**DOCUMENTATION**

ASSIGNMENT 1

STUDENT NAME: MARCU HOREA RADU

GROUP: 30423

# CONTENT

[1. Assignment Objective 3](#_Toc95297885)

[2. Problem analysis, modeling, scenarios, use cases 3](#_Toc95297886)

[3. Design 3](#_Toc95297887)

[4. Implementation 4](#_Toc95297888)

[5. Results 5](#_Toc95297889)

[6. Conclustions 5](#_Toc95297890)

[7. Bibliography 5](#_Toc95297891)

1. **Assignment objective**

The main aim of this assignment is to design and implement a polynomial calculator with a dedicated graphical interface through which the user can insert polynomials, select the mathematical operation to be performed and view the result. The secondary objectives of this assignment are:

* Analyze the problem and identify requirements
* Design the polynomial calculator
* Implement the polynomial calculator
* Test the polynomial calculator

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

Firstly, we have to think about all the methods which we are going to need in order to fully implement this polynomial calculator. These include all the mathematical operations, but also the addition of a monomial into a polynomial, the implementation of graphical user interface and also a test method for each operation so that we can make sure that it works as intended. The uses cases for this polynomial calculator are: addition, subtraction, multiplication, derivative and integration. In order to compute these operations, the user has to:

* For addition, the user has to input two polynomials, one in each of the panels, having the format of each monomial ax^b, where ‘a’ is the coefficient and b is the power. Even if one of them is 1, or 0 for instance for the power, the user still has to put that number in so that the operation gets done as intended. The user then has to select the addition from the operation list and click on compute to see the result of the addition.
* For subtraction, the process is very similar with the one for addition, in the sense of the introduction of the polynomials. The only difference is the selection of the subtraction from the operations list. After this, by pressing compute the user will see the result of the subtraction.
* For multiplication, the process is the same: input the correct polynomials, select the multiplication operation and press compute to see the result.
* For derivative, the process is slightly changed, meaning that the user has to input only one polynomial, of the same format as previously mentioned, in the box for the first polynomial. After this, by selecting the derivative operation and pressing compute, the user will see the result.
* For integration, the process is the same as for derivative, the only change being the selection of the integration from the operation list instead of the derivative.

So, the functional requirements of this polynomial calculator are: to allow the user to insert polynomials, to allow the user to select the wanted operation, to perform correctly the operation on the given input if the polynomials are written correctly. The non-functional requirements are: easy and intuitive use by the user, the result to appear on the screen when the compute button is pressed, so that the user can easily see the result of the operation wanted to be performed.

1. **Design**

The overall system design is just a “black box”, having as inputs the two polynomials and the operation, and as output the result.

The application is split into three main packages, plus another one for testing. The three packages are: Graphical User Interface, Data Models and Business Logic. The Graphical User Interface packages consists of only two classes, Controller and View, the Data Models package consists of 3 simple classes, Polynomial, Monomial Int Coefficient, Monomial Double Coefficient, the latter 2 extending the super class Monomial which is also part of the mentioned package. The last package, Business Logic consists of only one class, Operations, as it is the class which deals with all the operations performed on the polynomials.

As for the data structures used, I have decided to use array lists instead of arrays, as they are much easier to access, especially when trying to delete an element from the list, or when adding one, without having to worry about changing the size of the array.

1. **Implementation**

The Operations class contains all the mathematical operations which can be performed by the polynomial calculator. Its first method solves the addition of the two polynomials which it takes as parameters. Its implementation is quite simple, in the sense that it tries to add each monomial of the second polynomial into the first one, by calling the method add Monomial from the Polynomial class. After all the monomials are added into the first polynomial, it gets sorted so the monomial with higher power appears first, having the powers in a descending order. The second method from this class solves the subtraction of the two polynomials given as parameters. Its implementation is again quite simple, by simply changing the coefficients of all the monomials from the second polynomial with their negated value, and then calling the addition method prior described, to get the wanted result. The third method from the Operation class solves the multiplication of two polynomials. Its implementation consists in going through the lists of monomials of the first polynomial, while at each monomial going through the list of monomials of the second polynomial. At each step, a new monomial is created, having the power as sum of the powers of the two monomials from the current step, and the coefficient as their multiplication. By doing so we get the wanted result of the multiplication of the two polynomials. All that is needed left is to sort the resulted polynomial so that the result has its monomials in reverse order of the powers. The next method of this class solves the derivative of its parameter. For its implementation all that is needed is to go through all the monomials of the polynomial and to decrease its power with one, and to multiply its coefficient with the original power. Any monomial with the power equal to zero is ignored as its derivative would be zero which does not need to be added to the resulting polynomial. The last method of the Operations class is the one which solves the integration of a polynomial. The main difference from all the other methods is that it will generate a polynomial which has monomials with coefficients of type double, as the new coefficient of the polynomial is highly unlikely to be an integer. So, while going through the list of the monomials of the given polynomial, on each step a new monomial is created, having the power increased by one of the current monomial, and the coefficient divided with its power.

The Monomial class stores the power of the monomial, having a construct, setter ang getter, and also an overridden compareTo method, to be able to sort the list of monomials of a polynomial. The next two classes, Monomial Int Coefficient and Monomial Double Coefficient extend the Monomial class, having only a setter, getter and constructor, with a coefficient of the type suggested by their name. The polynomial class stores the list of monomials of the polynomial, having a setter and a getter for it. Its first method is used to check if the current polynomial is equal to the one given as parameter. This is useful in the testing environment to check if the operations work as intended. It also has an addMonomial method, which adds the given monomial into the monomial list. It checks if there already exists a monomial with the same power and if not, it simply adds it to the monomial list, unless its coefficient is zero. If there already exists a monomial with the same power, it makes the sum of the two coefficients, and sets the result as the coefficient of the current monomial. If this coefficient is zero it gets removed from the list. The next method from the polynomial class is the readPolynomial one, which creates a polynomial from a given string (if the string is of the format ax^b+cx^d+…). In this method we simply go through the string character by character and compute the coefficient, power, of the current monomial before going to the next one. The last method of this class is the getStringFromPolynomial one, which does the exact opposite of readPolynomial: from a polynomial it returns the string which should be displayed (with the correct signs between each of the monomials).

The next two classes (Controller, View) are very similar with the ones which were given to us as model on the Microsoft Teams channel for this lecture.

The View class stores all the labels and the text fields which are needed in the graphical user interface. It also sets the size and the layout of the GUI in the prepareGui method. In the next method it sets what is needed for the result panel, while in the prepareNumbersPanel it creates the list of operations which can be selected, and also creates the text fields for the user to be able to input the polynomials in the form prior mentioned in this documentation. The Controller has an instance of the view, to be able to have the working GUI, and also a method to perform the action determined by the user. It gets the polynomials from the strings given by the user and performs the wanted operation by calling the needed method. It also uses the getStringFromPolynomial method to be able to display the result in the same way as the polynomials were given in the input by the user.

1. **Results**

The results of all the operations are being tested with Junit, in the AppTest class. This firstly has an AddMonomialTest, which tests if in a polynomial the monomials are being inserted as intended, in its list of monomials. The next performed test is for the equals method, which checks if by reading a polynomial from a string we get the same result as by adding the monomials directly in the polynomial. This test is very useful because the equals method is called in all the other tests to see if the result is the correct one.

The next test is for the addition operation. It reads three polynomials from strings and checks if the sum of the first two is equal with the third one, giving an error message in the opposite case. The next test is done for the subtraction operation, being the same as the test for addition, but performing the subtraction of the first two polynomials, expecting the result to be equal to the third polynomial read from a string. In case these are not equal, a different error message will be displayed. The following test is for the multiplication operation, again taking three polynomials from strings and checking if the multiplication of the first two is equal to the third one, giving an error message in case the result is not the expected one. The next test only takes two polynomials from strings, checking if the derivative operation on the first one is equal to the second one. In case they are not equal, an error message will be shown. The last test is the one for integration, again having only two polynomials, and checking if the integration of the first one is equal to the second one. Just like in all the other tests, if the result is not the wanted one, an error message is going to be shown.

1. **Conclusions**

In conclusion, by doing this assignment I have learnt how to work with polynomial operations, and how to get them from a given string if its format is the correct one. The main objective, and also the secondary ones, of the assignment were completed, as described in this documentation. The main future upgrades of this applications would be for it to be able to also implement the division of polynomials, or for it to work with other types of inputs, not only if the monomials are of type ax^b, being separated by exactly one sign(+/-).

1. **Bibliography**

For the documentation the only places from which I took inspiration were:

- <https://dsrl.eu/courses/pt/materials/A1_Support_Presentation.pdf>

- <https://dsrl.eu/courses/pt/materials/PT2021-2022_Assignment_1.pdf>

- the documentation template.

Also, for the code, I have taken inspiration from the GUI example from the Microsoft Teams channel, and from what indications we were given during the Labs (for the casting used in addMonomial for instance).