DOCUMENTATION

ASSIGNMENT *1*

STUDENT NAME: TUDOR ALEXIA-CRISTINA

GROUP: 30424

# 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

Design and implement a polynomial calculator with a dedicated graphical interface that allows the user to insert polynomials, select the mathematical operation (addition, subtraction, multiplication, division, derivative, integration), and display the result.

# Problem Analysis, Modeling, Scenarios, Use Cases

* Problem Analysis

In mathematics, a polynomial is an expression consisting of indeterminates (also called variables) and coefficients, that involves only the operations of addition, subtraction, multiplication, and positive-integer powers of variables. An example of a polynomial of a single indeterminate x is x^2 − 4x + 7. An example with three indeterminates is x^3 + 2xyz^2 − yz + 1.

Polynomials appear in many areas of mathematics and science. For example, they are used to form polynomial equations, which encode a wide range of problems, from elementary word problems to complicated scientific problems; they are used to define polynomial functions

Adding, subtracting and multiplying polynomials is very similar to adding, subtracting and mutiplying exponents and radicals. We have to make sure we are doing the operations on the terms that have the same exponents on matching variables.

* Modeling

To use the calculator, given the operations I need to implement, I thought about the user to simply enter the polynomial expressions into the input Text Fields and select the desired operation using the corresponding button. The result of the operation will be displayed in the output text field.

-Addition

-Subtraction

-Multiply

-Divide

-Derivate

-Integrate

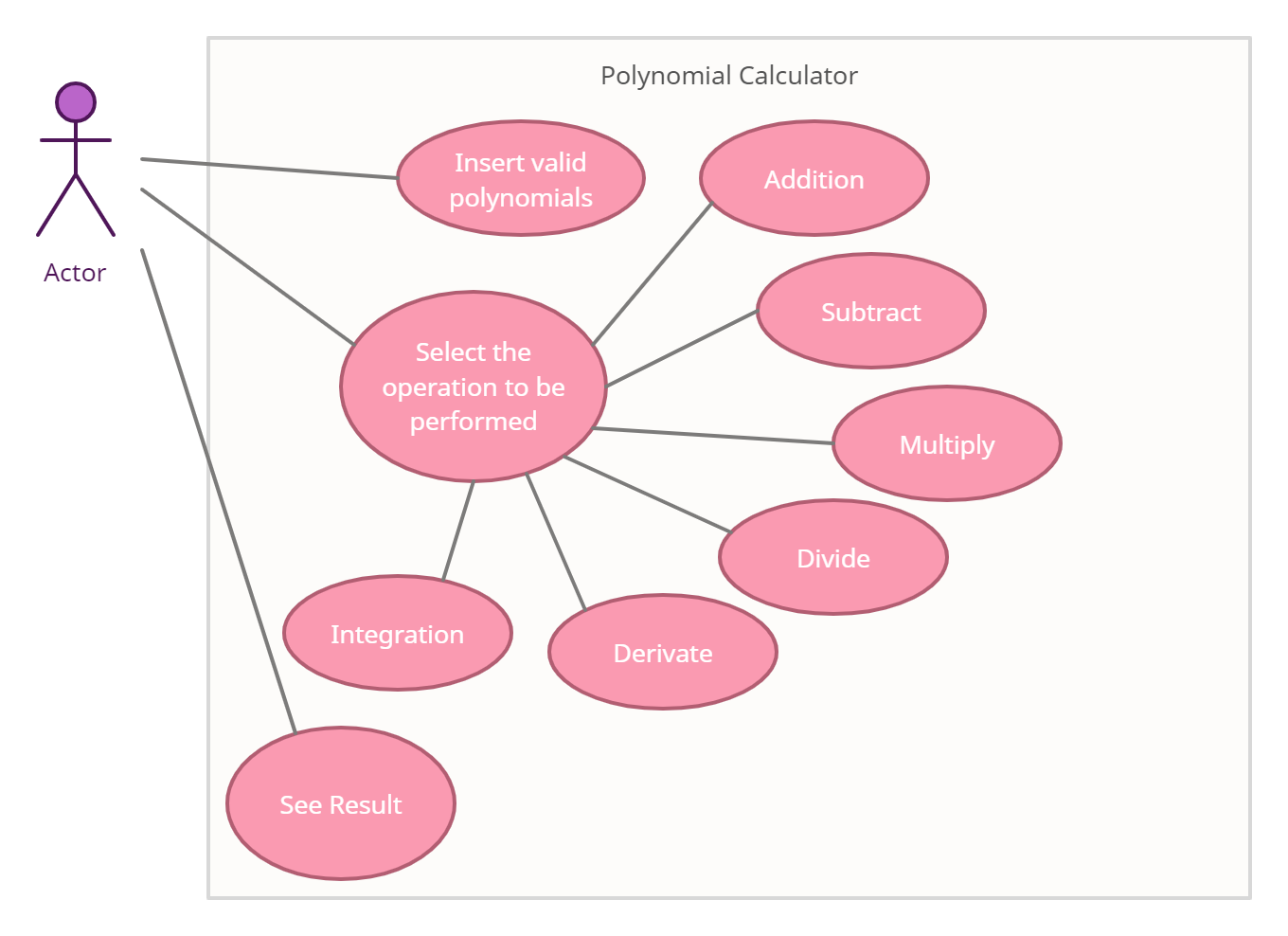
If the given input is not a proper representation of a polynomial, an Exception will be thrown and an operation wouldn’t be done unless the input is valid.

* Scenarios and Use Cases

They add value because they better clarify how the system should behave while also brainstorming what could go wrong.

A well-designed interface and user-friendly experience can greatly enhance the interaction between the user and the polynomial calculator system, making it easier and more intuitive for the user to perform mathematical operations on polynomial expressions.

The user should insert valid written polynomial in the Text Field and select an operation in order for the third Text Field to display the result. If the given input is not a proper representation of a polynomial, an Exception will be thrown and an operation wouldn’t be done unless the input is valid.



# Design

* Packages:

Java packages are used to group together multiple modules and to connect related classes and interfaces.

The packages and classes are designed based on the Model-View-Controller(MCV) design pattern for a better implementation.

-Graphical User Interface: contains the classes implementing the graphical user interface

-Data Models: contains the classes modeling the application data

-Business Logic: contains the classes implementing the mathematical operations functionality

* Class Diagram:



I’ve split the problem into smaller classes for a better visualization and interpretation of the solution. This is a general rule for Java app development.

* Data Structures:

The data structures that I’ve used are primitive data types: integers and doubles, and maps such as HashMaps and TreeMap. I have created new objects such as Polynoms. Within that sense, I used HashMaps instead of Arrays due to being more efficient, since it has a faster access of elements due to hashing technology and the fact that allows duplicate values, but not duplicate keys. This is perfect for storing monomials.

* Algorithms:

Operations on polynomials are done in a similar manner as for real numbers, but we need to consider the degree of each monomial.

-Addition: obtained by summing together the coefficients with the same degree and adding the other elements as they are

Ex:

-Subtraction: obtained by subtracting the coefficients with the same degree and adding the other elements as they are, multiplying by –1 the second polynomial.

Ex:

-Multiplication: obtained by multiplying every term from one polynomial to the other polynomial’s terms. We check if the result’s degree is already present in the HashMap, if so we add the coefficient to the already existing one.

Ex:

-Division: obtained by following a series of steps.

Step 1 - Order the monomials of the two polynomials of the two polynomials P and Q in descending order according to their degree.

Step 2 - Divide the polynomial with the highest degree to the other polynomial having a lower degree (let’s consider

that P has the highest degree)

Step 3 – Divide the first monomial of P to the first monomial of Q and obtain the first term of the quotient

Step 4 – Multiply the quotient with Q and subtract the result of the multiplication from P obtaining the remainder of

the division

Step 5 – Repeat the procedure from step 2 considering the remainder as the new dividend of the division, until the

degree of the remainder is lower than Q.

-Derivation: obtained by going through every term and multiplying the coefficient with the degree and decrementing 1 from their degree.

Ex:

-Integration: obtained by going through every monomial and dividing their coefficients with the value of degree plus 1. The exponent also increments by one.

Ex:

# Implementation

* Class Design:

1. Data Models

a. Polynomial: this class has a variable of type HashMap<Integer, Double>, where I store the polynomial’s degrees and coefficients. The HashMap is hashed after the degrees, since there can’t be duplicates and the coefficients are stored in the values.

Constructors:

-public Polynomial(){}: default constructor.

-public Polynomial(String input){}: constructor that transforms using the stringToInt method the given input String into a HashMap with the received values.

Methods:

- public HashMap polynomialStringToHash(String input): using pattern matching we extract the coefficients and the degrees from the given String and store them into a HashMap with degree as key and coefficient as value. We return the resulted polynomial.

- public TreeMap orderHashMap(HashMap<Integer, Double> polynomials): transforms the HashMap into a TreeMap that stores the pairs of key and values in descending order by degree.

- public String finalPolynomial(Polynomial a): print the polynomial as a string in reverse order, from highest to lowest degree.

- public String listTree(TreeMap<Integer, Double> poltree): list the TreeMap, depending on the degrees and coefficients.

- public String listPolynomial(): list the polynomial as a string

- public Double getCoeff(TreeMap<Integer, Double> poltree): gets the first coefficient of the first key from a TreeMap.

- public Integer getHighestGrade(TreeMap<Integer, Double> poltree): gets the highest degree of a TreeMap- the degree of the first key.

- public Integer getNextHighestGrade(TreeMap<Integer, Double> poltree): gets the next degree from a TreeMap. Deletes the first key and returns the degree of the now first key.

1. Business Logic

a. Operations: this class has a variable operations of type String and an object Polynomial. In this class we stored all the performable operations.

Constructors:

- public Operations(): default constructor

Methods:

- public void setP1(Polynomial p1): sets the first polynomial for the operation

- public Polynomial compute(Polynomial a, String operation): gets the second polynomial and the operations to perform in the String. Using a switch we call the corresponding method for the operation.

- private Polynomial add(Polynomial a, Polynomial b): adds the two given polynomials and returns the result as polynomial

- private Polynomial substraction(Polynomial a, Polynomial b): subtracts the two given polynomials and returns the result as polynomial

- private Polynomial division(Polynomial a, Polynomial b): divides the two given polynomials and returns the result as polynomial

- private Polynomial integration(Polynomial a): integrates the given polynomial and returns the result as polynomial

- private Polynomial multiplication(Polynomial a, Polynomial b): multiplies the two given polynomials and returns the result as polynomial

- private Polynomial derivative(Polynomial a): derivates the given polynomial and returns the result as polynomial

b. ValidString: this class is called when the user introduced a String in the TextField and we want to validate it.

Constructors:

- public ValidString(){}: default constructor

- public ValidString(String text): gets the String and calls the method isStringValid to check if it’s correctly written.

Methods:

- public static boolean isStringValid(String text) throws InputException: verifies if the string is correctly written. Checks if it contains a x and uses a pattern, matcher in order to identify the coefficients and degree. If not then it returns false and throws the Exception.

c. InputException: the exception thrown for bad inputs

Constructors:

- public InputException(String text): prints the string when the exception is thrown

1. Graphical User Interface

a. Graphic

Constructors:

- public Graphic(): the default constructor, that generates the JButtons, JFrames, JTextFields, JButtons and JPanel

Methods:

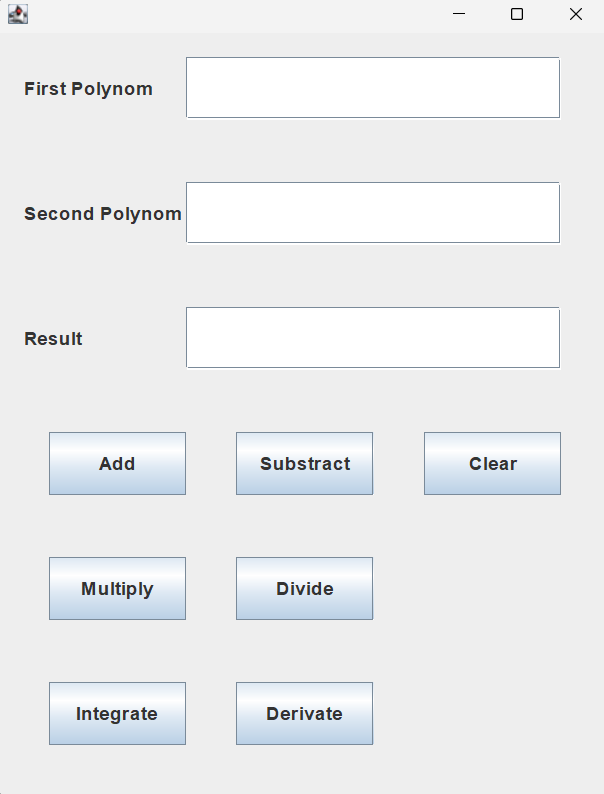
- public void actionPerformed(ActionEvent e): interprets the JButton pressed by the user and computes the given operation.

1. App Class:

The class is only used for creating the interface of the application and to easily run it.

* Graphical user interface:

The interface it’s simple and user friendly, done with Java Swing:



The user can enter their information into any of the text fields. If an invalid input is entered, an exception is thrown when a button is pressed; otherwise, if both Text Fields have valid inputs, the user can press any of the Operations buttons to compute the selected operation.

# Results

After implementing the operations I’ve run some test to check the correctness of the algorithms.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operation | Polynomial one | Polynomial two | Expected Output | Test Result |
| Addition | 3x^2-8x+1 | 5x^2+x-2 | 8.0x^2 - 7.0x - 1.0 | Passed |
| Subtraction | 5x^2+x+2 | 5x^2+x+2 | 0 | Passed |
| Multiply | 3x^2-8x+1 | 5x^2+x-2 | 15.0x^4 - 37.0x^3 - 9.0x^2 + 17.0x - 2.0 | Passed |
| Divide | x^3-2x^2+6x-5 | 2x^2-1 | 0.5x - 1.0 | Passed |
| Derivate | 3x^2-8x+1 | - | 6.0x - 8.0 | Passed |
| Integrate | 3x^2-8x+1 | - | x^3 - 4.0x^2 + 1.0 | Passed |

# Conclusions

The Polynomial Calculator was a great opportunity for me to test and enhance my software development and OOP abilities. Along with fundamental OOP knowledge and programming skills, these kinds of projects also necessitate excellent time management skills and a good understanding of the problem's requirements and constraints. It was a challenging exercise that demanded significant effort from me to complete, but it was impressive how I learned about project management - designing, implementing, and testing a product - in a short time.

One of the lessons I took from this project was working with JavaSwing more. I’ve wanted to further customize my program, but I focused on the algorithms more. Furthermore, working with regex proved to be one of the most challenging parts of the assignment. Nevertheless, I was able to gain a lot of knowledge and insight into their use and implementation.

Overall, this project was an excellent opportunity for me to test my programming skills and review polynomial operations quickly.

For future improvements, the application can perform some of the following:

- finding the roots of a polynomial

- computing the plot of a polynomial

# Bibliography

1. *Java Regex -* [*https://www.javatpoint.com/java-regex*](https://www.javatpoint.com/java-regex)
2. *Java StringBuilder -* [*https://docs.oracle.com/javase/7/docs/api/java/lang/StringBuilder.html*](https://docs.oracle.com/javase/7/docs/api/java/lang/StringBuilder.html)
3. *Java Swing- Jpanel with examples -* [*https://www.geeksforgeeks.org/java-swing-jpanel-with-examples/*](https://www.geeksforgeeks.org/java-swing-jpanel-with-examples/)
4. *Java Swing -* [*https://www.javatpoint.com/java-swing*](https://www.javatpoint.com/java-swing)
5. *Java Junit -* [*https://www.vogella.com/tutorials/JUnit/article.html*](https://www.vogella.com/tutorials/JUnit/article.html)
6. *TreeMap -* [*https://www.javatpoint.com/java-treemap*](https://www.javatpoint.com/java-treemap)
7. *Java Swing Dialogs -* [*https://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html*](https://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html)
8. *Java HashMap -* [*https://www.w3schools.com/java/java\_hashmap.asp*](https://www.w3schools.com/java/java_hashmap.asp)
9. *Polynomial -* [*https://en.wikipedia.org/wiki/Polynomial*](https://en.wikipedia.org/wiki/Polynomial)
10. *Use Cases -* [*https://www.usability.gov/how-to-and-tools/methods/use-cases.html*](https://www.usability.gov/how-to-and-tools/methods/use-cases.html)
11. *Use Case Diagram -* [*https://app.creately.com*](https://app.creately.com)
12. *Class Diagram Drawing application - https://app.diagrams.net*