Programming Techniques
HOMEWORK 1

Student: Tudorica Andrei
Group: 30421

0. Contents

1. Objective of the homework

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

2. Problem Analysis

A polynomial is described in the mathematics field as being an expression consisting of variables (or indeterminates) and coefficients. It involves only operations like addition, multiplication and subtractions. They only involve non-negative integer exponents.

When a polynomial is used as an expression, like in our case, ‘x’ is a fixed symbol which does not have any value, also known as an indeterminate. In this application we will use polynomials with a single indeterminate symbol.

Polynomials can also be considered as a series of smaller expressions, called monomials. Monomials have the following structure:

$c\,x^g$ where **c** is the coefficient, **g** is the degree and **x**, as denoted before, is the indeterminate.

The polynomials we use will only have integer coefficients. For instance a polynomial accepted as input for the application described in this document would look like this: x^2-9x-10.

As polynomials are used in a wide variety of domains, a tool that computes different operations on this mathematical structure can be very useful.

The system I developed, and that is described here computes the following operations on onea or more polynomials:

-Addition
-Division
-Subtraction
-Multiplication
-Differentiation
-Integration

3. Modelling

The process of modelling is defined as the process of modularising a big problem into smaller ones, that are easier to understand and debug. This also helps making

an abstract idea more clear. In software development, modelling is essential in order to build an application that has a strong background structure.

Our problem, as stated above in the “Problem Statement” paragraph, includes polynomials that we look at as a series of monomials, that is why the basic unit of our polynomial processing system will be the monomial.

4. Use cases

The developed software application is a basic one, therefore it only includes six operations. The operation are: Addition, Division, Subtraction, Multiplication, Differentiation, Integration.

4.1. Scenarios

Let's consider two examples:one for Addition and one for Differentiation. Before stating the steps of the application usage, the most important thing is that the user must be able to successfully launch the program.

IMPORTANT NOTES:

- The data input format should have the following pattern: “ C1 D1 C2 D2 C3 D3 C4 D4 . . . Cn Dn”. The elements are separated by a **single space character.** Ci is the coefficient of the i’th monomial and Di is the degree of that monomial.
- There is no input data validation method, so any data that is not respecting the rules will make the application stop.

Steps for the Addition Example:

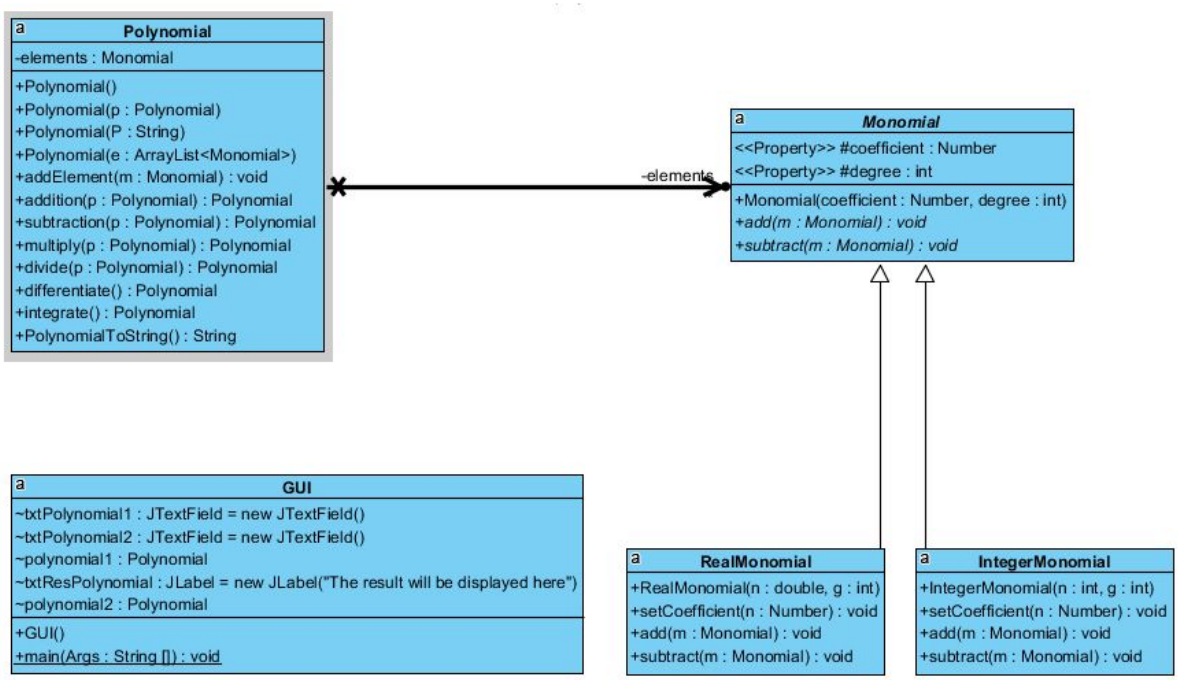
1. The app is launched.
2. The user introduces the first polynomial in the left text box (see the rules for introducing the input data in the program above).
3. The user introduces the second polynomial in the right text box (see the rules for introducing the input data in the program above).
4. The user clicks on the first button of the user interface, the one that has the word “Addition” written on it.
5. The application reads the first string given by the user.
6. The application converts the first string in an array of integer numbers.
7. The application converts each pair of numbers in a monomial, using the first number as the coefficient of the monomial and the second one as the degree of the monomial.
8. The application adds all these monomials into the first Polynomial.

9. The application reads the second string given by the user.
10. The application converts the second string in an array of integer numbers.
11. The application converts each pair of numbers in a monomial, using the first number as the coefficient of the monomial and the second one as the degree of the monomial.
12. The application adds all these monomials into the second Polynomial.
13. The application computes the sum of the 2 Polynomials.
14. The application outputs the result in label below the text boxes.
15. The user closes the window.

Steps for the Differentiation Example:

1. The app is launched.
2. The user introduces the polynomial in the left text box (see the rules for introducing the input data in the program above).
3. The user clicks on the first button of the user interface, the one that has the word “Differentiation” written on it.
4. The application reads the string given by the user.
5. The application converts the string in an array of integer numbers.
6. The application converts each pair of numbers in a monomial, using the first number as the coefficient of the monomial and the second one as the degree of the monomial.
7. The application adds all these monomials into the first Polynomial.
8. The application differentiates the previously built Polynomial.
9. The application outputs the result in label below the text boxes.
10. The user closes the window.

5. Design



As stated above, the monomial is the basic unit of this model, so, obviously, there we have the Monomial abstract class, which has 2 members: a coefficient (Number) and a degree (int). This class is extended by two other classes IntegerMonomial, for monomials that have integer coefficients, and RealMonomial, for monomials that have real coefficients.

Another class defined is the Polynomial class that contains an ArrayList of Monomials.

5.1. The Monomial class

```
package polynomial;

public abstract class Monomial {
    protected Number coefficient;
    protected int degree;
    public Monomial(Number coefficient, int degree)
    public Number getCoefficient()
    public void setDegree(int g)
    public int getDegree()
    public abstract void setCoefficient(Number n);
    public abstract void add(Monomial m);
    public abstract void subtract(Monomial m);
}
```

As stated above, the Monomial class is the atomic unit of this project. It contains a constructor, a setter and a getter for the coefficient property and a getter and a setter for the degree property. It also contains 2 abstract methods for adding and subtracting Monomials from the current instance of Monomial.

5.1.1 The IntegerMonomial class

```
package polynomial;

public class IntegerMonomial extends Monomial{
    public IntegerMonomial(int n,int g)

    public void setCoefficient(Number n) {

    public void add(Monomial m)

    public void subtract(Monomial m)



}
```

It implements the abstract methods defined in the Monomial class.

- The constructor gets two integers as arguments. The first one, n, becomes the coefficient of the Monomial being constructed and the second one, g, becomes the degree of the same Monomial.
- The setCoefficient method gets a Number as an argument and sets the coefficient of the current instance to that Number.
- The add method gets a Monomial as an argument and checks if this argument is an IntegerMonomial or a RealMonomial. If it is an IntegerMonomial it adds it to the current instance of the the IntegerMonomial class. Otherwise, it calls the add function int the RealMonomial class, to make the sum as real numbers.
- The subtract method gets a Monomial as an argument and checks whether it it is or it isn't a RealMonomial. If it is real, the function calls the subtract method in the RealMonomial class, to process the operation for real numbers. On the other hand, if the Monomial is an integer one, it multiplies its coefficient by -1 and calls the add function of the same class, as the difference between two monomials m1 and m2 equals the sum between m1 and m3, where m3 has the same degree as m2 but the coefficient is the one of m2 multiplied by -1.

5.1.2 The RealMonomial class

```
package polynomial;

public class RealMonomial extends Monomial {
    public RealMonomial(double n,int g)

    public void setCoefficient(Number n) {

    public void add(Monomial m) {

    public void subtract(Monomial m) {
}
```

It implements the abstract methods defined in the Monomial class.

- The constructor gets a double and an integer as arguments. The double one, n, becomes the coefficient of the Monomial being constructed and the second one, g, becomes the degree of the same Monomial.
- The setCoefficient method gets a Number as an argument and sets the coefficient of the current instance to that Number.
- The add method gets a Monomial as an argument and adds it to the current instance of the the RealMonomia class.
- The subtract method gets a Monomial as an argument and it multiplies its coefficient by -1 and calls the add function of the same class, as the difference between two monomials m1 and m2 equals the sum between m1 and m3, where m3 has the same degree as m2 but the coefficient is the one of m2 multiplied by -1.

5.2. The Polynomial class

```
package polynomial;

import java.util.ArrayList;

public class Polynomial {
    private ArrayList<Monomial> elements;
    public Polynomial()

    public Polynomial(Polynomial p)

    public Polynomial(String P)

    public Polynomial(ArrayList<Monomial> e)

    public void addElement(Monomial m)

    public Polynomial addition(Polynomial p)

    public Polynomial subtraction(Polynomial p)

    public Polynomial multiply(Polynomial p)

    public Polynomial divide(Polynomial p)

    public Polynomial differentiate()

    public Polynomial integrate()

    /*public void ResolvePolynomial()

    public String PolynomialToString()
}
```

In this class the entire functionality of the system is implemented.(the word 'this' will be used in the following to describe the Polynomial that is currently being processed) In the following I will explain each element defined in this class:

- Elements is an ArrayList of Monomial instances. As described in section 2, a polynomial is defined by a set of monomials. An ArrayList is like a List but with the implementation of a resizable-array. It implements the entire set of optional list operations. It also permits all sort of elements, the null element also being included. Not only it implements the List interface, but this class allows you to modify and change the size of the array used. The get, set, isEmpty, size, iterator, and listIterator operations run in constant time. All of the other operations run in O(n) time. The elements in this Array List will be the Monomials with non zero coefficients, sorted in descending order. In order

to assure this order for certain algorithms used in the code, an additional sort operation may be required.

- The constructor that takes no arguments, instantiates the elements Array List with a new Array List, which is empty. (elements = new ArrayList <Monomial> ();)
- The second constructor takes another Polynomial as an argument and instantiates the Polynomial being built with copies of the elements in the received Polynomial.
- The third constructor Takes the String P as an argument. This string must respect the rules enunciated in the “IMPORTANT NOTES” part of the 4.1 section, otherwise it will not work. The constructor splits the string by the space character, and adds the piece into an array of strings called parts. Further on, this array will be iterated through two elements by two elements and parsed to pairs like (int , int) or, (double , int), depending on the case. These pairs, as expected, have the form of (coefficient, degree), and Monomials are created on their behalf. These Monomials can be IntegerMonomial or RealMonomial, depending on the datatype of the first number of the pairs previously described. This constructor is used for converting the string from the text fields where the user gives his input, into Polynomials for them to be easier to process.
- The last constructor Builds the polynomial on the behalf of an array list of Monomials, directly linking the elements of this.elements to the elements of e.
- The addElement method receives a Monomial as an argument and adds it to the array list of Monomials in the current Polynomial.
- The addition method gets another Polynomial as an argument and basically computes the sum of this and that polynomial. It iterates through the elements of the polynom to be added and for each Monomial in that Polynomial, it looks for a Monomial of the same degree in the this Polynomial. If there is one the method sums up the two Monomials using the methods in the IntegerMonomial and IntegerMonomial classes. If there is no Monomial of the same degree, it adds the Monomial to the list of elements if this. The result is returned and it can also be found in this. The method does not return a sorted Polynomial.
- The subtraction method gets a Polynomial as an argument and basically computes the difference of this and that polynomial. It iterates through the elements of the polynom to be subtracted from this and for each Monomial in that Polynomial, it looks for a Monomial of the same degree in the this Polynomial. If there is one the method subtracts the Monomial from the received Polynomial from the found Monomial in this using the methods in the IntegerMonomial and INtegerMonomial classes. If there is no Monomial of the same degree, it adds the Monomial to the list of elements if this. The result is

returned and it can also be found in this. The method does not return a sorted Polynomial.

- The multiply method receives a Polynomial as an argument and returns the product of this and the received Polynomial. The method creates a new Monomial for each pair of monomials, taking one from this and the other one from the received Polynomial. It then creates a new Polynomial p1 containing only the Monomial we just computed. Finally it computes the sum between the prod Polynomial (the result Polynomial) and p1. This assures us that there will be no duplicate degree after the multiplication process.
- The divide method takes as an argument a Polynomial and returns this divided by the received polynomial. The explanation for the algorithm used on this is very long but there will be a link in the bibliography at point 1 with the step by step algorithm.
- The integrate method returns the integrated version of this instance of Polynomial. It takes each Monomial of the Polynomial and creates a new Monomial with the degree increased by 1 and the coefficient divided by the previous degree. It adds the new Monomial in the res Polynomial, which will be returned.
- The differentiate method returns the differentiated version of this instance of Polynomial. It takes each Monomial of the Polynomial and creates a new Monomial with the degree decreased by 1 and the coefficient multiplied by the previous degree. It adds the new Monomial in the res Polynomial, which will be returned.
- The last method of this class is the PolynomialToString method. It returns a user readable version of the current Polynomial. It first sorts the elements list in descending order by degree. Then, using a StringBuilder for every monomial with a degree greater than 1, creates and appends the string adding plus or minus between the monomials, depending on the case, the coefficient, the multiplication sign, the “x^” string and the degree. For the element with degree 1 it only creates and appends the string formed by the coefficient and “x”. For the element of degree 0 it only appends the coefficient. The coefficients are appended as doubles or as integers, depending on the type Monomial. The String is returned for it to be outputed.

5.3. The GUI class

```
import polynomial.*;

public class GUI extends Frame
{
    JTextField txtPolynomial1=new JTextField();
    JTextField txtPolynomial2=new JTextField();
    JLabel txtResPolynomial=new JLabel("The result will be displayed here");
    Polynomial polynomial1;
    Polynomial polynomial2;

    public GUI()

    public static void main (String Args[])
}
```

The GUI class is the class that contains the public static void main method, so this is where the application will start.

- The txtPolynomial1 JTextField is the graphical user interface text box where the user will introduce the coefficients and degrees of the first Polynomial.
- The txtPolynomial2 JTextField is the graphical user interface text box where the user will introduce the coefficients and degrees of the second Polynomial.
- The txtResPolynomial JLabel is the graphical user interface where the user readable version of the result Polynomial will be displayed.
- The polynomial1 is the structure in which the Polynomial from the txtPolynomial1 JTextField will be stored.
- The polynomial2 is the structure in which the Polynomial from the txtPolynomial2 JTextField will be stored.
- The GUI constructor contains the entire functionality of the graphical user interface. The following are done in the context of this method:
    1. I initialize the frame with the title Polynomial.
    2. The default close operation of the frame is when it is closed.
    3. The prefered sizes of the JTextFields txtPolynomial1 and txtPolynomial2 are set.
    4. The graphical user interface buttons btnAddition, btnSubtraction,btnMultiplication, btnDivision, btnIntegration, btnDifferentiate are defined and their texts are set to anounce the operation that will be performed if each button is clicked.
    5. The btnAddition button action listener reads and initializes the two Polynomials polynomial1 and polynomial2 using the texts in txtPolynomial1 and txtPolynomial2 text boxes. It then sets the text of the txtResPolynomial to the readable version of the result of the addition of the two Polynomials.
    6. The btnSubstraction button action listener reads and initializes the two Polynomials polynomial1 and polynomial2 using the

texts in txtPolynomial1 and txtPolynomial2 text boxes. It then sets the text of the txtResPolynomial to the readable version of the result of the polynomial1 - polynomial2.

7. The btnMultiplication button action listener reads and initializes the two Polynomials polynomial1 and polynomial2 using the texts in txtPolynomial1 and txtPolynomial2 text boxes. It then sets the text of the txtResPolynomial to the readable version of the result of the multiplication of the two Polynomials.
8. The btnDivision button action listener reads and initializes the two Polynomials polynomial1 and polynomial2 using the texts in txtPolynomial1 and txtPolynomial2 text boxes. It then sets the text of the txtResPolynomial to the readable version of the result of the polynomial1 / polynomial2.
9. The btnIntegration button action listener reads and initializes the Polynomials polynomial1 using the text in txtPolynomial1 text box. It then sets the text of the txtResPolynomial to the readable version of the result of the integration of polynomial1.
10. The btnDifferentiate button action listener reads and initializes the Polynomials polynomial1 using the text in txtPolynomial1 text box. It then sets the text of the txtResPolynomial to the readable version of the result of the differentiation of polynomial1.
11. A panel is created.
12. The buttons and text boxes are added to that panel.
13. The panel is added into the frame.
14. The frame is set visible.

6. Results

The result of the development of his application, that lasted for two weeks together with writing the documentation, is a software easy to use that makes it easy for the user to perform a couple of basic computations on polynomials. This tool can be, of course improved in several ways that will be described in the next section.

7. Further development

A couple features can be modified or added in order to improve the functionality of the application:

- A better graphical user interface
- A simpler way of introducing the data
- Computations on more than 2 polynomials

- More operations

8.Conclusions

During the development of this project i learned and improved of several skills:

- A better use of Eclipse
- A better use of the Java coding language
- A better understanding and usage of the object oriented programming paradigms
- More organized and clear code

9. Bibliography

1. Polynomial division tutorial http://www.purplemath.com/modules/polydiv2.htm
2. Java general documentation https://docs.oracle.com/javase/7/docs/api/
3. Lecture and laboratory guidelines http://www.coned.utcluj.ro/~salomie/PT_Lic/3_Lab/
4. Word counter https://wordcounter.net/