DOCUMENTATION

ASSIGNMENT *1*

STUDENT NAME: AVRAM ALEXANDRU-ANDREI
GROUP: 30422

CONTENTS

1. Assignment Objective 3

2. Problem Analysis, Modeling, Scenarios, Use Cases 3

3. Design 5

4. Implementation 5

5. Results 6

6. Conclusions 7

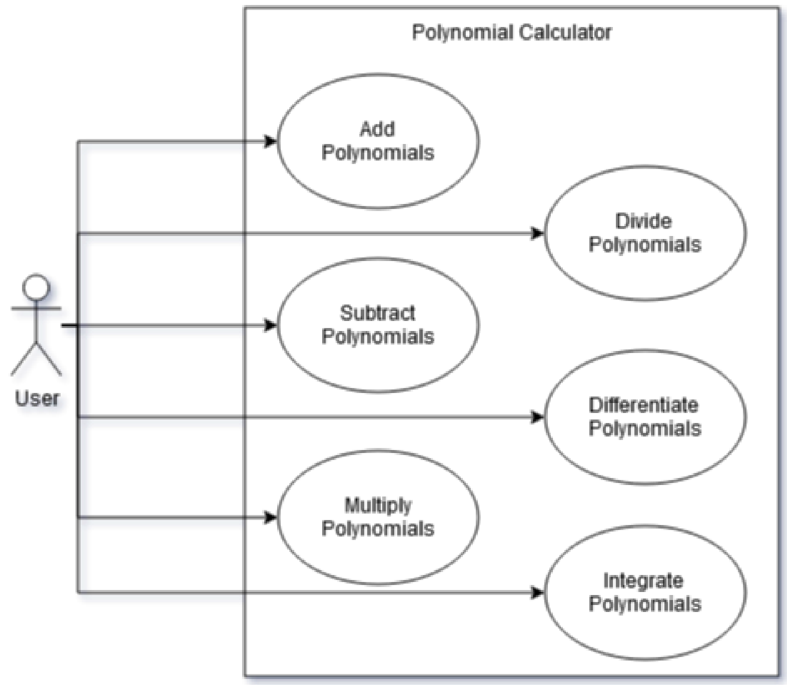
7. Bibliography 7

1. Assignment Objective

The main objective of this assignment is to create a calculator which can perform a range of operations on polynomials.
Sub-objectives:

(1) Implement Polynomial Addition Operation
(2) Implement Polynomial Subtraction Operation
(3) Implement Polynomial Multiplication Operation
(4) Implement Polynomial Division Operation
(5) Implement Polynomial Derivation Operation
(6) Implement Polynomial Integration Operation
(7) Test the code using JUnit
(8) Create a GUI using JavaFX
(9) Handle Errors

2. Problem Analysis, Modeling, Scenarios, Use Cases



The polynomial calculator should be able to carry out all the important operations one can do on polynomials: term by term addition, subtraction, multiplication, division, partial differential and integration.

Use case descriptions:

(1) Polynomial Addition

- User inputs the first polynomial
- User inputs the second polynomial
- User selects the "Add" option from the menu
- The system performs the addition operation between the two polynomials
- If there is no error, the output is displayed
- If there is an error, the program lets the user know with an appropriate message in the GUI

(2) Polynomial Subtraction

- User inputs the first polynomial
- User inputs the second polynomial
- User selects the “Subtract” option from the menu
- The system performs the subtraction operation between the two polynomials
- If there is no error, the output is displayed
- If there is an error, the program lets the user know with an appropriate message in the GUI

(3) Polynomial Multiplication

- User inputs the first polynomial
- User inputs the second polynomial
- User selects the “Multiply” option from the menu
- The system performs the multiplication operation between the two polynomials
- If there is no error, the output is displayed
- If there is an error, the program lets the user know with an appropriate message in the GUI

(4) Polynomial Division

- User inputs the first polynomial
- User inputs the second polynomial
- User selects the “Divide” option from the menu
- The system performs the division operation between the two polynomials
- If there is no error, the output is displayed
- If there is an error, the program lets the user know with an appropriate message in the GUI

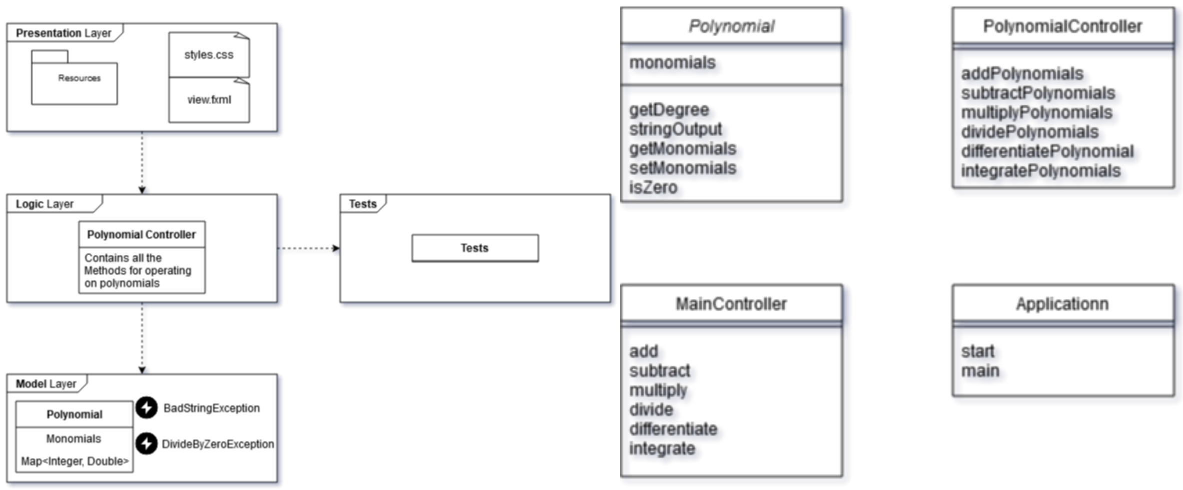
(5) Polynomial Derivation

- User inputs the polynomial in the first field
- User selects the “Differentiate” option from the menu
- The system performs the partial derivation operation on the polynomial
- If there is no error, the output is displayed
- If there is an error, the program lets the user know with an appropriate message in the GUI

(6) Polynomial integration

- User inputs the polynomial in the first field
- User selects the “Integrate” option from the menu
- The system performs the integration operation on the polynomial
- If there is no error, the output is displayed
- If there is an error, the program lets the user know with an appropriate message in the GUI

3. Design



The main class of the calculator is the “Polynomial” class, which describes a polynomial as a Map<Integer, Double>, between the power and coefficient of each term. To do this, the HashMap data structure has been used here.
The reason behind using a hash map is that it makes certain operations, like polynomial addition easier, by grouping the terms by their exponent.
The “PolynomialController class” implements the addition, subtraction, multiplication, division, derivation and integration operations on polynomials as static methods, so they can be easily accessed in other classes. Each method in this class is specialized for a single operation, offering the possibility for easier further development.

4. Implementation

(1) Polynomial

The “Polynomial” class contains a “monomials” field, defined as a map between the exponent of each monomial (Integer) to the coefficient of the same term (Double).
This implementation enables for quick coefficient operations, by searching the terms by power, as well as easy ordering of powers in the polynomial.

(2) PolynomialController

This is the class where all of the operations are implemented. Each method describes an operation which can be performed on one (derivation, integration), or between two (addition, subtraction, multiplication, division) polynomials.
The division method behaves fairly different from the rest, in the sense that it does not return a Polynomial, but an array of 2 Polynomials, where result[0] is the quotient of the division and result[1] is the remainder, in order to account for all cases of division, even when the polynomials do are not exactly divisible.

(3) MainController

“MainController” is the class that links the frontend and the backend of the application. It is responsible for taking input from the user, passing it to the function that is required, and outputting it to the FXML Graphical User Interface.

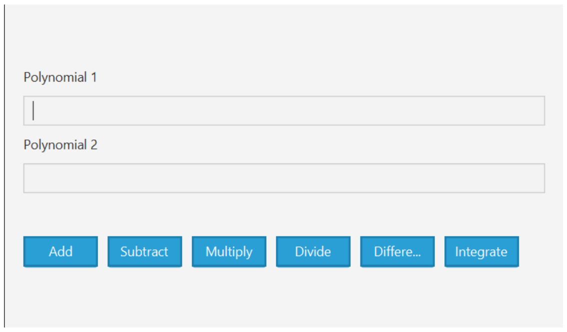
(4) Applicationn

The “Applicationn” class extends “Application” and takes care of the runtime of the app. It gets the resources needed, loads them and starts the application in a new window.

(5) Exception Classes

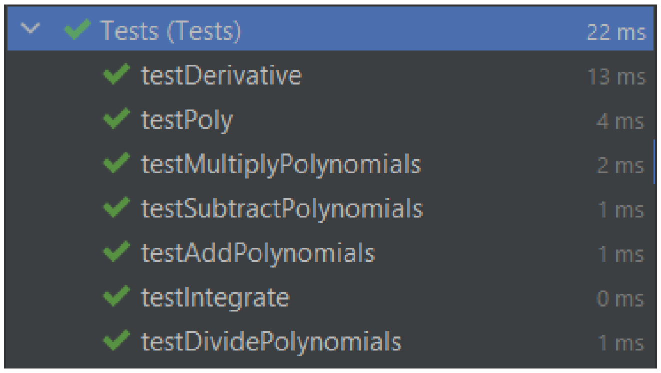
These classes are implemented specifically for error handling. There is two of them: “BadStringException” and “DivideByZeroException”. The first one manages user inputs and is thrown when a string which cannot be broken down correctly by the regex patterns is given, while the latter is thrown when the “dividePolynomials” method is called with the divisor parameter as an empty or “0” polynomial.

(6) Graphical User Interface



The application has 2 text fields, one for inputting each polynomial.
6 buttons are also present at the bottom of the interface: one for each operation which the calculator can perform.
If the Differentiate or Integrate operation is selected, only the first polynomial is taken into consideration.

5. Results



```
Tests (Tests)                 22 ms
    testDerivative            13 ms
    testPoly                   4 ms
    testMultiplyPolynomials    2 ms
    testSubtractPolynomials    1 ms
    testAddPolynomials         1 ms
    testIntegrate              0 ms
    testDividePolynomials      1 ms
```

(1) testPoly()

Evaluates the polynomial. Checks that if we input a polynomial, transform the data into a Map and turn it back into a string, the output will be correct and the polynomial will hold its values and properties.

(2) testAddPolynomials()

Checks correctness of addition operation between two polynomials.

(3) testSubtractPolynomials()

Checks correctness of subtraction operation between two polynomials.

(4) testMultiplyPolynomials()

Checks correctness of multiplication operation between two polynomials.

(5) testDividePolynomials()

Checks correctness of division operation between two polynomials

(6) testDerivative()

Checks correctness of derivation operation on a polynomial.

(7) testIntegrate()

Checks correctness of integration operation on a polynomial.

6. Conclusions

In conclusion, the assignment provided an opportunity to explore Collections in Java, a brief theory of design patterns and review knowledge on polynomials.

During the course of this assignment, I learned how to work with JavaFX along with CSS in order to create an interactive GUI for my application, and also understood the key features and functionalities of Junit and automated testing in Java.

Moving forward, there are several potential developments that could be explored related to this topic. For example, the operation algorithms could be improved for better complexities, the calculator could receive other features, such as arithmetic calculations or the derivative and integral functions could be extended to work with any function, not just polynomials. Additionally, the GUI could be made more pleasant and easy to use, while also adding more exceptions and messages prompted to the user. Overall, the assignment was informative and established a sturdy groundwork for delving deeper into these subjects in the future.

7. Bibliography

Example:

1. *Bruce Eckel, Thinking in Java (4th Edition), Publisher: Prentice Hall PTRUpper Saddle River, NJUnited States, ISBN:978-0-13-187248-6 Published:01 December 2005.*
2. *What are Java classes? - www.tutorialspoint.com*
3. *Jakob Jenkov, Java Map, 21 September 2020 - https://jenkov.com/tutorials/java-collections/map.html*
4. *Using FXML to Create User Interface - https://docs.oracle.com/javafx/2/get_started/fxml_tutorial.htm*
5. *Refactoring Guru - https://refactoring.guru/design-patterns*
6. *How to Add a CSS Stylesheet in FXML - https://stackoverflow.com/questions/23975897/how-to-add-a-css-stylesheet-in-fxml*
7. *Drawio - https://app.diagrams.net/*