DOCUMENTATION

ASSIGMENT NUMBER 1

-Programming Techniques-

Polynomial Calculator

NUME STUDENT: Tat Teodora

GROUP: 30423

CONTENT

I. Assigment Objectives

The main objective of this theme is the implementation of an application that performs various operations on some polynomials introduced by the user, such as addition, subtraction, multiplication and division of two polynomials, but also the derivation, respectively their integration.

The implementation of the application was aimed at meeting some secondary objectives, which will be presented in the following chapters, as follows:

- Correct implementation of as many different use cases as possible, starting from the main objective: A use case is a methodology used in the analysis of a problem to identify, clarify and organize the requirements to be implemented by the developer. The use-case consists of a set of possible sequences of interactions between applications and users in a specific environment and related to a specific purpose.
- Organize the code into different classes and packages so that it is as easy to understand as possible: An M-V-C (Model - View - Controller) structure was used to create the GUI graphical interface. Each class contains specific methods, generally implemented in less than 30 lines, to ensure code readability. Classes have less than 300 lines, allowing for better content organization.
- Use of appropriate data structures: Each used data structure will be presented in the class method description.
- Use appropriate algorithms to implement the required functionalities. The algorithms used in the implementation of the methods will be described: Each algorithm will be described in the method's implementation.
- Implementation of the solution: Each class will be presented separately, with the fields and methods specifically used.
- Testing with JUnit, in different scenarios: The scenarios that were addressed in the testing with JUnit will be described.

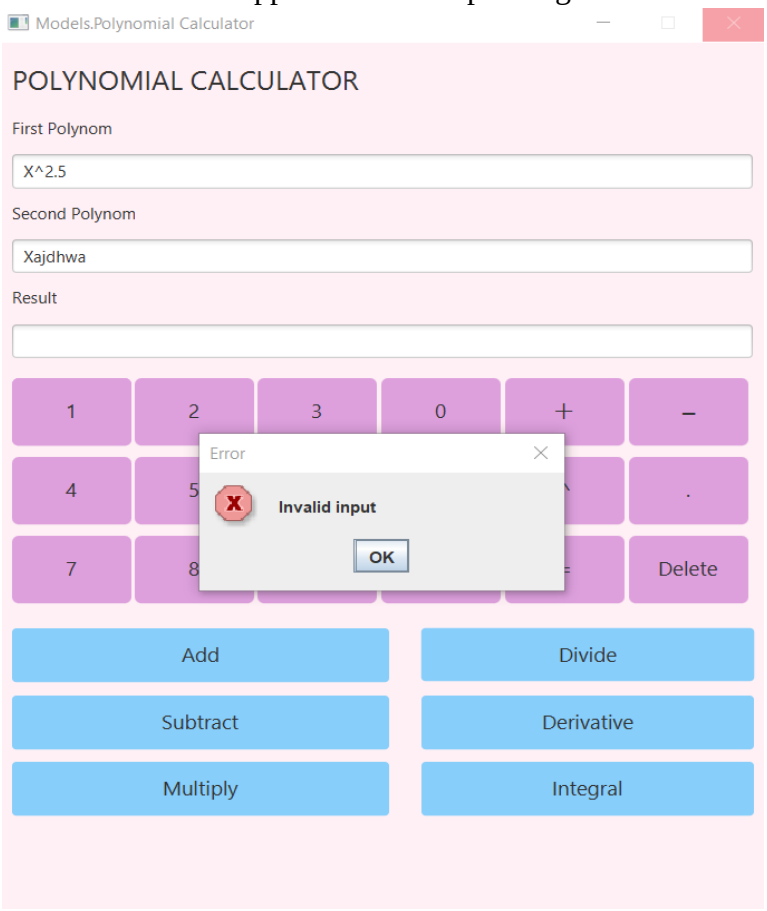
II. Problem analysis, modelling, scenarios, use cases

In the following lines, the formalized functional requirements framework and use cases will be presented, as well as diagrams and descriptions of use-cases. The descriptions of the use-cases will be made in the form of a flow-chart.

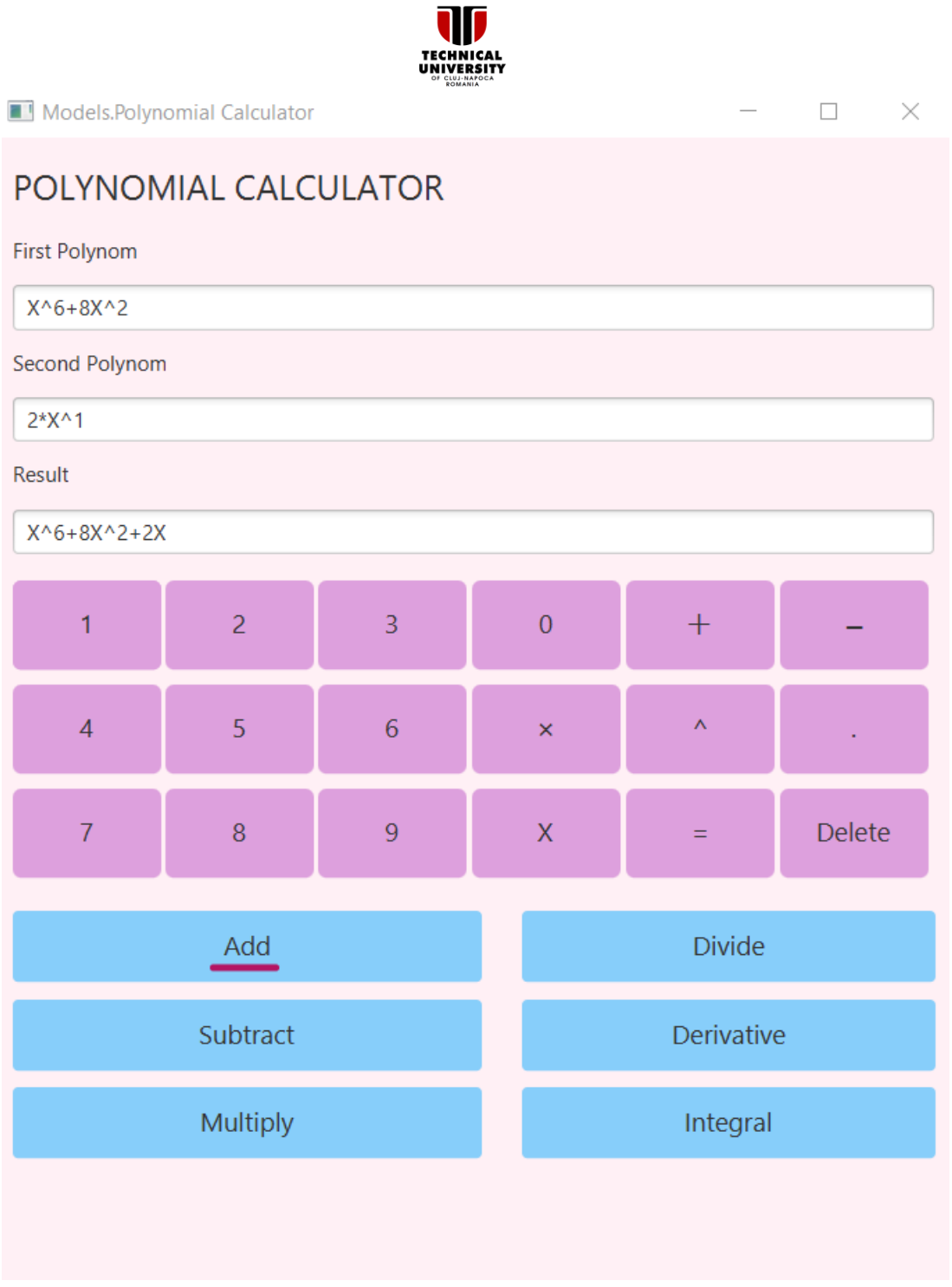
In order for the application to perform operations on polynomials, the user must enter valid polynomials either from the keyboard displayed in the graphical interface or from the keyboard itself. A polynomial is considered valid if, after performing some necessary processing, it can be written as: anXn + an-1Xn-1 + ... + a2X2 + a1X + a0, where an, an-1 , ..., a2, a1, a0 are real numbers, represented by two decimal places, and n is an integer representing the degree of the polynomial.

Therefore, after opening the GUI, the user has the option to enter two polynomials in two TextFields. He will then select from the buttons displayed on the screen the operation he wants to perform. If one of the entered polynomials is invalid, a Label will be displayed next to it with the message "Enter a valid polynomial" (fig. 1), and the user will have to re-enter a polynomial. If the entered polynomials are valid, the result of the selected

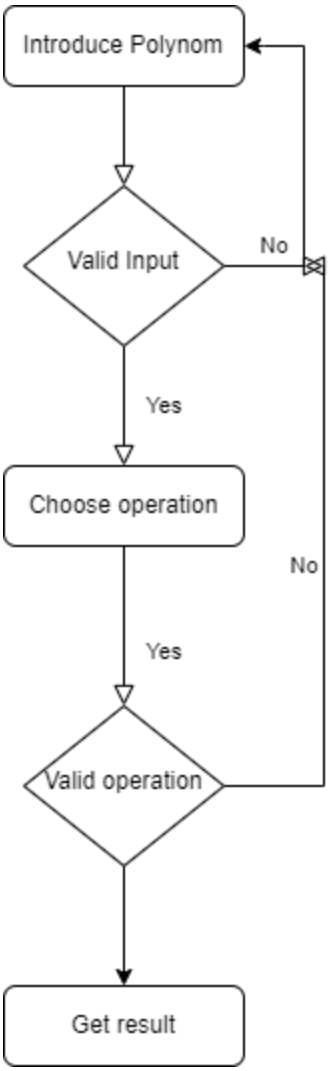
operation will be displayed in the TextField "Result". In the case of derivation or integration operations, respectively, each valid polynomial will be derived, or integrated, and the result will appear in a corresponding TextField.



It can also be seen in Figure 2 that the user can enter a polynomial, not necessarily in the standard form. Thus, even if the powers are not arranged in descending order, the application will be able to recognize if the user has entered a valid polynomial and then process it to bring it back to its usual form. It should be noted that both monomial variants: a * X and aX, where a is a constant, are accepted as valid.


Models.Polynomial Calculator
POLYNOMIAL CALCULATOR
First Polynom
X^6+8X^2
Second Polynom
2*X^1
Result
X^6+8X^2+2X
1
2
3
0
+
–
4
5
6
×
^
.
7
8
9
X
=
Delete
Add
Divide
Subtract
Derivative
Multiply
Integral

The steps the user takes to perform operations on polynomials can be described in the flow chart below:



Use cases

The problem statement lets us identify 6 use cases, namely performing the following polynomial operations: addition, subtraction, multiplication, division, integration, and derivation.

1) Adding Polynomials

Primary Actor: User

Success scenario steps:

1. The user inputs the polynomials
2. The program extracts the polynomial data from the inputs
3. The user presses the "Addition" button
4. The program performs the addition operation
5. The program shows the result

2) Subtracting Polynomials

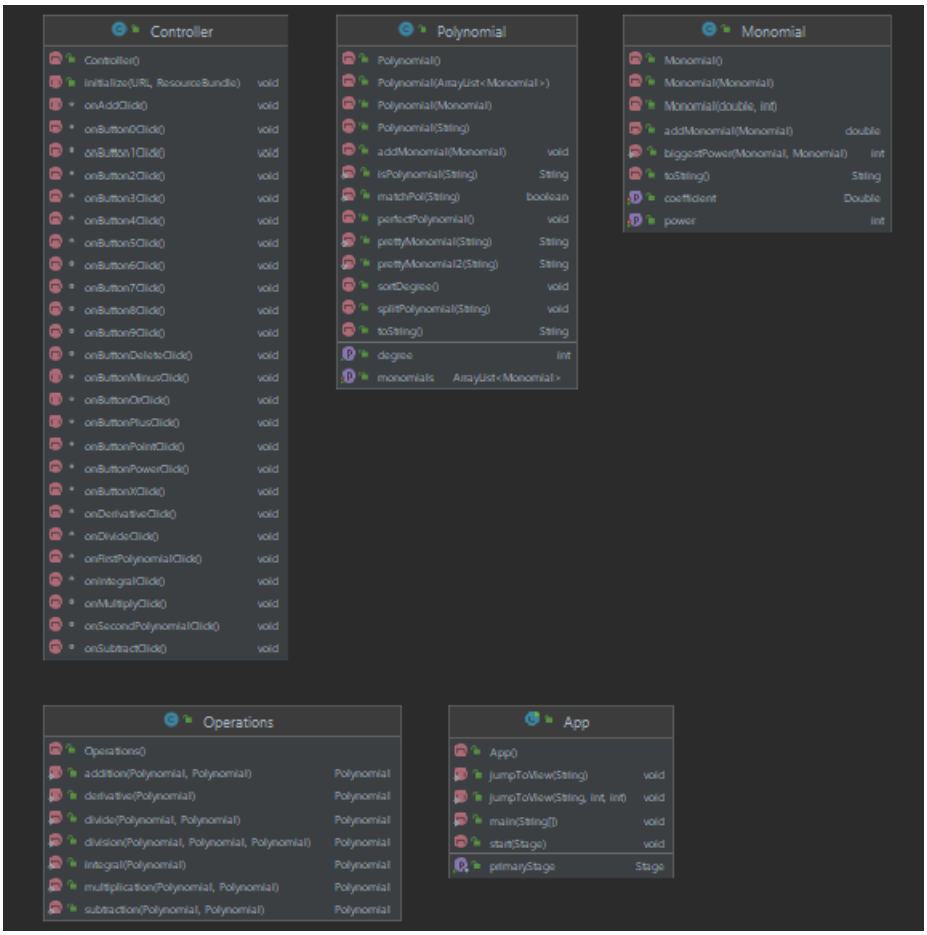
Primary Actor: User
Success scenario steps:
1. The user inputs the polynomials
2. The program extracts the polynomial data from the inputs
3. The user presses the "Subtraction" button
ASSIGNMENT 01 - POLYNOMIAL CALCULATOR 6
4. The program performs the subtraction operation
5. The program shows the result
3) Multiplying Polynomials
Primary Actor: User
Success scenario steps:
1. The user inputs the polynomials
2. The program extracts the polynomial data from the inputs
3. The user presses the "Multiplication" button
4. The program performs the multiplication operation
5. The program shows the result
4) Dividing Polynomials
Primary Actor: User
Success scenario steps:
1. The user inputs the polynomials
2. The program extracts the polynomial data from the inputs
3. The user presses the "Division" button
4. The program performs the division operation
5. The program shows the results
5) Deriving Polynomials
Primary Actor: User
Success scenario steps:
1. The user inputs the polynomials
2. The program extracts the polynomial data from the inputs
3. The user presses the "Derivative" button
4. The program performs the derivation operation
5. The program shows the results
6) Integrating Polynomials
Primary Actor: User
Success scenario steps:
1. The user inputs the polynomials
2. The program extracts the polynomial data from the inputs
3. The user presses the "Integral" button
4. The program performs the integration operation
5. The program shows the results

III. Design

In this chapter, we will present the OOP design of the application, the UML diagram, as well as the data structures and algorithms used.

To create the graphical interface I chose to use the JavaFX tool, and JUnit was used for testing the application.

The application is organized in several different packages, which respect the structure M-V-C (Model - View - Controller). The "Application" package contains the "App" class, which contains the main method, through which the execution of the program starts. The "Model" package is also the most important package of the program, because it contains the basic classes: "Monomial", "Polynomial" and "Operations". The "Controller" package contains the "Controller" class, which is used to link the Model to the View. Also the View is represented by the FXML tab "interface" through which the graphical interface is made. Unit testing is performed in the "test" package, for each operation in a different class.
As data structures, we chose to store the polynomial as an arrayList of monomials, to facilitate its processing in the methods used. This can be easily seen in the UML diagrams below.



Regarding the algorithms used, for the division of polynomials we used the algorithm described by the pseudocode below:

```
function n / d is
    require d ≠ 0
    q ← 0
    r ← n             // At each step n = d × q + r

    while r ≠ 0 and degree(r) ≥ degree(d) do
        t ← lead(r) / lead(d)       // Divide the leading terms
        q ← q + t
        r ← r − t × d

    return (q, r)
```

his algorithm also works if the first polynomial has a lower degree than the polynomial to which it is divided, in which case the quotient has the value 0. The +, -, × signs in the above pseudocode represent the corresponding arithmetic operations performed on the polynomials, while / represents the simple division of two monomials (it was implemented in the Monomial class). After pressing the Divide button, the division result and remainer of the two polynomials will be displayed in the "Result" field.

IV. Implementation

This chapter will describe each class with important fields and methods, as well as the implementation of the user interface.

Monomial Class:
It has the fields "power" and "coefficient" which retain the degree, respectively the coefficient of a monomial.
The "toString ()" method is an Override method that transforms an object of type Monomial into an object of type String that can later be displayed on the screen. Here, several cases are noted. One of the cases is where the coefficient is 0, so the resulting string will be "0". If the coefficient is positive, the "+" sign will be placed on the first position in the string. To prevent polynomials such as "1X" or "X ^ 1" from appearing on the screen, we have also dealt with cases where the coefficient or degree of the monomial is 1, in which case they are no longer written to the returned String.
Polynomial Class:
It contains a list of monomials, but also a constant that represents the maximum degree that the polynomial can have (we chose to be 100, but we can choose any value).
The "prettyMonomial (String polynomial)" method will be used later in the "polynomialAdjust" method. Its role is to write a constant in the form a = + aX ^ 0.
The "prettyMonomial2 (Polynomial String)" method takes as a parameter a String that represents a polynomial written in some form and writes this polynomial in a form that is easier to process. Thus, the polynomial will be written as sums and differences of monomials of the form "aX ^ n", adds the sign "+" in front of every minus for an easier split and removes the "*" sign.
The "isPolynomial" method verifies that the String entered by the user is a valid polynomial, and if so, transforms this String into an object of type Polynomial, by calling the "splitPolynomial"

and "groupMonomials" methods. If the String entered in TextField is invalid, a RunTimeException will be thrown. To implement this method we used regular expressions, ie a template string that describes the set of possible words that can be composed, following certain rules (we used patterns from regex).
The "splitPolynomial" method is the one that actually performs the conversion from a String to a polynomial. First the string is separated at the meeting of the '+' character, in substrings that will represent the monomials of which the polynomial is composed. Thus, the first term in each substring represents the monomial coefficient, while the second term represents its degree.
The "perfectPolynomial ()" method gathers terms with the same degree in a polynomial, using two foreach methods and a bool to verify that the operation was performed.
The "getDegree ()" method returns the highest power in the polynomial (ie its order).
The "sortDegree ()" method sorts the polynomial with monomials in descending order of powers for easier operations.
Operations Class:
Implements arithmetic operations on polynomials. Each operation is implemented in a separate method.
The "addition (...)" method performs the operation of adding two received polynomials as a parameter. For the implementation of this method, we took into account the fact that polynomials have monomials written in descending order. Thus, using the interclassing method, an auxiliary list called newMonomialList, performed the operation of gathering polynomials.
The "subtract (…)" method changes the signs of the coefficients of the second polynomial and then calls the add method.
The "multiplication (...)" method performs the operation of multiplying two received polynomials as a parameter, by means of two for loops.
The "division (..)" method divides two polynomials according to the algorithm described in Chapter 3.
The "derivative ()" and "integral" methods perform the derivation, respectively the integration of a polynomial received as a parameter.
Controller Class
Connects between the methods in the Model and View packages.
This class specifies the action of each button in the GUI, sets the contents of different labels, which can be made visible or invisible depending on the context, sets the text in the TextFields according to the result of the methods called in the Model.
Graphic interface
It was made using JavaFX. Buttons, TextFields, respectively Labels have been declared in the FXML tab called: interface.fxml, which has the role of View for our application.

V. Results and testing

For each addition, subtraction and multiplication operation, we performed 4 tests with JUnit, and for derivation and integration 6 tests each.
We tested the cases in which the polynomials received as input are not written in the classical form, they are subsequently processed by the program. We also tested that the operations work on polynomials of different degrees, as well as for constants, respectively for 0.
In the division operation, we have performed 5 tests but without the remainer shown. We also notice that the division operation works both if the first polynomial has a higher degree than the

second, and in cases where the degree is lower, respectively in the case where the degrees are equal.
In derivation and integration, we notice that both operations work, regardless of the form in which the polynomials are written, or their degree. Also, both the derivation and the integration of a constant give a correct result.

VI. Conclusions

The application performed correctly performs various arithmetic operations on polynomials. One of the possibilities for further development would be to display the res when dividing two polynomials (the rest are already saved in memory). Another improvement that could be made to the application would be that it also works with negative exponents, or with polynomials of several variables. By making this application we have accumulated a lot of knowledge related to object-oriented programming, unit testing with JUnit, the use of regular expressions and the creation of graphical interfaces. I also learned how to better organize my code, following the Model-View-Controller structure, so that it is as readable as possible.

VII. Bibliography

- https://junit.org/junit5/docs/current/user-guide/
- https://docs.oracle.com/javase/tutorial/essential/regex/index.html
- http://tutorials.jenkov.com/java-regex/matcher.html
- https://en.wikipedia.org/wiki/Polynomial_long_division
- http://csfaculty.tcu.edu/comer/20803/MVCDemo.pdf