

Fundamental Programming Techniques

- Assignment 1 -

POLYNOMIAL CALCULATOR

Stânea Georgiana Grațiela

Group 30424

Table of contents

[1. Objective 3](#_Toc97837904)

[2. Problem analysis, modelling, scenarios, use cases 3](#_Toc97837905)

[3. Design 4](#_Toc97837906)

[4. Implementation 6](#_Toc97837907)

[5. Results 9](#_Toc97837908)

[6. Conclusions 10](#_Toc97837909)

[7. Bibliography 10](#_Toc97837910)

# Objective

The objective of this project is to design and implement a polynomial calculator of one variable and with integer coefficients. The application will be able to perform operations that manipulate one or two polynomials, namely: addition, subtraction, multiplication, division (of two polynomials) and differentiation, integration (for one polynomial).

The communication between the user and application is done through the graphical user interface, where one can enter the desired polynomials and select which operation to be performed.

# Problem analysis, modelling, scenarios, use cases

* Problem analysis

The program should compute all the above mentioned operations based on simple mathematical algorithms and using Object Oriented Programming. This offers us an advantage, because we can treat the problem at a higher level, without being constrained by technical characteristics.

The user should enter the polynomials, select the operation and see the final result on the graphic interface. To make this process easy, the polynomials will be represented using their coefficients and powers and entered like a text.

* Modelling

Polynomials have always been an important part of science. They are used in nearly every field of mathematics to express numbers as a result of mathematical operations.

Polynomials are made of terms named monomials which have a single coefficient and one or more variables. Each variable has a power that gives the degree of the monomial. The degree of a polynomial is represented by the largest power of a monomial consisting it.

To model these structures, polynomials are represented by objects of type Polynomial that consist of a list of Monomial objects. The entered powers are always integers, but we can deduce that some computed coefficients can be of double type. This problem is solved by creating two constructors for the class Monomial, one with integer coefficients and one with double.

* Scenarios and use cases

The user will enter two polynomials in the corresponding text fields, select the operation and then the result will automatically be displayed in the “Result” textbox. In the case of differentiation and integration, only the first text field is used. We assume that the user’s input is written correctly, otherwise an error will appear on the screen (for example, a polynomial of more than one variable is not accepted). The order in which the monomials are entered does not matter, because the result will be displayed in correct format.

Use case description

Use case: Computing a polynomial operation

Primary actor: The user

Success scenario steps:

1. The user enters two (or one) polynomial in the specific text fields.
2. The application checks if the input has the correct format and transpose it into objects of type Polynomial
3. The user selects the desired operation.
4. The application computes the result.
5. The result is displayed in the specific text box.
6. The user can enter other polynomials or change the operation.

Alternative sequences:

1. The entered polynomial does not respect the standard format. The application displays “Please enter a valid polynomial” and the scenario returns to step 1.

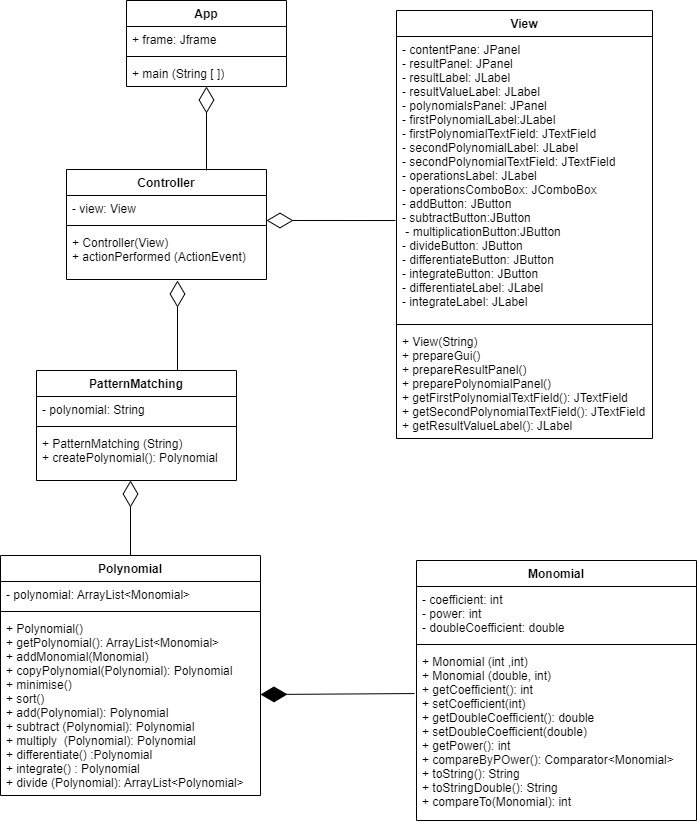
Post-conditions:

We have no post-conditions. The application continues to compute operations with polynomials until it is closed.

# Design

* UML Diagram for classes

This diagram was created using draw.io :



* Packages

Packages in Java are a mechanism to encapsulate a group of classes, sub packages and interfaces. In my application I used the model-view-controller (MVC) design pattern to organize my data.

* *Model*: is the application's dynamic data structure, independent of the user interface It directly manages the data, logic and rules of the application. In my case, I defined the Polynomial and Monomial structures.
* *View*: is represented by the gui package where I created methods and classes for the user interface. This class extends JFrame and inherits all the methods and options that let you create something visually.
* *Controller*: represents the classes that connect the view and the model. In my application, this is represented by the package “logic” where I have the class “PatternMatching” that transforms my input to an object of type Polynomial.
* Data structures

This application uses both primitive data types like integers or of double type,strings and more complex ones, like ArrayList. I chose to use this data structure instead of arrays because it is more efficient in the case of memory management. We also get rid of the problem of overflow because we do not have to specify the length (like in the case of simple arrays).

I also created some new objects to help me store the entered data: Monomial and Polynomial.

# Implementation

* ***Monomial class***

This is the main object my application is based on, polynomials being formed of a list of monomials. This class has three attributes: an int power, an int coefficient and a double doubleCoefficient. I also defined two different constructors, for both the cases of coefficients:

* public Monomial(int coefficient, int power) : the constructor the initialize the monomial with the transmitted power and integer coefficient. This one in most of the cases.
* public Monomial(double doubleCoefficient, int power): the constructor that initialize the monomial with the transmitted power and double coefficient. We need this constructor because in the case of division and integration, we might have the coefficient of the result different from an integer value.

Methods:

* public static Comparator<Monomial> compareByPower() : compares the powers of two given monomials;
* public String toString() : this method is used to display each monomial in the desired format;
* public String toStringDouble(): same as the oane before but it handles double coefficients;
* ***Polynomial class***

This class has only one instance variable which is represented by an ArrayList<Monomial> named “polynomial”. This is the main object I am working with.

Constructor:

* public Polynomial() { this.polynomial = polynomial; }

Methods

* public void addMonomial(Monomial mon){

polynomial.add(mon);}

This is used to add the transmitted Monomial to the list of the current Polynomial;

* public Polynomial copyPolynomial(Polynomial pol) : is used to create a copy of the transmitted Polynomial. The returned result is also a Polynomial. I created this method to help me computing the division;
* public void minimise() : has the role of computing operations between monomials that have the same power. After applying this method, we will have a minimized polynomial;
* public void sort() : sorts the monomials from a polynomial with respect to their powers. After applying this method we will have a Polynomial the highest power Monomial first.
* public Polynomial add( Polynomial pol2) : this method adds the current Polynomial with the transmitted one. The result is also a Polynomial.
* public Polynomial subtract( Polynomial pol2): the function for subtraction. This takes the second Polynomial, reverse its signs and then performs the addition operation.
* public Polynomial multiply(Polynomial pol2) : function that computes the multiplication of two polynomials. In this method we also need the minimize() and sort() functions mentioned before.
* public Polynomial differentiate(): computes the differentiation of the current Polynomial. This is done with respect to the mathematical formula;
* public Polynomial integrate(): computes the integration of the current Polynomial, this is also done with the mathematical formula.
* public ArrayList<Polynomial> divide(Polynomial pol2) : computes the division. The result is an ArrayList<Polynomial> because we have a quotient and a remainder. First, this method makes sure that the entered polynomials are sorted by power and minimized, in order to have a correct algorithm. The computation takes place until the power of the remainder is less or equal to the power of the second Polynomial or until the remainder is null. In order to compute this operation correctly, I needed assign a double coefficient to the Monomials of the result returned here.
* ***PatternMatching class***

This is the class where the application transforms the input given as a String to an valid Polynomial formed of Monomials. We have just one variable, a String “polynomial” which is transmitted through the constructor.

Constructor:

* public PatternMatching(String polynomial) {

this.polynomial = polynomial; }

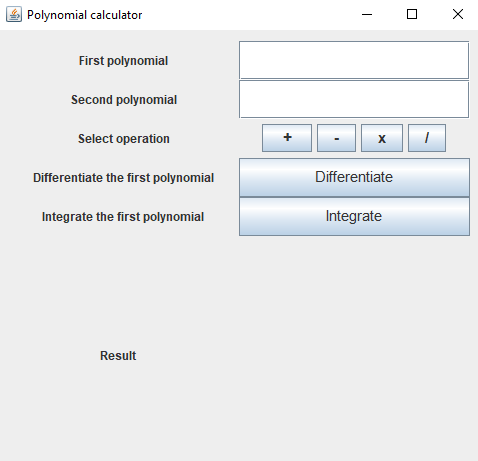
Methods:

* public Polynomial createPolynomial() : this method takes a given String and parse it in order to obtain an object of type Polynomial. In this scope, I used the Java Regex library that lets select parts of a String according to a given pattern. After it separates the String into Monomials, it makes another test so find the position of the “x” and splits the resulted String in two parts: the first part will be the coefficient and the second part represents the power. In the end, it creates a new Polynomial and returns it.
* ***View class***

This is the class where I implemented the Graphic User Interface (GUI). I chose to work with Java Swing because I consider it much easier and simple to understand than others GUI builders. I made “View” extend JFrame in order to have all the options and methods of this superclass.

Methods:

* public void prepareGui(): creates a new panel with a grid layout, sets the size and the default exit operation;
* private void prepareResultPanel(): prepares the lower part of the Panel, where the result will be displayed;
* private void preparePolynomialsPanel(): prepares the upper part of the Panel, where the user can introduce the polynomials and select the operation;



* ***Controller class***

This is the class that links the Graphical User Interface to the uses of the application. It only has one method that interprets the actions performed on the GUI. The result is computed depending on which button is pressed and then is displayed in the corresponding Panel.

* ***App class***

Is the core of the application, here we initialize the frame and start the program.

# Results

For testing this application I used Junit. I created a new class named “PolynomialTest” where I simulated the operations implemented for a Polynomial. The condition is checked using the “assertEquals” API of Junit and if the test was successful, the message “The test was successful" will be written in the console.

* Addition

Polynomial1: 4x^3 - 2x + 5

Polynomial2: 2x^2 + 2x

Result: + 4x^3 + 2x^2 + 5

* Subtraction

Polynomial1: x^2 - x

Polynomial2: x^3 – x + 1

Result: - 1x^3 + x^2 – 1

* Multiplication

Polynomial1: x + 1

Polynomial2: x + 2

Result: + x^3 – 4x^2 + 5x - 2

* Differentiation

Polynomial1: 8x^4

Result: + 32x^3

* Integration

Polynomial1: - x + 2

Result: - 0.5 x^2 + 2x

* Division

Polynomial1: x^2 + 2x + 2

Polynomial2: x + 1

Result: Q: + x + 1.0 R: + 1.0

# Conclusions

In the end, this project made me improve my knowledge regarding Object Oriented Programming and the ability to handle different data structures. After developing this application I came to the conclusion that it was harder to design than I was expecting, but this only made me search solutions to my problems and challenged me to get to the final result. I also learned how to implement a Graphical User Interface using Swing and how to deal with String parsing using Regex. Some functionalities were easier to code, for example addition and subtraction, but division was the one that put me in difficulty.

This program could be improved by adding some more functionalities like computing the value of a polynomial in a given point, supporting polynomials of more than one variable, Fourier transform multiplication, finding the roots of a polynomial, plot the graphic and for a better performance there should be implemented all cases where exceptions can occur and the application stops working due to an error made by the user.

Also, the graphic user interface could be improved by making it more interactive, adding more text boxes for the error messages and some new fields for operations.

# Bibliography

* Programming Techniques – Lectures of prof. Cristina Bianca Pop
* <http://www.stackoverflow.com/>
* <http://www.wikipedia.org>
* <https://www.draw.io>