

**PROGRAMMING TECHNIQUES**

Homework 1

**Student:Nicolescu Alexandru**

**Group:** 30422

**Table of Contents**

**1. Objective 3**

**2. Problem analysis, modeling, scenarios, use cases 3**

**2.1. Problem Analysis 3**

**2.2. Modeling 4**

**2.3. Scenario and use cases 4**

**3. Design 5**

**3.1. UML Diagram 5**

**3.2. Data Structures 6**

**3.3. Class Projection 6**

**3.4. User Interface 8**

**4. Implementation and use of cases 9**

**5. Results 10**

**6. Conclusion and possible updates 10**

**7. References 10**

**Documentation polynomial processing system**

1. Objective

This homework’s aim is to use elements of object-oriented programming to implement an efficient polynomial processing system, covering all operations between two polynomials and those that apply to a single polynomial.

1. Problem analysis, scenarios, use cases

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

* 1. Problem analysis

If we take a closer look to the problem of polynomial processing we shall find that it is a more complex problem that it may seem at first glance.

First of all we need to find an OOP style way to store the data that comes from the graphic user interface. Even if the problem of storing polynomials may seem easy to solve using arrays of double, where the index is the power and the value of that array index is the coefficient, this is not OOP and if we analyze the problem a little bit further we find out some cases where it isn’t even efficient. For example if we have the polynomial x1000+1 we have to create an array of 1000 elements which is a very inefficient way of solving this problem. Therefore we created a Polynomial object which has an Arraylist of Terms which is another object. The Term object has 2 characteristic values which are easy to deduce: a coefficient and a degree. So, using this structure we can now store and use the Polynomials which are part of the operations.

Another problem concerning the polynomial processing would be the way the user gives us the data and the way we tell him the result of the processing. After doing some research on the internet and looking at online applications that provide these services that cover the polynomial processing problem I have reached the conclusion that the best way in terms of String processing and also creating user-friendly application is that the input should be like this: 5x^5+2x^3+2. This way the user will be able to use the facilities of this application fast enough, but the input-output problem will not take so much time to resolve.

Further on we shall analyze all the aspects which needed to be managed in order for the right functioning of this project.

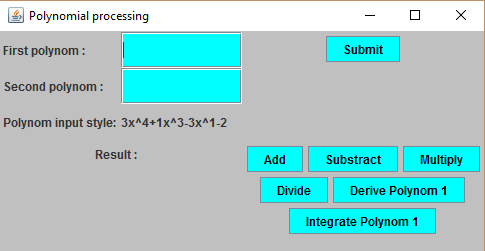
* 1. Modeling

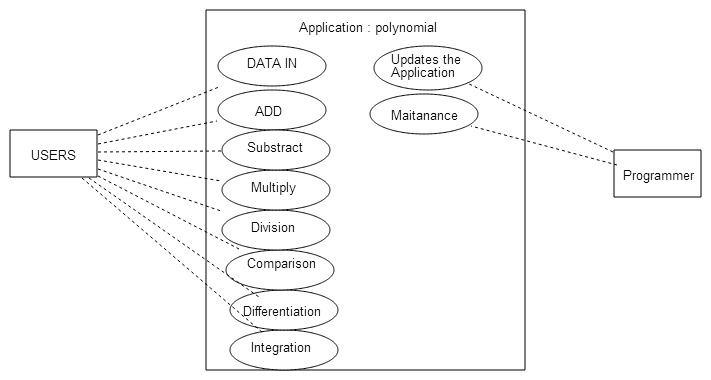
The implementation of this project is the same one as the one presented in the section 2.1. The „Operations” class holds all the methods for the operations with the polynomials, when the construction of the polynomials happens after pressing the Submit button on the user interface implemented in GUI class. I have tried to make all the classes clear and make them work together as good as possible in order for this application to run smoothly and be easy to modify in the case of development.

2.3 Scenarios and use cases

The scenarios were already mentioned, but I will present the details here. Firstly I thought about how the data will be managed by the application and my idea was: GUI (graphic user interface class) -> PolynomProcessing -> GUI (managing the operations depending on which buttons are pressed on the interface) -> -> Addition, Subtraction, Multiplication, Division, Integration, Derivation -> GUI (displaying the result on the user interface). I followed this plan and I did not have any unexpected surprise when it comes to the implementation and the structure of the program.

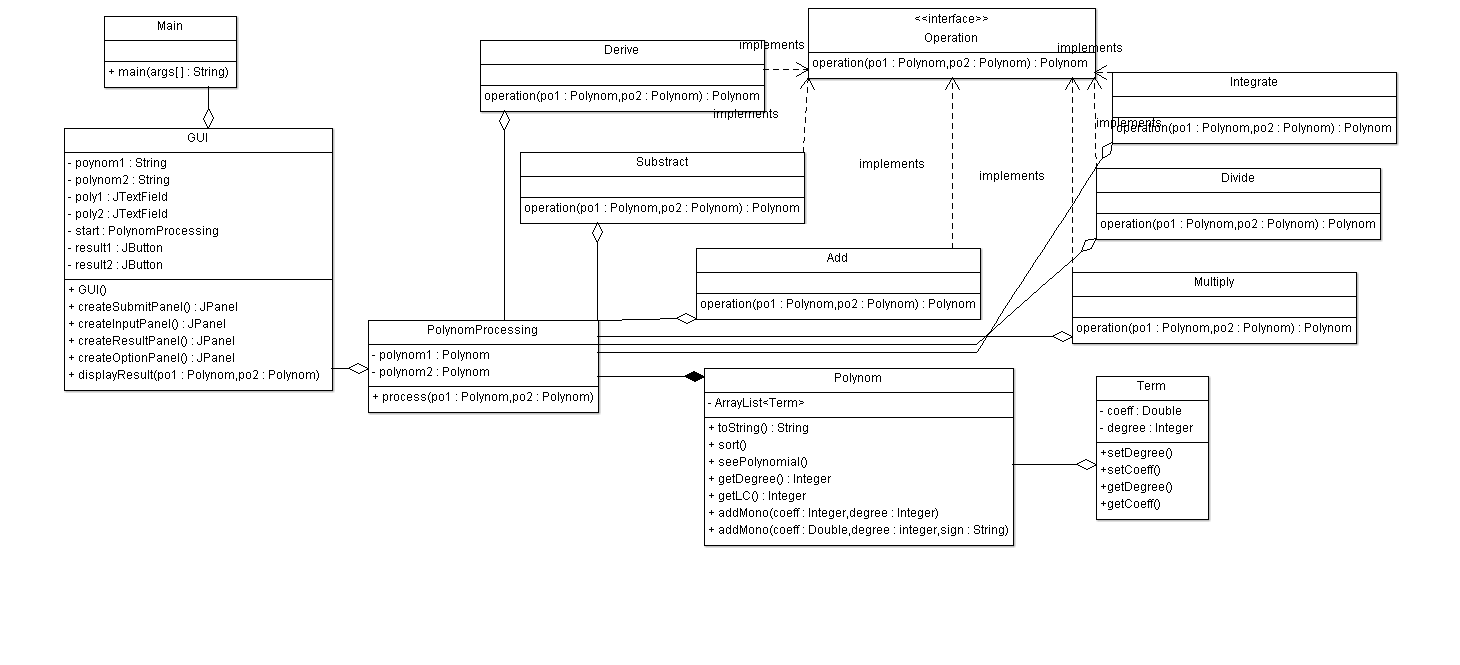
The use cases are strictly dependent on the user, and finally I order to make the application as user friendly as possible I decided to implement the following user interface:



First of all, the user introduces the first and second polynomial and when he presses Submit, the application processes those polynomials and transforms them into Polynomial objects. The next step is to choose what operation he wants to do with the recently entered polynomials. For example, if he pressed Add the result will appear instantaneously in the “Result:” section displayed as a String (the Polynomial class has a toString method overridden made in order for the application to be able to print whatever polynomial so that the user is able to understand it. I have presented the use case diagram below:

1. Design

3.1 UML Diagram

The UML diagram is a class diagram in which we can find the relationship between classes and also the elements that the specified class contains.

One could observe that for joining the classes I used several types of relationships. Between the class Term and the class Polynomial, I have used composition because I think that the Term cannot survive without the Polynomial. I have also used aggregation between the Operation classes and the PolynomProcessing due to the fact that the objects of the class PolynomProcessing contain references to objects of this other classes. Last but not least, I have implementation between the classes that compute the desired calculations and the Operation interface.

3.2 Data structures

The data structures used at this problem are either primitive data types such as integers or floats or more complex objects such as ArrayList type objects or new created objects such as Term, Polynomial or Operations. The object Term was introduced for representing a term in the sequence of terms that form a polynomial. This terms were added in a list of type ArrayList<Term>, Term being the base-type. Again the object Polynomial has been introduced in order to obtain a list with such terms that form together a polynomial; adding several such terms we obtain the desired polynomial.

3.3 Class projection

Class projection refers mainly to how the model was thought, how the problem was divided in sub-problems, each sub-problem representing more or less the introduction of a new class. First I will start by mentioning exactly how my problem was divided into packages and afterwards each package with its own classes. I begin by creating the two packages I used: the first one being called “gui” and the second one being called “logic” and the “main” package that only starts the execution. I named them intuitively because the first one handles the interface; the part that deals with the user and the second one handles the implementation, the part that is hidden from the user.

1. *gui* package

This package only contains the GUI class which creates the User Interface and moreover it is used for input-output functions. Using JTextField it takes the input from the user and sends it to processing am using two JButtons result1 and result2 it displays the results of the operations performed. The GUI class is also used to enable the user to choose what operations he wants to be done.

JPanel inputPanel = createInputPanel();

**this**.add(inputPanel);

JPanel buttonPanel = createSubmitPanel();

**this**.add(buttonPanel);

JPanel resultPanel = createResultPanel();

**this**.add(resultPanel);

JPanel optionPanel = createOptionPanel();

**this**.add(optionPanel);. . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . .. . . .. . . . . . . . . .

This is the main code of the GUI class which describes perfectly how everything is done. There are 4 methods used to create the main Frame, one for each Panel which is added to the window.

2) *logic* package

This package contains all the classes which do the calculation (addition, subtraction, multiplication, division, integration and derivation).

It contains one class for each operation and all these classes implement the Operation interface which obliges them to have a method which receives as parameters two polynomials and return one polynomial which is the result of that operation.

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

This implementation varies from operation to operation. A good example of how efficient is the usage of the operations is the Multiplication and Division. We shall analyze their code:

**public** Polynom operation(Polynom p1, Polynom p2) {

Add addition = **new** Add();

Substract substraction=**new** Substract();

Multiply multiplication=**new** Multiply();

Polynom q = **new** Polynom();

Polynom r = **new** Polynom();

r = p1;

**int** d = p2.getDegree();

**double** c = p2.getLC();

**while** (r.getDegree() >= d) {

Polynom s = **new** Polynom();

s.addMono(r.getLC() / c, r.getDegree() - d);

q = addition.operation(q, s);

r=substraction.operation(r, multiplication.operation(s, p2));

}

GUI.*displayResult*(q,r);

**return** r;

}

In this problem of division of two polynomials I have used the Euclidean Division of Polynomials algorithm and therefore I had to use the addition, subtraction and multiplication that were already implemented.

**public** Polynom operation(Polynom p1, Polynom p2) {

**int** length = p1.getMonomials().size();

Polynom result[] = **new** Polynom[length];

**for** (**int** i = 0; i < length; i++)

result[i] = **new** Polynom();

ArrayList<Term> monomials1 = **new** ArrayList<Term>();

ArrayList<Term> monomials2 = **new** ArrayList<Term>();

monomials1 = p1.getMonomials();

monomials2 = p2.getMonomials();

**int** i = 0;

**for** (Term t1 : monomials1) {

**for** (Term t2 : monomials2) {

result[i].addMono(t1.getCoeff() \* t2.getCoeff(), t1.getDegree() + t2.getDegree());

}

i++;

}

Add addition= **new** Add();

**while** (length > 1) {

Polynom aux = **new** Polynom();

aux = result[0];

result[0]=addition.operation(aux,result[length-1]);

length--;

}

GUI.*displayResult*(result[0],**null**);

**return** result[0];

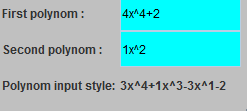
}

As another example, in the problem of Multiplication I needed to use the addition so that there will be no duplicate Term (Term which have the same degree and are not added). As you can see, this resulted in a perfect combination between these operations and I managed to define each operation using normal algebraic operations between coefficient and power or even using other polynomial operations.

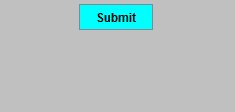
A class which I did not talk about is the PolynomialProcessing which is the class that links the user input (3x^5) to the application’s logic which is Polynomial.Monomials.add(new Term(3,5);. As you can see there is a huge difference between the code that is actually executed and the input which is as user friendly as I could make it. I managed to do so using Java regex, which managed to decompose the String which was inputted by the user into small parts which were taken as degrees, coefficients or were ignored (x^).

3.4 The interface

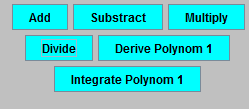
The interface is made out of 4 JPanels which are included in the JFrame in a GridLayout(2,2). We shall describe each JPanel in order to understand how the user interface is created.



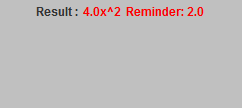
In this first JPanel the user inputs the data (polynomials) according to the example in Polynom input style.



After clicking the Submit button from JPanel2 the user will choose what operation to do from JPanel4:



Therefore the result will be printed in the 3rd JPanel which looks like this:



And this is how the user interface is made out of 4 JPanels which work together to make a user friendly in order for the application to work properly.

1. Implementation

In what the implementation is concerned this project was developed in Eclipse and it was only tested in this environment. However the program should maintain its portability. Concerning the code implementation I did not make use of laborious algorithms, but I have rather stayed faithful to the classical algorithms of computing polynomials learned in high school. However I have tried to implement my problem in a way that appears to me as being the most efficient one, this is why I have changed my model at first. On the subject of this I have decided to use ArrayList instead of arrays because I believe that they are more efficient related to performance and memory management. Testing implies checking for any errors in the program or limitations of this program. Due to the fact that the program is rather simplistic, they are few errors that might generate this program to work wrong or to stop. These errors are mostly related to the interface. I have assumed that the user reads the instructions from the interface and respects them, otherwise if he enters data with invalid format the program will probably generate some bugs and will stop. Hence this part with checking all the possible scenarios will be seen as future development.

1. Results

The application is an user friendly and useful application to perform basic polynomial

operations such as: addition, subtraction, multiplication, division, differentiation and

integration. As the application is developed on a Java platform, it is highly portable and allows it to run on several operating systems (as long as they have the Java SDK installed). The application is straightforward an easy to understand and to use by any user who respects the instructions given in the interface and who has some basic knowledge of polynomials operations, of course. Even though being limited, this application can be considered as being a helpful tool that can be used when dealing with such polynomial operations.

1. Conclusions

All in all, the application works perfectly on the required operations and it is a user friendly interface which gives the users a simple and efficient answer to any problems related to polynomials. As in possible updates I would modify the input and output as I would make it even more easier to add a polynomial.

1. References
2. <https://en.wikipedia.org/wiki/Polynomial_greatest_common_divisor#Euclidean_division>
3. http://stackoverflow.com/questions/28859919/java-regex-separate-degree-coeff-of-polynomial