Technical University of Cluj-Napoca

Assignment 1

Polynomial Calculations

Name: Tudor Alexandru Haba

Group:30425

Table of contents

1. Introduction

1.1 Task objectives

1.2 Personal approach

2. Problem description

2.1 Modelling

2.2 Scenarios

2.3Use cases

3. Projection

3.1 UML diagrams

3.2 Data structures

3.3 Class projection

3.4 The algorithms

4. Implemetation and Testing

5. Results

6.Conclussions

1. Introduction

1.1 Task objectives

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

1.2 Personal approach

The aim of the assignment was to solve different polynomial operations. The most basic polynomial operations(addition, subtraction, multiplication, division, integration, derivation etc.).

2. Problem description

Polynomials seem easy enough to understand but some of the operations regarding them are trickier than they look. The polynomial as a mathematical expression looks like this:

a_n x^n + a_{n-1}x^{n-1} + \dotsb + a_2 x^2 + a_1 x + a_0,

There are two aspects to look at when thinking about a polynomial: the degree(n) and the coefficient an . That is, a polynomial can either be zero or can be written as the sum of a finite number of non-zero terms. Each term consists of the product of a number—called the coefficient of the term[2]—and a finite number of indeterminates, raised to nonnegative integer powers. The exponent on an indeterminate in a term is called the degree of that indeterminate in that term; the degree of the term is the sum of the degrees of the indeterminates in that term, and the degree of a polynomial is the largest degree of any one term with nonzero coefficient.

When looking at two polynomials the operations of addition and subtraction look pretty straightforward with multiplication implying repeated addition…at least in theory. Division starting to become a bit different from what we understand of division of 2 integer numbers.

2.1 Modeling

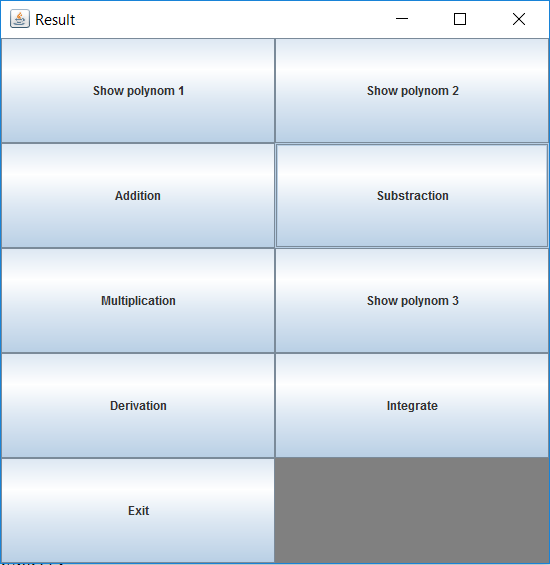
This is where it all becomes interesting. There are many possible ways of solving this problem. I will now talk about data structures. The two ways of thinking this problem consist of: arrays or linked lists. When I first started working on the problem I tried using linked lists because I saw them as being a more elegant way of tackling the issues. However after some time after testing what I can and cannot do with linked lists I decided to go back to the old way of thinking: arrays. My one issue that I had when I tried implementing linked lists and that I couldn’t see myself was storing the degrees and the coefficients in such a way that I could use both at the same time. So I decided to use arrays because I could store the coefficients inside the given array and I would safely keep the degrees in the index part.

2.2 Scenarios

For solving the problem I wanted to use 4 classes at first. One class where I would read the polynomials, one class where I would do the operations on the given polynomials (addition, subtraction, multiplication, division, integration, derivation) one class for the user interface and one class “Main”. When I first started I was not too sure about using a GUI- not having good prowess so at first I used console input and output of the polynomials and the operations around them. Also as I worked on the operation class I saw the fact that almost everything used there came from the Polynomial class and because I either used static methods/ variables or face the problem of entering an endless loop between instantiating Polynomial objects that went directly to the Operations class and from there it would instantiate polynomial objects I found out that I would rather use a “OperationsPolynomial “ class that did both jobs. I will talk about the GUI shortly.

2.3 Use cases The use cases are the ones the user will benefit from so they have to be simple enough so that the user won’t lose him/herself in but they have to be complex enough to give the user the ability to solve polynomial operations with relative ease.

This is the interface I came up with:

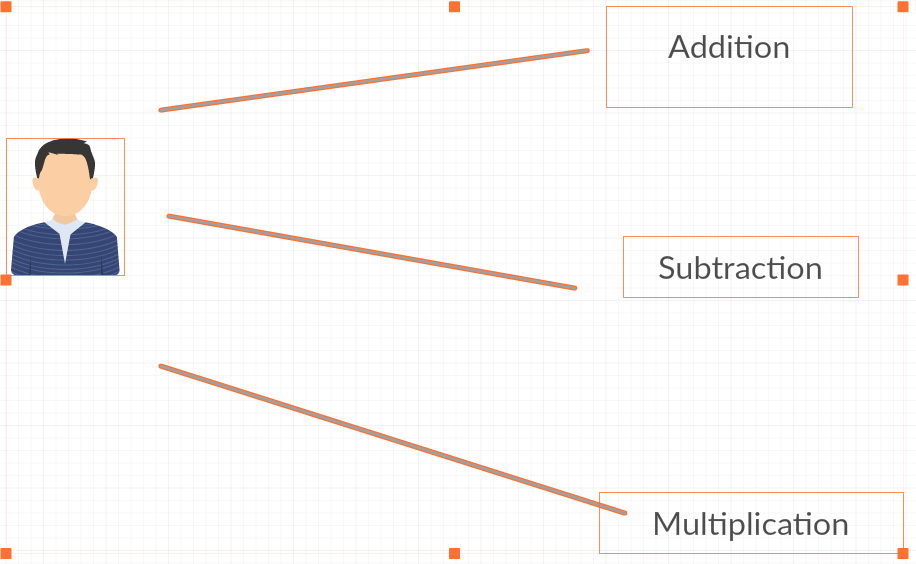


The issue that this user interface has to do with the fact that the user still has to use to console for the input of the degree of the polynomials and the coefficients. But from that point forward the operations are done with the click of a button. In the bottom right corner the polynomials are displayed by clicking the “Show polynom” 1 or 2. The “Show polynom 3” only has meaning after a subtraction /addition /multiplication occurs. The Deivation and the Integrate functionality are used upon the first polynomial. The “Exit” button is just a quick way of closing the application in a correct way.

3. Projection

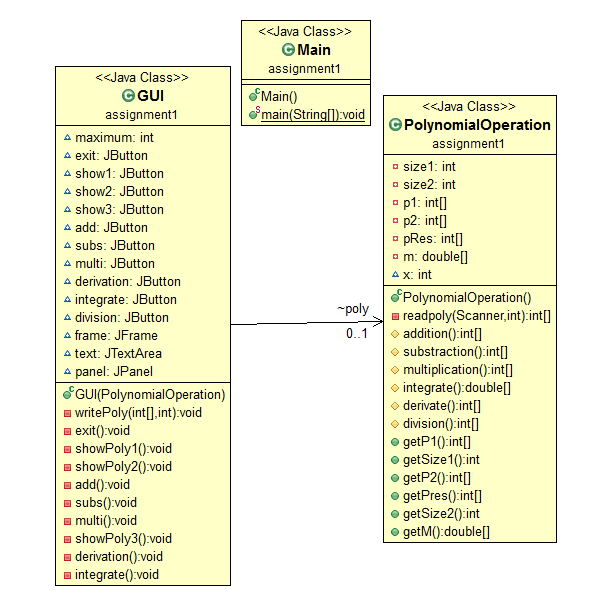
3.1 UML diagrams

a)Use Case Diagram:



In here I depict the user as being able to interact with the application. He can apply the operations that I have given as example (addition, subtraction, multiplication) on two polynomials. This is just a small example. There are more operations that the user can do.

b) Class Diagram:



The dependency that I have is between the “GUI” class and the “PolynomialOperation” . The GUI uses and object of the type “PolynomialOperation” for getting the data that is stored there so it can be utilized in the graphical part of the problem. All the operation methods are in the “PolynomialOperation” class and they are called by the GUI so they can be displayed on the screen.

3.2 Data structures

I have used simple data types such as integers, integer arrays and on one ocasion a double and more complex one such as class objects, panel, buttons and so on. All the coefficients that are read from input are stored in either one of the 2 arrays P1 and P2. Their operations are stored either in a new array P3 or in P1(derivation,integration).

3.3 Class projection

I do not have huge amounts of classes but the ones i have are worth mentioning and talking about.

The classes(explained):

**GUI.java:**

This class has the job of implementing the interface that stands between the user and the application. It has quite a few buttons,a frame and a panel where all the buttons go on.

**Private Jbutton show1, show2, show3, add, subs, multi, derivation, integration,** are used to do the operations: addition, subtraction, multiplication, integration, derivation. The **exit** button does not need explanation.

**Private JTextArea** is the variable used for diplaying the Polynomials on the screen.

**Private JFrame** frame is used to delimitate the panel.

**Private JPanel** panel is the container for all the buttons and the textfield.

The constructor of this class designes the frame. I have based the layout on a Grid Layout divided into 10 panels ( 5 rows and 2 colums).

This class has quite a lot of methods because every button does a significant job and with 8 buttons in place I will explain what they do. The methods are:

**public void actionPerformed(ActionEvent e**) –this method has the job of handleing all the events in the class( clicking of buttons to be more precise ).

**private void writePoly(int[] polynom, int size)** – is used for displaying the Polynomials

code snippet:

**for** ( **int** j = size ; j >= 0; j-- ) {

**if** ( j != 0 )

**if** ( polynom[j] >= 0 ) {

text.append( polynom[j] + "x^" + j + "+" );

} **else** {

text.append( polynom[j] + "x^" + j + "-" );

}

**else** {

text.append( polynom[j] + j + "=0\n" );

}

}

**private void exit()**- self explanatory

**private void showPoly1()-** this method is used for displaying the first Polynomial \*

code snippet: show1. addActionListener (**new** ActionListener () {

@Override

**public** **void** actionPerformed ( ActionEvent e ) {

text.append ( "Polynomial 1:" );

writePoly ( poly.getP1(), poly.getSize1( ) );

}

});

**private void showPoly2()-**this method is used for displaying the second Polynomial \*

**private void showPoly3()-**this method is used for displaying the third Polynomial \*

**private void add()-** handles the addition of the two polynomials \*

**private void subs()-**handles the subtraction of the two polynomials \*

**private void multi()-**handles the multiplication of the two polynomials \*

code snippet: multi.addActionListener( **new** ActionListener ( ) {

@Override

**public** **void** actionPerformed( ActionEvent e ) {

maximum = poly.getSize1( ) + poly.getSize2( );

poly.multiplication( );

}

});

**private void derivation()-** calculates the derivative of teh first polynomial and displays it on the screen \*

**private void integrate()-** claculates the integral of the first polynomial and displays it on the screen\*

\*- all of these methods use getters for taking the values of the polynomials and their respective sizes

**Main.java**

This class contains the main method in whch the GUI isgenerated and the PolynomialOperation instanciates an onbject polynom.

**PolynomialOperation.java**

This is the class where most of the calculations are done and where the psudo-interface aka console input of the polynomial degrees and their coefficients are kept, creating the two Polynomials P1 and P2. P1 has the degree in size1(int variable) and P2 has the degree kept in size2(int variable). Also this is where the resulting polynomial is being created ( pRes ) . For the input from keyboard I have used a simple ‘Scanner keyboard = new Scanner(System.in)’ and using an int x the degrees of the two polynomials are being read from input.

The methods in this class are:

**private int[] readpoly –** this methods creates the two polynomials;

**protected int[] addition()-** this methods is the one that adds the two polynomials and saves the result in the variable pRes;i will explain the algorithm later

**protected int[] addition()-** this methods is the one that adds the two polynomials and saves the result in the variable pRes;i will explain the algorithm later

**protected int[] substraction()-** this methods is the one that subtracts the two polynomials and saves the result in the variable pRes;i will explain the algorithm later

**protected int[] multiplication()-** this methods is the one that multiplies the two polynomials and saves the result in the variable pRes;i will explain the algorithm later

**protected int[] multiplication()-** this methods is the one that multiplies the two polynomials and saves the result in the variable pRes;i will explain the algorithm later

**protected int[] integrate()-** this methods is the one that integrates the first polynomial ;i will explain the algorithm later

**protected int[] derivate()-** this methods is the one that derivates the first polynomial ;i will explain the algorithm later

There are also a number of getters that are used in the GUI class. The getters are: getP1 (), getP2 (), getSize1(), getSize2(), getPres(), getM().

3.4 The algorithms

I will describe the algorithms that i have used for making this program:

Addition:

Addition is one of the simplest polynomial operation. A math example would be he following:

(4x 5 + x 4 -12x 3 + x - 6) + (3x 4 +8x 3 +6x 2 - x) = ?

= 4x 5 + (1 + 3)x 4 + (- 12 + 8)x 3 +6x 2 + (- 1 + 1)x - 6

= 4x 5 +4x 4 -4x 3 +6x 2 + 0x - 6

= 4x 5 +4x 4 -4x 3 +6x 2 - 6

The method that i have applied using arrays:

protected int[] addition()- this is mainly the addition of 2 vectors from an index to index standpoint; the only difference is that if one of the arrays has a larger coefficient that should be copied first in the new array because errors can occur.

Code snippet: **int** i;

**if** (size1 > size2) {

**for** ( i = size1; i > size2; i-- )

pRes[i] = p1[i];

**for** ( i = size2; i >= 0; i-- ) {

pRes[i] = p1[i] + p2[i];

}

} **else** **if** (size2 > size1) {

**for** ( i = size2; i > size1; i-- )

pRes[i] = p2[i];

**for** ( i = size1; i >= 0; i-- ) {

pRes[i] = p1[i] + p2[i];

}

} **else**

**for** ( i = 0; i <= size1; i++ ) {

pRes[i] = p1[i] + p2[i];

}

**return** pRes;

}

protected int[] substraction()- this is mainly the subtraction of 2 vectors from an index to index standpoint; the only difference is that if one of the arrays has a larger coefficient that should be copied first in the new array because errors can occur. Careful at the sign.

protected int[] multiplication()- this multiplies all terms of the two arrays and stores them in the new one at the correct index; this operation is a for in for so the complexity is rather bad on large arrays.

Code snippet: **int** i;

**int** j;

**for** ( i = 0; i < size1 + size2; i++ ) {

pRes[i] = 0;

}

**for** ( i = 0; i <= size1; i++ ) {

**for** ( j = 0; j <= size2; j++ ) {

**int** c = i + j;

pRes[c] = pRes[i + j] + p1[i] \* p2[j];

}

}

protected double[] integrate()- this method integrates the first polynomial by converting it to an array of double; afterwards the new array copies the value of every value in the original array after it is divided by the next in line index ( i+1 ). This conversion is a bit controversial but it does the job.

**code snippet: int** i;

**double**[] m = Arrays.*stream*(p1).asDoubleStream().toArray();

**for** (i = 0; i <= size1; i++) {

m[i] = (**double**)p1[i] / (i + 1);

}

**for** (**int** j = size1+1; j >= 1; j--) {

System.***out***.print(m[j-1] + "x^" + j + "+");

}

protected int[] derivate()- this methods calculates the derivative of the first polynimial by multipliyng every element by its respective index and then at the end reducing the size of the array by 1.

Code snippet: **int** i;

**for** (i = size1; i>0; i--) {

p1[i] = p1[i] \* i;

System.***out***.println("p["+i+"]="+p1[i]);

}

4.Implementation and testing

I have implemented this program in Eclipse and I have used algorithms that could do with some improvement. The classic arrays that I have used might not be the most time efficient or memory efficient solutions but on the examples that a user can input from console will hold well. If they would be tested for high arrays the situation might me different.

The input is not checked by try and catch so if the user does something strange errors can appear. The graphical user interface cannot really generate errors because it does not let the user go that crazy. Checking for all possible bugs will be done in a future version.

Also I have not been able to calculate the division of two polynomials but I believe I will do it In future versions.

5.Results

This is a useful application to perform basic polynomial operations such as: addition, subtraction, multiplication, differentiation and integration. Because it is made in Java, the application is highly portable on any operating system that has Java JDK.

6. Conclusions

This has been a rather interesting assignment to work on. Even though it took a while I believe I have done a decent job.