*Technical University of Cluj-Napoca  
Faculty of Automation and Computer Science  
Department of Computer Science  
2nd Semester 2017-2018*

*Student: Crisan Adrian   
Group: 30433*

Programming Techniques

Homework 1

Polynomial Calculus

**Table of Contents:**

1. Problem Specification . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
2. Example of working . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
3. Problem Design. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

3.2.1 Monom Class . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

3.2.2 Polinom Class . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

3.2.3 Operations Class. . . . . . . . . . . . . . . . . . . . . . . . . . 7

3.2.4 PolDemo Class . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

3.2.5 PolTest Class. . . . . . .. . . . . . . . . . . . . . . . . . . . . . 9

3.3 Packages and Interfaces. . . . . . . . . . . . . . . . . . . .. . . 10

3.4 User Interface. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

1. Using and testing the application. . . . . . . . . . . . . . . . . . . . . . . . 12
2. Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
3. Bibliography. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

**1. Problem Specification**

(EN) Lab – Homework 1

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

The designed system is able to perform certain arithmetic operations on one or two polynomials, such as :

* Addition
* Subtraction
* Multiplication
* Division
* Differentiation
* Integration

The System will have a graphical interface through which the user can enter the polynomials to be used in the desired arithmetic operation, which can be chosen as well through the command buttons of the GUI.

Using only Java for both the implementation and the user interface, one was able to develop a standalone Polynomial Operations application which includes multiple functions and operations.

**2. Example of working**

For the following examples we assume that the user entered the following polynomials: 1X^2+2X^1+1X^0 and 2X^1+2X^0 . The program will output, for the given polynomials, the following values:

* Addition result : +1X^2+4X^1+3X^0
* Subtraction result: +1X^2-1X^0
* Multiplication result: +2X^3+6X^2+6X^1+2X^0
* Differentiation(applied on the first polynomial) result: +2X^1+2X^0
* Integration(applied on the first polynomial result : +0.333333334X^3+1X^2+1X^1

The program has a calculator appearance where the user can introduce the two polynomials and can select the desired operation by clicking on one of the existing buttons. The result will be printed at the bottom of the window in the text field: “Result:” .

**3. Problem Design**

**3.1 Class .Design**

***.***

3.1.1. The Monom Class:

**public class Monom implements Comparable<Monom>**

This class is the representation of a monomial.

The Fields of the Monom Class:

* A public float field “coefficient” corresponding to the coefficient of the monomial
* An public int field ”degree” corresponding to the degree of the monomial

The Constructor of the Monom Class:

**Monom(int degree, float coefficient)**()

* It will initialize the above mentioned fields ( the coefficient and the degree )

The Methods of the Monom Class:

* Setters and Getters method:
  + **public float getCoefficient()**
  + **public int getDegree()**
  + **public void setCoefficient(float y)**
  + **public void setDegree(int x)**
  + **public void setMonom(int d, float c)**
* In addition to the setters and getters method I’ve also implemented the **public int CompareTo(Monom other)**, which I’ve used in order to sort the resulting polynomial in a decreasing order of degrees.

3.1.2. The Polinom Class: **public class Polinom**

This class is a representation of a polynomial, which is basically a List<> of Monom objects. Using objects of type Polinom, the application is able to perform the required operations .

The Fields of the Polinom Class:

* A String field called “polinom” .
* A List<> of monomials called “pol”.

The Constructors of the Polinom Class:

* **public Polinom(String x)** :The constructor receives a String argument called “x” and is used to pass the input polynomial read as a string to the object of Polinom type.
* **public Polinom()** : used to create empty Polinom objects.

The Methods of the Polinom Class:

* **void addMonom(int grad, float coef):** this method is used to add a Monom type object, which represents the monomials of the polynomial, in the List<> of Monom.
* **void removeMonom(Monom m):** this method is used to remove a Monom object from the List<> pol.
* **void clearPol():** this method is used to empty the List<> pol.
* **void showPolinom():** I used this method before creating the GUI in order to see the List<> pol.
* **void String toString():** this method overrides the toString method of the java.lang.Object library in order to convert the List<> pol of Monom in a string that can be printed in a textfield.

3.1.3. The Operations Class: **public class Operations**

This class contains all the operations to be performed on polynomials.

The Fields of the Operations Class:

* This class has no fields.

The Constructor of the Operations Class:

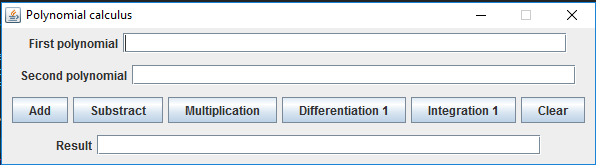
* **Operations():** this constructor creates an empty instance of the Operations class.

The Methods of the Operations class:

* **public Polinom addPol(Polinom pol1, Polinom pol2)**– Performs the addition of two polynomials and returns the resulting polynomial. It adds the coefficients of the Monom objects that have the same degree and afterwards adds to the resulting polynomial the Monom objects that have degrees which do not belong to both input polynomials and it does not add to the resulting polynomial Monom objects that have the coefficient equal to 0.
* **public Polinom subPol(Polinom pol1, Polinom pol2)**– Performs the subtraction of two polynomials and returns the resulting polynomial. It subtracts the coefficients of the Monom objects that have the same degree and afterwards adds to the resulting polynomial the Monom objects that have degrees which do not belong to both input polynomials and it does not add to the resulting polynomial Monom objects that have the coefficient equal to 0.
* **public Polinom mulPol(Polinom pol1, Polinom pol2) –** Performs the multiplication of two polynomials and returns the resulting polynomial. It multiplies the coefficients of each Monom object and adds their degrees.
* **public Polinom derive(Polinom pol1)-** Performs the differentiation of a polynomial. It multiplies the coefficient of each Monom object with its own degree and subtracts 1 from the degree.
* **public Polinom integrate(Polinom pol1) –** Performs the integration of a polynomial. It adds 1 to each Monom object’s degree and it divides each coefficient with the initial degree.

3.1.4. The PolDemo Class : **public class PolDemo**

Using elements from multiple packages I managed to create a graphical user interface, which can be seen below:



Note the two text fields used for reading the polynomials at the top of the window. Below them there are a few buttons, each of them corresponding to a specific operation. The result can be seen in a third text field, which is used only for displaying it.

The Fields of the PolDemo Class:

* **JFrame appFrame;**
* **JLabel jlbFirstPol, jlbSecondPol, jlbResult;**
* **JTextField jtfFirstPol, jtfSecondPol, jtfResult;**
* **JPanel panel1, panel2, panel3, panel4;**
* **JButton addPol, subPol, mulPol, derPol, integPol, clear;**
* **Polinom pol1 = new Polinom(“”);**
* **Polinom pol2 = new Polinom(“”);**
* **Operations o = new Operations();**
* **String resText = null;**

The Constructor of the PolDemo Class:

* **public PolDemo() {**

**createGUI();**

**} –** which instantiates the GUI object type.

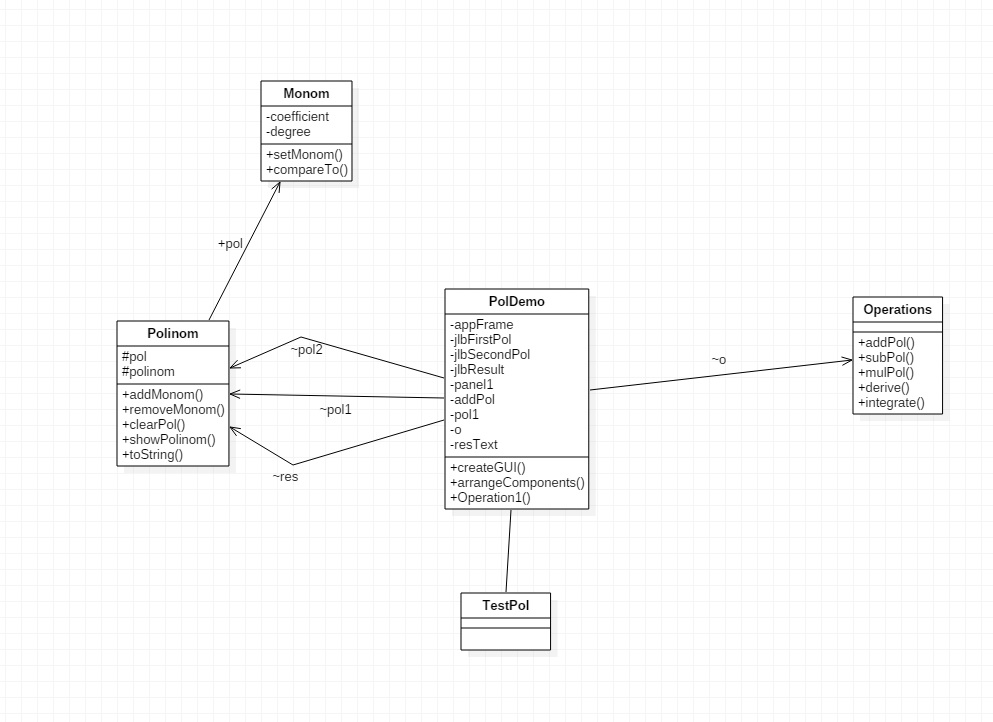
The Methods of the PolDemo class:

* **public void createGUI() –** this method creates the frame of the window and sets its parameters.
* **public void arrangeComponents() –** this method effectively implements all the fields and the buttons used in the GUI and also implements the functionality of the buttons using the help of ActionListener. So after the user introduced correctly two polynomials in the first two text fields, if the user presses the Addition button for example, then the two input polynomials are taken as strings and processed using the help of Pattern and Matcher packages from java.util.regex import and the resulting polynomial is printed as a string in the “Result:” text field.

3.1.5. The main Class: **public class TestPol**

This class contains only the  **public static void main(String args[])** method which makes the whole application run . It only creates an object of type PolDemo and the GUI is shown with all its functionalities.

3.1.6. Class diagram



* 1. **Packages and Interfaces**

A Java package is a mechanism for organizing Java [classes](http://en.wikipedia.org/wiki/Class_%28computer_science%29) into [namespaces](http://en.wikipedia.org/wiki/Namespace_%28computer_science%29). Each type present in a package has its own namespace. Classes in the same package can access each other’s package-private and protected members. The classes and interfaces grouped in a package are related in one way or another.

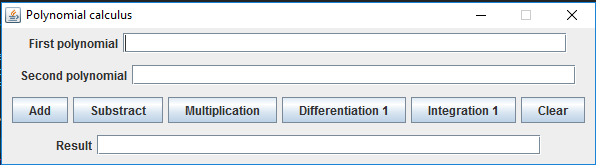
For this application certain packages were imported, most of them related to the graphical user interface and its functionallity.

* **import java.awt.event.\*;**
* **import java.util.regex.\*;**
* **import java.swing.\*;**

Typical swing applications do processing in response to an event generated by the user. For example, clicking a Jbuttion will notify all the ActionListeners added to that JButton.

**3.3 User Interface**

When running the application a window “Polynomial Calculus” will pop-up and the user will have the possibility to enter input polynomials as predefined strings and to choose which operation to be performed . The user interface is based on characteristics of the packages before mentioned. All the attributes needed are created by the PolDemo *c*lass.



In order to perform operations, the application uses predefined functions from the *ActionListener* interface. The *ActionListener* interface is for receiving action events. When the action event occurs, that object's actionPerformed method is invoked. In this case the only events that occur are when the user clicks on one of the operation buttons from the graphical interface.

For each implemented button I have implemented inside its ActionListener the methods required for processing the strings, getting the coefficients and degree of each monomial thus creating the List<> of Monom which corresponds to the input polynomial.

**4. Using and testing the application**

Rules for the input data:

* The input string must be give following the next pattern:

sign (+/- ) + coefficient + “X^”+ degree for each monomial

* The sign of the first monomial can be omitted if it is positive.
* If you are required to enter constants, they should be written like “X^0” because of the conditions of the parsing algorithm.

Testing examples :

* Valid example : 2X^3+3X^2-2X^1+1X^0
* Invalid example : 3X^7-2X^1+2

For a better testing a JUnit Test Case was created in order to verify the results of each operation;

* I created a new class, PolJUnit;
* We test the methods from Operations class using *assertEquals* in order to verify the result of the specific operation on two predefined polynomials.

Take as an example the *add* method. All the other methods are tested in a similar way.



**5.Conclusions**

This first assignment that requests object-oriented concepts from the students that are relatively new to it proved to be very difficult for me, as I had a great lack of knowledge in this field. But after taking again the basics of Java and OOP I have finally managed to complete it as best as I could.

Speaking about implementation and the algorithms used, the only big problem that occurred was the parsing algorithm and the implementation of the ActionListeners.

The reading and printing of a polynomial processes which seemed easy in the first place required some special methods that also increased the difficulty of the project, taking a lot of time testing, trying to remove and setting its restrictions.

Speaking of the assignment again, a better performance for my implementation would be given by the existence of all the cases where exception can occur and handling them. Also the graphical user interface can be heavily improved.

**6.References**

* Prof. Joldos’ laboratory papers and course notes from the first semester;
* <https://docs.oracle.com/javase/tutorial/collections/algorithms/index.html>
* <https://courses.cs.washington.edu/courses/cse143/11wi/eclipse-tutorial/junit.shtml>