Laboratory Work –Assignment 1

Polynomial Processing

Table of contents

1.Introduction

1.1 Task objectives………………………………………………………………………….page 3

1.2 Personal approach…………………………………………………………………….page 3

2.Problem description

2.1 Problem analysis……………………………………………………………………….page 3

2.2 Use cases………………………………………………………………………………….page 4

3.Projection

3.1 UML diagrams…………………………………………………………………………….page 7

3.2 Data structures…………………………………………………………………………..page 8

3.3 Class projections………………………………………………………………………...page 8

3.4 Algorithms………………………………………………………………………………….page 15

4.Implementation and testing……………………………………………………………..page 20

5.Results………………………………………………………………………………………………page 20

6.Conclusions……………………………………………………………………………………….page

7.Bibliography………………………………………………………………………………………page

1. Introduction

1.1 Task objectives

The task of the problem is defined as follows: “Propose, design and implement a system for polynomial processing. Consider the polynomials of one variable and integer coefficients.”

1.2 Personal approach

This paper aims to present one way of solving the problem of polynomial processing by means of implementing several operations specific for polynomials such as: addition,subtraction,multiplication,division,integration,evaluation,exponentiation,equality test ,root finding and graph representation which are the most common operations between polynomials.

2.Problem description

2.1 Problem analysis

The analysis of a problem starts from examining the real model or the model we confront with in the real world and passing the problem through a laborious process of abstractization. Hence we identify our problem domain and we try to decompose it in modules easy to implement. We should always keep in mind that if we do not have a good model we have to do more complex programs.

So we have the problem domain that is defined by the mathematical definition of a polynomial:

P ( x )=a_n x^n + a_{n-1}x^{n-1} + \dotsb + a_2 x^2 + a_1 x + a_0, n∈N

In the application, as it will be shown later we will use the mathematical representation for entering the polynomials and the result will be displayed mathematically as well.

2.2 Use cases

The use cases are strongly related to the user . I tried to design the interface so that it could be as user friendly as possible , so as it can be seen in the figure below , the execution begins with an information message that describes the way in which the user should enter the polynomials.

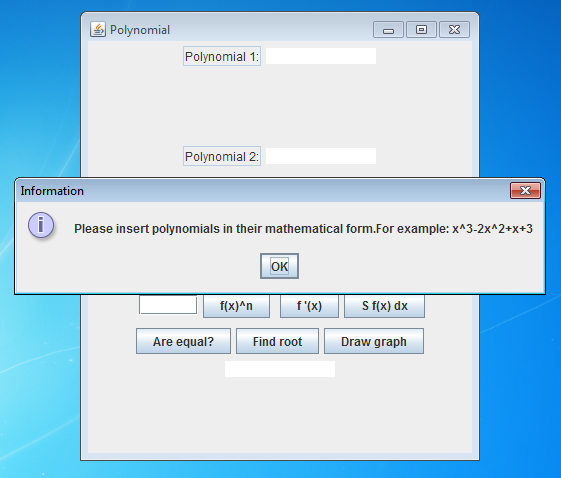


Figure 1: The Graphical User Interface

After pressing “OK” in the information bar,the user can start the execution of the application,so it can use its operations.

In the next figure we can see all the operations and options that this application has.

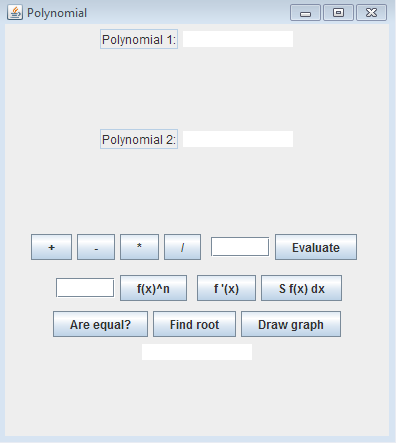
.

Figure 2: The Graphical User Interface

The user must enter the the 2 polynomials or only the first one in the case of evaluation,exponentiation,derivation,integration,root finding and graph drawing and if necessary, in the case of evaluation and exponentiation, to enter the corresponding values.

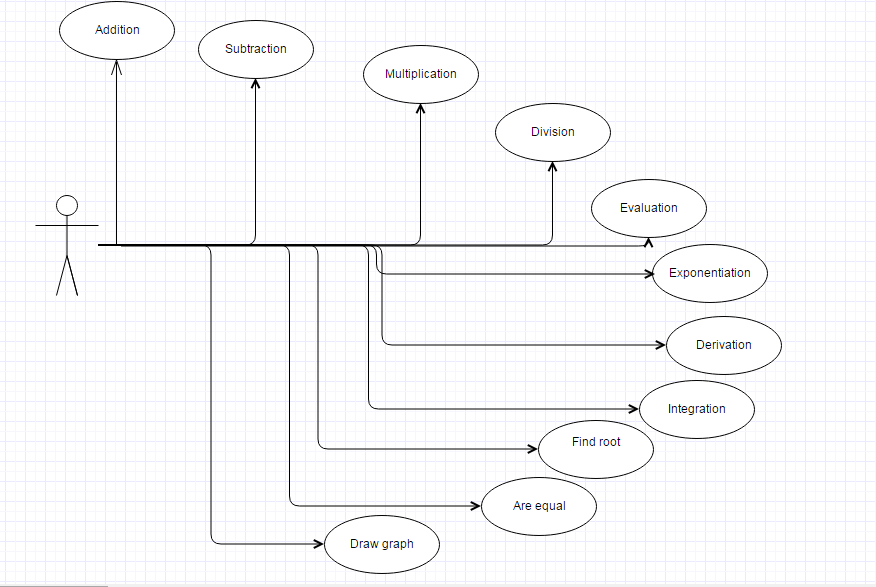
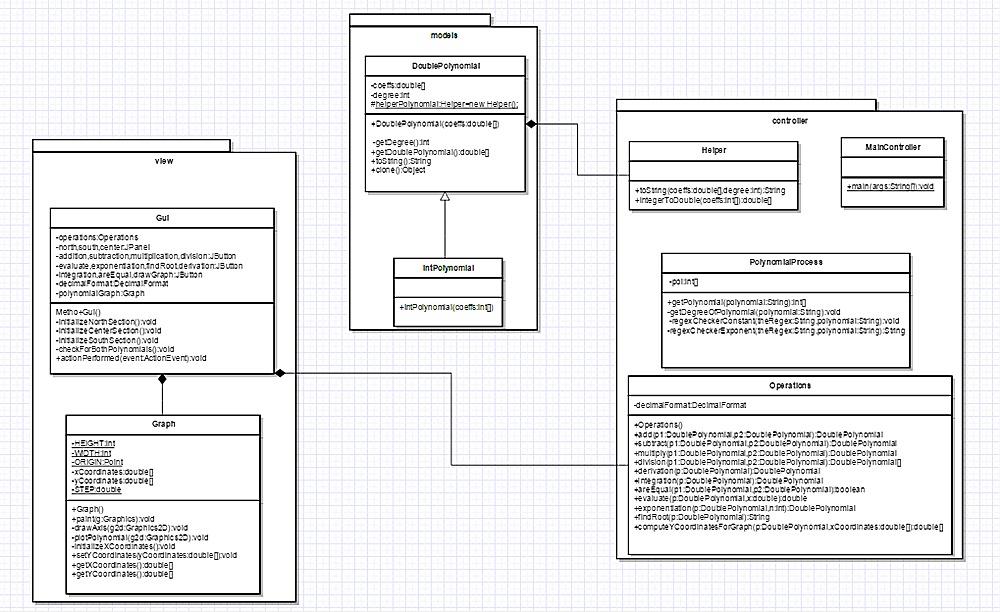


Figure 3: The Use Case diagram

3.Problem description

3.1 UML Diagrams

Class diagram:



3.2 Data structures

The data structures used at this project are primitive data types such integers and doubles and more complex types such as DoublePolynomial or IntPolynomial.In our case, we use IntPolynomial structure just to make sure that the user enters a polynomial with integer coefficients and then in all our operations we use the DoublePolynomial type and at output we take care to format the coefficients such that they will appear in their proper way .

3.3 Class projection

Class projection refers mainly to how the model was thought, how the problem was divided in sub-problems, each sub-problem representing more or less the introduction of a new class . First I will start by mentioning exactly how my problem was divided into packages and afterwards each package with its own classes . So I have 3 packages called : models , view , controller related to the MVC software architectural pattern .

The models package contains the classes : DoublePolynomial and IntPolynomial , the last one being the subclass of the first .

In the DoublePolynomial class we have as field variables:

private double[] coeffs ; -which will memorise the coefficients of our polynomial

private int degree ; -which will represent the degree of our polynomial

protected static Helper helperPolynomial ; - which is a helper field of type Helper defined in the controller package .

The methods of this class are:

public DoublePolynomial(double... coeffs) –which is the constructor of this class

public int getDegree()-method used to get the degree of the polynomial

public double[] getDoublePolynomial()-method used to return the coefficients of the polynomial

public String toString()-method used to print the polynomial in its mathematical form with the help of the helper class

public Object clone()-method provided by the Cloneable interface that makes a deep clone of the DoublePolynomial object.

The IntPolynomial class has just a constructor in which we call the super constructor explicitly with the corresponding coefficients.

The controller package has the following classes:Helper,MainController,Operations,PolynomialProcessing.

The Helper class has no instance variables and the following methods:

public String toString(double[] coeffs, int degree)-which is the class that does the logic for the DoublePolynomial toString() method

public double[] integerToDouble(int... coeffs)-method that helps in the super constructor call in the IntPolynomial class

The MainController class has just the public static void main(String[] args) method wich will launch the application.

The Operations class has also fields and methods that are used in computing the operations.

The instance variables are used only locally so we don’t pay too much attention to them but the methods are:

public Operations()-the constructor of the class

public DoublePolynomial add(DoublePolynomial p1, DoublePolynomial p2)-the method that computes the addition between 2 polynomials and returns the resulting polynomial

public DoublePolynomial subtract(DoublePolynomial p1, DoublePolynomial p2)- the method that computes the subtraction between 2 polynomials and returns the resulting polynomial

public DoublePolynomial multiply(DoublePolynomial p1, DoublePolynomial p2)- the method that computes the multiplication between 2 polynomials and returns the resulting polynomial

public DoublePolynomial[] division(DoublePolynomial p1, DoublePolynomial p2)- the method that computes the division between 2 polynomials and returns the resulting polynomial

public DoublePolynomial derivation(DoublePolynomial p)- the method that computes the derivation of a polynomial and returns the resulting polynomial

public DoublePolynomial integration( DoublePolynomial p )- the method that computes the primitive of a polynomial and returns the resulting polynomial

public boolean areEqual ( DoublePolynomial p1, DoublePolynomial p2 ) - the method that verifies the equality of 2 polynomials and returns true or false

public double evaluate( DoublePolynomial p, double x ) - the method that computes the value of a polynomial in a specific point and returns the resulting value

public DoublePolynomial exponentiation(DoublePolynomial p, int n) - the method that computes the exponentiation of a polynomial and returns the resulting polynomial

public String findRoot ( DoublePolynomial p ) - the method that finds the root of a polynomial and returns the corresponding root value or a text in case that the polynomial doesn’t have real roots

public double[] computeYCoordinatesForGraph(DoublePolynomial p, double[] xCoordinates ) - the method that computes the values of a polynomial in the x coordinates of points and returns the corresponding y coordinates values used in plotting the polynomial

For reading the polynomial in its mathematical form we used Regular Expressions which offer a great flexibility to our application . This logic is implemented in the class PolynomialProcessing .

The PolynomialProcessing class has the following features:

It has only one field:

private int[] pol –which will memorise our result

and more methods:

public int [] getPolynomial ( String polynomial ) – returns the integer coefficients from the processed polynomial

private void getDegreeOfPolynomial ( String polynomial ) – computes the degree of the polynomial entered at input

private void regexCheckerConstant ( String theRegex , String polynomial ) – uses the Regular Expressions to read the term of degree 0 , it exists .

private String regexCheckerExponent ( String theRegex , String polynomial ) - uses the Regular Expressions to read the terms of degree different from 0 if they exist .

In our view package we describe the graphical user interface of the program.

So , it has 2 classes called : Gui and Graph . In the Gui class we connect all the components from the project and in the Graph class we display the graph of the polynomial . The Gui class extends the JFrame class and implements the ActionListener interface .

The Gui class has many fields , but we will describe here some of them :

private static String ERROR = " Error " ; -message that will be displayed on the screen in the case of an error

private static String ERROR\_POLYNOMIAL = " You didn't enter the polynomial . " ; - message that will be displayed on the screen in the case when our operation needs the first polynomial to be read but we didn’t enter it .

private static String ERROR\_POLYNOMIALS = " You didn't enter both polynomials . " ; - message that will be displayed on the screen in the case when our operation needs both polynomials to be read but we didn’t enter them or we have enter just one .

private JTextArea pol1, pol2, result ; - the place from where we will read our polynomials and where we will display the result

private Operations operations ; - marks the composition relationship between Operations class and the Gui class . This field will help us in solving our tasks .

private DecimalFormat decimalFormat ; - this field helps us in displaying the correct form of the coefficients in the case they are integer or doubles.

private Graph polynomialGraph ; - marks the composition relationship between Graph class and the Gui class . This field will help us displaying the polynomial graph .

private JButton addition , subtraction , multiplication , division , evaluate , exponentiation , findRoot , derivation ,integration , areEqual , drawGraph ; - this fields help us in executing and after that displaying , the correct operation that the user requires .

As methods we have :

public Gui () – the class constructor

private void initializeNorthSection () – this method organizes the north section of our graphical user interface

private void initializeCenterSection () - this method organizes the center section of our graphical user interface

private void initializeSouthSection () - this method organizes the south section of our graphical user interface

private void checkForBothPolynomials () – this method checks to see if we enter the both polynomials in the case that we need both of them

public void actionPerformed ( ActionEvent event ) - this method helps us in executing and after that displaying the correct operation that the user requires by pressing the specific button for the specific operation .

The Graph class extends the JPanel class and has some fields like :

private static final int HEIGHT = 400 ; - which sets the default height of the panel at 400 pixels

private static final int WIDTH = 400 ; - which sets the default width of the panel at 400 pixels

private static final Point ORIGIN ; - defines the origin of the axis .

private double [] xCoordinates ; - defines the pair of x coordinates .

private double [] yCoordinates ; -defines the pair of y coordinates .

private static double STEP = 0.2 ; - defines the step in which the polynomial will compute its value .

As methods we have :

public Graph () –the class constructor

public void paint ( Graphics g ) – the method that will paint the graph to the screen .

private void drawAxis ( Graphics2D g2d ) - the method that will draw the graph axis on the screen .

private void plotPolynomial ( Graphics2D g2d ) –the method that will paint the line of the polynomial graph on the axis .

private void initializeXCoordinates () – this method initializes our x coordinates of the points so that we can compute their values .

public void setYCoordinates (double[] yCoordinates ) – this method is a setter for the y coordinates of the points .

public double [ ] getXCoordinates ( ) – this method is used to return the x coordinates vector of the points in which we compute the value of the polynomial .

public double [ ] getYCoordinates ( ) - this method is used to return the y coordinates vector of the points in which we have the values of the polynomial in the corresponding x coordinates .

3.4 Algorithms

In this section I will describe mainly the method that are implementing the operations as I consider them the most important ones :

Addition:

The mathematical model for the addition operation is defined as follows :

- we have two polynomials to add , meaning Polynomial 1 : and Polynomial 2 : both of one variable x and with integer coefficients .

- the first and second polynomials are defined by the expressions :

f(x) = \sum_{i=0}^n a_ix^i; g(x) = \sum_{i=0}^m b_ix^i , n , m ∈ N

-the result is defined by:

f(x)+g(x)= \sum_{i=0}^m (a_i+b_i)x^i  where m > n, n , m ∈ N

The mathematical model for the multiplication operation is defined as follows:

-we have two polynomials to multiply, meaning Polynomial 1: and Polynomial 2 :

f(x) = \sum_{i=0}^n a_ix^i; g(x) = \sum_{i=0}^m b_ix^i, n , m ∈ N

-the resulting polynomial is expressed by the expression:

f(x)\times g(x)=\sum_{i=0}^{n+m} c_ix^i , n , m ∈ N

• Division

For the division I applied the classical algorithm of division presented below with an example . For the division the following mathematical relations take place:

P(x) = D(x) \* Q(x) +R(x) where

P(x)- dividend, D(x)- divisor, Q(x)- quotient, R(x)- remainder

The classical algorithm for division is described as follows:

-we have two polynomials P(x) the dividend and Q(x) the divisor

-we divide the most significant term of the dividend to the MST of the divisor

- this way we are obtaining the first term of the quotient

- we multiply the result with the divisor and we subtract this result from the dividend

- this computation gives us the first remainder of the division

- we repeat this procedure taking now the obtained remainder as the dividend

-the algorithm ends when the deg (remainder) is less than the deg ( quotient )

Example :

Polynomial 1 : x^5-x^2+1

Polynomial 2 : x^2+3

Quotient: x^3-3x-1

Rest: 9x+4

Differentiation

The mathematical model for the differentiation operation is defined as follows :

- we have one polynomials to differentiate , meaning P(x)

- the polynomial is expressed by the expression :

P ( x ) = a_n x^n + a_{n-1}x^{n-1} + \dotsb + a_2 x^2 + a_1 x + a_0, n ∈ N

For an easier description we will use an example:

P(x) = 6x^ 5 + 3x^ 4 − 9x + 7

- the differentiated polynomial will have the following form :

df/dx = 30^ 4 + 12x^ 3 – 9

Integration

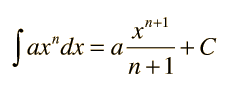
The mathematical model for the differentiation operation is defined as follows:

- we have one polynomials to differentiate, meaning P(x)

- the polynomial is expressed by the expression :

P ( x ) = a_n x^n + a_{n-1}x^{n-1} + \dotsb + a_2 x^2 + a_1 x + a_0, n ∈ N

-therefore the integrated polynomial will have the following form :

 , where we apply the same method for any term ,discarding at last the C constant .

Root finding :

Root finding is done using the bisection method described above .

The method is applicable for numerically solving the equation f(x) = 0 for the real variable x, where f is a continuous function defined on an interval [a, b] and where f(a) and f(b) have opposite signs. In this case a and b are said to bracket a root since, by the intermediate value theorem, the continuous function f must have at least one root in the interval (a, b).

At each step the method divides the interval in two by computing the midpoint c = ( a + b ) / 2 of the interval and the value of the function f(c) at that point. Unless c is itself a root (which is very unlikely, but possible) there are now only two possibilities: either f(a) and f(c) have opposite signs and bracket a root, or f(c) and f(b) have opposite signs and bracket a root.[5] The method selects the subinterval that is guaranteed to be a bracket as the new interval to be used in the next step. In this way an interval that contains a zero of f is reduced in width by 50% at each step. The process is continued until the interval is sufficiently small.

Explicitly, if f(a) and f(c) have opposite signs, then the method sets c as the new value for b, and if f(b) and f(c) have opposite signs then the method sets c as the new a. (If f(c)=0 then c may be taken as the solution and the process stops.) In both cases, the new f(a) and f(b) have opposite signs, so the method is applicable to this smaller interval.

The input for the method is a continuous function f, an interval [a, b], and the function values f(a) and f(b). The function values are of opposite sign (there is at least one zero crossing within the interval). Each iteration performs these steps:

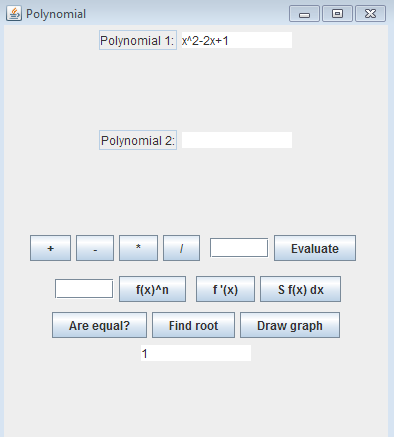
1)Calculate c, the midpoint of the interval, c = 0.5 \* (a + b).

2)Calculate the function value at the midpoint, f(c).

3)If convergence is satisfactory (that is, a - c is sufficiently small, or f(c) is sufficiently small), return c and stop iterating.

4)Examine the sign of f(c) and replace either (a, f(a)) or (b, f(b)) with (c, f(c)) so that there is a zero crossing within the new interval.

Example from running program :



4. Implementation and testing

In what the implementation is concerned this project was developed in Eclipse and it was only tested in this environment. However the program should maintain its portability. Concerning the code implementation I did not make use of laborious algorithms, but I have rather stayed faithful to the classical algorithms of computing polynomials learned in high school. However I have tried to implement my problem in a way that appears to me as being more efficient than other methods.

Testing implies checking for any errors in the program or limitations of this program. Due to the fact that the program is not very complex , they are few errors that might generate this program to work wrong or to stop. This errors are mostly related to the interface and to the finding root method which may have somehow a logical error because the algorithm may not return always the best solution . I have assumed that the user reads the instructions from the interface and respects them, otherwise if he enters data with invalid format the program will probably generate some bugs and will stop or it will give wrong results . Hence this part with checking all the possible scenarios will be seen as future development .

5. Results

The application is an user friendly and useful application to perform basic polynomial operations such as : ddition , subtraction , multiplication , division , integration , evaluation , exponentiation , equality test , root finding and graph representation. As the application is developed on a Java platform, it is highly portable and allows it to run on several operating systems (as long as they have the Java SDK installed). The application is straightforward an easy to understand and to use by any user who respects the instructions given in the interface and who has some basic knowledge of polynomials operations, of course. Even though being limited, this application can be considered as being a helpful tool that can be used when dealing with such polynomial operations.

5. Conclusions

All in all, this project seemed to me a new experience. I say it from many points of view. From time management point of view I should say that I have learned and both felt that time is precious and we should have a daily plan overall and also a plan to work for our project . Even though was a laborious work, all steps in completing it are all important and neither should be or can be skipped. Out of all steps I considered modeling as being the most important one because once you have the right model half of the work is done .

I have learned that is better first to think on paper, make different scenarios and only when we are sure about the model start to implement it.

I have also learned that a good modularisation will provide both a good implementation and a better view of the whole image, especially when we apply the MVC design type for our project .

7.Bibliography

Head First Java – Second Edition

[*http://users.utcluj.ro/~jim/OOPE/*](http://users.utcluj.ro/~jim/OOPE/)

[*http://docs.oracle.com/javase/7/docs/api/overview-summary.html*](http://docs.oracle.com/javase/7/docs/api/overview-summary.html)

[*http://stackoverflow.com/*](http://stackoverflow.com/)