Logo, company name

Description automatically generated

DOCUMENTATION

**Polynomial calculator**

Programming Techniques

Laboratory - Assignment 1

Student: Paul Petruț-Betuel

Group: 30421

# CONTENTS

[1. Assignment Objective 3](#_Toc128043139)

[2. Problem Analysis, Modeling, Scenarios, Use Cases 3](#_Toc128043140)

[3. Design 3](#_Toc128043141)

[4. Implementation 3](#_Toc128043142)

[5. Results 3](#_Toc128043143)

[6. Conclusions 3](#_Toc128043144)

[7. Bibliography 3](#_Toc128043145)

# Assignment Objective

*The objective of this assignment is to develop a polynomial calculator in Java that can perform arithmetic operations on polynomials. The calculator should be able to add, subtract, multiply, divide, integrate and derivate polynomials. The calculator should provide an intuitive user interface and give accurate results for all operations.*

# Problem Analysis, Modeling, Scenarios, Use Cases

Arithmetic operations on polynomials: The polynomial calculator should be able to perform the mentioned operations on polynomials with positive integer power. This will involve parsing user input, manipulating polynomial data structures, and implementing the relevant algorithms for each operation.

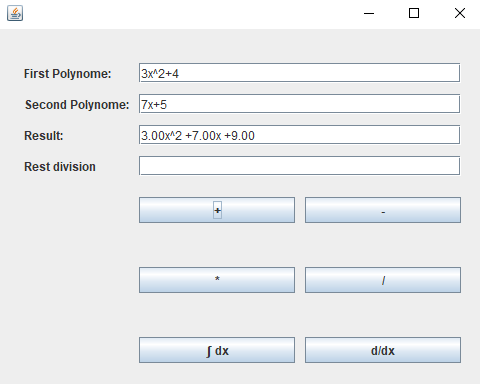
Intuitive user interface: The calculator should provide an easy-to-use interface for users to enter and manipulate polynomials. This involve creating a graphical user interface (GUI) with buttons, text fields and labels.

Accuracy and precision: To be useful for mathematical calculations, the calculator must provide accurate and precise results for all operations. This involve using appropriate data types, avoiding numerical errors, and handling edge cases and corner cases carefully.

For writing the polynomials the user needs to write the two polynomials in any order, using only the **‘*x*’** as the unknown variable. The polynomial can be written in any order just to respect the order of the monomial, coefficient variable and then the power. If there is just x then it will be taken as x1and if there isn’t any element in front of the x for the coefficient it will be taken as 1.00. If nothing is written it will be taken as the null polynomial: 0.00x.

For operations after the user has input the needed data, he needs to press the button for that, for adding he needs to press the ‘*+*’ button, ‘-’ for subtraction of the first polynomial minus the second polynomial, for multiplying is the ‘\*’ button, ‘/’ is for division witch gives a warning for division with 0.

The integration and derivatives are done only for the polynomial in the box next to the text First Polynome.

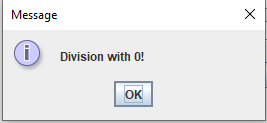


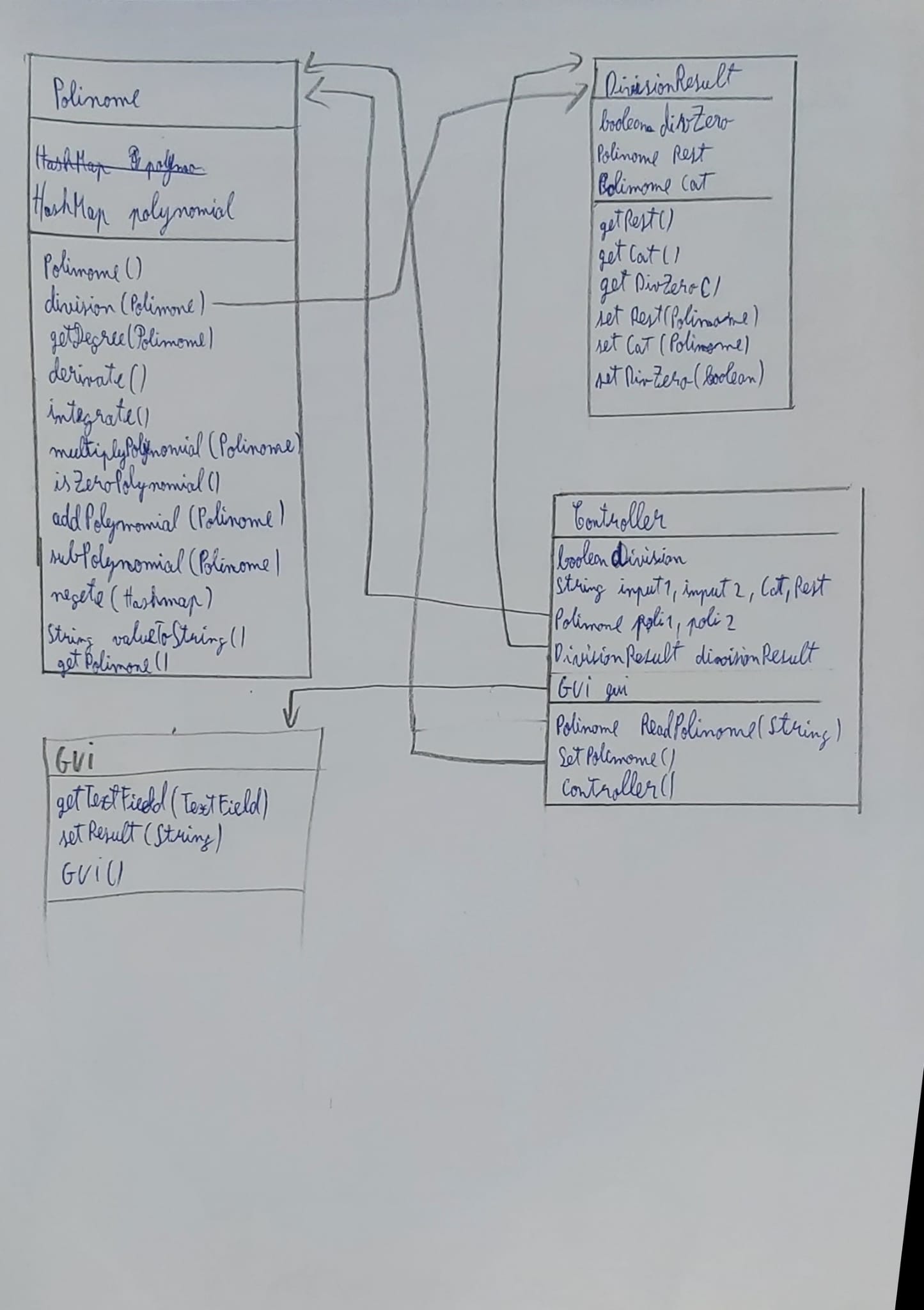
# Design

*The following should be presented: OOP design of the application, UML package and class diagrams, used data structures, defined interfaces and used algorithms (if it is the case).*

*In the design phase of this project, we will focus on developing a clear object-oriented design for the polynomial calculator application.*

*The store the polynomial as monomials in a hash map where the key is the power of each monomial, and the value is a float for the coefficients.*

O imagine care conține masă

Descriere generată automat

# Implementation

The class “Polinome” is the one that occupy with the operations on polynomials. The methods do:

* Polinome() make new polynomial with a new HashMap
* Division(Polinome) take the polynomial form its object and divide it by the given polynomial and set a flag if the division is with the null polynomial. The result is a new object from the class DivisionResult.
* getDegree() return the biggest power which has a coefficient different of 0
* derivate and integrate subtract, respectively add 1 to the power and multiply, respectively divide with the new power the coefficient in a new Polinome object
* multiply, add and sub use the same logic the difference is the operation done. It takes each element and check if it can be operated with the other polynomial if it can than the result is added to a new Polinome object.
* ValueToString() returns a string with the polynomial in a classical writing style of it

The class DivisionResult has just getters and setters for its values.

The class Controller open the GUI and it has all the listeners for the buttons and it takes the action about them, the other methods do:

* readPolynome() is parsing the user input in a new polinome object
* SetPolinome() is just reading both polinomials and “translate” then from string to Polinome

# Results

Based on the result from the Junit testing all the operations are correct implemented.

# Conclusions

In conclusion, developing a polynomial calculator in Java involves several key tasks, including implementing arithmetic operations, designing an intuitive user interface, and ensuring the accuracy and precision of the calculations. By completing each of these tasks carefully and thoroughly, we can create a polynomial calculator that is both functional and user-friendly.

Looking ahead, there are several potential areas for future development of the polynomial calculator, including the addition of new operations or functions, the incorporation of graphical or interactive elements to enhance the user experience, and the optimization of the code for faster and more efficient calculations.

# Bibliography

The references that were consulted by the student during the implementation of the homework:

1. <https://regex101.com/>
2. Lab teacher explanations
3. <https://dsrl.eu/courses/pt/>