Logo, company name

Description automatically generated

Programming Techniques

Laboratory - Assignment 1

Polynomials Calculator

Student: Onaci Andreea-Maria

Group: 30421

Contents

[1. Objective 3](#_Toc130506696)

[2. Analysis of the problem 3](#_Toc130506697)

[a) Polynomial theory 3](#_Toc130506698)

[b) Modelling the polynomial 3](#_Toc130506699)

[c) Scenarios & Cases 4](#_Toc130506700)

[d) Cases & Case Diagram 4](#_Toc130506701)

[3. Design 5](#_Toc130506702)

[OOP Design 5](#_Toc130506703)

[UML Package Diagram 5](#_Toc130506704)

[Classes Diagram 6](#_Toc130506705)

[Data Structures 7](#_Toc130506706)

[4. Implementation 7](#_Toc130506707)

[Polynomial 7](#_Toc130506708)

[PolynomialController 7](#_Toc130506709)

[ReturnDivision 8](#_Toc130506710)

[View 8](#_Toc130506711)

[5. Results 8](#_Toc130506712)

[6. Conclusions 11](#_Toc130506713)

[7. Bibliography 12](#_Toc130506714)

1. Objective

Design and implement a polynomial calculator with a dedicated graphical interface through which the user can insert polynomials, select the mathematical operation (i.e., addition, subtraction, multiplication, division, derivative, integration) to be performed and view the result. Consider the polynomials of one variable and integer coefficients.

1. Analysis of the problem
2. Polynomial theory

A polynomial is composed by one or more monomials. A monomial has a coefficient, a variable and a degree.

1. Modelling the polynomial

For representing the polynomial, we use HashMaps. The powers are represented as keys (the powers in a polynomials are unique and integer) and a coefficient is associated to a specific key.

The operations which the user can compute for 2 given polynomials are addition, substraction, multiplication and division and for one polynom, the differentiation and integration operations. The polynomials are given through a GUI and the result will also be displayed through the same GUI. If the input is not proper, the application throws an error box.

1. Scenarios & Cases

The manner I implemented the GUI is a friendly one and it is easy to use.

A picture containing chart

Description automatically generated

The user will introduce the 2 polynomials in the text field destinated. The operation performed will be chosen through the button corresponding to the operation. The result will be display in the right part of the “Result”. On the right side of “Result 2” will be displayed the remainder for the division operation and the quotient will be displayed on the “Result” field. To make sure that the quotient and the remainder are the correct ones, on the right part of each polynomial returned as the result will appear the text “quotient”/” remainder” for clarifying the user.

The user must respect the format of the polynomial, otherwise the error message will be displayed. I treated the cases if the polynomial is null, if the text is not a polynomial at all, if the polynomial has more than one x (like xx), if the polynomial has another variable than x and the division by 0 which means already that polynomial 2 is null. For the degree 0, the user must simply introduce the coefficient as a number and for the monomial of order 1, the user must introduce “x” and the calculator will recognize it as a monomial of degree 1.

d) Cases & Case Diagram

Enter the polynomials.

Press the button for establishing the operation performed

See the final result

User

Press the button for establishing the operation performed

* Addition
* Subtraction
* Multiplication
* Division
* Differentiation
* Integration

User

The case diagram is used for establishing the relation between the user and the characteristics of the program. In our particular case, the user can introduce polynomials, can choose the operation to be performed with the input and can see the result.

1. Design

OOP Design

The calculator respects the OOP concepts. The class *Polynomial* is inherited by the class *ReturnDivision*, class which returns the quotient and the remainder for the division operation. Also, the calculator provides the encapsulation and abstractization concepts, by the organization in different methods of the operations and the manipulation of input data is hidden in the background, so that the user can compute the operations in a very easy and confortable manner.

UML Package Diagram

This Calculator is divided in 2 packages, *Operations* which contains the classes *Polynomial* and *ReturnDivision* and the package *GUI* which contains the classes *PolynomialController* and *View*.

Package Operations

Polynomial

ReturnDivision

Package GUI

Polynomial Controller

View

Classes Diagram

A white board with writing on it

Description automatically generated with low confidence

The diagram contains all the classes implemented for computing the calculator. *View* is the class which contains the elements from the GUI, *PolynomialController* is the class which realizes the relation between the GUI and the class which implements the operations, *Polynomial* is the class which implements all the operations and *ReturnDivision* is the class which stores the result of the division operation.

Data Structures

The polynomials are introduced through the GUI. There is a method implemented to convert the given polynomials (which are represented as Strings) into HashMaps. For computing this, I used regex for getting the coefficients and the powers of the given polynomial. The coefficients are stored as values in the HashMap and the powers are the unique keys. The values are stored in a list, corresponding to an unique key.

1. Implementation

Polynomial

This class contains all the operations that can be performed for a given polynomial. It contains the method *sum* for computing the sum of 2 polynomials which has one parameter, containing the HashMap resulted by concatenating the HashMap of the first polynomial and the second one. The second method is the *diff* which is implemented by multiplying with (-1) all the coefficients from the second polynomial and compute the sum of the new polynomial concatenating with the first polynomial. Another method implemented is the *multiplication* which computes the multiplication of the second polynomial with every monom from the first one and then computes the sum of the resulted polynomial. The *divison* method is implemented as a long division, in which the remainder becames the divident at each step. The *differentiation* is implemented for a given polynomial after the formulae and the *integration* is implemented in the same way. Also, this class has the methods *degree* and *concatenate*, the *degree* function returns the highest power of the polynomial (even if the order of the monomials are random distributed) and it is used in computing the division of the 2 polynomials and the *concatenate* method is used in the *sum* method, concatenating 2 maps into one (using chaining approach).

PolynomialController

This class has the method *hasMultipleX* which returns true if the input has more than one x one after the other one and false otherwise. Also, the class has the method *checkError* which verifies if the input is properly to be converted into a HashMap (checks if the input is null, if the input is different from a digit or it has or not the variable x (required) , if the method *hasMultipleX* returns true, and if the polynomial has a good form). Another method is *convertPolToHashMap* which implements the conversion from a String to a HashMap (the input is verified through regex methods and there is applied the *checkError* method for the input). The method *convertHashMaptoPol* does the inverse path, it converts the HashMap resulted by the operation chosen to a String which will be displayed on the GUI. For the buttons, there are implemented the methods which realizes the relation between the button pressed in the GUI and the corresponding implementation of the operation.

ReturnDivision

This class is implemented for returning the pair of quotient and remainder for the *division* method. This class contains the getters for the quotient and remainder, passed to the *View* class for displaying them.

View

This class implements the GUI, it contains the buttons, the labels, the buttons and the main panel. Also, it has the method for computing the dimensions of the window and the methods for listeners of the buttons.

1. Results

The input is given through the 2 JTextField defined. If the input is null, the error *The polynom is null!* will be displayed. If the input has more than one x one after the other one, the error message *Invalid polynomial!* will be displayed. If the polynomial has another variable than x, the message *The output should be either a digit or a polynomial with the variable x!* is displayed. If the input does not have the proper form, in the view will be displayed the following error message *The polynomial has not a good form, it should be of form a1x^n+a2x^(n-1)+..+a(n-1)x^1+an!*. If the input contains a sequence like „xx”, the error message *Invalid input, duplicated keys!* will be displayed.

If the input is valid, for the sum the result will be displayed on the first field named *result*, the same will happen for diff, differentiation, integration and multiplication. For the division, the quotient will be displayed in the first field, and the remainder on the second one, and on the right part of each result will appear the label named after the term which is displayed (quotient and remainder), in order to make it easier for the user to understand and to see the results given by the calculator.

For checking the correctness of the operations performed on polynomials, I used Junit. I tested there a basic case and an exceptional case for each of the operations. These method helped on verifying the result passed through convertion methods from controller and helped on checking the result of operations.

The results for implementing the function will be presented below:

Sum:

A picture containing background pattern

Description automatically generated

Subtraction:

Graphical user interface

Description automatically generated with low confidence

Multiplication:

A picture containing background pattern

Description automatically generated

Division:

A screenshot of a computer

Description automatically generated with medium confidence

Differentiation:

A screenshot of a computer

Description automatically generated with medium confidence

Integration:

Graphical user interface

Description automatically generated with medium confidence

The coefficients are double, for displaying the proper values of the coefficients of the integrated polynomial.

1. Conclusions

This assignment used the OOP concepts learned in the first semester, so it was a good way for revising what I learned in the first semester. Also, it was a challenge to manage my time in order to have time for implementing it.

In conclusion, this assignment helped me revising my skills in writting Java code and OOP concepts.

7. Bibliography

https://www.geeksforgeeks.org/convert-string-or-string-array-to-hashmap-in-java/

https://www.scaler.com/topics/string-to-float-in-java/

<https://java2blog.com/format-double-to-2-decimal-places-java/>

https://www.w3schools.com/java/java\_regex.asp