DOCUMENTATION

ASSIGNMENT 1

STUDENT NAME: Persea Oana - Mihaela
GROUP: 30424

CONTENTS

1. Assignment Objective

The main objective of the assignment is to implement a polynomial calculator, more specifically, to perform operations on two polynomials. For this calculator to be complete, it should perform addition, subtraction, multiplication, division, derivatives and antiderivatives. The sub-objectives include:

- understanding the requirements thoroughly

- finding a way to transform the input given by the user into a format that the computer can understand

- developing the algorithms necessary to perform the required operations

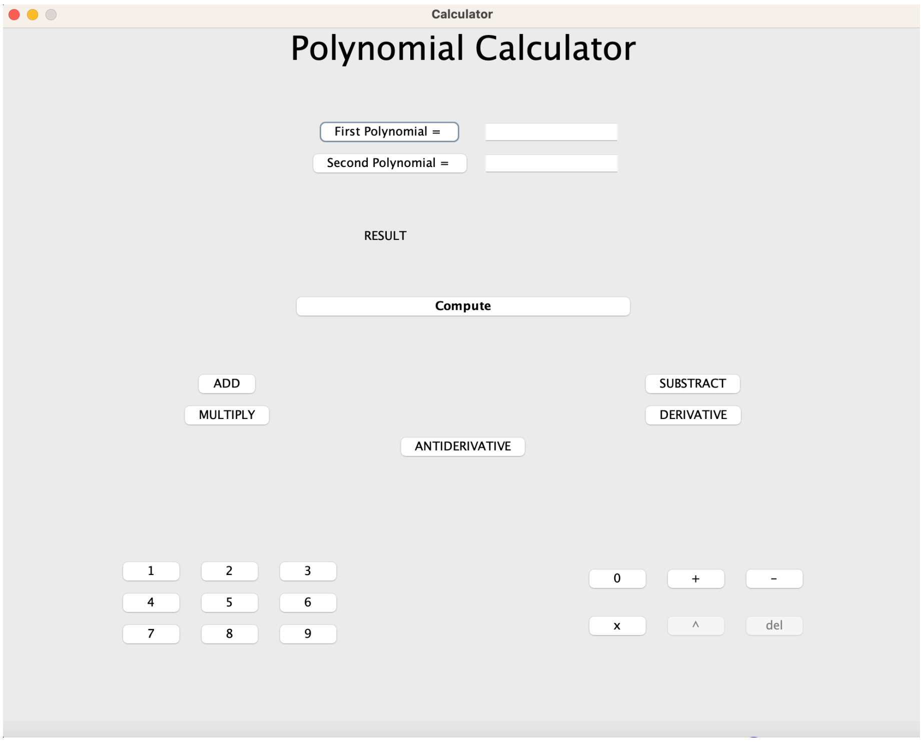
- finding a way to transform the result given by the program into a format that the user can read and understand

- developing the user interface

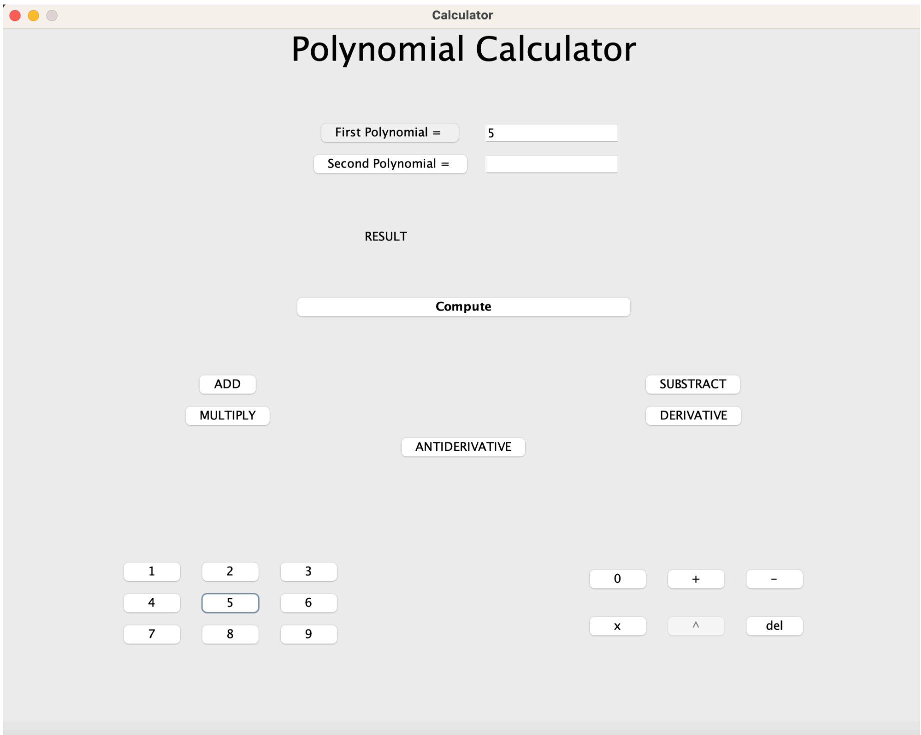
- a lot of testing to make sure that the program will respond to multiple variations of the way a polynomial can be written

2) Problem Analysis, Modeling, Scenarios, Use Cases

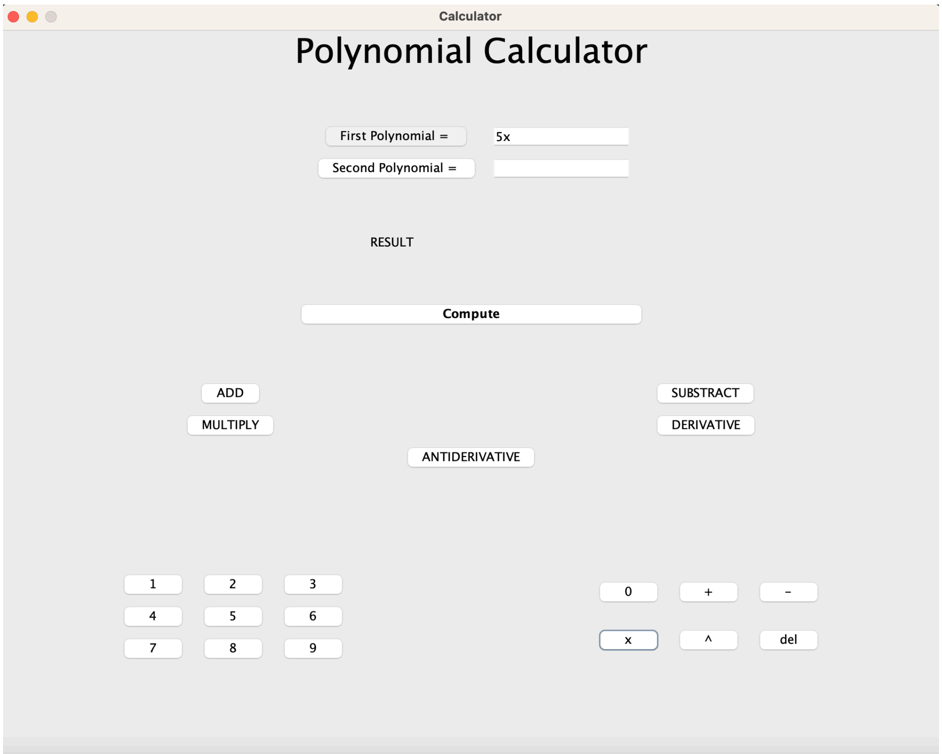
Firstly, the user has to choose the polynomial he/she wants to introduce (the delete button is inactive since there is no input yet)



Then, the buttons are used to introduce the coefficients (here the delete button becomes inactive)


First Polynomial = 5

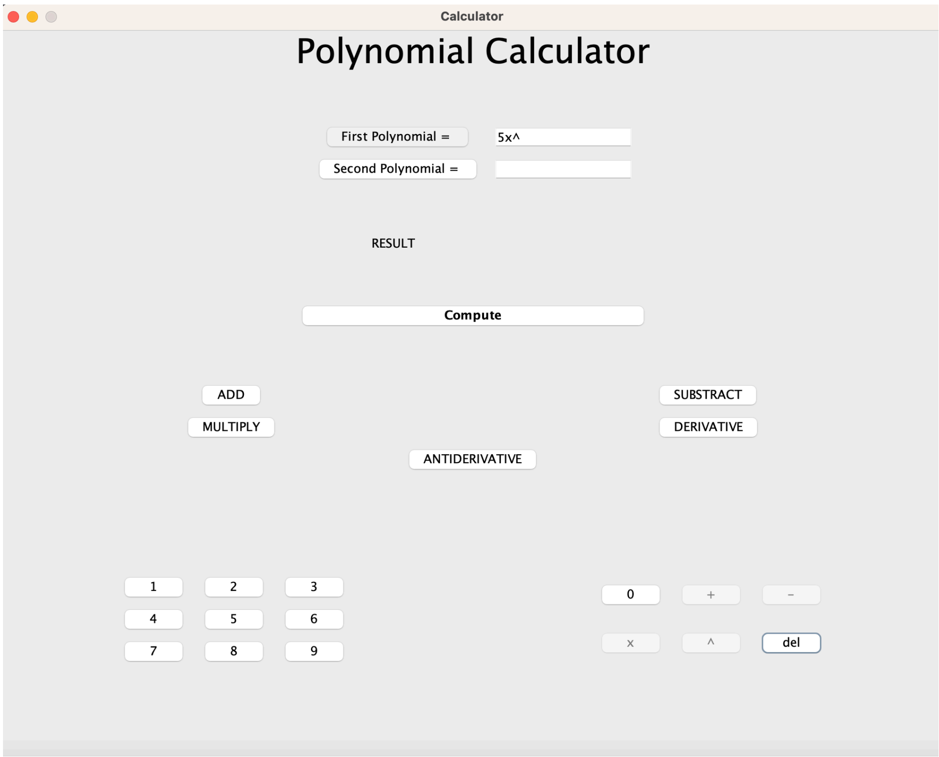
After the user adds x, the button corresponding to the power sign becomes active (it was inactive to avoid wrong input)



First Polynomial = 5x

Second Polynomial =

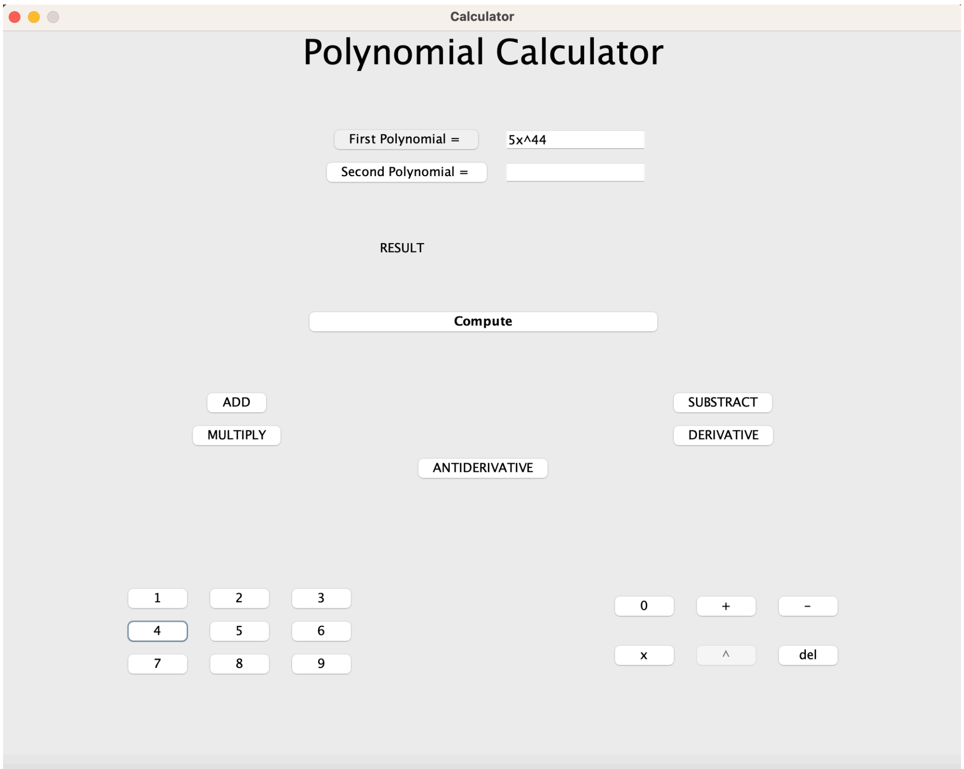
When the power sign is added, it becomes inactive again



First Polynomial = 5x^

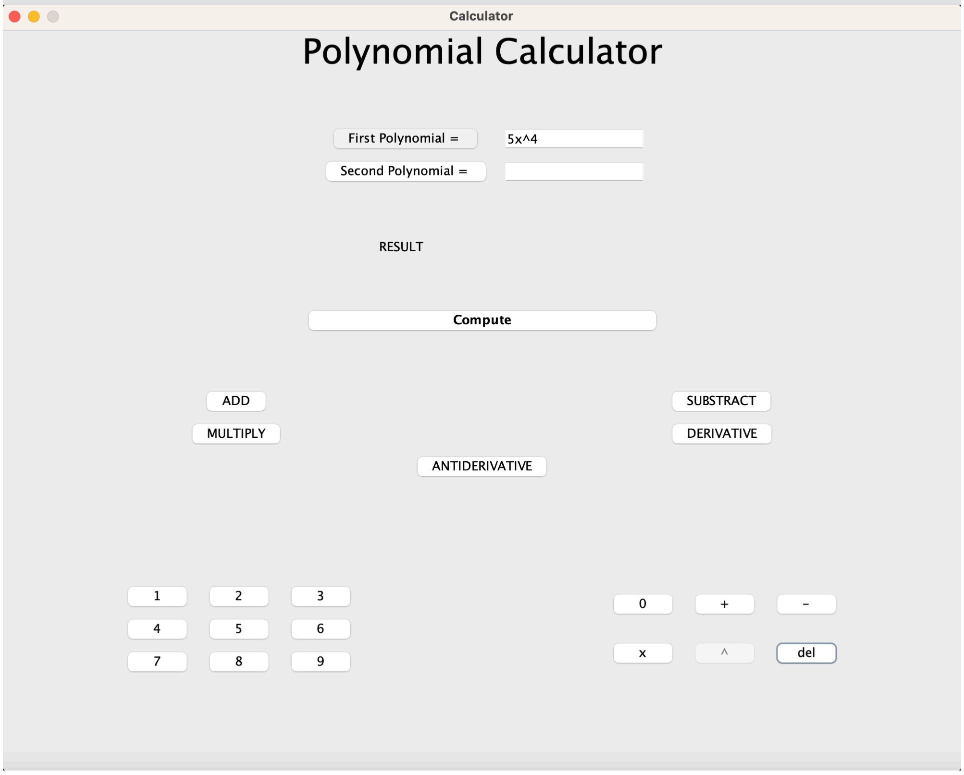
Second Polynomial =

If the user makes a mistake, the del button can be used, it will delete the last introduced character



First Polynomial = 5x^44

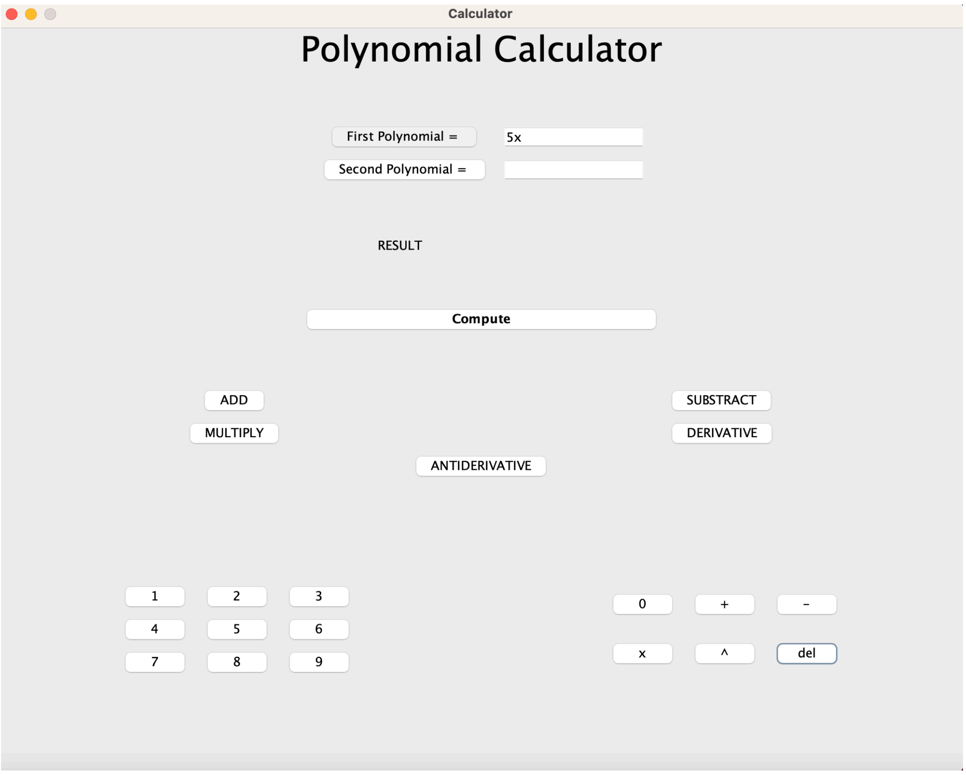
Second Polynomial =



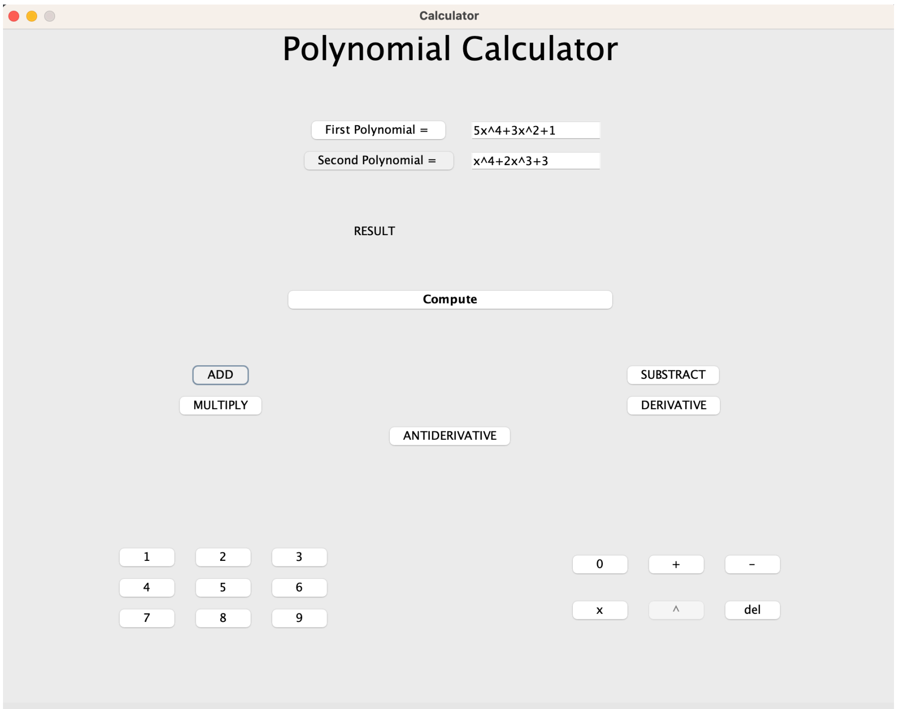
First Polynomial = 5x^4

Second Polynomial =

If the last element in the text field after using “del” button is x, the power sign button becomes active once again



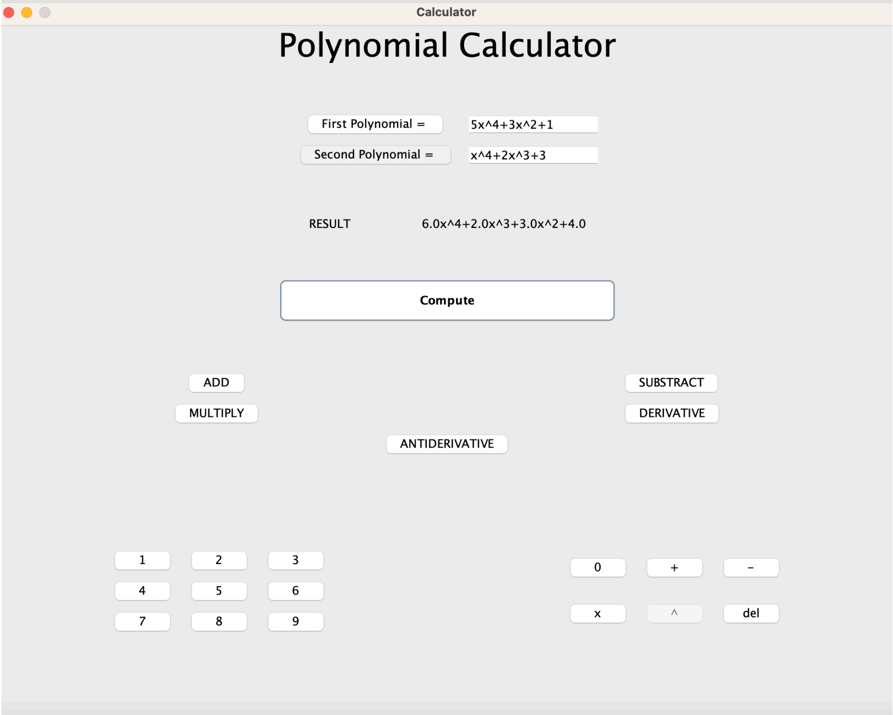
After adding the two polynomials, one of “ADD”, “MULTIPLY”, “SUBTRACT”, “DERIVATIVE”, “ANTIDERIVATIVE” buttons have to be selected, and when “Compute” is pressed, the result will appear on the screen



First Polynomial = 5x^4+3x^2+1

Second Polynomial = x^4+2x^3+3

RESULT

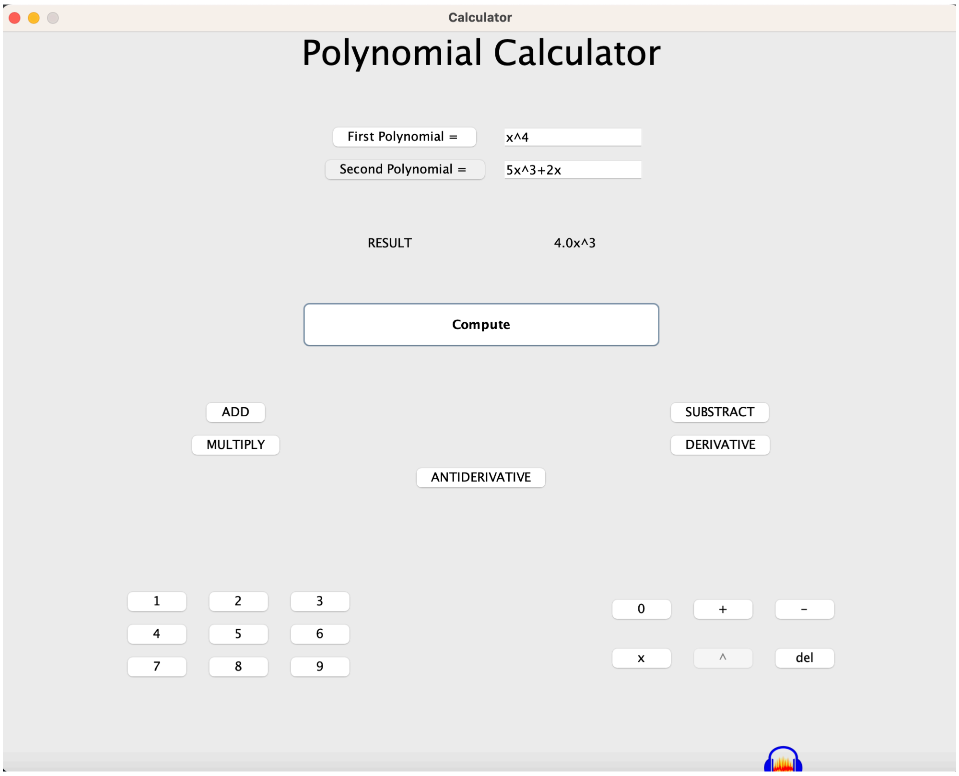


First Polynomial = 5x^4+3x^2+1

Second Polynomial = x^4+2x^3+3

RESULT 6.0x^4+2.0x^3+3.0x^2+4.0

For derivatives and antiderivatives, in case in both fields there is a polynomial, the operation will be performed on the first one



Calculator

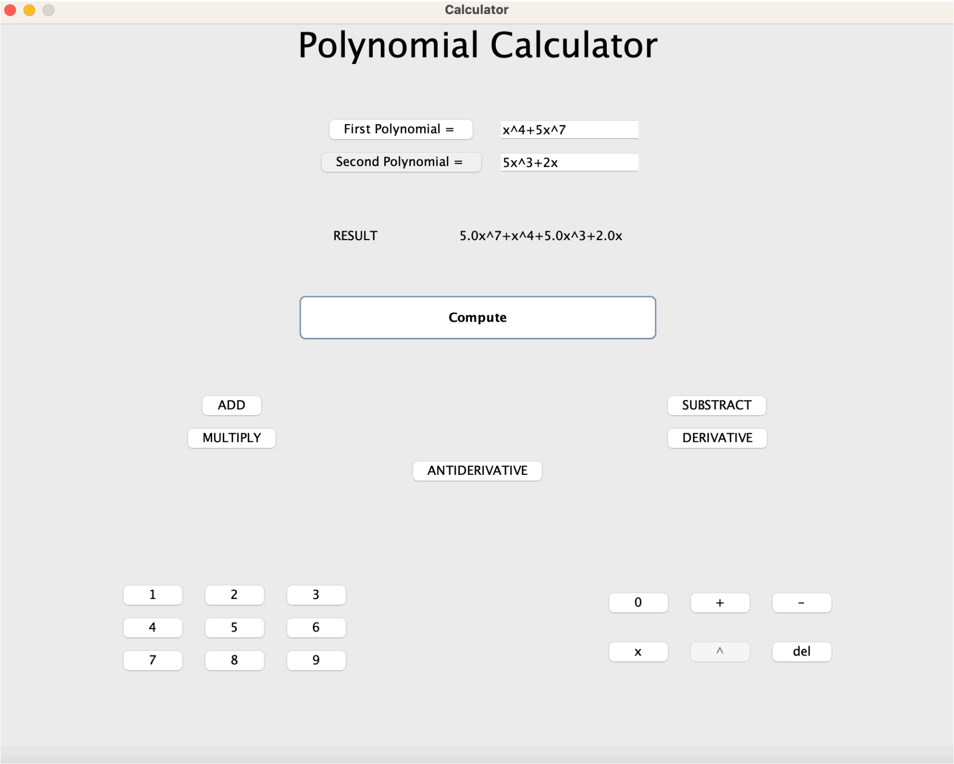
Polynomial Calculator

First Polynomial = x^4

Second Polynomial = 5x^3+2x

RESULT 4.0x^3

In the case when the elements of the polynomial are not in a specific order, the result will always show them in descending order of the powers



Calculator

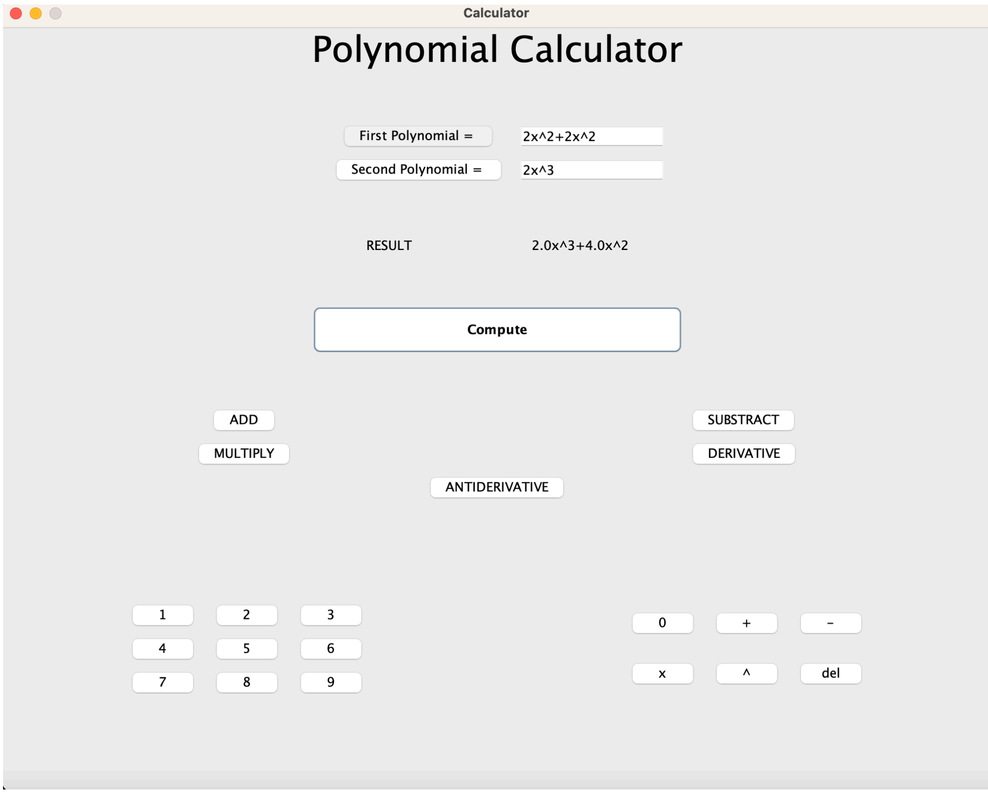
Polynomial Calculator

First Polynomial = x^4+5x^7

Second Polynomial = 5x^3+2x

RESULT 5.0x^7+x^4+5.0x^3+2.0x

If in the same textfield there are more terms with the same coefficient, those are added together and saved as a sum in the map. Also, coefficients that start with 0 (0005) are also accepted. For power 0 the user can enter both the coefficient alone or the coefficient followed by x^0. For power 1, also we can use x or x^1.



Calculator

Polynomial Calculator

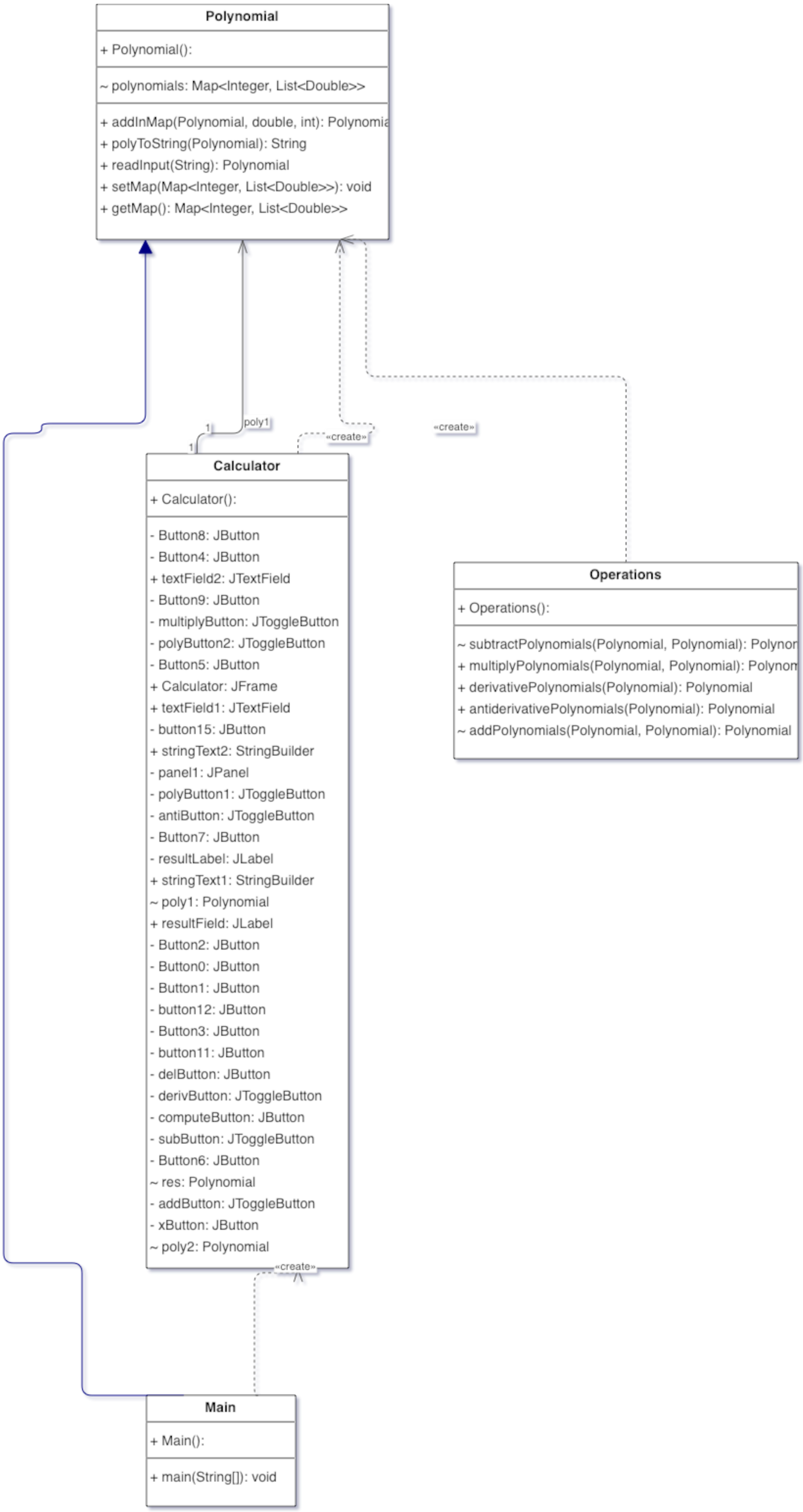
First Polynomial = 2x^2+2x^2

Second Polynomial = 2x^3

RESULT 2.0x^3+4.0x^2

3. Design

—UML Diagram—


Polynomial
+ Polynomial():
~ polynomials: Map<Integer, List<Double>>
+ addInMap(Polynomial, double, int): Polynomia
+ polyToString(Polynomial): String
+ readInput(String): Polynomial
+ setMap(Map<Integer, List<Double>>): void
+ getMap(): Map<Integer, List<Double>>
1
poly1
1
«create»
«create»
Calculator
+ Calculator():
- Button8: JButton
- Button4: JButton
+ textField2: JTextField
- Button9: JButton
- multiplyButton: JToggleButton
- polyButton2: JToggleButton
- Button5: JButton
+ Calculator: JFrame
+ textField1: JTextField
- button15: JButton
+ stringText2: StringBuilder
- panel1: JPanel
- polyButton1: JToggleButton
- antiButton: JToggleButton
- Button7: JButton
- resultLabel: JLabel
+ stringText1: StringBuilder
~ poly1: Polynomial
+ resultField: JLabel
- Button2: JButton
- Button0: JButton
- Button1: JButton
- button12: JButton
- Button3: JButton
- button11: JButton
- delButton: JButton
- derivButton: JToggleButton
- computeButton: JButton
- subButton: JToggleButton
- Button6: JButton
~ res: Polynomial
- addButton: JToggleButton
- xButton: JButton
~ poly2: Polynomial
«create»
Operations
+ Operations():
~ subtractPolynomials(Polynomial, Polynomial): Polynor
+ multiplyPolynomials(Polynomial, Polynomial): Polynon
+ derivativePolynomials(Polynomial): Polynomial
+ antiderivativePolynomials(Polynomial): Polynomial
~ addPolynomials(Polynomial, Polynomial): Polynomial
Main
+ Main():
+ main(String[]): void

—Data Structures—

This application uses both primitive data types like integers or double type, strings and more complex ones, like HashMap. I chose to use this data structure instead of arrays because it is more efficient in the case of memory management. We also get rid of the problem of overflow because we do not have to specify the length (like in the case of simple arrays).

I also created a new object to help me store the entered data: Polynomial.

4. Implementation

- Class **Polynomial**, that contains a HashMap used to store a single polynomial. It implements the method addInMap, which makes sure that even if the same power appears multiple times in the same polynomial, in the map we only save the final value. The second method used is readInput which uses multiple Patterns and Matchers to transform the text given by the user into a format the computer is able to read (a map with powers and coefficients). The method polyToString does the opposite, it transforms the Map into a text that can be written so the user can see the result.
- Class **Operations** that implements methods for the following operations: addition, subtraction, multiplication and computing the derivative and the antiderivative of a polynomial.
- Class **Calculator** which is in charge of the UI (calls the operations when the corresponding buttons are pressed). It doesn’t have any methods, mostly actionListeners to implement the behaviour of all the objects in the UI.

5. Conclusions

In the end, this project made me improve my knowledge regarding Object Oriented Programming and the ability to handle different data structures. After developing this application I came to the conclusion that it was harder to design than I was expecting, but this only made me search solutions to my problems and challenged me to get to the final result. I also learned how to implement a Graphical User Interface using Swing and how to deal with String parsing using Regex. Some functionalities were easier to code, for example addition and subtraction, but division was the one that put me in difficulty.

This program could be improved by adding some more functionalities like computing the value of a polynomial in a given point, supporting polynomials of more than one variable, Fourier transform multiplication, finding the roots of a polynomial, plot the graphic and for a better performance there should be implemented all cases where exceptions can occur and the application stops working due to an error made by the user.

Also, the graphic user interface could be improved by making it more interactive, adding more text boxes for the error messages and some new fields for operations.

6. Bibliography

1. Programming Techniques – Lectures of prof. Cristina Bianca Pop
2. regex101.com
3. stackoverflow.com
4. geeksforgeeks.org