DOCUMENTATION

ASSIGNMENT *1*

NAME : STOICA IRINA

GRUPA:30422

**TABLE OF CONTENTS**

1. Assignment Objective……………………………………………………………………………………3
2. Problem analysis, modelling, scenarios, use cases………………………………………………………3
3. Design……………………………………………………………………………………………………5
4. Implementation…………………………………………………………………………………………..6
5. Results……………………………………………………………………………………………………8
6. Conclusions………………………………………………………………………………………………8
7. **Assignment Objective**

The purpose of the project is to create a Java Application which is able to compute basic operations on polynomials, like addition, subtraction, multiplication, division with quotient and remainder, derivation and integration.

Secondary objectives

* First, we need to divide the problem in more subproblems and create classes that interact with each other and solve the different subproblems such that in the end we can solve the entire problem. We also need to decide how the data will be represented (integers, strings, objects etc.). We can split the elements of a polynomial in more monomials and perform the operations on the monomials which will compose the polynomial that we actually want.
* Next, we need to read the polynomials from the user. The user can make mistakes when entering the polynomials, so there is a need for input validation, which also has to be implemented in order to avoid further problems during the run of the application. After we know for sure that the entered polynomials are valid and that we can perform the desired operations on them, we implement the functionality of the application. In this case the functionality is the operations performed on the polynomials.
* In the end we convert the polynomial data into information that can be displayed to and read by the user of the application.

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

Functional requirements

1. The polynomial calculator should allow the user to enter two polynomials.
2. It should allow the user to choose an operation to perform on the entered polynomials.
3. It should validate the input from the user to make sure the entered data is correct.
4. It should be able to add two polynomials.
5. It should be able to subtract two polynomials.
6. It should be able to multiply two polynomials.
7. It should be able to divide two polynomials.
8. It should take care of the care of the potential division by zero that can come as user input, as well as the division by a divisor whose power is greater than the power of the divisor.
9. It should be able to multiply two polynomials.
10. It should integrate a polynomial.
11. It should derivate a polynomial.
12. It should simplify the expression of a polynomial if it more elements have the same degree (i.e., 2x + x should be written as 3x).
13. It should display the result of the chosen operation to the user of the application.

Diagram

Description automatically generatedUse-cases

1. The application starts.
2. The user enters polynomials as text in the first two available text boxes.
3. The user chooses and clicks one of the six buttons available depending on the operation he wants to perform.
4. The result of the operation is computed and displayed as text in the third text box in the application.
5. The user can click another button in order to perform other operations on the already entered polynomials. The result is computed again and displayed in the third text box.
6. The user can also delete the polynomials entered in the text boxes and press one of the buttons to compute an operation on the new polynomials added.
7. The user can press one of the clear buttons which are available to delete the content of the text boxes easier.
8. The user can press the exit button to terminate the application.
9. **Design**

Diagram

Description automatically generated

Class design

The application has 6 classes: App, Monomial, Polynomial, Operations, Controller and the Enum class OperationType. The App class loads an fxml file which is the view and the user interface. It displays the Labels, Text boxes and buttons with which the user interacts. The user inputs command the model which is formed with the classes Monomial, Polynomial and Operations through the Controller class.

Class relationships

The Monomial and the Polynomial classes form a composition. A Polynomial object has one or multiple monomials of type Monomial, but they can exist outside the Polynomial class. The Polynomial class also forms a composition with the Operations class, which has two polynomials. The Controller class calls the methods from the model formed by the three classes: Monomial, Polynomial and Operations and receives results to display in the user interface. The Controller class depends on both Polynomial and Operations classes and the App class with the view interacts with the controller

Design decisions

For the integration and differentiation of the polynomials, I chose to only perform the operations on the first polynomial introduced by the user. Thus, the calculator displays the integration and differentiation of the first polynomial in the result box. In this case, the user is not obliged to enter the second polynomial, the application will not notify anything. However, if the user wants to perform one of the other operations which requires two polynomials, he must enter two valid polynomials. The user will be notified if one polynomial is incorrect or missing.

The calculator will take care of inputs which can be reduced. If one or both polynomials have more monomials with the same degree, they will be added before performing any of the operations, so that the result is the correct one.

The division operation is a particular operation, because is requires more computation and more care. The result of the division is displayed as quotient and remainder. For the division, the calculator will assume that the first polynomial is the dividend and the second entered polynomial is the divisor. If the second polynomial has a higher degree than the first polynomial, the division cannot be performed, and the application will alert the user. The user can modify the polynomial and try the division again. Another problem is the division by 0. In this case the application will also alert the user if the input is 0 or if the polynomial entered can be reduced to 0. The calculator will not perform the division and is ready to perform another operation requested by the user.

The input polynomials can have positive or negative integer coefficients and positive integer degrees. The polynomials will be entered as text, with the symbol “^” for notifying the power and “+” or “-” before the coefficient. The coefficients are written before the variable “x” with no other signs in between. If no “x” is entered, the calculator will assume the monomial is a constant with x at the power 0. If no coefficient is entered before the “x” character, the calculator will assume that the coefficient of the monomial is either -1 or +1 depending on the previous character. The entered polynomial cannot contain spaces, or other characters than digits, “x”, “^”, “+” and “- “.

Examples: *-x^2+3x+1, +2x^3-x+7*

When computing the operations on the polynomials, the coefficients are of float type. This is because some operations do not guarantee to have results with integer coefficients (division, integration). The result polynomial will be displayed with int type coefficients if the float type can be converted to integers and with float type coefficients if it is not possible to write them as integer.

For validating the input, I thought of and used a regex expression such that I cover a variety of possible inputs.

"(([+-][0**-**9]\*x\\^[0**-**9]+)|([+-][0**-**9]\*x)|([+-][0**-**9]+))+"

The brackets form a group. The expression basically matches a string which contains one or more groups specified in the brackets. In the big group there are more groups delimited by a logic OR. The groups correspond to the most common possible inputs of the user. I will list an example for each group inside the big group.

[+-][0-9]\*x\\^[0-9]+ => *+11x^3*

[+-][0-9]\*x => *-2x^7*

[+-][0-9]+ => *+6*

User interface

The user interface is simple and easy to use. There is only one scene which is displayed on the stage and contains the six buttons for the operations, the three text boxes and three labels for entering the polynomials.

1. **Implementation**

Monomial Class

The Monomial class has three constructors. The first constructor is empty. The second constructor receives the string form of the monomial and calls the method *public void convertMonomialFromString ().* The input of this method is known to be valid because the monomials are created when validating the input (we do not need to check again if the user input is valid, we just need to convert it to a float coefficient and an integer power). It checks for the first character of the string which is the character for the sign and after that checks for the character “x”. Next, it parses the string for the characters that are not digits. If there is no character between the sign and “x” the coefficient is automatically set to the sign and if there is, the coefficient is set to the corresponding integer value of the first string. If the substring “x^” is not found, the power is assumed to be 1, else it is converted from the string. If the string does not contain “x” the power will be 0 and the coefficient will be converted from the string. The second constructor creates a Monomial with the coefficient and the power specified.

The *public boolean hasSamePowerAs (Monomial x)* is a helper method, returning true if the *x* monomial has the same power. I added it in order to increase readability in some parts of the code.

The method *public String toString ()* is an overwritten method of the Object class in order to convert a monomial in a nicer way for the user display. If the coefficient is positive, there needs to be a “+” added before the coefficient. If the power is 0 it means that the monomial is a constant, so only the coefficient will be displayed. If the power is 1, then we do not need to print the character “^” after “x”, so we only use it for powers greater or equal to 2.

Polynomial Class

The polynomial class contains a list of all its monomials and a string for a potential error message that may have to be displayed during the usage of the application. It has an empty constructor and a constructor which receives the string form of a polynomial. Before anything, the string is checked in the method *public boolean checkPolynomial (String polynomialString)* with a regex expression and if it matches it, then the polynomial is parsed and converted, else, the error string is updated with a message explaining the problem. After a successful check, the string is then further parsed into substrings containing monomials. Then the corresponding monomials are created and added to the monomial list the polynomial.

The class contains a method for sorting the monomials in descending degree order. The method *public void reduce ()* checks for each monomial in a polynomial to see if there are other monomials with the same degree and if there are, adds them to the current monomial and removes them from the list after that. If after the reducing, there is a monomial with coefficient 0, it is also removed.

The methods for integration and differentiation are performed directly on the polynomial, not in the Operations class, because they do now require another polynomial. The coefficients and powers are modified correspondingly to the rules of integration and differentiation.

Operations Class

This class contains two polynomials and a string for a potential error message that should be displayed to the user. The addition and subtraction are done through the same method, with different parameters for the *operation* parameter.

*public Polynomial addOrSubtractPolynomials(Polynomial firstPolynomial, Polynomial secondPolynomial, Integer operation)*

By convention, if the parameter is 1, the method will perform addition and for 0 it will compute the subtraction. In the case of the addition, the method simply adds all elements of the two polynomials to the same list and calls the method for reducing the polynomial. For the subtraction, the coefficients of the monomials of the second polynomial are first multiplied with -1 in order to invert them, after that added to a list and reduced.

The division is performed according to the algorithm of division with quotient and remainder of two polynomials. The method *public Polynomial multiplyPolynomial (Polynomial p1, Polynomial p2)* multiplies each element from the first polynomial with each element in the second polynomial, adds it to a list, and reduces it after that.

Controller Class

In the controller class we have a function for each button. The function *performOperation(OperationType operationType)* receives from the functions responsible for the actions of the buttons for the operations the operation type in order to compute the wanted result and return it to the view. I also added functions in order to handle the possible errors and display them on the label in the view.

App Class

The app class loads the fxml file which contains the declarations for the buttons, text fields and labels and also connects to the Controller class which interacts with the model and updates the view. The fxml file was generated automatically when using the SceneBuilder for JavaFX applications.

Graphical user interface

Description automatically generated

1. **Results**

I created a test for each possible operation that can be performed on the polynomials and one for the addition which is used throughout the application to sort the polynomials before computations that need them sorted. I wrote a separate function which returns a true value if all the monomials of two separate polynomials have the same coefficients and powers. Each test was run and completed successfully.

Graphical user interface, text

Description automatically generated

1. **Conclusions**

The application implemented is a polynomial calculator which can perform some basic operations on polynomials and display the results to the user. In terms of functionality, it does the required tasks. It could be improved in terms of user interface and efficiency. The sorting and reducing methods can be slow for a large number of monomials. In terms of user interface, the displaying of the polynomials could be a little nicer. The current version displays the integer coefficients as float coefficients, and also the coefficients which are 1 could be dropped so that it is more readable. All in all, I think that this assignment helped me develop my programming capabilities and problem-solving abilities because it required to take care of a lot of aspects in comparison to the usual short or medium-length problems that are not so complex.