Technical University of Cluj-Napoca

Programming Techniques

Assignment 1

Polynomial Calculator

STUDENT NAME: CRISTIAN CASIAN-CRISTI

GROUP: 30422

CONTENTS

1. Assignment Objective 3
2. Problem Analysis, Modelling, Scenarios, Use Cases 3
3. Design 4
4. Implementation 7
5. Results 9
6. Conclusions 9
7. Bibliography 9

1. Assignment Objective

Propose, design and implement a system for polynomial processing. Consider the polynomials of one variable and integer coefficients.

2. Dimension of the problem

2.1 Problem analysis

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.

A polynomial in a single indeterminate x can always be written (or rewritten) in the form.

$$\sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

where $a_0, a_1, a_2 \ldots a_n$ are constants and x is the indeterminate. The word "indeterminate" means that x represents no particular value, although any value may be substituted for it. The polynomial consists of a list of certain terms, also called monomials, for example $3x^2$ is a monomial. The coefficient 3, the indeterminate is x and the degree is 2. Forming a sum of several terms produces a polynomial, like the following one: $3x^2 - x + 4$, which has three terms with different exponents: the first is degree two, the second is degree one, and the third is degree zero.

This way of representing polynomials can be used to perform the most common operations on polynomials: addition, subtraction, multiplication, division, differentiation and integration.

2.2 Modelling the problem

The user will be able to use the functions of calculator by introducing in the interface, two polynomials. The client will have to fetch the two polynomials so the application can store values in "ordinary polynomials" then the client will can choose a specific operation to perform, such as:

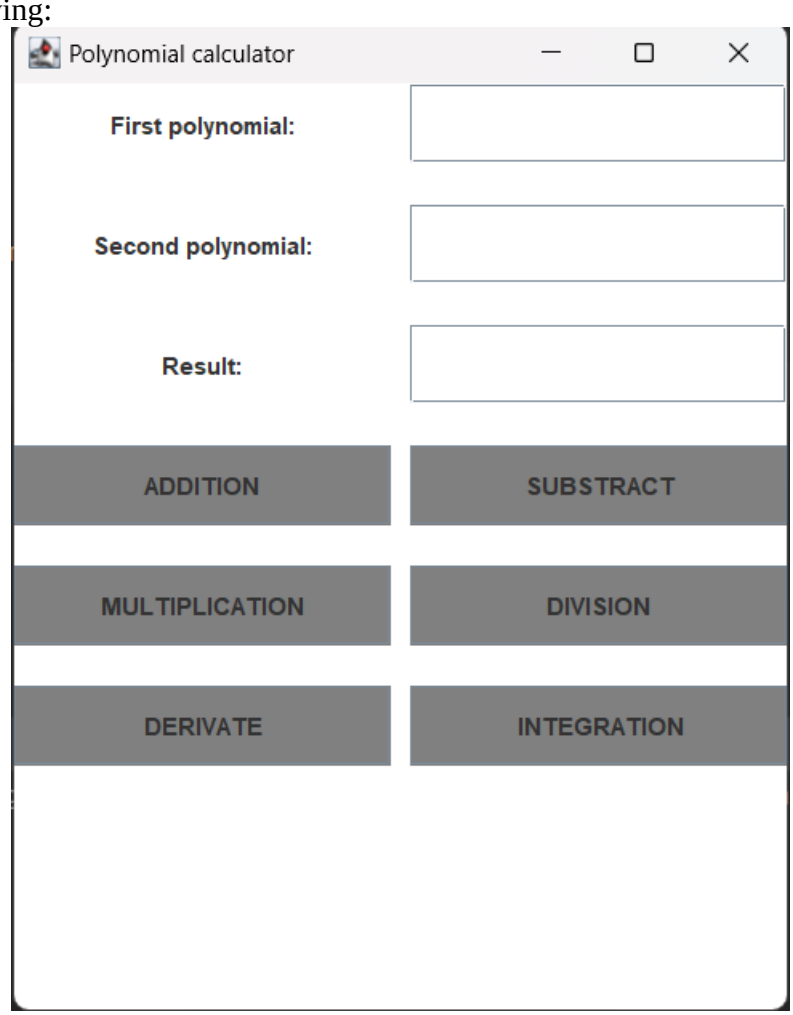
- Addition of two polynomials
- Subtraction of two polynomials
- Multiplication of two polynomials
- Division of two polynomials
- Differentiation of a polynomial
- Integration of a polynomial

The result of the chosen operation will be displayed in the interface. The user will also have the option to set the current result as one of the operands for the following operation, but the client must fetch again in the new operand, if the previous result is not proper polynomial, the application will throw an error box.

2.3 Scenarios and use cases

A use case is a methodology used in system analysis to identify, clarify and organize system requirements. The use case is made up of a set of possible sequences of iterations between system and users in a particular environment and related to a particular goal.

The use cases are strongly connected with the user steps. This is the reason why I tried to design my interface in a very friendly mode and the result is the following:



The user will introduce the two polynomials in the corresponding TextFields, and the will fetch them so that the application will generate internally those two polynomials. If he/she wants to perform a certain operation, the user will have just to press the right button, and the result will be written in the result TextField. If the user wants to make the differentiation or integration, he/she must write the polynomial in the first TextField. The user must pay attention to follow the format of the polynomial, which appears in the GUI.

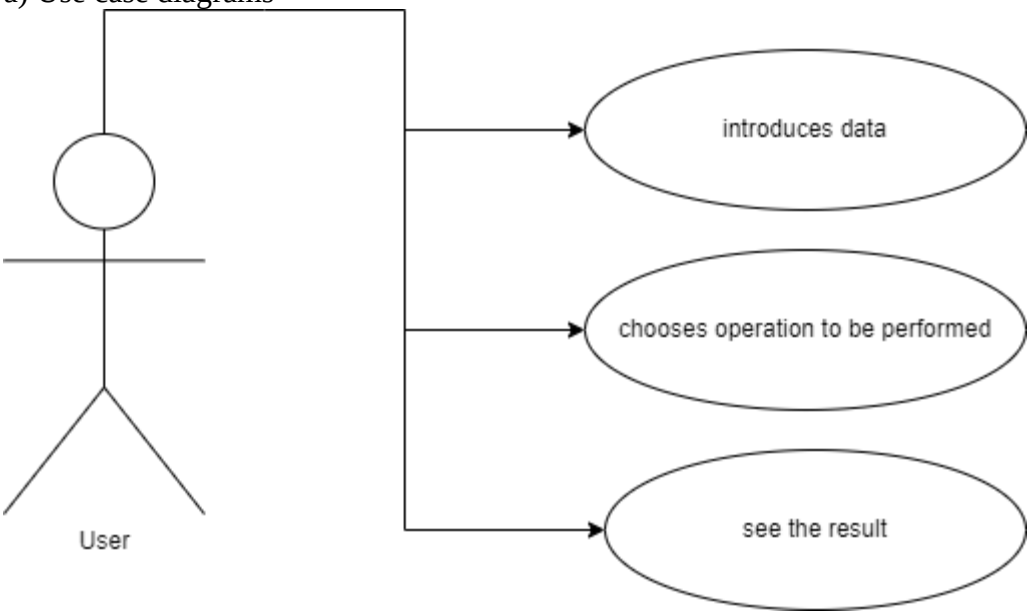
I assume that the user's input is written correctly, otherwise an error dialog will be displayed on the screen, telling you to respect the format, these includes the scenarios that can appear as result of not respecting the corresponding type, also a polynomial mustn't be equal with "0" or "0x^0" – this is not permitted, otherwise the application will not work.

The format of the polynomial must be in the following way: "$a_n \times (n) + a_{n-1} \times (n-1) + \cdots + a_0$".

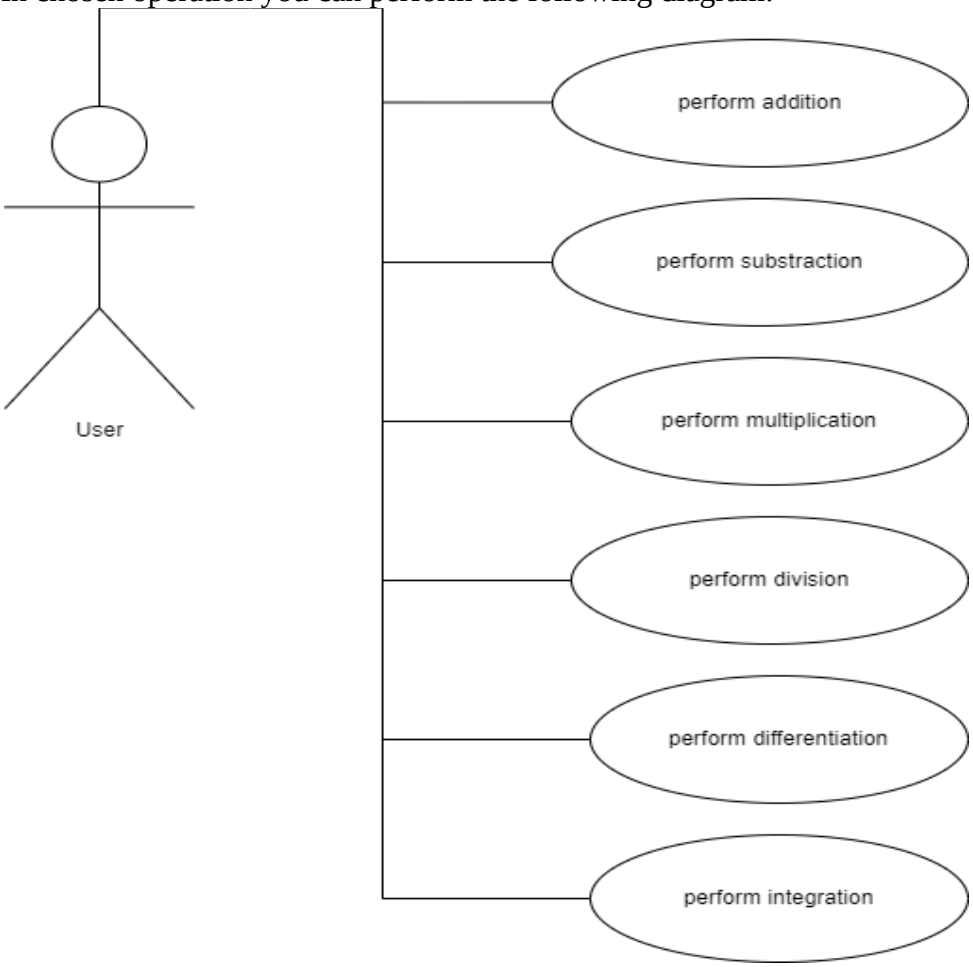
3. Design

3.1 Diagrams

a) Use case diagrams

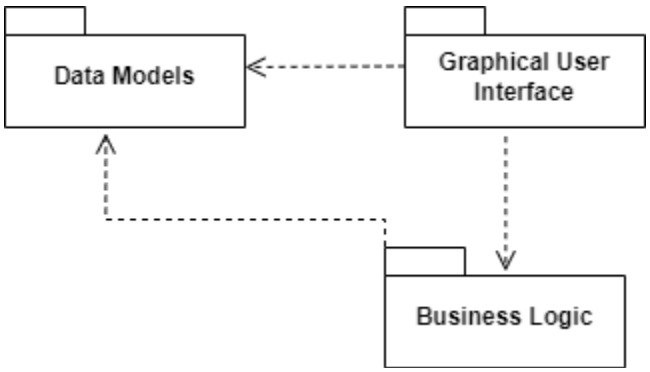


In chosen operation you can perform the following diagram:



In our case the user interacts with the application. He/she can perform operations with two polynomials like: addition, subtraction, multiplication, division, differentiation, and integration.

b) Packages diagram

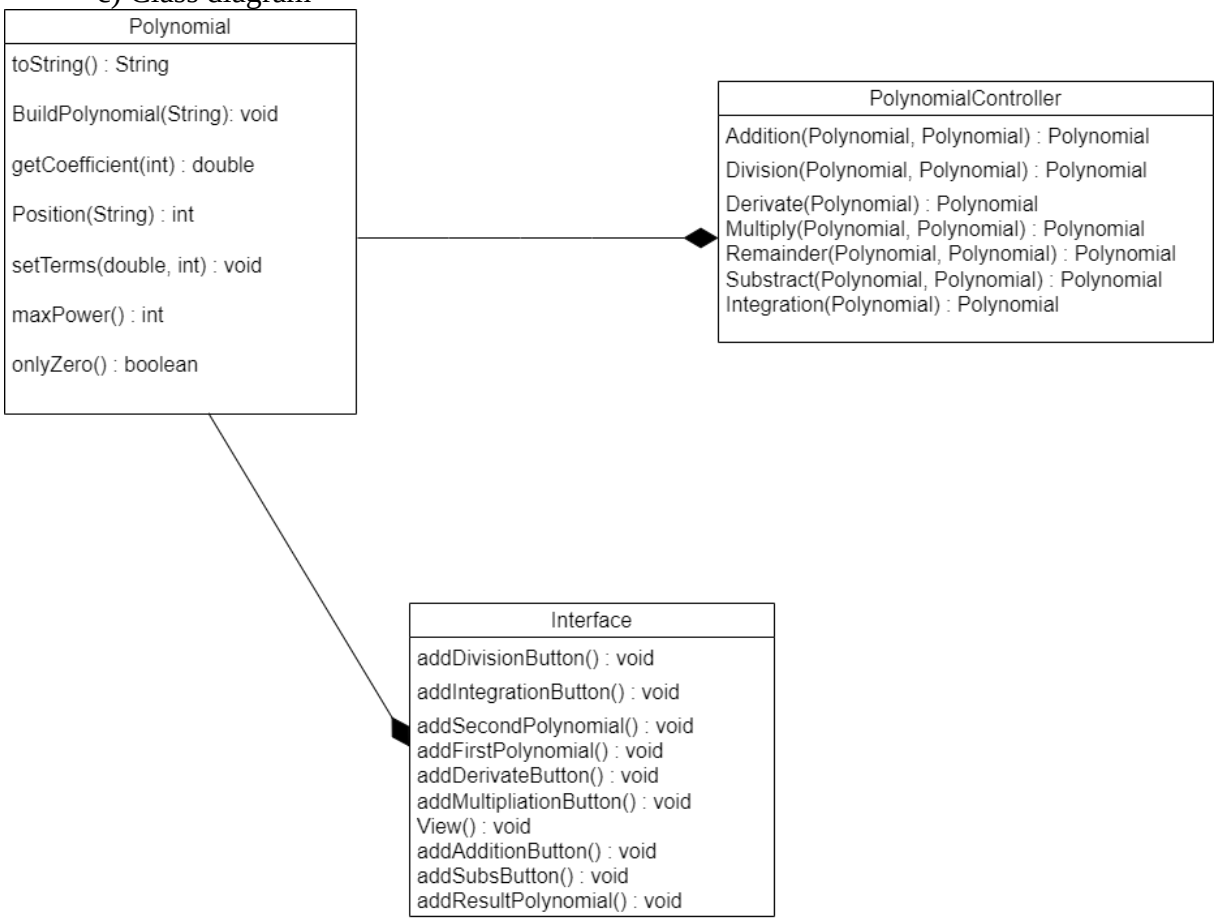


Graphical User Interface package contains the classes implementing the graphical user interface.

Business Logic package contains the classes implementing the mathematical operations functionality.

Data Models package contains the classes modelling the application data (e.g., polynomial, monomial, …).

c) Class diagram



d)Data Structure

Map: A map is a data structure that allows for the storage of key-value pairs, where each key is unique. In our polynomial calculator, you may use a map to store the coefficients of each term in a polynomial, with the exponent of the term serving as the key and the coefficient as the value.

HashMap: It is a collection type that stores key-value pairs. HashMap is implemented using an array of buckets, where each bucket contains a linked list of entries. When a key-value pair is added to the HashMap, it is stored in a bucket based on the has code of the key, and if there is a collision, the entry is added to the linked list in that bucket. The key is used to calculate the hash code, which determines the index of the bucket in which the entry is stored. Therefore, HashMap is a useful data structure for many applications that require fast access to data, including data indexing and searching.

4. Implementation

Polynomial Class

This class represents a polynomial and has the following fields and methods:

Fields:

terms: a **Map<Integer, Double>** to store the polynomial terms as keys (exponents) and values (coefficients).

Methods:

Polynomial(): a constructor that initializes the terms map with an empty HashMap.

Position(String string): a static method that takes a string as input and returns the position of the first occurrence of the character 'x' in the string. If 'x' is not present in the string, it returns -1.

BuildPolynomial(String string): a method that takes a string as input and parses it to populate the terms map with the polynomial terms. The input string should be in the form of a space-separated list of terms, where each term is a coefficient followed by 'x' (optional) followed by '^' (optional) followed by the exponent. For example, "2x^2 + 3x - 4" represents the polynomial 2x^2 + 3x - 4. If the exponent is not present, it is assumed to be 0. If the coefficient is not present, it is assumed to be 1. If a term has a minus sign in front of it, the sign of its coefficient is set to negative.

toString(): a method that returns a string representation of the polynomial in the standard form. For example, "2.0*x*^*2* - *3.0*x + 4.0" represents the polynomial 2x^2 - 3x + 4. If a term has a positive coefficient, the sign is omitted. If a term has a coefficient of 1, the coefficient is omitted. If a term has a coefficient of 0, it is omitted.

getExponents(): a method that returns a Set<Integer> containing the exponents of the polynomial terms.

getCoefficient(int exponent): a method that takes an integer exponent as input and returns the coefficient of the term with that exponent. If there is no such term, it returns 0.

setTerms(double coefficient, int exponent): a method that sets the coefficient of the term with the given exponent. If the coefficient is 0, the term is removed from the terms map.

maxPower(): a method that returns the highest exponent in the terms map.

onlyZero(): a method that returns true if all the coefficients in the terms map are 0, false otherwise.

PolynomialController Class

This class contains static methods to perform various operations on polynomials. It has the following methods:

Methods:

Derivate (Polynomial polynomial): a static method that takes a Polynomial object as input and returns its derivative as another Polynomial object.

Integration (Polynomial polynomial): a static method that takes a Polynomial object as input and returns its indefinite integral as another Polynomial object.

Addition (Polynomial polynomial1, Polynomial polynomial2): a static method that takes two **Polynomial** objects as input and returns their sum as another Polynomial object.

Substract(Polynomial polynomial1, Polynomial polynomial2): a static method that takes two **Polynomial** objects as input and returns their difference as another Polynomial object.

Multiply (Polynomial polynomial1, Polynomial polynomial2): a static method that takes two Polynomial objects as input and returns their product as another Polynomial object.

Division(Polynomial polynomial1, Polynomial polynomial2): this method takes two polynomials as input and returns their quotient as another Polynomial object.

Remainder(Polynomial polynomial1, Polynomial polynomial2): this method takes two polynomials as input and returns their remainder as another Polynomial object. Is exactly the same like Division method but we know that in first polynomial every time remains the remainder after we calculate the quotient. The only difference between Division and Remainder methods is the last return.

Interface Class

This class is a Java GUI application that implements basic arithmetic operations on polynomials. The interface consists of text fields for inputting the two polynomials, a text field for displaying the result, and buttons for performing addition subtraction, multiplication, division, derivate, and integration.

Methods:

All the methods are almost the same. When a button is clicked, the corresponding method is called which creates two Polynomial objects from the input fields, performs the operation using the PolynomialController class, and displays the result in the output text field. If the input is not valid, and exception is caught and an error message is displayed in pop-up window. The methods in this class are: addAdditionButton(), addSubsButton(), addMultiplicationButton(), addDivisionButton(), addDerivateButton(), addIntegrationButton(). addFirstPolynomial(), addSecondPolynomial(),

addResultPolynomial(), are methods for creating the text fields where we pun the first and second polynomials and display the result after the operation we made. In View() method I put together and create the Interface.

5. Results

What I test	First Polynomial	Second Polynomial	Expected Output	The effective result	Pass/Fail
Addition	4x2 + 3x1 + 10	5x3 + 6x1 + 12	22.0 + 9.0*x^1 + 4.0*x^2 + 5.0*x^3	22.0 + 9.0*x^1 + 4.0*x^2 + 5.0*x^3	Pass
Subtraction	4x2 + 3x1 + 10	5x3 + 6x1 + 12	-2.0 -3.0*x^1 + 4.0*x^2 – 5.0*x^3	-2.0 -3.0*x^1 + 4.0*x^2 – 5.0*x^3	Pass
Multiplication	4x2 + 3x1 + 10	5x3 + 6x1 + 12	120.0 + 96.0*x^1 + 66.0*x^2 + 74.0*x^3 + 15.0*x^4 + 20.0*x^5	120.0 + 96.0*x^1 + 66.0*x^2 + 74.0*x^3 + 15.0*x^4 + 20.0*x^5	Pass
Division	3x2 + 2x1	1x1 + 1	-1.0 + 3.0*x^1	-1.0 + 3.0*x^1	Pass
Remainder	3x2 + 2x1	1x1 + 1	1.0	1.0	Pass
Differentiation	x4 + 4x2 + 2x1 + 10	-	2.0 + 8.0*x^1 + 4.0*x^3	2.0 + 8.0*x^1 + 4.0*x^3	Pass
Integration	x4 + 4x2 + 2x1 + 10	-	10.0*x^1 + 1.0*x^2 + 1.33333*x^3 + 0.2*x^5	10.0*x^1 + 1.0*x^2 + 1.33333*x^3 + 0.2*x^5	Pass

This are the inputs and outputs form the PolynomialControllerTest class where I wrote tests for every operations that I have in PolynomialController class.

6. Conclusions

This project was a good exercise to remembering the OOP concepts learned in the first semester, which are very important for the future as a software development, but I learned also new things, like using Map. It was challenging and also a nice project in which I worked with love.

This project can be improved with:

- To add new operations like square of the polynomial
- Find the roots of the polynomial
- Make the graphic of the polynomial
- To change the form of the input, like instead of having 2x2 + 10x1, to have 2*x^2 + 10*x

7. Bibliography

- Object-Oriented Programming – Lecture and Laboratory slides of prof. Marius Joldos

- Programming Techniques – Lecture and Laboratory slides of prof. Cristian Pop
- https://stackoverflow.com/
- https://app.diagrams.net/
- https://www.wikipedia.org/
- https://www.geeksforgeeks.org/