**DOCUMENTATION**

ASSIGNMENT NUMBER 1

STUDENT NAME: VOIVOD OVIDIU – FLORIN

GROUP: 30423

**CONTENT**

1. Assignment’s objective ………………………………………………………………………………………3
2. Problem analysis, modeling, scenarios, use cases ……………………………………………………………3
3. Design ………………………………………………………………………………………………………..7
4. Implementation ………………………………………………………………………………………………9
5. Results ………………………………………………………………………………………………………10
6. Conclusion ……………………………………………………………………………………………….…11
7. Bibliography ………………………………………………………………………………………………..11
8. **Main purpose of the project**

**1.1 Main objective**

The main purpose of the project is to implement a polynomial calculator with a GUI (graphical user interface) which would let the user to insert polynomials and do some mathematical operations on those input polynomials. The application performs mathematical operations on polynomials and provides the result of the operations to the user.

**1.2 Sub - objectives**

* Analyze the problem and identify the requirements and implementation
* Design the solution for the polynomial calculator
* Implement the functionalities and the GUI for the polynomial calculator
* Test the functionalities and assure the correctness and good functioning of the application

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

**2.1 Functional requirements**

* The polynomial allows the user to insert polynomials as strings
* The polynomial calculator allows the user to select the mathematical operation which he wants to perform
* The polynomial calculator should add, subtract, multiply or divide two polynomials, and show the result of the operation
* The polynomial calculator should integrate or give the result of the derivative for one polynomial
* The polynomial calculator should allow user to reset the input text fields faster
* The polynomial calculator should provide user an easy way to exit the application

Diagram

Description automatically generated

**2.2 Use cases**

**Use Case:**  add polynomials

**Primary Actor:** user

**Main Success Scenario:**

1. The user inserts two polynomials in the graphical user interface.
2. The user clicks the “Add” button
3. The application performs the addition operation between the 2 polynomials and displays the result in a label field (First Polynomial + Second Polynomial = Result)

**Alternative Sequence:** Incorrect polynomials with variables different than “x”

* The user inserts incorrect polynomials with variables different than “x”
* The GUI provides an “Incorrect input” popup
* The scenario returns to step 1

**Alternative Sequence:** Incorrect polynomials with characters that are not allowed

* The user inserts incorrect polynomials with characters different than numbers, “x” variable or “^,\*,+,-” special characters
* The GUI provides an “Incorrect input” popup
* The scenario returns to step 1.

**Use Case:**  subtract polynomials

**Primary Actor:** user

**Main Success Scenario:**

1. The user inserts two polynomials in the graphical user interface.
2. The user clicks the “Subtract” button
3. The application performs the subtract operation between the 2 polynomials and displays the result in a label field (First Polynomial – Second Polynomial = Result)

**Alternative Sequence:** Incorrect polynomials with variables different than “x”

* The user inserts incorrect polynomials with variables different than “x”
* The GUI provides an “Incorrect input” popup
* The scenario returns to step 4

**Alternative Sequence:** Incorrect polynomials with characters that are not allowed

* The user inserts incorrect polynomials with characters different than numbers, “x” variable or “^,\*,+,-” special characters
* The GUI provides an “Incorrect input” popup
* The scenario returns to step 4.

**Use Case:**  multiply polynomials

**Primary Actor:** user

**Main Success Scenario:**

1. The user inserts two polynomials in the graphical user interface.
2. The user clicks the “Multiply” button
3. The application performs the multiplication operation between the 2 polynomials and displays the result in a label field (First Polynomial \* Second Polynomial = Result)

**Alternative Sequence:** Incorrect polynomials with variables different than “x”

* The user inserts incorrect polynomials with variables different than “x”
* The GUI provides an “Incorrect input” popup
* The scenario returns to step 7

**Alternative Sequence:** Incorrect polynomials with characters that are not allowed

* The user inserts incorrect polynomials with characters different than numbers, “x” variable or “^,\*,+,-” special characters
* The GUI provides an “Incorrect input” popup
* The scenario returns to step 7.

**Use Case:**  divide polynomials

**Primary Actor:** user

**Main Success Scenario:**

1. The user inserts two polynomials in the graphical user interface.
2. The user clicks the “Divide” button
3. The application performs the division operation between the 2 polynomials and displays the Quotient in a label field and Remainder in another label field (First Polynomial / Second Polynomial = Quotient (first label field), = Remainder (second label field) )

**Alternative Sequence:** Incorrect polynomials with variables different than “x”

* The user inserts incorrect polynomials with variables different than “x”
* The GUI provides an “Incorrect input” popup
* The scenario returns to step 10

**Alternative Sequence:** Incorrect polynomials with characters that are not allowed

* The user inserts incorrect polynomials with characters different than numbers, “x” variable or “^,\*,+,-” special characters
* The GUI provides an “Incorrect input” popup
* The scenario returns to step 10.

**Use Case:**  integrate polynomial

**Primary Actor:** user

**Main Success Scenario:**

1. The user inserts one polynomial in the graphical user interface, in the First Polynomial text field.
2. The user clicks the “Integrate” button
3. The application performs the integration operation of the polynomial and displays the result in a label field ( ∫ (First Polynomial) dx = Result)

**Alternative Sequence:** Incorrect polynomials with variables different than “x”

* The user inserts incorrect polynomials with variables different than “x”
* The GUI provides an “Incorrect input” popup
* The scenario returns to step 13

**Alternative Sequence:** Incorrect polynomials with characters that are not allowed

* The user inserts incorrect polynomials with characters different than numbers, “x” variable or “^,\*,+,-” special characters
* The GUI provides an “Incorrect input” popup
* The scenario returns to step 13.

**Use Case:**  derivate polynomial

**Primary Actor:** user

**Main Success Scenario:**

1. The user inserts one polynomial in the graphical user interface, in the First Polynomial text field.
2. The user clicks the “Derivate” button
3. The application performs the derivation operation for the polynomial and displays the result in a label field ( (First Polynomial)’ = Result)

**Alternative Sequence:** Incorrect polynomials with variables different than “x”

* The user inserts incorrect polynomials with variables different than “x”
* The GUI provides an “Incorrect input” popup
* The scenario returns to step 16

**Alternative Sequence:** Incorrect polynomials with characters that are not allowed

* The user inserts incorrect polynomials with characters different than numbers, “x” variable or “^,\*,+,-” special characters
* The GUI provides an “Incorrect input” popup
* The scenario returns to step 16.

**A screenshot of a computer screen

Description automatically generated with medium confidence**

1. **Design**
   1. **OOP Design**

The application is designed based on the MVC design pattern, where the model contains all the functionalities of the program, the view contains all the code which implements the GUI, and the controller makes the linking between functional part and GUI.

* 1. **Class Diagram**
  2. **Package Diagram**

**Graphical user interface, application

Description automatically generated**

* 1. **Data Structures**

Array List is the main data structure used in the Polynomial Calculator application’s implementation. Array List in Java is used to store dynamically sized collection of objects. Compared to Arrays which have fixed sizes, array lists size grows automatically when new elements are stored in it.

Array Lists is a part of Java’ s collection framework which implements List interface. They have almost the same characteristics as the traditional array, with some improvements and features which facilitate a correct and easy implementation of the applications. Array Lists are dynamic arrays which grow or shrink their sizes according to the number of elements stored in the collection. They maintain the insertion order of the elements and allow us to access the stored elements by their index, like in the traditional array.

Java’s Array Lists cannot store primitive types (int, double, char, etc.) but can store the boxed types of the primitive ones.

In conclusion, for small applications with small data structures it is recommended to use lists instead of arrays because they favor the ease of use but not the optimization. So, in our case, for the polynomial calculator, which doesn’t use large amount of data, we can trade up the program efficiency for development time efficiency.

* 1. **Defined interfaces**

The Monomial class of the application implements comparable interface. This interface is used to order the object of the user-defined class. In this application, it is used to facilitate the overriding for the “compareTo” method which was used for comparing monomials based on their power values.

1. **Implementation**

The application implementation part is developed after the **MVC** (**model view controller**) design pattern and it is divided in 3 main packages: Models, Views and Controllers. Every Package mentioned before contains important components which contribute to the good and complete functioning of the application.

**4.1 Model**

Encapsulates the core data and functionality of the application. It contains all the classes and methods which contribute to the functioning of the application.

**4.1.1 Monomial**

Is the object class in which we implement the functionalities of a monomial. A monomial is composed of a coefficient and a power. It can be represented in “a\*x^b” form where a is the coefficient and b is the power. In our project the coefficient will be of Number type and the power of int type. Number is an abstract class which represents numeric values that are convertible to the primitive types (int, double, char, etc.). Int type is used for representing integer numbers.

In the Monomial class I’ve implemented 2 constructors, one with parameters and one without parameters (considered the default constructor).

This class implements comparable interface which will facilitate the overriding of “compareTo” method. This method is used to compare the current monomial with another one based on their power values.

The division method is used to divide the current monomial with another monomial and store the result in the current monomial. The overridden toString method is used to represent the monomial as a string.

**4.1.2 Polynomial**

Is the class for polynomial objects. Here we store the monomials of a polynomial in a list of monomials.

The first constructor of this class takes a string as parameter and using a special regex, with pattern and matcher classes it converts the string into a list of monomials which form the polynomial inputted as a string.

The overridden equals method is used to compare two polynomials. I’ve implemented this method especially for the testing, to compare the given result polynomial with the true result polynomial.

The “toString” method is used to convert the polynomial stored as a list of monomials intro a string, for friendly displaying it to the user.

**4.1.3 Operations**

Operations class is the object class in which I’ve implemented all the mathematical operations done on polynomials, plus the sort method in which we sort the list of monomials of a polynomial in decreasing order, after their monomials power values. All the method implemented in this class are static method. I’ve used this approach because I won’t make any changes to any object from this class during the implementation of the application. I will only use the methods of this class so if they are static methods I don’t have to create a new object of Operations class to do the operations needed. This class has only one attribute which is an instance of PolynomialService class which is static and final because, I won’t make changes to that object, I only use it to perform some “repair” operations on the used polynomials.

The ”addOrSubstractOperation” method takes as parameters two Polynomial objects and an int. The int indicates whether the method should add or subtract the polynomials. If the operation parameter has value zero the method will add the polynomials an return a result polynomial object. If the operation parameter is different than zero the method performs the subtract operation.

The “multiply” method is a static method which takes as parameters two polynomial objects and performs the multiplication term-wise. After that, it verifies whether the result polynomial has redundant terms (terms which have same power and their coefficients can be added) and after that it checks if the polynomial has monomials with zero coefficients. This verification is done by the methods implemented in Polynomial Service.

The “integrate” and “derivate” methods are static methods which iterate through the list of monomials of the input polynomial given as parameter and performs the integration or derivative operation of every monomial. This method returns a result polynomial.

The “division” method takes as parameters two polynomials. It performs the polynomial division algorithm which takes the first monomial of the first polynomial and divides with first monomial of second polynomial. The result is then multiplied with the second monomial and we subtract this result from first polynomial and we repeat this operation on the resulted polynomial after subtraction until this polynomial has the degree smaller than the second polynomial. Division method return a list containing two polynomials, first is the quotient and second is the remainder.

**4.1.4 PolynomialService**

This class has two important methods. First one, “fixRedundantTerms” checks whether the polynomial has monomials of same degree, and if true, it adds their coefficients. Second method, “checkForZeros” checks whether the polynomial has monomials with 0 coefficients, is so, it eliminates those monomials.

**4.1.5 StringValidator**

This class contains only one method which takes a string as parameter, which would be the input polynomial, and determines whether it has variables different than “x” or characters that are not allowed in the input. It return true if the input is correct or throws an exception if not.

**4.2 Controller**

**4.2.1 MainController**

This is the class from controller component of the MVC. Here the linking between the functionality of the program and the GUI is made. In this class we have a constructor in which we implement the linking between functional part and the user interface. Every button, text filed, text label, more precisely, every action that happens on the GUI is controlled in the controller part.

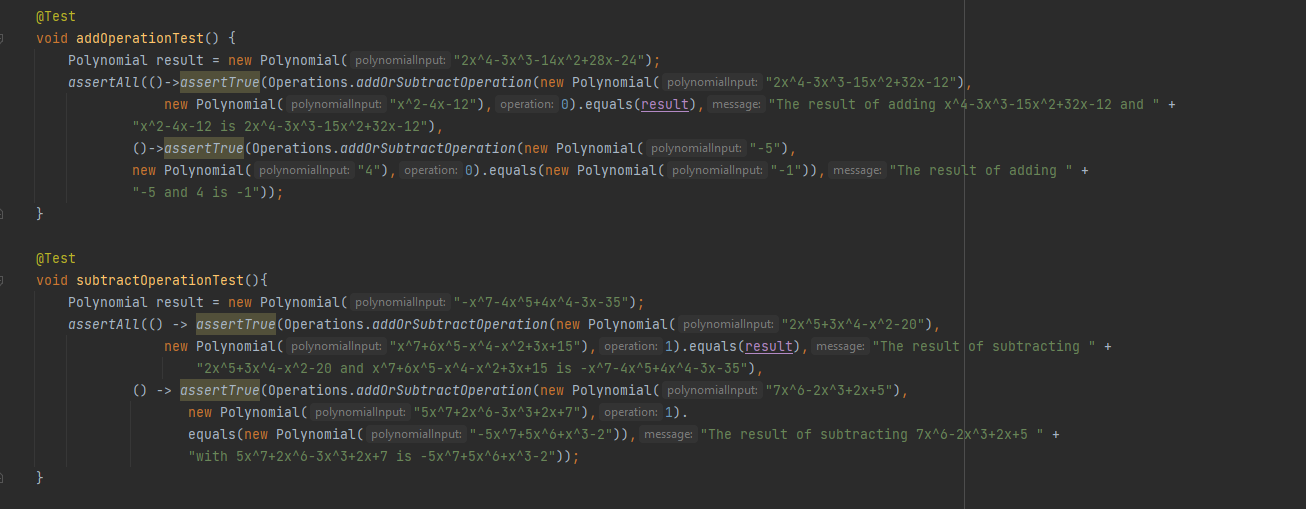
**4.3 Model**

**4.3.1 MainView**

The MainView class has a constructor in which every part of the GUI’s code is implemented. The GUI is implemented in Java SWING architecture. The main layout is a border layout. In the North of the border layout, it is positioned a title “Polynomial Calculator” as a label. In the South part of the main layout, we have a new panel, which is a Grid Layout, with 2 rows and 2 columns on which we will add 2 other grid layouts panels. The polynomial input panel is a grid layout with 4 rows and 2 columns, where we have 6 labels (2 which name the text fields 2 which name the result labels and 2 result labels, one for result and another one for remainder, where is the case) and 2 text fields. In the text fields the user will input the polynomials. The operation buttons panel is another grid layout with 4 rows and 2 columns where we placed the operation buttons along with reset and exit buttons.

**5.Results**

In the testing part, I’ve used Junit testing framework. Here is placed the OperationsTest class in which are placed the methods for testing every operation on the polynomials. In the testing part I’ve used the “assertTrue” function where I perform every operation and compare the result polynomial with the true result polynomial. This function returns true if the results are correct and false if not. Every operation method was provided with 2 test cases. In every case, the application gave the correct result.



A screenshot of a computer

Description automatically generated

1. **Conclusions**

Implementing this application, I’ve learned how to work with MVC design pattern, and how to structure my code using this pattern. I’ve also learned how to make a GUI interface because I’ve never worked with one before. I’ve also learned how to work with Junit testing. In my opinion, making this project helped me a lot in improving my skills in developing applications using Java.

Developing this application taught me that, the final step of implementing an application is the toughest one, because this is when you can find bugs or things that don’t work as expected and you have to modify the code, but modifying it can cause more things to break, so the debugger will be the developer’s best friend in those rough times.

**Future improvements**

As a future improvement I would like to make the string validator more precise, to allow user to input polynomials with single variables different than “x”. I would also like to improve the GUI design. I woulsd also like to add an image recognition feature, which would let the user scan a written polynomial and the application would detect it and take it as an input.

**7.Bibliography**

<https://regex101.com>

<https://stackoverflow.com>

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

<https://www.symbolab.com/solver/polynomial-long-division-calculator>

<https://www.symbolab.com/solver/polynomial-long-division-calculator>

<https://www.javatpoint.com/java-swing>

<https://tutorial.math.lamar.edu/Solutions/Alg/DividingPolynomials/Prob3.aspx>