Monomial object.

* monoFormatter

It is a method used to correctly convert a Monomial object into a string representing it

* polyFormatter

It is a method which uses monoFormatter in order to convert a Polynomial object into a string representing it

* polyCombine

It is a method which combines all the monomials of the same power contained in a Polynomial, it is very important for input processing, multiplication and other methods.

* deriveMono

It is a method which receives a Monomial and derives it, returning a new Monomial representing the derived one, it is used in the derivation of a polynomial operation.

* integrateMono

It is a method which receives a Monomial and integrates it, returning a new Monomial representing the integrated one, it is used in the integration of a polynomial operation.

* divideMono

It is a method which receives two monomials and divides them, returning a new Monomial representing the result of the division, it is used in the division of two polynomials operation.

* Operation: is a class used for performing different operations on polynomials. It has no attributes.
* addP

It is a method used for adding two polynomials and it returns a new Polynomial representing their sum.

* substractP

It is a method used for subtracting two polynomials and it returns a new Polynomial representing their difference.

* multiplyP

It is a method used for multiplying two polynomials, the Monomials with the same power in the resulting Polynomial are then added together using the polyCombine method from the PolyUtil class.

* divideP

It is a method used for dividing two polynomials, if one of the terms is null or if the first Polynomial’s degree isn’t bigger than the second’s null is returned. Otherwise the Long Division algorithm is used to divide the two polynomials. It returns an array of two Polynomials, the first one representing the Quotient and the second representing the Remainder.

* deriveP

It is a method used to derive the first Polynomial, returning a new Polynomial representing the result.

* integrateP

It is a methodused to integrate the first Polynomial, returning a new Polynomial representing the result.

* ViewCalc: it is a class used for implementing the GUI with Java Swing. It contains different graphic elements. Attributes: resultPanel, resultLabel, resultValueLabel, contentPanel, polyPanel, firstPolyLabel, firstPolyTextField, secondPolyLabel, secondPolyTextField, multiplication, substraction, division, derive, addition, integrate. All the attributes and methods and private.
* preparePolyPanel

This method is used to place the general bulk of components. It contains buttons for each operation, each button has a representative action command and it has an action listener. Each button also has a representative name and it represents a different operation.

Two text fields are used to receive the user input, each text field also has a representative label. At first the two text fields contain some information about the correct format for the input polynomials. The components are placed in a grid layout (6 by 2) in a pleasing visual order which makes the user experience intuitive. The panel in which these components are placed is being called polyPanel.

* prepareResultPanel

This method is used to place result related components. The panel in which these components are placed is called resultPanel and it has a grid layout (1 by 2). It contains two label, the first one indicating the panel role and the second the actual result of the operations performed.

* prepareGUI

Diagram

Description automatically generatedGraphical user interface

Description automatically generatedThis method is used to create the main panel and place inside it the two previous panels. It uses a grid layout (2 by 1). It also calls the two previous metho

1. **Results**

Initially the testing was done using input given to the GUI and after each newly implemented operations tests were written in order to quickly verify each scenario that could go wrong. This proved to be very useful to check for correctness after changes were made in different areas of the code. Some tests had to be rewritten because of changes in input formatting and processing.

All the operations implemented were testes thoroughly and also the input formatting and processing part. Parameterized tests were used mainly.

Note: one test doesn’t work because of floating point rounding error and I didn’t find a fix for it. (ex: 0.3333333 is expected but it receives 0.33333334).

1. **Conclusions**

Implementing the operations wasn’t the most difficult part of the project, the single difficult operation to implement was the multiplication because I had to group the terms after multiplying them. At first glance division seemed difficult but after implementing the rest of the operations it was quite easy. Extracting the Monomials and creating correctly the Polynomials was difficult for me because I didn’t work priorly with regex expressions. I’ve also learned how to work with Maven and Junit which at first was a bit challenging.

The GUI could be improved a lot and be made more appealing to the eye. Also, a “RESET POLYNOMIALS” button could be added. The app could also be made to work for Polynomials with real coefficients, which would be a minor change in the input formatting part.

Main aspects learned during the making of this project: how to work with regex expressions, a better understanding of the MVC model, gained experience working with Java Swing and Maven.

1. **Bibliography**

* <https://docs.oracle.com/javase/tutorial/uiswing/>
* <https://www.baeldung.com/junit-5>
* <https://www.vogella.com/tutorials/JUnit/article.html>
* <https://www.geeksforgeeks.org/multiply-two-polynomials-2/>
* <https://en.wikipedia.org/wiki/Polynomial_long_division#:~:text=In%20algebra%2C%20polynomial%20long%20division,division%20problem%20into%20smaller%20ones>.
* <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>
* Lots of stackoverflow threads

Diagram

Description automatically generated