**TECHNICAL UNIVERSITY OF CLUJ-NAPOCA**

**Laboratory Work: Polynomial Processing**

**Name: Nistor Rares Constantin**

**Group: 30425**

**Content:**

1. **Introduction**

**1.1 Homework objective……………………………………………………... page 3**

1. **Problem solving**

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

**2.2 Modeling …………………………………………………………………….....page 3**

**2.3 Scenarios and use cases…………………………………………………..page 3**

1. **Projection**

**3.1 UML diagram…………………………………………………………………..page 4**

**3.2 Data structures………………………………………………………………..page 6**

**3.3 Class projection……………………………………………………………….page 6**

**3.4 User interface………………………………………………………………….page 9**

**3.5 Packages……………………………………………………………………….page 11**

**3.6 Algorithms…………………………………………………………………….page 11**

1. **Implementation and testing……………………………………………page 13**
2. **Results……………………………………………………………………………page 14**
3. **Conclusions…………………………………………………………………….page 16**
4. **Bibliography…………………………………………………………………..page 17**
5. **Introduction.**
   1. **Homework objective**

The objective of the first homework is the following: **“**Propose, design and implement a system for polynomial processing. Consider the polynomials of one variable and integer coefficients**”**. In other words we have to create a program that does specific operations (addition, subtraction, division, multiplication, differentiation, integration, etc) on a set of polynomials.

1. **Problem solving**

**2.1 Problem analysis**

To solve this problem we need former java and OOP knowledge and some mathematical formulas. So I started to think how the operations are done and what types of data should I use to make it easier to implement them. The powers of the polynomial can be integers but since the coefficients must be divided they must be a float or double type. Unfortunately because I have a limited experience with java and OOP I didn’t did such a good job dividing the program into classes and methods and using the right kind of variables. Hopefully this will change in the future.

**2.2 Modeling**

As I said above because of my limited knowledge the modeling process was affected. I started by creating the user interface and making it user friendly. Then I started thinking how shod I extract data from the interface and store in and finally how to work with it. There are many ways to introduce polynomials you can introduce the coefficients and its power depends on its position. But in this case is someone wanted to introduce a polynomial like “8x^6+1x^0” it would be pretty uncomfortable for the user because he would have to write”1 0 0 0 0 0 6” so I decided to let the user writhe the coefficient and its power. Then I had to store the data somewhere so I decided to use vectors to store the coefficient and powers (not the greatest idea).

**2.3 Scenarios and use cases**

The scenarios and use cases are strongly related to the user and the graphic interface so I tried to make it as simple and friendly as possible. The user introduces the two polynomials in two text fields (for ex: 5x^6-2x^0) the he presses the button corresponding with the operation he wishes to execute and the output will be displayed. In case he enters the wrong polynomial or in a wrong way he can just reenter it.

1. Projection

**3.1 UML**

**Use case diagram:**

Actor

Addition

Subtraction

Division

Multiplication

Differentiation

Integration

In the previous figure the Use case diagram is represented. This diagram shows what actions the actor (user) can perform using the program. In this particular case the user can perform the addition, subtraction, multiplication, division of two polynomials and the integration and differentiation of the first polynomial after he introduces the respective polynomial in the specific fields found in the interface.

**Class diagram:**

**GUI**

jp: JPanel

jp2: JPanel

jp3: JPanel

l1: JLabel

l2: JLabel

l3: JLabel

l4: JLabel

t1: JTextField

t2:JTextField

t3:JTextField

bAdd: JButton

bMul: JButton

bDiv: JButton

bSub:JButton

diff: JButton

integ: JButton

p1: Polynomial

p2: Polynomial

o : Operations

actionPerformed(e: ActionEvent): void

<<create>> GUI()

**Main**

main(arg: string): void

**HelpfulFunctions**

+polynomialToString(p: Polynomial): string

+getCoefficient(i:int, s: string): double

+getPower(i: int, s: string): int

+sort(p: Polynomial): void

**Polynomial**

#coefficient: double[]

#power: double[]

#size: int

-createCoefficient(s: String) : void

-createPower(s: String): void

+createPolynomial(s: String): void

**Operations**

+add(p1,p2 : Polynomial): string

+sub(p1,p2 : Polynomial): string

+mul(p1,p2 : Polynomial): string

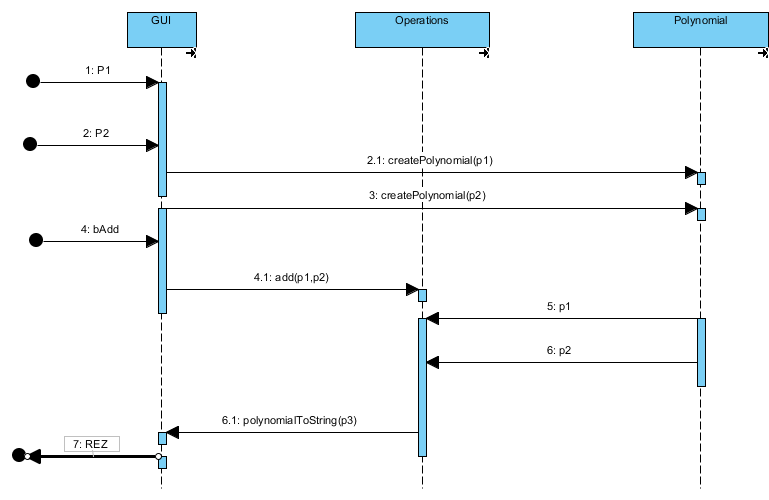
+divi(p1,p2 : Polynomial): string

+diff(p1,p2 : Polynomial): string

+integration(p1,p2 : Polynomial): string

The class diagram show for each class the variables and methods that are contained by that class, how visible they are, what type they are(int, double, string, etc) and what type of parameters they have(only for methods). The class diagram also show how the classes are connected one to each other and what type of relation is between them. For example the classes “Operations” and “Polynomial” extends “HelpfulFunctions” because they use methods from that function, between the classes “Operations”, ”Polynomial” and “GUI” there is an aggregation relation because GUI contains references to objects from the first two classes.

**Sequence diagram:**

 The sequence diagram show what sequence of methods the program calls when he is given the task to add two polynomials.

**3.2 Data structures**

The data structures that are used are data types like int, double, int vector, double vector, and objects like Polynomials and Operations. I used two vectors to store the data of the Polynomial: a double vector for the coefficients and an int vector for the powers. It would have been better to use an ArrayList to store the data because it doesn’t have a fixed dimension, like a vector, but unfortunately I realized this too late and don’t have time to make the changes. The object Operations is used to call the adding, subtracting, and all the other operations on polynomials.

**3.3 Class projection**

Now we will talk about how the program was divided in subprograms that are easier to complete. I divided the solution of the problem in five classes(1 class for testing). These five classes are: Polynomials, Operations, GUI, HelpfulFunctions, and the Main (the test class).

***Polynomials***

In this class the polynomials are created from the string recived as input in the user interface. Polynomials extends another class HelpfulFunctions(about which it will be talked later )and has the following variables:

* **protected coefficient**: a double vector in which the coefficients of the elements of the polynomial are store.
* **protected power**: a int vector in which the powers of the elements of the polynomial are store.

And a variable

* **protected size**: represents the number of elements in the vectors

These two vectors have the same length and the coefficient at the position i has the corresponding power on the same position.

The class also contains 3 methods:

* **private void createCoefficient(String s)**: this methods receives as a parameter a string which is the polynomial and extract from it, using a parsing function from the class HelpfulFunctions, the coefficients that are stored in the coefficient vector.
* **private void createPower(String s)**: this method receives as a parameter the same string that is received by the previous method and is extracts, using another parsing method from HelpfulFunctions, the powers of the elements of the polynomial and stores them in the power vector.
* **private void createPolynomial(String s)**. This class just calls the previous two classes (createPower and createIndex).

***Operations***

In this class all the operations on the polynomials are extends. It also extends HelpfulFunctions. This class has no variable and has six methods and all the methods have as parameters 2 polynomials and use a method from HelpfulFunctions to convert the resulting polynomial into a string.

* **public String add (Polynomial p1, Polynomial p2)**: this function computes the addition of 2 vectors and returns the result as a string of type ”ax^b+cx^r+…”.
* **public String sub (Polynomial p1, Polynomial p2)**: this function subtracts the 2 polynomials and return a string like above like all the other functions in this class
* **public void mul (Polynomial p1, Polynomial p2)**: this function multiplies the two polynomials.
* **public void divi (Polynomial p1, Polynomial p2)**: divides the two polynomials
* **public void diff(Polynomial p1)**: differentiates the polynomial
* **public void integration(Polynomial p1)**: integrates the polynomial

***HelpfulFunctions***

This class is extended by Polynomials and Operations. It contains the following methods:

* **public double getCoefficient(int i, String s)**: this functions has as parameters the string s which is a polynomial(ex: 5x^2+3x^7) and the index i which represents the position where ’x’ is. This method extracts the number before x and returns it as a double.
* **public int getPower(int i, String s)**: this function receives as parameters the string s like above a the index i which represent the position where ‘^’ is and returns the power of x as a int.

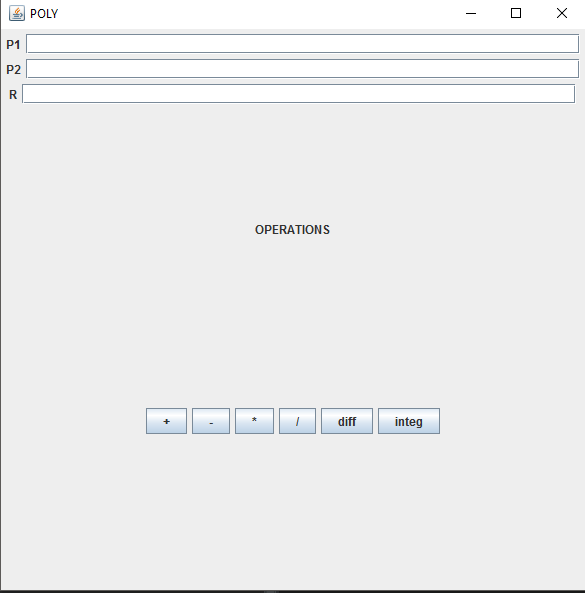
These two methods are used by the functions createCoefficient and createPower from the Polynomials class.

* **public String polynomialToString(Polynomial p)**: this method has as parameter a polynomial and converts it to a string. This method is used by all the methods in the class Operations.
* **sort(Polynomial p)**: this method is used by thepolynomialToString method(to be sure that the first el has the biggest power, the last the smallest and so on) and by the divi method in the Operations class (we need to be sure that the first el of each polynomial has the biggest power ).

***GUI***

The GUI is the class where the user interface is implemented. It has a constructor in which all the JButtons (which are used for selecting the operation: addition, substraction,etc), JPanels (on which the buttons and text fields are added) and JTextFields (where the polynomials are introduced and the output is displayed) are added. Also in the constructor we specify information about the Frame : the layout(Grid), its size(600,600), etc. This class also implements the ActionListener so here is also where the ActionPerformed method is implemented. The ActionPerformed method allows us to use certain methods or parts of code when a button is pressed or data is introduced in the textfield.

**3.4 User interface**

****

This is the user interface in which the user will introduce data and receive the result. The interface has three JTextFields P1, P2 and R. The first JTextField (P1) is the place where the user will introduce the firs polynomial; the second JTextField (P2) is where the user will introduce the second polynomial. The polynomials can must have the following form (“ax^b +() cx^d+(-)ex^f……”). The polynomial can be introduce starting with any power (5x^3 + 2x^2 +6x^7), it is not mandatory to introduce them in the decreasing order of its elements. After introducing each polynomial the user must press Enter for the polynomial to be saved. The third JTextField (R) shows the result of the operations done to the polynomials. The result will always be shown in the decreasing order of the powers.



In the interface there are also six JButtons for the operations. After introducing the polynomials the user can press any button and the result will be displayed.



**3.5 Packages**

The program has 3 packages the Test package which contains the Main class. The main class has a GUI object for testing. The gui package has the GUI class in which the user interface is implemented and the program package which has the classes HelpfulFunctions, Operations and Polynomial. These classes are the one that solve the problem.

**3.6 Algorithms**

The main algorithms are in the Operation class and in the UsefullFunction class.

Now for the Operations class the methods will be described. In every method there are auxiliary objects of type polynomial so that the ones that were introduced will not change their value. This way it won’t be necessary to introduce the same polynomials again after each operation effectuated.

* **add()**

In the add method the value of the first polynomial (p1) is saved in auxiliary polynomial (p3). After the elements in the vector power from p3 are compared with the elements of the power vector from p2. If 2 elements have the same power the coefficient from p3, at the same index as the power, increases its value with the coefficient of p2,which is found at the index where the power was equal to the power of p3. If there is a element in p2 which power is not found in p3 the size of p3 is increased by one and it is added at the end of p3. In the end the method polynomialToString is called to get p3, which is now the sum of the 2 polynomials, sorts it and transforms it to a string to be displayed in the JTextField R.

* **sub()**

The sub method is very similar to the add method the only difference is that the elements are subtracted not added.

* **mul()**

In the multiplication method the again an auxiliary polynomial, p3, is created. Every element of p2 is multiplied with every element of p1, the coefficients are multiplied and powers added, and the resulting coefficient and power are saved in the vectors in p3, then the value of p3 is incremented. This way p3 becomes the result of the multiplication and it’s sent as a parameter to the function polynomialToString.

* **divi()**

The division algorithm is the most complex algorithm in the program. First the two polynomials are sorted, because the user can introduce its elements in any order and for division the polynomials must be sorted, then two auxiliary polynomials p3 and p4 take the values of p1 and p2, the original polynomials, and another polynomials p5 is created, in which the final result will be stored. While the power first element of p3 (at this point p3 is equal to p1) is greater or equal to the power of the first element if p2 we divide the 2 elements and the result is saved in p5, this is the first element of the result. Now the newly added element of p5 is multiplied with p4, which initially has the value of p2, and the result is stored p4. Then we subtract p3 with p4 , using the sub method, and the result is saved in p3. P3 is the sorted to be sure that p3 has the element with the highest power at the beginning. And while the power of the first element of p3 is grater that the power of the first element of p2 the same operations are repeated. When the condition is no longer fulfilled p5 will be the result of the division and the function polynomialToString will be called to create a string after p5.

* **diff()**

The differentiation function has only one parameter p1. And again its value is saved in an auxiliary polynomial p3. Each coefficient of p3 is multiplied with its corresponding power and then the power is decremented. p3 will store the result of the differentiation.

* **integrations()**

This method also has one parameter, p1. Like above p3 takes the value of p1. The powers of the elements of p3 are incremented the each coefficient is divided by their corresponding power and like above polynomialsToString is called to create the output string.

For UsefullFunction class almost all the methods, besides sort, are used for extracting the power and coefficient from a string or creating a string when we have the coefficient and the power. One such method is polynomialToString. This method has as parameter a polynomials and it starts verifying its coefficient if it is different than 0 we add that coefficient to a string followed by the characters ‘x’, ‘^’ and its power. Next we verify if the next coefficient, different from 0, is positive, if it is we add the character ‘+’ to the string. We repeat this steps untilled we reach the last coefficient of the polynomial. In the end we verify something was added to the string if not this means that the result is zero so we add ‘0’. This function is called by the ActionPerformed method in the GUI to give the output to the text field R.

The other two functions are getPower and getCoefficient. They recive as a parameter the string that is introduced by the user. Then we check the string for certain characters like ‘x’ in case of getCoefficient and ’^’ in the case of getPower. After we found this character we extract the digits near them form a number and store it in the vector. This is the function that is used to put elements in the coefficient vector and power vector of a polynomial.

1. **Implementation and Testing**

The implementation in not so faithful to the OOP principles, It resembles a more C++ implementation. As I said before the user must introduce the polynomials in the two text fields in the following form ”ax^b+cx^d-…..” and then press Enter after introducing the data in the first text field and then press again after introducing the data in the second text field. If the users doesn’t press enter that polynomial will have the value 0 and the result will be 0 or a random value. If he introduces o polynomial in a different form again the result will be a random value, but the program will not crash and he can simply reintroduce the data in the correct form and the right result will be displayed. We can consider the polynomial just a normal number if its introduced in the right way ( nrx^0). The testing part was pretty limited. It consisted in introducing a set of and presiing the buttons that represent the operations in different order or a button repeatedly to see if the value is the right one or if at some point I am given a random value. From what I can tell so far there are no errors or bugs.

1. **Results**

The result of this homework is a user friendly application which does polynomials operations: addition, subtraction, multiplication, division, differentiation, integrations. The graphic interface of the program is a simple and clean one easy to use and to understand.

1. **Conclusions**

In conclusion this homework helped me to get more familiar to the OOP way of thinking. I learned new thinks and was remained about forgotten ways of how to work with java.awt and JFrame. Also this homework helped to see that I need to learn a lot more about how to work with java or about the OOP principles and how to write a good code.

For example some ways in which this application can be greatly improve are are:

* + Using ArrayList instead of vectors
  + Increasing the algorithms performance
  + Additional operation like plotting the functions graphic or finding its roots

1. Bibliography

- <http://stackoverflow.com/>

- the lectures from the course and laboratories