Polynomial Calculator

Programming Techniques – First Project

Group: 30424

First\_Name: Daiana

Second\_Name: Spatacean

Teacher: Claudia Pop

Summary:

1**. Introduction**

1.1 Problem specification . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

2. **Description of the project**

2. 1 Problem analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .4

2. 2 Modeling . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .4

3. **Projection**

3. 1 UML diagrams . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . 5

3. 1. 1 Use-case diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . .5

3. 1.2 Class diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . 6

4**. Implementation**

Packages . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

4. 1 Model package . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7

4. 2 GUI package . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

4.3 View package . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

5. **Implementation** **and testing** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

6. **Further developments .** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .16

7. **Conclusions** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

8. **References**  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

1. **Intoduction**

Using only Java for both the user interface and background programming, one was able to develop a standalone application. This field, the one of Polynomials is of great importance both in Mathematics and Computer science with a wide range of applicability. It is no simple matter dealing with Polynomial operations (excluding, of course, the basic operations of addition and subtraction) thus the need for a more comprehensive application grew greater as time went by.

Firstly, there are basic operations such as: addition, subtraction, multiplication and division (they use two polynomials – binary operations). The others, which were considered to be a bit more difficult, are done on a certain polynomial: differentiation, computing an indefinite integral or a definite one, evaluating the polynomial at a given value.

Considering the fact that a polynomial in a single indeterminate *x* can be written in the form:

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

there are many methods of solving the given problem, for example : using an ArrayList of **monomials** (each monomial being composed of two terms: coefficient and power of x) or using an **array of coefficients** .

**2.Description of the project**

The purpose of this assignment was to make some operations with polynomials. Some binary operations like: addition, subtraction, multiplication and division and unary operations need to be implemented. This project contains only addition and subtraction. We have to read two polynomials ( or one depends on the type of operation that we implement) and make the operations that the user requires. The application needs to have an interface that can be managed by any user and is not hard to understand where the input has to be write or which button he has to use for each operation.

**3.Diagrams**

**3.1 Use-case diagram**

A **use case diagram** at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different [use cases](https://en.wikipedia.org/wiki/Use_case) in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

The user can be any person that wants to use the application and the interface must be pretty friendly.

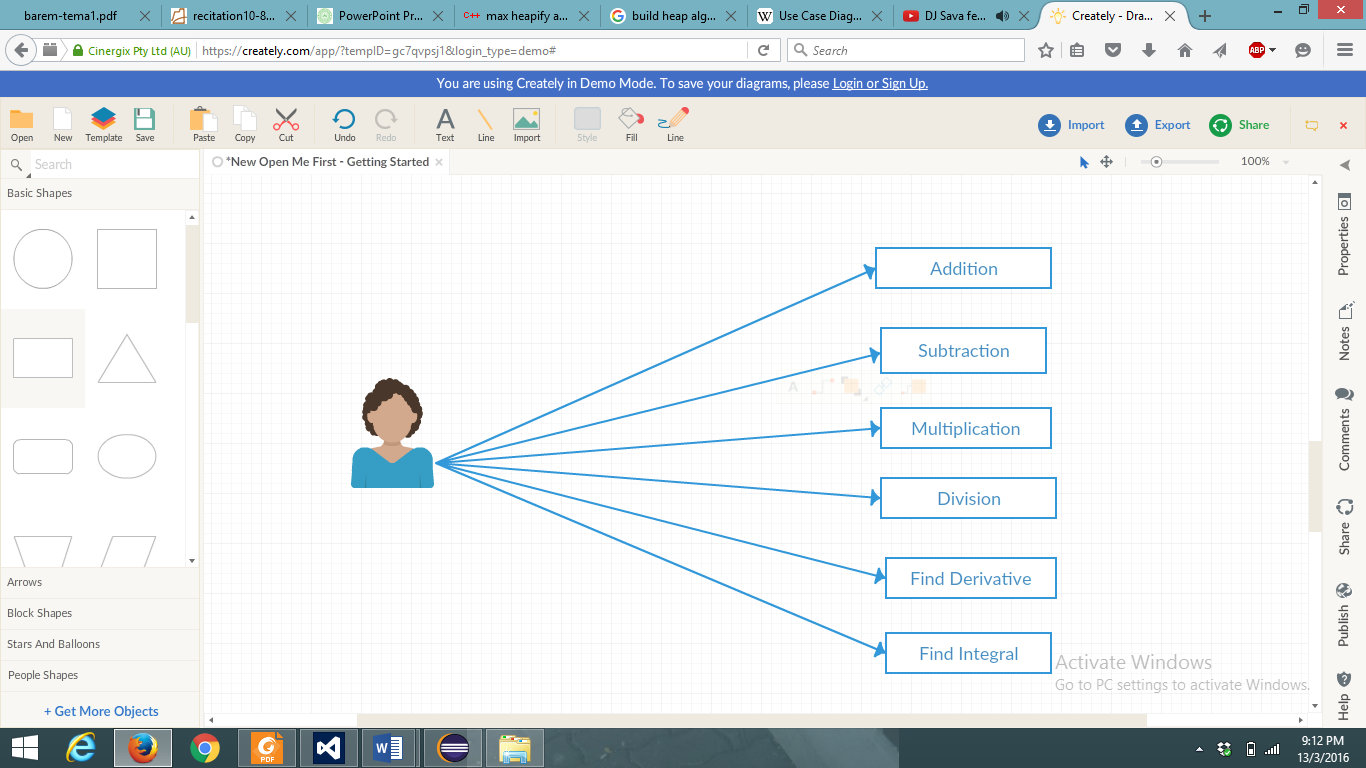
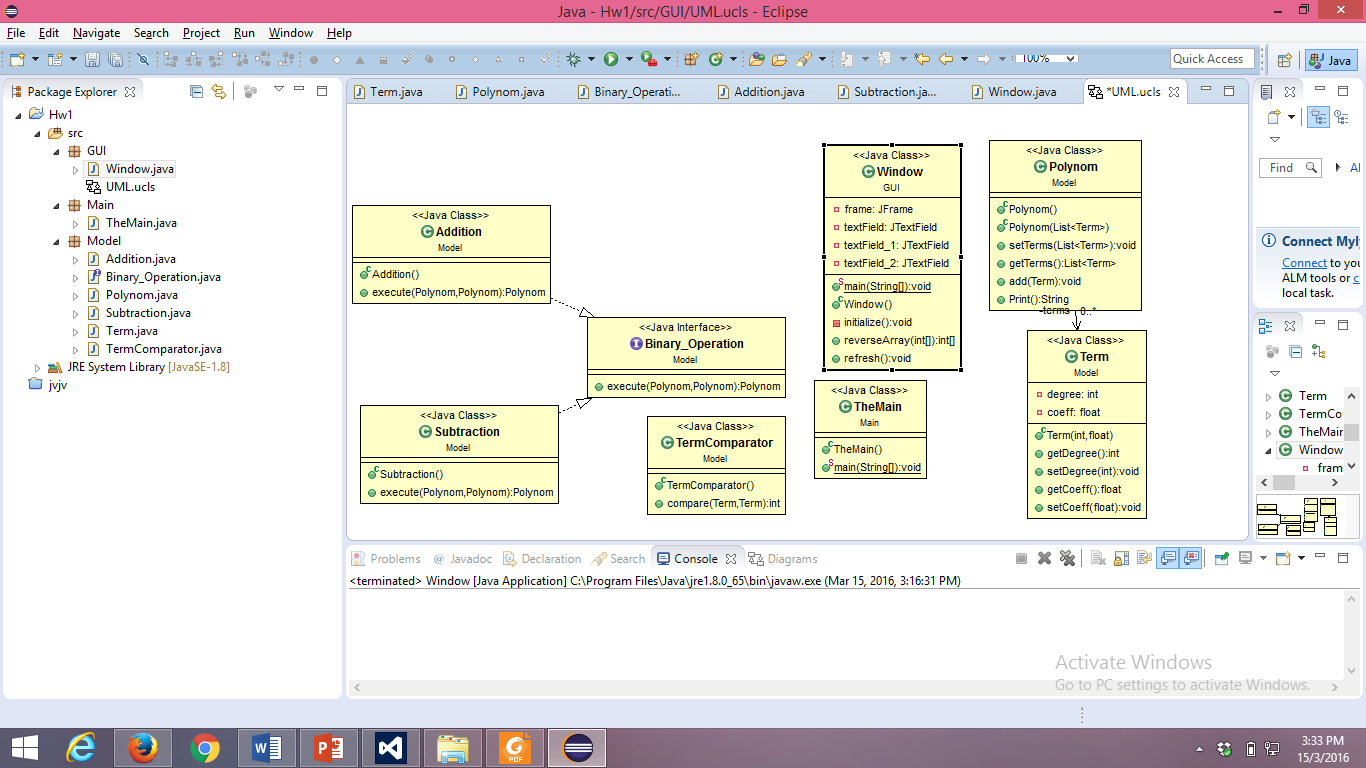


Fig. User- diagram

**3.2 Class Diagram**



**4.Implementation**

This project is made following the OOP structure(classes, methods) and principes:

- **inheritance** when one object acquires all the properties and behaviours of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism

- **polymorphism** when one task is performed by different ways i.e. known as polymorphism. For example: to convense the customer differently, to draw something e.g. shape or rectangle etc.

- **abstraction** hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing

- **encapsulation** binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.

The project has the following packages:

* + **Model**- it contains the classes for implementing the operations that will be performed on the polynomials and also the one who build the polynomials
  + **GUI-** is has only one class for the interface
  + **Main-** the main
* **Model Package**
* **Term class**
* **Polynomial class**
* **TermComparator**
* **Binary\_Operation**
* **Addition**
* **Subtraction**
* **Term class**

**public** **class** Term {

**private** **int** degree;

**private** **float** coeff;

* in this class I implement each term of a polynomial with its coefficient and its degree as private attributes
* also I made getter and setter for each attribute and the constructor of the class: Term (int degree, float coeff)
* so every time I call this class I will have the attributes for a monomial: coefficient\* X^degree
* **Polynom class**

**Public class Polynom**

**-** in this class I have a list of terms which represents a polynomial

**-** I sort the terms and import the Collention class because I need a sort method and use TermComparator for ascending order

**-**in this class is the methos add which will be used when I perform the addition of the polynomials in which is the coefficients of the polynomial are equal then the addition will be performed

**-** also I made a method for printing the polynomial( one is for asscenting order and one dor descending, depending if the polynomial is the input or the output ) :

**Public void print()**

**-** I take a sting and nme it “str” if the string is empty and when we make a for through the elements from the list and just get the coeff (if it positive then we don’t need “+” and if is negative we already have a “–“ )

- I add the terms in the String

- on the else branch we have the case when is not the first element and if the coeff is “ -“ don’t print “+” because the term will already have a minus and if the coefficient is positive then we have to prin “ +”

* **TermComparator class**

**public** **class** TermComparator **implements** java.util.Comparator<Term>

**-** this class hass only one method: compare in which we have a. getDegree- b. getDegree

**-** a comparator for Term objects to be used when sorting the List< Term> in class Polynomial

**-** use when we need to input the polynomials for the operations

* **Binary\_Operation class**

**public** **interface** Binary\_Operation {

// simple interface to be implemented by Addition and Subtraction classes

**public** Polynom execute(Polynom p1,Polynom p2);

}

* **Addition class**

public class Addition implements Binary\_ Operation{ }

- here is the method execute from the binary\_ operation

- here I have a polynomial r which will return the result of the addition and 2 list of terms(which are the 2 polynomials that will be used to compute te operation )

- get greatest degree and at this moment the list should be already sorted

- get term with degree i for each polynomial and if we found it we add them and form the resulting polynomial

* **Subtraction Class**

public class Subtraction implements Binary\_ Operation{ }

- this is very similar to addition and also implements the execute method from the class Binary\_Operation

- the difference is that when the coefficients from the terms which are in the second polynomial are greater or equal to 0 then we make 0 minus them and if not we multiply the coefficients with – 1

- and the if we found 2 terms with the same degree in both polynomials we “add” them

* **GUI**
* **Window**

**public** **class** Window

* this is the class in which the interface is made
* here are three text fields named: textField, textField\_ 1, textField\_ 2 and 3 buttons for: addition, subtraction and refresh such that the user can delete what has been compiled without opening other window
* first I make an object of type addition
* I input the polynomial like this: 1 2 3 1 and I make split by space and this will be equal to: 1X^2+3X^1
* I take a list of terms and I transform the string into inetegers and the I create a term( which will be inserted in the list of terms to compute the polynomial ) and the coefficient and degree will be the elements in the position i and i+1 and then I will do the same for the second polynomial
* And at the end I compute the addition and prit it in the textFiled that is made for that
* The same thing is done for subtraction( it is a lot of duplicate code I know)
* And id the input doesn’t correspond an error message will appear
* **Main**
* **The Main**
* This class contains only:

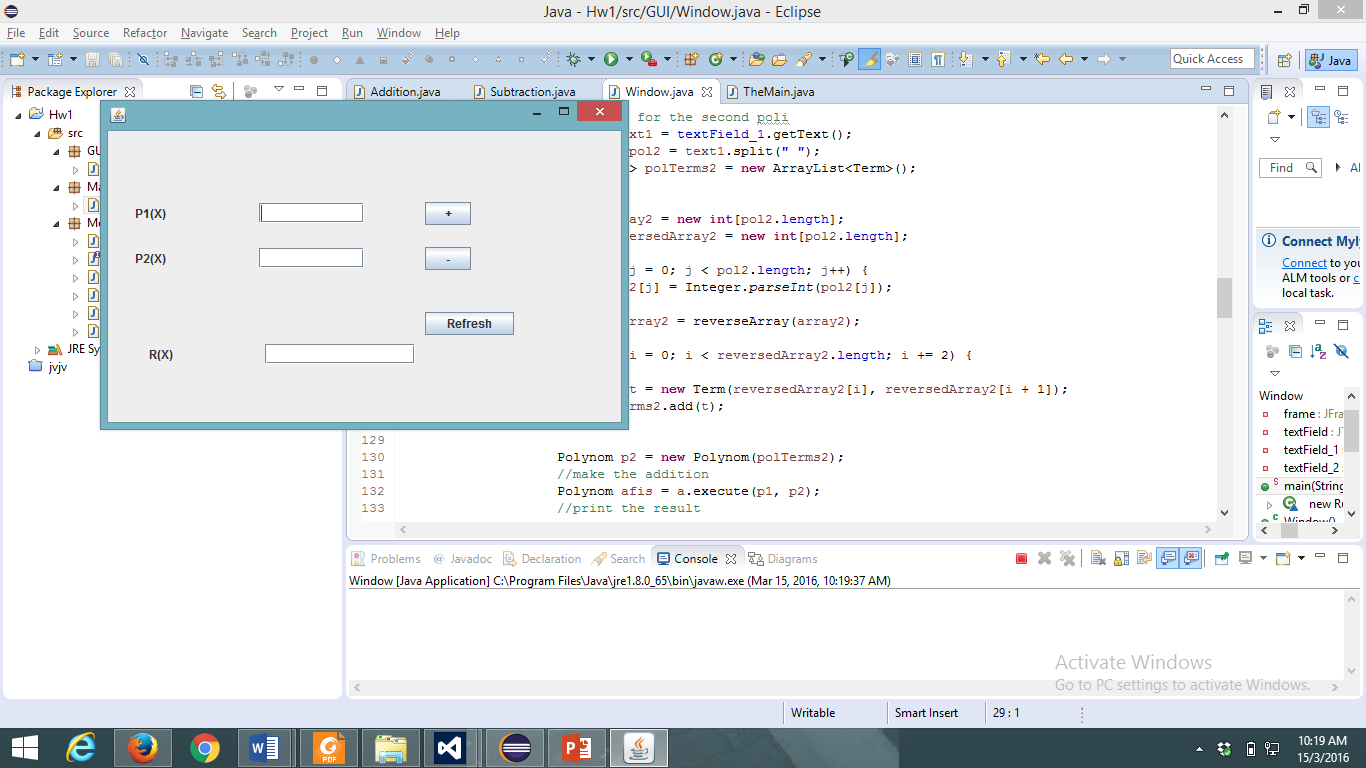
**public** **static** **void** main(String[] args) {

**new** Window();

}

5. **Implementation** **and testing**

This is how the interface looks like:



* The **user** should:
* Enter the coefficients and the degree of first polynomial separated by space
* Enter the coefficients and the degree of first polynomial separated by space
* Press the button with the desired operation
* the result will be displayed in the third field
* in order to perform a new operation, the refresh button should be used

Regarding the implementation process, I used as program Eclipse IDE. During the implementation of the project I made a lot of changes. Frst my program was into one package and then I split it. The classes had many methods. I also start the project not based on OOP principles: input the array of coefficients and degrees from the keyboard and then I figure out theat it was not the best ideea. Before the interface was made I print the results in the console to see if they are correct and to know if I continue in that manner or I had to make some changes.

5. **Results**

I am pretty satisfied with my work and I think that the project is very easy to understand and the most important thing is that any type of user is able to use the interface and will have no problem with it. And in the end I think that I implemented all the petitions of the “client”.

6. **Further developments**

I think that I can make a very long list with the things that I whant to improve related to my project with the polynomial calculator:

* starting from the “back” of the project: I like to make more operations like multiplication, division and so on
* I also want to implement the unary\_operations and why not other operations with polynomials like the root finding and so on
* More OOP with more classes and method with less lines of code an not so many duplicate code in some
* Also the addition and subtraction are very similar and this means duplicate code and this has to be changed
* I wan to have the structure for the packaages: Model( with all the operations, term class and so on ), GUI( with the interface ) and Controller( in which the model and the GUI are tied )
* And for the interface I want to make it much complicated and when the window appears it will have a message like: “Welcome” and than the window with the calculator will appear
* For the buttons if I will make them with colors and with another type of writing and for the toolbar I want to implement help and menu and so on
* It will be nice if the size of the window cand be modified from the user

7. **Conclusions**

To conclude, I can say that this project meant hard work, a lot of new things learned, focusing, development and creativity. Even if I encountered a lot of problems, I was able to fix them after all, by searching on the internet or asking a colleague for advice. I think that my application satisfies the requirements and the users will have at their disposal all its functionalities.

8. **References**

* <https://en.wikipedia.org/wiki/Polynomial>
* <http://stackoverflow.com>
* <http://www.oracle.com/technetwork/articles/java/index-137868.html>
* <http://www.tutorialspoint.com/java/java_data_structures.htm>

And others!

.