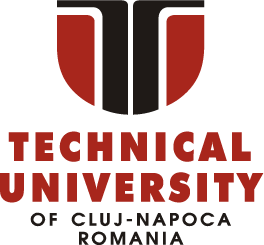
Technical University of Cluj-Napoca

Programming Techniques

Laboratory Assignment One Documentation

Polynomials Calculator

**

Teacher: prof. Dr. Ioan Salomie

Laboratory teacher: Dr. Cristina Bianca Pop

Student: Nemeș Mihnea-Andrei

Group: E\_30424

Contents

1. [Assignment objective.](#Assignment_Objective)
2. [Problem analysis, modeling, scenarios, use cases.](#Problem_Analysis)
3. [Design (design decisions, UML diagrams, data structures, class design, interfaces, relationships, packages, algorithms, user interfaces).](#Design)
4. [Implementation](#Implementation)

1. [Results](#Results)
2. [Conclusions](#Conclusions)
3. [Bibliography](#Bibliography)

1.Assignment objective

The objective of the assignment “Polynomial Calculator” is to familiarize ourselves with the object oriented programming paradigms by creating a simple program that processes polynomials. We consider these polynomials to be of one variable and to have integer coefficients. Another personal objective is to familiarize ourselves with creating a graphical interface and connecting it with the rest of the program.

2. Problem analysis, modeling, scenarios, use cases.

1. **Problem analysis**

A polynomial is an expression built like a list of monomials, which in turn are mathematical structures composed of constants and symbols, usually “x”, called indeterminates or variables on which we can perform simple mathematical operations such as addition, subtraction, multiplication, derivation etc. or a combination of these simple operations.

A polynomial’s general form can be written as a sum of monomials, a0+a1x+a2x^2+...+anx^n, where a0,a1,..., an are constants and x is the indeterminate(variable). An example of a monomial is 3x^2 where 3 is the coefficient and 2 is the power/degree of x. By representing polynomials in this way, we can perform some simple operations on polynomials: addition, subtraction, multiplication and so on.

1. **Modelling the problem**

The user will be able to introduce in the interface two polynomials, submit them to the system for them to be parsed into variables, and then select which operation they want to be performed on these polynomials, such as:

-Addition of the polynomials

-Subtraction of the polynomials

-Multiplication of the polynomials

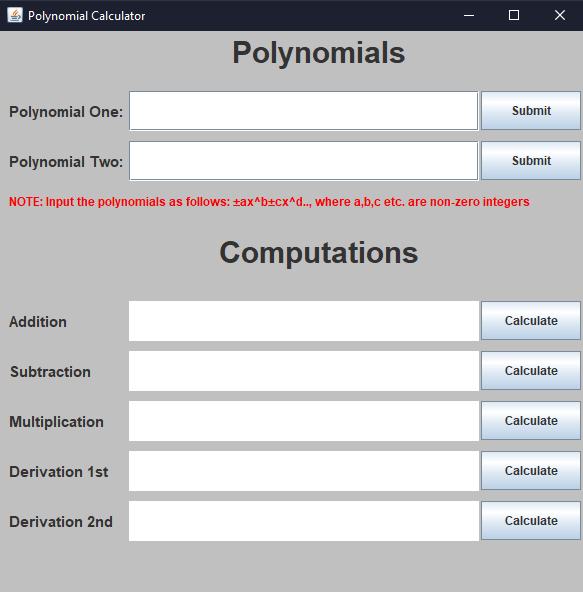
-Derivation of the first polynomial

-Derivation of the second polynomial

The result of the wanted operation will be seen in the corresponding field after pressing the correct Calculate button for that operation. The user is able to change the value of one or both polynomials, but he/she will have to submit them again in order for the system to parse the new value and assign it to the variable. If one of the polynomial boxes is left empty the system will throw an error both when trying to submit the polynomial and when trying to perform on the the operations that include said polynomial

1. **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 comprised of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. The use cases are strongly connected with the steps the user will have to take to achieve his/her desired result. For this exact reason I tried to design my interface in as friendly of a way possible to make sure there is no confusion.

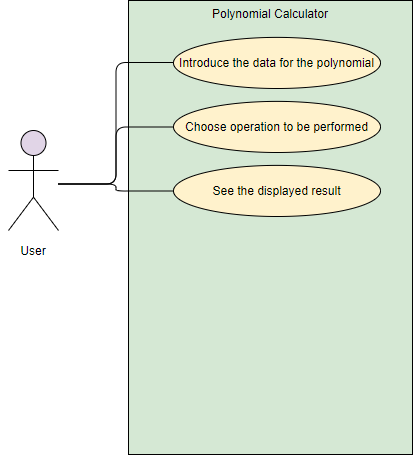


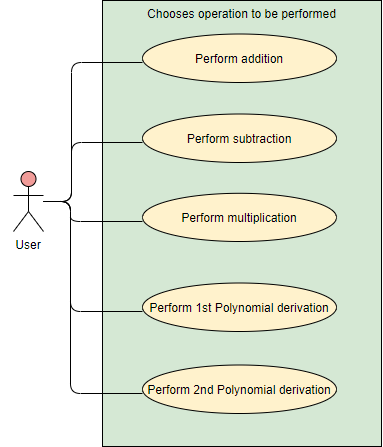
The user will have to introduce two polynomials, one in the first JTextField after which he/she will submit it, then introduce the second in the second JTextField and submit it in order for the application to be able to parse the given strings and save the corresponding polynomials in variables to be used for the computations. For every operation I have implemented there is a label informing the user which operation this is concerning, a JTextArea where the result will be displayed, and a Calculate button which will perform the operation and display the result in the corresponding JTextArea. Most operations use both polynomials, but in the case of derivation, the user will have to choose which polynomial he/she wants to be derived and click the corresponding Calculate button. When inputting the polynomials, the user will have to pay attention to follow the required format which appears on the interface in red letters.

I assume the input string to be in the correct format, otherwise, if one of the polynomials is left empty and tried to be parsed, the program will throw an error telling the user that said polynomial cannot be left empty. In the same way, if one of the polynomials is left empty or is written correctly but the user forgot to click Submit, the program will also throw an error telling the user that one of the polynomials is empty.

3. Design (design decisions, UML diagrams, data structures, class design, interfaces, relationships, packages, algorithms, user interfaces).

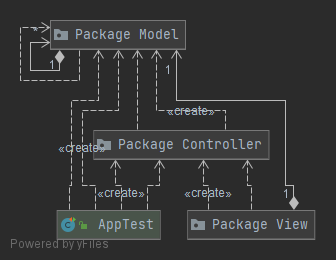
1. Use Case Diagrams

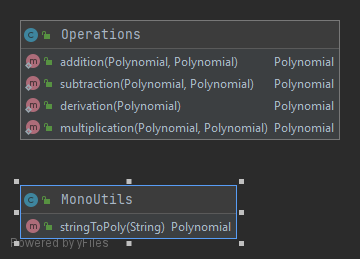


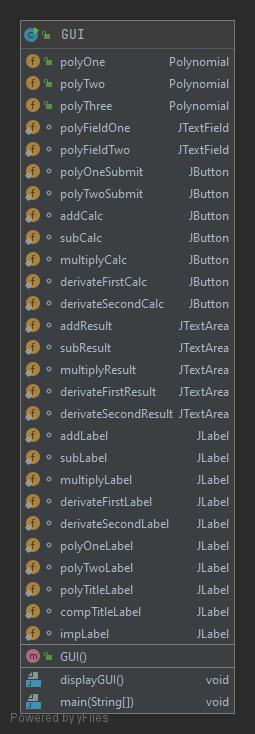
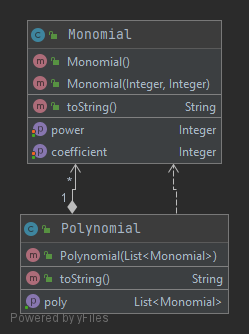


These use cases present the actor, which in our case is the user that interacts with the program. The user can perform several actions on the two given polynomials, such as addition or subtraction.

1. **Class & Packages Relationship diagrams**







1. **Data structures**

The data structures used to create the application are either primitive data types, like Integer, and complex ones, like ArrayList or the ones created by myself, specifically Monomial and Polynomial.

For implementing the polynomials there are two different ways, one in which for each polynomial you store the coefficients in a vector, and the second where each polynomial is a list of monomials. The first is not efficient because it does not use the principles and paradigms of the object-oriented programming, and especially because problems when performing operations can occur. For example, for the polynomial x^25, an array with 25 cells has to be created, even though 24 of those will not be utilized.

Thus, I have decided to create two classes, one Monomial class which stores the coefficient and the power of the monomial, and one Polynomial class which creates a list of monomials and because lists have variable length, the polynomial x^25 will only have one element (one monomial) in the list.

1. **Packages**

The use of Java Packages can help organize multiple modules and group together related classes and interfaces. In object-oriented programming, the MVC (model-view-controller) is the name of the methodology used in order to successfully and efficiently relate the user interface with the data models defined. It is a method designed to efficiently reuse object code and to more easily developing applications with user interfaces, thus reducing the time needed to implement such a program. The components of the MVC are:

* Model-represents the logical structure of the data in a software application and the high-level class associated with it. This part of the application does not contain and information about the user interface. It consists of the Monomial and Polynomial classes.
* View-consists of a collection of classes representing the elements in the user interface (all the things the user can see and interact with), such as labels, text fields, buttons and so on. It consists of the GUI class.
* Controller-consists of classes connecting the Model and the View, is used to communicate between the other two packages. It consists of the MonoUtils and Operations classes.

1. **Algorithms**

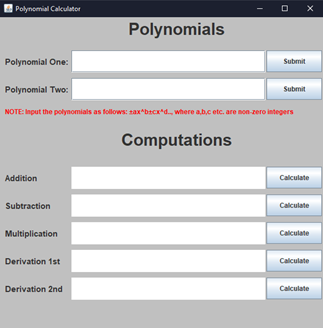
1.Addition-The sum of two polynomials is obtained by adding the coefficients of the monomials with equal powers of x. The result will have order equal to the maximum order from the two polynomials or, in the case that some terms are equal to 0 after the addition (for example in the case of 3x^2 and -3x^2), the resulting polynomial will have smaller order.

2.Subtraction-The difference of two polynomials is obtained by subtracting the coefficients of the second polynomial from the coefficients of the first polynomial in the case of monomials with equal powers of x. As with the addition, the order of the resulting polynomial will be equal to or less than the biggest order of the two polynomials in the case some terms are equal to 0 (ex. 3x and 3x when subtracting will give 0).

3.Multiplication-The product of two polynomials is given by multiplying term by term the coefficients of the terms and adding their powers, and then adding together the terms that have the same power of x after the multiplication. The order of the resulting polynomial is equal to the sum of the orders of the original polynomials.

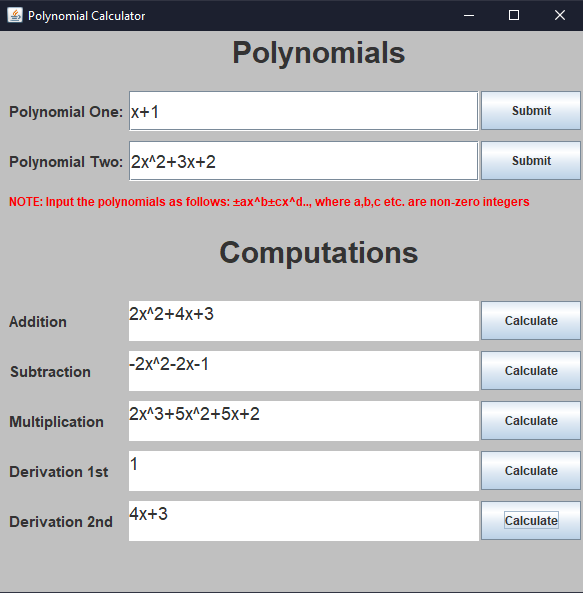
4.Derivation-The derivative of a polynomial is a simple algorithm: in the case of terms with power 0(constants) they are not considered. For the other terms, the coefficient is multiplied with the power and the power is decremented by 1.

**F**. **User interfaces**

The graphic interface has the purpose of connecting the user with the application.

<-The user introduces data for the first and second polynomial according to the form given below in red letters.

<-The user selects which of the given operations he/she wants to be computed and pressed the corresponding buttons, after which the value of the operation will be computed and displayed in the text area.



<-After the user introduces the polynomial, he desires in both of the fields, he click submit for the program to fetch the data and manipulate it

<-After the user clicks calculate on the corresponding operation, the result will be displayed in the text area.

4. Implementation

**Class design**

The idea of working with classes is based on the idea that you can divide a problem into smaller problems that are easier to solve with simple and casual algorithms. Having followed the MVC(Model-View-Controller), my application has 3 parts:

**1. Model**

a. Monomial class

A polynomial is composed of one or more terms called monomials. The monomial class has two integer fields, coefficient and power, a constructor with two parameters that creates a new monomial with the given coefficient and power, setter and getter methods for the coefficient and power fields and a toString() method for displaying a monomial on the screen. The monomial class is a part of the Model Package.

b. Polynomial class

The polynomial class is constructed as a list of monomials. It contains one constructor that gets as a parameter a list of monomials and assigns it to a polynomial, it also has a getter for the polynomial which returns the list of monomials of that polynomial and a method toString() for displaying the polynomial on the screen, method which will be used by the Graphical User Interface(GUI).This class has no setters because I have used the static keyword when defining the list of monomials that comprise the polynomial. The polynomial class is part of the Model Package.

**2. View**

The view package has only one class, the GUI class. It forms the graphical interface with the user. The graphic interface, also called graphic user interface or GUI, consists of a frame, on which I have a number of elements.

* For the polynomial introduction part, I have 2 labels, 2 text fields and 2 buttons, one of each for each polynomial. When the button is pressed, the string from the text field is parsed into a polynomial and stored into a variable. In the case that the text field is empty, the application throws an error message to the user saying that the polynomial cannot be empty.
* For the operations part I have 5 labels, 5 text areas and 5 buttons, one for each implemented operation (addition, subtraction, multiplication, derivation of the first polynomial and derivation of the second polynomial). When pressed, a function is called to calculate the operation and the result will be displayed in the text area. Also, when submitting a new polynomial, the results of the previous operations involving said polynomial will be erased.

**3. Controller**

a. MonoUtils

The MonoUtils class is a simple class consisting of only one method, the stringToPoly() method used to parse the string argument that comes when submitting a polynomial and transforms it into a list of monomials. I have employed the use of regular expressions in order to more easily recognize a monomial of the form ±ax^b and to split the given array into corresponding monomials, each with its coefficient and power. The method is able to detect monomials with coefficients both positive and negative but is not currently able to distinguish a wrongly formatted polynomial.

b. Operations

The operations class consists of methods for every operation considered (addition, subtraction, multiplication, derivation of the first polynomial and derivation of the second polynomial):

* addition()-takes as arguments 2 polynomials and returns a polynomial representing the addition of the 2 given polynomials(p1+p2)
* subtraction()-takes as arguments 2 polynomials and returns a polynomial representing the subtraction of the 2 given polynomials(p1-p2)
* multiplication()-takes as arguments 2 polynomials and returns a polynomial representing the multiplication of the 2 given polynomials(p1\*p2)
* derivation()-takes as argument 1 polynomial and returns a polynomial representing the derivation of the given polynomial(p1’)

5.Results

The testing scenarios are in close relation with the use cases of the application. I have decided to use Junit to test my program for a few scenarios. I have tested each operation on a general case that poses no problem for the application. Additionally, I have also tested some “special” cases like in the case of multiplying a polynomial with 0, where the result should be 0. Testing was also done on the Monomial and Polynomial classes, to check whether the toString() methods from each class work accordingly. Lastly, I have written a few tests to test whether the parsing method stringToPoly() is also working fine and also whether after the parsing, the monomial terms appear in the polynomial in decreasing order of the power, regardless of the order given in the input. These tests will be presented in tables below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| What is to be tested | Input Data | Expected Output Data | Actual Output Data | Test Result |
| Monomial  toString() | 3x^2 | 3x^2 | 3x^2 | Success |
| Monomial  toString() | 4x^5 | 4x^5 | 4x^5 | Success |
| Polynomial  toString() | 3x^2+3x+2 | 3x^2+3x+2 | 3x^2+3x+2 | Success |
| Polynomial  toString() | 5x^4+3x^3+2x^2+3x+2 | 5x^4+3x^3+2x^2+3x+2 | 5x^4+3x^3+2x^2+3x+2 | Success |
| Polynomial  toString() | 5x^4+3x^2+2x^3+3x^5+2x^6 | 2x^6+3x^5+5x^4+2x^3+3x^2 | 2x^6+3x^5+5x^4+2x^3+3x^2 | Success |
| Addition | P1= 3x^2+3x+2  P2= 5x^4+3x^3+2x^2+3x+2 | 5x^4+3x^3+5x^2+6x+4 | 5x^4+3x^3+5x^2+6x+4 | Success |
| Subtraction | P1= 3x^2+3x+2  P2= 5x^4+3x^3+2x^2+3x+2 | -5x^4-3x^3+x^2 | -5x^4-3x^3+x^2 | Success |
| Multiplication | P1= 3x^2+3x+2  P2= 5x^4+3x^3+2x^2+3x+2 | 15x^6+24x^5+25x^4  +21x^3+19x^2+12x+4 | 15x^6+24x^5+25x^4  +21x^3+19x^2+12x+4 | Success |
| Derivation of the 1st Polynomial | 3x^2+3x+2 | 6x+3 | 6x+3 | Success |
| Derivation of the 2nd Polynomial | 5x^4+3x^3+2x^2+3x+2 | 20x^3+9x^2+4x+3 | 20x^3+9x^2+4x+3 | Success |
| Multiplication with 0 | P1=0  P2=5x^4+3x^3+2x^2+3x+2 | 0 | 0 | Success |
| Addition where result is 0 | P1=x+1  P2=-x-1 | 0 | 0 | Success |
| Subtraction where result is 0 | P1=x+1  P2=x+1 | 0 | 0 | Success |
| Derivation of constants | 6 | 0 | 0 | Success |

6.Conclusions

This first assignment was a good exercise to remember the object-oriented programming paradigms I have learned in the first semester, but also to get familiar with new methods and techniques. At first it seemed really challenging given the pretty tight schedule, but once you start to actually research and structure the program on paper with classes and interfaces, it becomes manageable and quite enjoyable. Thus, it is very important for us as future programmers to be able to efficiently manage our time and to model the problem in such a way to help you implement it as fast and as easy as possible. Moreover, I have found that researching by yourself how to solve the problems that arise along the way when coding an application is a good way to better understand the concepts you have to employ in order to construct the program.

For future improvements, a more variety of operations can be implemented, such as the division and integration of polynomials, and even more complex operations like plotting a graph for the polynomial, or even solving complex equations using a combination of these simple operations.

7.Bibliography

* Object-Oriented Programming – Lecture Slides by prof. Marius Joldos
* Programming Techniques – Lecture Slides by prof. Ioan Salomie
* Assignment 1 Support Presentation – Programming Techniques Teams
* <https://www.baeldung.com/java-immutable-object> - working with final functions/variables
* <https://www.baeldung.com/regular-expressions-java> - understanding and employing the use of regular expressions
* <https://bit.ly/3qL9z6h> - Operations on Polynomials
* <https://www.javatpoint.com/java-swing> - Java Swing tutorial for GUI