DOCUMENTATION

ASSIGNMENT *1*

STUDENT NAME: SĂVIANU BOGDAN-CASIAN
GROUP: 30223

CONTENTS

1. Assignment Objective

The objective of this first assignment was ultimately to create an application with the functionality of a polynomial calculator, able to make operations such as addition, subtraction, all the way to integration. But in order to achieve this final objective, I had to divide the problem into smaller sub-objectives:

- Analyzing the problem and identifying the requirements
- Designing the application, making the UML diagram
- Implementing the application
- Testing the calculator and making the necessary refinements

2. Problem Analysis, Modeling, Scenarios, Use Cases

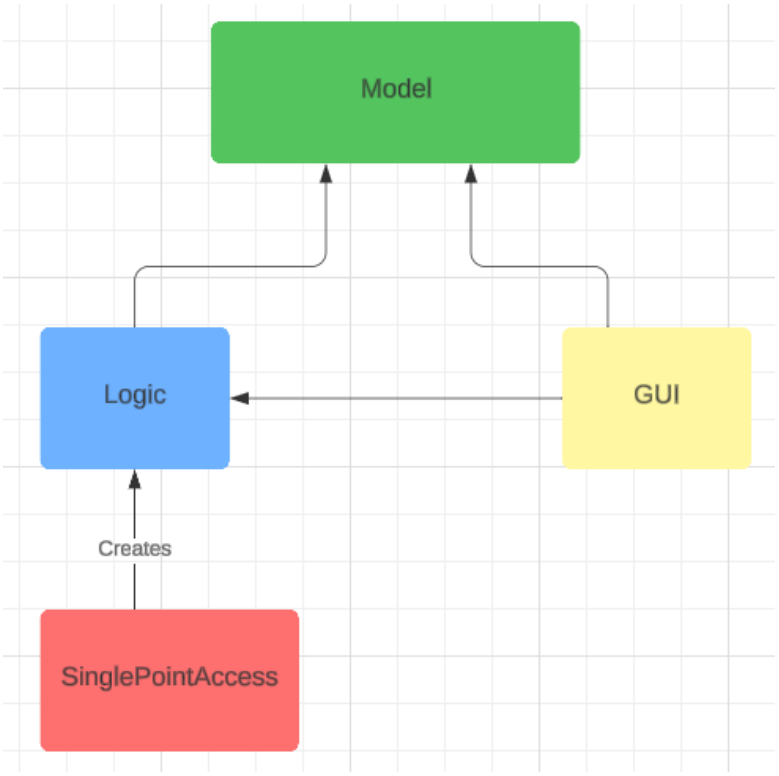
The first sub-objective I took on was modelling the polynomial. At first I thought about having just the Polynomial class, and while that would have been fine, I realized that since each term of the expression has a coefficient and a degree, I could go even further and model that into its own class called Monomial. Then the Polynomial class would contain a list of monomials or a map <Integer, Monomial>, I will go more in-depth on which I chose and why when I get to the implementation chapter.

Having modelled my polynomials, I could then start implementing operations on them. The addition, subtraction, multiplication take in two terms and return the resulting polynomial, The division is very similar but it can sometimes return a remainder as well so we have to take that into consideration. Unlike these first operations, the differentiation and integration operate on only one term and the result is a single polynomial as well.

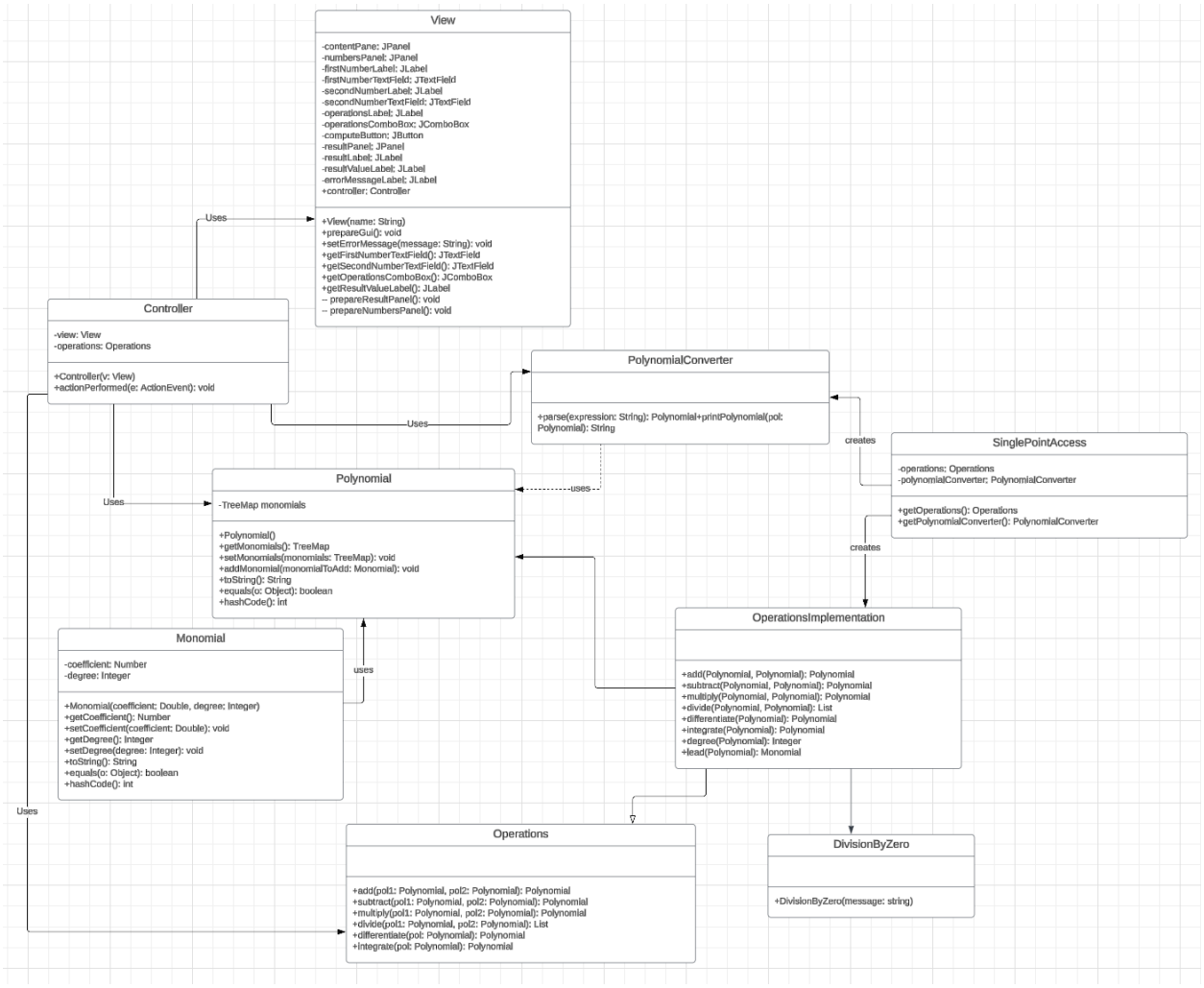
In the real world we often need to a calculator for a plethora of reasons. Maybe an engineer wants to make absolutely sure the calculations are correct so the engine he is designing won't fail miserably or maybe we have a friend who needs help with his homework or is just too lazy to do it by himself. Whatever the circumstance may be, this application is designed to provide as much help as possible when it comes to polynomials.

3. Design

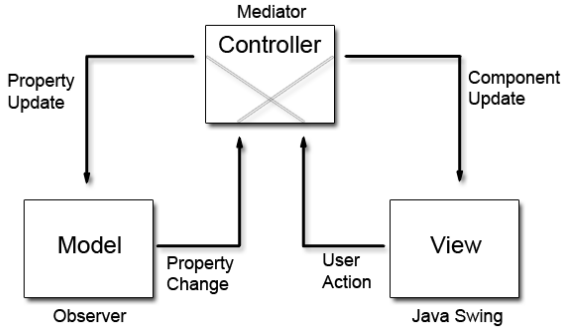
Package Diagram:



UML Diagram



I divided my source code into 3 main packets, one for modelling the equation, one for the logic which consists of the operations, the parsing of the input and an error I throw in case of division by zero, and one last packet for the graphical user interface where I included the Controller and View classes in order to respect the MVC architecture.



As fas as data structures go, I opted to model my polynomials as TreeMaps containing a key equal to the degree of the monomial, and the monomial itself as the value. I chose this data structure due to its ability to keep data sorted after the *natural ordering* of the keys.

4. Implementation

In the **model** package I represented the following classes:

- Monomial

Attributes:

- Coefficient
- Degree

Methods:

- equals, hashCode and toString

- Polynomial

Attributes:

- TreeMap<Integer, Monomial>

Methods

- addMonomial
- toString, equals and hashCode

In the **logic** package I represented:

- Operations - Interface

Methods:

- Headers for all operations to be implemented

- OperationsImplementation

Methods

- Implementations for all 6 methods in the interface

- PolynomialConverter

Methods

- printPolynomial
- parse – turns a String input into a Polynomial with the map<Integer, Monomial>

In the **single_point_access** package I have the following class:

- SinglePointAccess

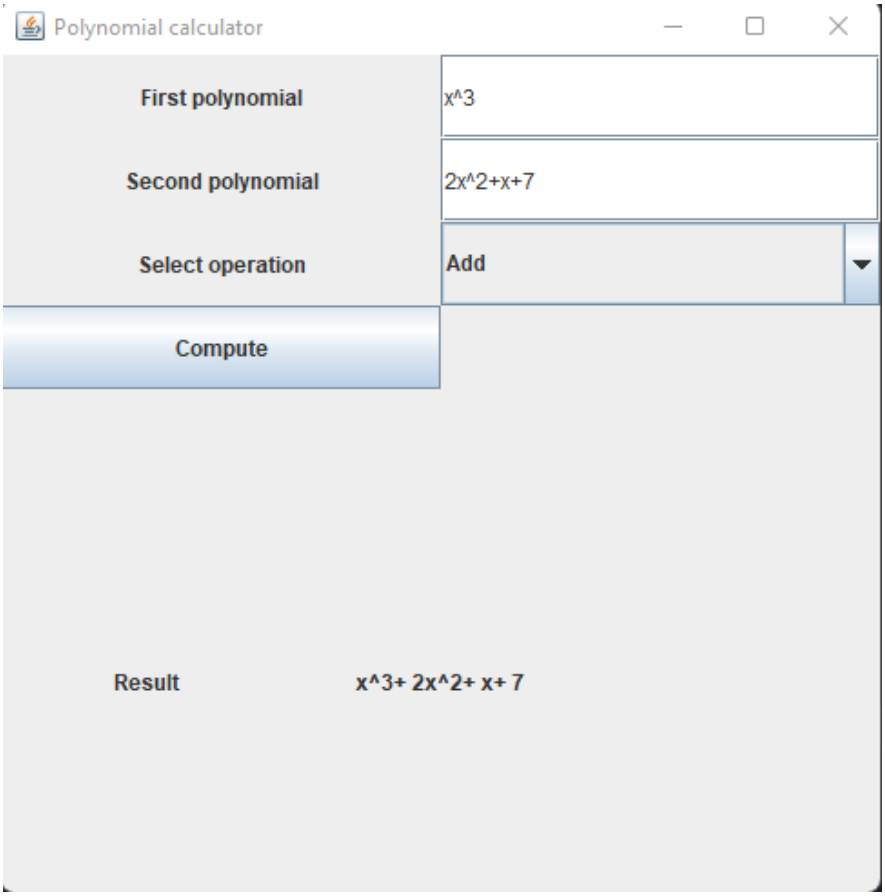
Provides a centralized way to access the operations and the parser using a static block.

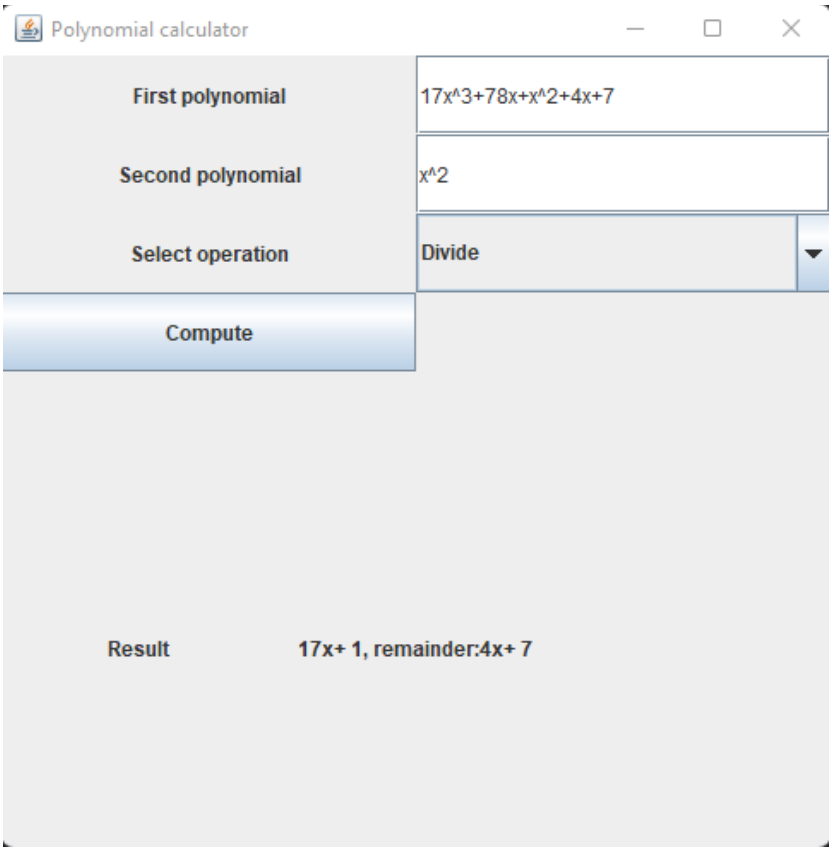
In the **gui** package:

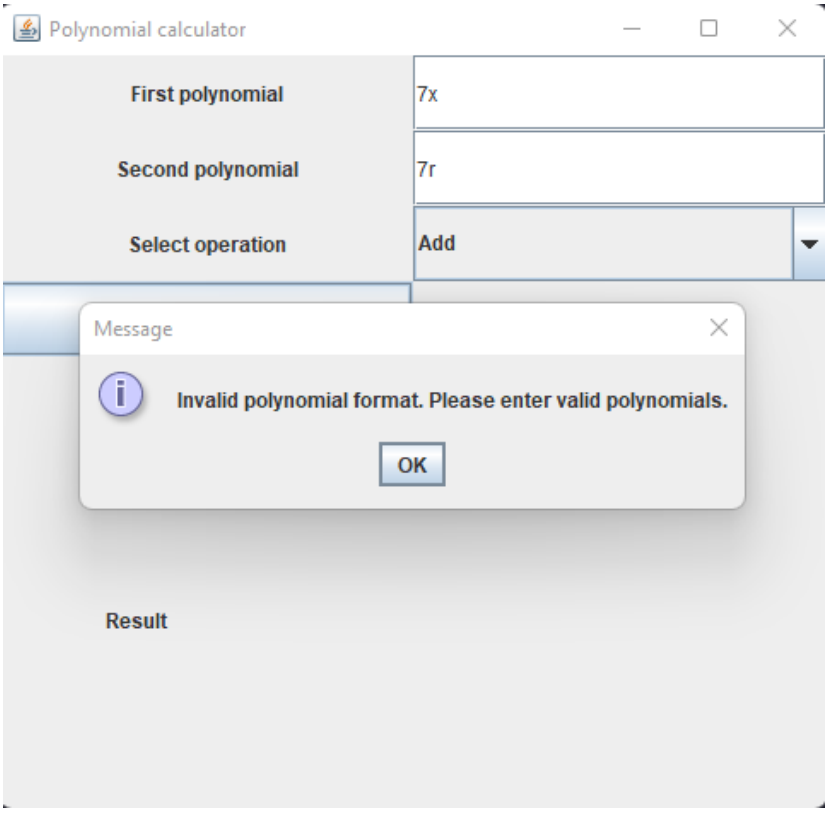
- View

Attributes

    - It contains the buttons and text fields of the GUI.
- Controller
    - It contains the logic behind the GUI.




Polynomial calculator
First polynomial
17x^3+78x+x^2+4x+7
Second polynomial
x^2
Select operation
Divide
Compute
Result
17x+ 1, remainder:4x+ 7


Polynomial calculator
First polynomial
7x
Second polynomial
7r
Select operation
Add
Message
Invalid polynomial format. Please enter valid polynomials.
OK
Result

5. Results

Using JUnit I conducted several tests per operation and 5 tests for the regular expression. Those tests should cover most of the edge cases both for the operation and the RegEx. In the end I was able to find some things that I missed in the implementation and quickly fixed them. The code now passes all of my tests.

6. Conclusions

For most of the things I used in this assignment, it was not my first contact with them, but the assignment still provided some valuable practice and more depth into unit testing, regular expressions, and, obviously, it was a good recap on polynomial long division.

7. Bibliography

https://www.w3schools.com/java/java_regex.asp
https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html
https://www.cuemath.com/algebra/long-division-of-polynomials/
https://dsrl.eu/courses/pt/materials/PT_2024_A1_S1.pdf