**DOCUMENTATION**

ASSIGNMENT 1

STUDENT NAME: APOSTOL HORIA

GROUP: 30227

# CONTENTS

[1. Assignment Objective 3](file:///C:\Users\40741\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043139)

[2. Problem Analysis, Modeling, Scenarios, Use Cases 3](file:///C:\Users\40741\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043140)

[3. Design 4](file:///C:\Users\40741\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043141)

[4. Implementation 5](file:///C:\Users\40741\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043142)

[5. Results 8](file:///C:\Users\40741\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043143)

[6. Conclusions 8](file:///C:\Users\40741\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043144)

[7. Bibliography 9](file:///C:\Users\40741\Downloads\PT2023_Documentation_Template_EN.doc#_Toc128043145)

# Assignment Objective

The project’s purpose is to create a polynomial calculator that can perform operations faster than a human being. All six operations such as adding, subtracting, multiplying, dividing and also single polynomial operations: integrating and deriving can be easily done. The user just has to simply input the two polynomials in the designated spaces and the result will be displayed at the top of them.

# Problem Analysis, Modeling, Scenarios, Use Cases

To use the polynomial calculator, the user must first launch the main program and enter the two polynomials. Once the polynomials have been entered, the user can then choose the desired operation.

Diagram

Description automatically generated

After the operation button is pressed, each polynomial is verified if it has the correct polynomial format that is also mentioned in the GUI.If one of them does not have a correct format, in the designated result area will appear a message that mentions the error. Otherwise in the same spot will appear the correct result of the operation.

# 3.Design

Diagram

Description automatically generated

# 

# Implementation

The classes have been organized into packages that are representative.

The MainClass takes on the task of initializing the Graphical User Interface and establishing the necessary connections between the models and the GUI.

***Data Models:***

**-Monomial Class:**

Inside the Monomial class, the coefficient and the power are set to be private attributes. Other than the getters and setters the class is provided with methods that can modify the coefficient or power by adding or subtracting. The class also includes a method for converting the java object to a String type.

-**Polynomial Class**

The Polynomial is constructed using a TreeMap with Integer keys (representing the power) and Monomial values for each key. The Polynomial class has three constructors: an empty constructor, one that takes a Monomial object as an argument and constructs a Polynomial with just that Monomial, and another constructor that takes a TreeMap of Monomials as an argument to create a Polynomial.

Public void addMonomial(Monomial)- adds to the treemap a new Monomial

Public void showMonomials()- prints all the values of the polynomial TreeMap

Public void removeZeroes()- removes all entries in the TreeMap with a coefficient of 0, as these entries are considered unnecessary for further calculations. The Polynomial class creates a Set of keys that correspond to entries with a coefficient of 0, and then iterates through the Set to remove the keys from the TreeMap. This approach is necessary because direct modification of the TreeMap during iteration would result in an exception.

Also a toString method appears.

**-GUI class**

The GUI class stores the 6 operation buttons and 3 textFields as private attributes, which are utilized for inputting and displaying data within the program's user interface.

The program creates a graphical user interface (GUI) by constructing a frame with specific size dimensions and a default close operation. The GUI is built using JPanels as building blocks to assemble the final interface.

The “upperPanel” is divided into three sub-panels, with one for result, one for polynomial 1, and one for the polynomial 2 . Each sub-panel is given a FlowLayout to arrange a label and a textField on the same row. A scroll is added to each sub-panel for easy viewing of longer results. The title label is added directly to the UpperPanel. The "allPanel”, which holds all the sub-panels, uses a BoxLayout with a Y-axis orientation to stack all rows vertically.

Below the UpperPanel, the required format for a valid polynomial is specified, along with the observation that only polynomial 1 supports integration and derivation. Two panels with three operation buttons each are added below this.

Finally, the frame is set to visible.

The GUI methods serve various purposes, such as retrieving JTextFields or attaching ActionListeners to buttons. Additionally, there is a function available that can display an error message in a new window stating "Invalid Polynomial x". Furthermore, there is another method that can obtain the result's JTextField and modify its content with a provided argument.

**-Operation Class**

*adding(Polynomial ,Polynomial): Polynomial*

Firstly I create a new result polynomial in which I put all the first Polynomial entries. Then I iterate through all the polynomial 2 keys and where result contains the current key(both have a monomial with the same power) then I add the coeficients, else I put a new entry in the result polynomial.

*subtracting(Polynomial ,Polynomial): Polynomial*

It works like the adding method but instead of adding I subtract the polynomial 2 entries from the result. I made it so the user can only subtract the second from the first polynomial.

multiplying(Polynomial, Polynomial): Polynomial

I iterate through the second polynomial’s monomials and I multiply each of them to the first polynomial. For each operation(“monMultPol= Monomial multiplying with Polynomial”) I add the obtained Polynomial to the result and at the end I return it.

deriv(Polynomial): Polynomial

I go through every key of the polynomial and if it is different from 0 I create a new Monomial which has the current power and coefficient values. I create the new power which is the power reduced by 1 and the new coefficient is the monomial coefficient multiplied by the monomial power. To the result I add a Monomial with these results, and the process is repeated until the loop has ended.

*integ(Polynomial ): String*

For this method I thought it is a better idea to return a String so the coeficients cand have the form of “(a/b)” if the division can not return an “int” value. If the value returns an int value to the String a value type “a” should be added.

*dividing(Polynomial, Polynomial):Polynomial[ ]*

Before doing the operation I should check if the dividend and divisor are equal by subtracting the divisor from the dividend. If they are, I return a Polynomial made from a Monomial of coefficient 0 and power 0.

Else I firstly check if the divisor is equal to 0 and throw an Exception if it is true. Otherwise the remainder polynomial gets the dividend’s polynomial form.

I create a loop to get the quotient and the remainder and it ends if the remainder’s polynomial TreeMap is empty or if the remainder’s degree is smaller than the divisor’s degree. I create Monomials that have the power received from subtracting the divisor’s biggest power from the remainder biggest power. The Coefficient is calculated by dividing the coefficient of the term with the highest power in the remainder polynomial by the coefficient of the term with the highest power in the divisor polynomial. This monomial is then added to the quotient, and from the remainder, this monomial multiplied by the divisor is subtracted from the remainder. Then from the remainder I remove the terms that have a coefficient equal to ‘0’.

The process is repeated until the remainder is an empty polynomial or if the remainder degree is smaller than the divisor’s. The returned variable is a Polynomial array of the remainder and the quotient.

**-Controller class**

The Controller is responsible for managing the functionality of the buttons.

In the constructor this thing is provided. Also here the method validatePolynomials(String,Polynomial) is found. Here I used regular expression and pattern matching to check if the user input is a valid Polynomial.

The regex has this form: **“^([+-]?\\d\*x\\^\\d+([+-]\\d\*x\\^\\d+)\*([+-]\\d+)?)$”**

**`^` and `$`-** are marking the start and the end of the string and are ensuring that the string matches the pattern.

**`([+-]?\\d\*x\\^\\d+`-** matches the first term of the polynomial which has an optional “+” or “-“ followed by the coefficient “\d+” followed by the variable x and the exponent “\d+” separated by the symbol “^”

**`([+-]\\d\*x\^\\d+)\*`-** here the “ \* ” operator allows for zero or more occurrences of this group.

**`([+-]\\d+)?`-** matches an optional constant term which consists of an optional sign followed by the constant coefficient. “?” marks this group optional.

I did not use a pattern to extract the Monomial’s values but I split into Monomials having the signs “+” and “-“ as splitters. To create Monomials I searched for the index of “x” and what was found before the x was the coefficient and after the”^” the power. Last but not least I added each Monomial to the polynomial sent as parameter in the method.

1. **Results**

The tests for the results are performed via JUnit. The following tests are destined for the polynomial operations. For each individual operation i made 2 polynomials and i verified if the polynomial String format takes the correct value. For example for the ***addingTest*** for the polynomial 1 that has the form “3x^2-x^1+1” I put into the Polynomial TreeSet the corresponding values of each monomial. I did the same for the polynomial 2 “x^1-2”. The test asserts true if the result after adding is “3x^2-1”.

I also inserted some cases where the tests fail.

1. **Conclusions**

As a conclusion, the project’s main objective is the efficient and accurate calculation operations of two polynomials. The implementation of the algorithms allowed for the creation of a user friendly interface that allowed users to easily input two polynomials and perform the desired operation. The resulting output can be seen in a clear and concise manner, so every user can quicky interpret the results.

Overall, the successful implementation of the calculator that provides a valuable tool for anyone that works with polynomials in fields such as mathematics, engineering or computer science.

The resulting output is can be seen in a clear and concise manner, allowing every user to quickly interpret the results.

1. **Bibliography**

<https://regexr.com/>

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

https://app.diagrams.net/