Programming Techniques

Assignment 1

Student : Rujita Alexandra

Group : 30422

Contents

[1. Problem specification . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3](#_Toc445751246)

[2. Problem analysis , modeling, scenarios , use cases . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3](#_Toc445751247)

[2.1. Problem analysis 3](#_Toc445751248)

[2.2. Modeling 3](#_Toc445751249)

[2.3. Scenario and use cases 4](#_Toc445751250)

[3. Design . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5](#_Toc445751251)

[3.1. UML Diagram 5](#_Toc445751252)

[3.1.1 Monom Class 6](#_Toc445751253)

[3.1.2. Polynomial class 6](#_Toc445751254)

[3.1.3 GUI Class 6](#_Toc445751255)

[3.1.4 orderByDegree Class 6](#_Toc445751256)

[3.1.5 OnePolynomOp and TwoPolynomOp Interface 6](#_Toc445751257)

[3.1.6 OpWith1 Class 6](#_Toc445751258)

[3.1.7 OpWith2 Class 6](#_Toc445751259)

[3.1.8. MainC Class 7](#_Toc445751260)

[3.2. Data Structures 7](#_Toc445751261)

[3.3. Relationships 7](#_Toc445751262)

[3.4. Algorithms 7](#_Toc445751263)

[3.5. User Interface 9](#_Toc445751264)

[4. Implementation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10](#_Toc445751265)

[5. Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12](#_Toc445751266)

[6. Conclusions and further developments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13](#_Toc445751267)

[7. References . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13](#_Toc445751268)

1. Problem specification

**Assignment 1 -** Propose, design and implement a system for polynomial processing. Consider the polynomials of one variable and integer coefficients.

The program will be able to perform the 6 operations ( with one or two polynomials ), that are described next:

* Addition of two polynomials
* Subtraction of two polynomials
* Multiplication of two polynomials
* Division of two polynomials
* Derivation of a polynomial
* Integration of a polynomial

It will have a graphical user interface through which the user will be able to insert the polynomials and then select the desired operation for one of the polynomials or for both of them. There will be different buttons for every operation the user wants to perform, even for choosing which polynomial he wants to differentiate or integrate , and the result will be displayed in a textbox near the message „Result: ”.

1. Problem analysis, modeling, scenarios, use cases

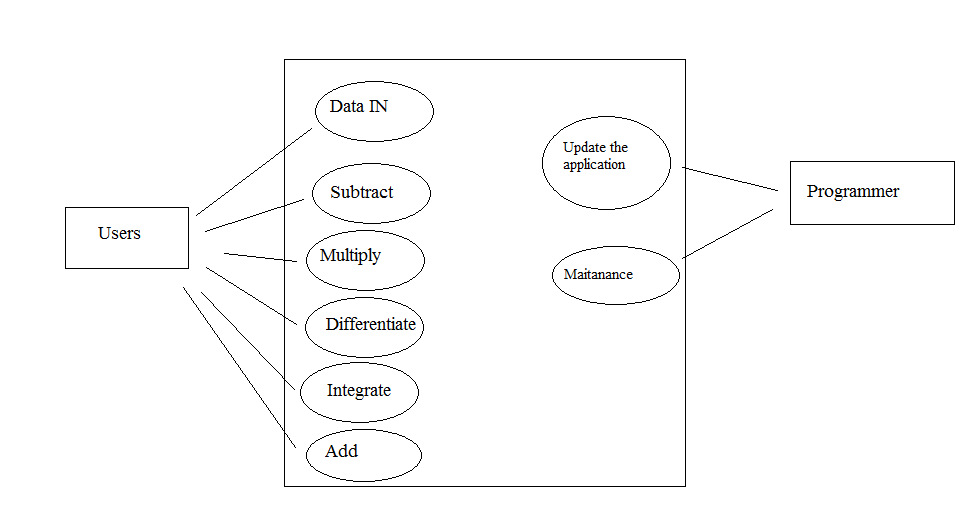
## Problem analysis

A polynomial can be modeled inside an application in several ways. The easiest way to store polynomials and work with them is storing them in 2 arrays, one for coefficients and one for degrees. Then the array will have the length of the maximum degree and where the power is missing the coefficient will be zero. Another way to model them is to store them as arrays of terms ( “monom” ) and each term will contain its’ coefficient and degree , which is a more “ real life” approach.

## Modeling

The implementation of this project is the second one presented at point 2.1. Each polynomial is broken down into terms , the terms are the Monom Class objects, and is ordered from the highest degree to the lowest into an List of Monom objects.

## Scenario and use cases



The user will enter the desired polynomials as shown in the example above the text fields. To be able to apply different operations on them he must press the “Submit” button, he can add new terms to the current polynomial but he must always press the “Submit” button in order to make the changes visible to the program.

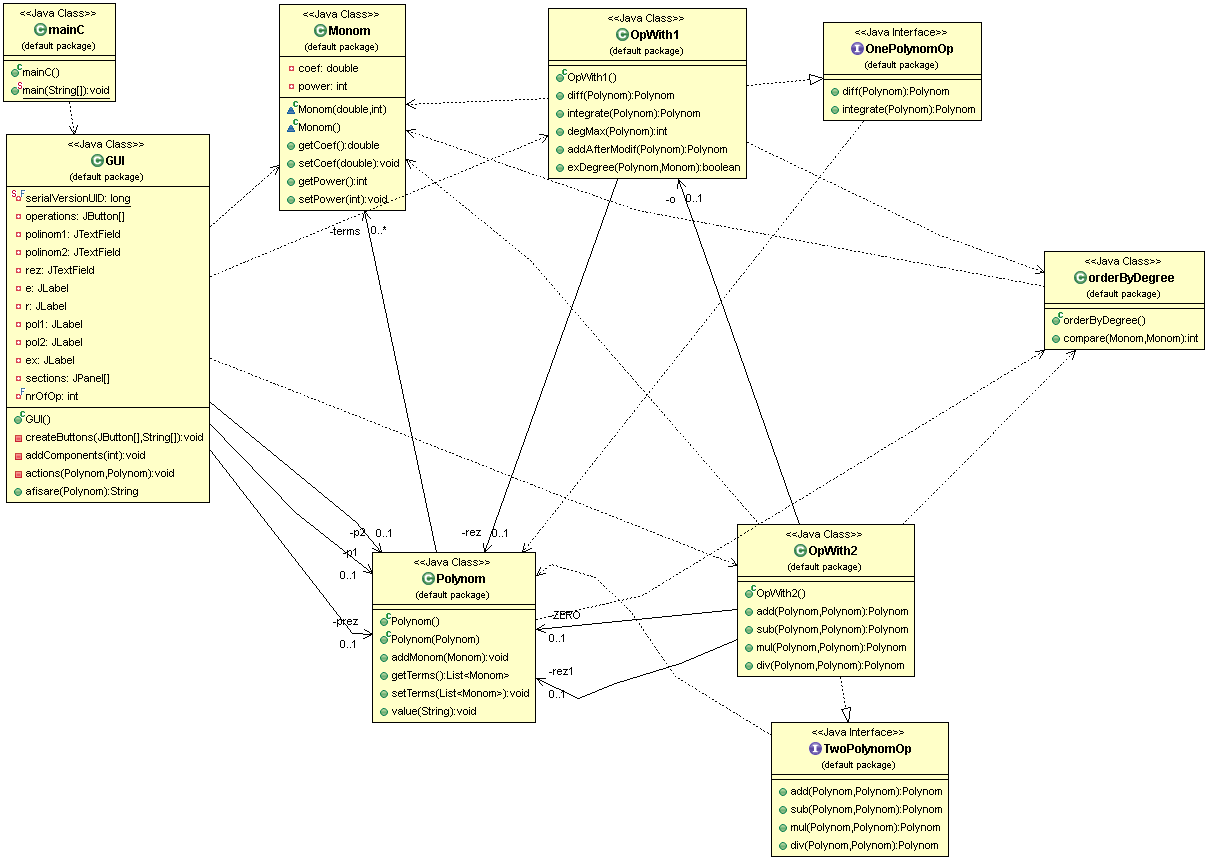
For addition, subtraction, multiplication and division the user has to input two polynomials, but is not necessary. If he does not introduce one it will be considered 0. For integration and differentiation he can enter either one of them, or both, but he will have to choose on which one to perform the operation.

In order to treat the problem imposed by division and integration, that the coefficients might be real the numbers, the result will also have as coefficients real numbers, displayed as numeric values, even if they are integer numbers. This happens because it is the only solution to treat all cases.

1. Design

## UML Diagram

The UML diagram is a A class diagram that shows a set of classes, interfaces and collaborations and their relationships. Class diagrams may also contain packages or sub-systems, both of which are used to group elements of the model.



### 3.1.1 Monom Class

The Monom Class holds only the getter and setter method for both the coefficient and the degree. This class objects represent the terms of the polynomial.

### 3.1.2. Polynomial class

This class transforms polynomial transmitted as a string to the actual polynomial on which we can perform operations. It has an ArrayList of Monom objects which togheter form the polynomial. The getTerms method is used to itterate through the terms of the ArrayList.

### 3.1.3 GUI Class

This class is designed to create a graphical user interface to simplify the use of the application, the interaction of the user with the actual code . This class is a subclass of the predefined class „JFrame” for the purpose of using objects like : buttons, frames, panels and text fields.

The panel next to the “Example” shows the user how the appropriate format of the introduced polynomials should be, such that he would not get an error. The next two text fields are where the user has to write the desired input and the third one is where the result of the selected operation will be shown.

### 3.1.4 orderByDegree Class

This class implements the Comparator<T> interface, where in this case T is Monom.

### 3.1.5 OnePolynomOp and TwoPolynomOp Interface

The use of this interfaces is to clearly separate the two types of operations that can be performed on the polynomials.

### 3.1.6 OpWith1 Class

In this class, which extends the OnePolynomOp interface , the operations that are performed on only one polynomial are hold:

* Differentiation of the introduced polynomial : returns the resulted polynomial after differentiating the sent one, one time.
* Integration of the introduced polynomial: returns the resulted polynomial after integrating the polynomial sent as a parameter.

This class also holds three more methods which facilitates the work of the programmer.

### 3.1.7 OpWith2 Class

In this class, which extends the TwoPolynomOp interface , the operations that need two polynomials to give the result are hold:

* Adding two polynomials : returning the resulting polynomial after adding the polynomials passed as parameters;
* Subtraction of two polynomials: returning the resulting polynomial after subtracting the polynomials passed as parameter from the object who called the method; if the resulting polynomial is equal to 0 then 0 will be shown to the user
* Multiplication of two polynomials: returns the resulting polynomial after multiplying the two ones sent as parameters
* Dividing two polynomials: returns the resulting polynomial after dividing the two ones sent as parameters; if the second one is 0 then an exception will be thrown; if the first one has the degree less than the second one the zero polynomial will be the result.

### 3.1.8. MainC Class

It is the main class that creates a GUI object, creating also the interface and enabling all the features of the program.

## Data Structures

The fundamental structure is the Monom ( the coefficient, the degree ) which are hold together in an ArrayList, making it more efficient to hold data, because its’ size is equal to the number of the terms in the original polynomial.

## Relationships

There is a strong depency reelationship between the Monom and Polynom Class, if the Monom class is deleted the polynom class can no longer exist because its’ fundamental object does not exist anymore.

Also there is a relaionship of association between the GUI class and the Polynom, OpWith1 and OpWith2 class because every when a button is pressed one of the operation is executed, operation to which the program has access through the OpWith1 or OpWith2 object that is created. The result is stored in an object of the class Polynom.

## Algorithms

The addition of two polynomials is done in two steps: first we add to the polynomial that will hold the result, all the terns that have powers that do not appear in the second one, and vice versa; after that we go through the polynomials at the same time and add the coefficients that have the same power. After the addition of all terms is completed we sort the ArrayList according to the power ( from the highest to lowest degree ) .

The subtraction is done in the same way as the addition, but in a separate method. If the result is equal to 0, the zero polynomial will be transmitted as a result.

The multiplication of two polyomials is done by going through both polynomials at the same time (using two nested for loops) and multiplying the coefficients and adding the powers. After the computation is done we check if terms with the same power occur in the result and if so, we add the coefficients and write the power only once. This verification is done in a separate method which also returns a polynomial.

The division of two polynomials is done using the polynomial long divison algorithm. I will use the notation polynomial 1 to refer to the polynomial divided, and polynomial 2 to refer to the polynomial I divide with. First we create a Monom object that has as coefficient the result of dividing the coefficient of the first term in polynomial 1 to the coefficient of the first term in the polynomial 2, and the power is the difference between the maximum degree of each polynomial. Afterwards we add the monom to the resulting polynomial and also in an auxiliary one. The auxiliary one is multiplied with polynomial 2 and the result is substracted from polynomial 1. Then the process is repeted recursevly until the degree of polynomial 1 is smaller then the degree of polynomial 2.

Also if the user tries to divide by 0 or by a polynomial with a higher degree than polynomial 1 an error message will be displayed.

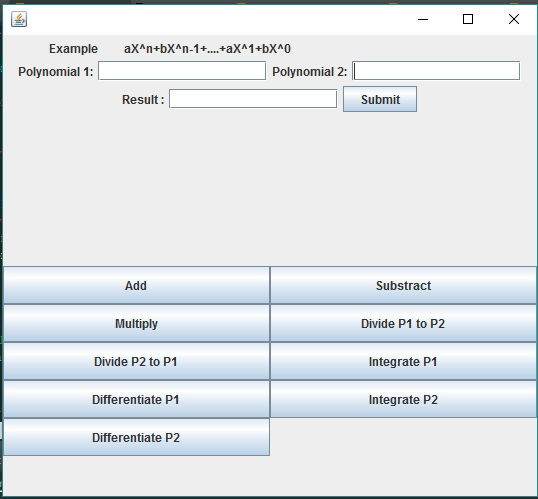
The integrate function of one polynomial is done going through the ArrayList terms and incrementing the degree with one, but before doing this dividing the coefficient with the actual degree, the result may be a real number, therefore the coefficients are stored in Double format.

The derivation of one of the polynomials is done in a similiar way to the integration, the difference being only the way the coefficients are obtained . If one power becomes 0 it will not be shown to the user.

The polynomial that results after doing any of the possible operations is then modified to look like the polynomials used in real life. For example: not showing the coefficient if it is equal to one or if the power is equal to zero then only the coefficient is printed, also when the power is equal to one it is not printed ( the user sees coefX) . Furthermore if the power is a negative number then it will pe printed between parathesis. If the coefficient is a positive number the ‚+’ sign is added.

Before printing the result if the first coefficient is greater than zero the ‚+’ sign will be deleted. After all the requierd fomatting is done the string that represents the result will be printed in the text field corresponding to the output. For formatting the polynomial object into a string the StringBuilder class is used, because is more effiecient in memory storage and it is easier to work on it.

## User Interface



The label at the top of the interface shows the user how the proper format of the introduced polynomial shoul look in order not to have any errors. If he leaves any of the polynomial text fields empty, the program will consider it as being the ero polynomial. Then in the next 2 text fields he has to introduce the desired polynomials, respecting the format, and then press submit. If he wants to modify the content he can, be he has to press the „Submit” button again.

Then when pressing any of the buttons to perform operations the result will be displaied in the third text field, next to : „Result”. He can do as many operations as he wants on the introduced polynomials but only the current operation result will be displayed and the old ones are not stored in history. There will be nothing displayed if the user operates on a non existing polynomial or „0” when is the case.

1. Implementation

In order to get the polynomial the user entered we store it in a string:

operations[0].addActionListener ( **new** ActionListener() {

***@Override***

**public** **void** **actionPerformed**( **ActionEvent** e) {

**String** **s1** = polinom1.getText();

**String** **s2** = polinom2.getText();

**if** (s1.equals(""))

p1.value("0x^0");

**else**

p1.value(s1);

**if** (s2.equals(""))

p2.value("0x^0");

**else**

p2.value(s2);

actions(p1, p2);

}

}

All the buttons are stored in an array of JButton making it easier to work with them. The “value” method is declared in the Polynomial class and does the splitting of the string, according to the next format:

**String** **degree**[] = polinomText.split("(-|\\+)?\\d+[xX]\\^?");

**String** **coef**[] = polinomText.split("[xX]\\^(-?\\d+\\b)"; }

});

The “actions” method “activates” the actionListeners for the buttons containing the operations. Therefore if the user does not press the “Submit” button he can not do any operation.

For dealing more easily with the operations I chose to create auxiliary methods to help me and also to make the code more logic and well structured. All these methods work only on one polynomial so they are created in the OpWith1 class.

**public** **int** **degMax**( **Polynom** polinom ) {

**Collections**.*sort*( polinom.getTerms() , **new** orderByDegree());

**if**  ( polinom.getTerms().size() > 0 )

**return** polinom.getTerms().get(0).getPower();

**return** 0;

}

Because the polynomials are always ordered from the highest to the lowest degree the maximum power will be on the first position of the polynomial, or index 0 of the ArrayList .

**public** **boolean** **exDegree** ( **Polynom** p1, **Monom** m ) {

**for** (**Monom** **m1** : p1.getTerms()) {

**if** (m.getPower() == m1.getPower())

**return** **true**;

}

**return** **false**;

}

This method returns the result of the question “ Is the monom M in my polynomial p1 ?” if yes return true, otherwise return false. This method facilitates the addition and subtraction because it tells me to add the current monom into the result because I will not have any other term with this degree in the second polynomial, so if I do not add it know it will be “forgotten”.

**public** **Polynom** **addAfterModif**(**Polynom** p1) {

**Polynom** **rez** = **new** Polynom();

p1.addMonom(**new** Monom(0, 0));

**Collections**.*sort*(p1.getTerms(), **new** orderByDegree());

**Monom** **m3**;

**int** **j** = p1.getTerms().size();

**int** **k** = 0;

**double** **c** = 0;

**boolean** **nuZero** = **false**;

**for** (**Monom** **m** : p1.getTerms()) {

**if** (m.getCoef() != 0)

nuZero = **true**;

}

**if** (!nuZero) {

**return** rez;

} **else** {

**while** (k < j - 1) {

c = 0;

m3 = **new** Monom();

**if** (p1.getTerms().get(k).getCoef() == 0)

k++;

**else** {

**if**  (p1.getTerms().get(k).getPower() == p1.getTerms().get( k + 1).getPower())

**while** ((k < j - 1)

&& (p1.getTerms().get(k).getPower() == p1.getTerms().get(k + 1).getPower())) {

c += p1.getTerms().get(k).getCoef() + p1.getTerms().get( k + 1).getCoef();

k++ ;

}

**else**

c = p1.getTerms().get(k).getCoef();

m3.setCoef(c);

m3.setPower(p1.getTerms().get(k).getPower());

rez.addMonom(m3);

k++ ;

}

}

}

**return** rez ;

}

This method is used to remove duplicate powers, problem which can occur after multiplication: more terms with the same power result but their coefficients are not summed.

Also this method resolves the problem when a coefficient becomes zero after some computation, it will remove the monom which has as coefficient the value zero from the ArrayList that is the polynomial. This case creates problem not in printing but when calling the degMax() method which will return the power of the first term even if its’ coefficient is zero.

The Polynom Class has two constructors:

**public** **Polynom**(**Polynom** p) {

terms = **new** ArrayList<Monom>();

**for** (**Monom** **m** : p.getTerms()) {

addMonom(m) ;

}

**Collections**.*sort*( terms , **new** orderByDegree() ) ;

}

**public** **Polynom** () {

terms = **new** ArrayList<Monom>();

}

The first one is used to create a clone of an already existing polynomial and the second one is used for the simple creation of an object of type Polynom.

1. Results

Given as inputs the following two polynomials:

Polynomial 1: 3x^3 – 5x^2 + 10x^1 – 3x^0

Polynomial 2: 3x^1+1x^0

We get the following results:

* Addition : 3.0x^3-5.0x^2+13.0x -2.0
* Subtraction : 3.0x^3-5.0x^2+10.0x -3.0
* Multiplication : 9.0x^4-12.0x^3+25.0x^2+x -3.0
* Dividing P1/P2 : x^2 - 2.0x + 4.0
* Dividing P2/P1 : 0
* Integrate P1 :
* Integrate P2 : 3.0x^2+x
* Differentiate P1 :
* Differentiate P2 : 3.0

1. Conclusions and further developments

After working on this project I found out the importance of interfaces and inheritance. Interfaces make your project more organized because you know what you have to implement and where, as for the inheritance it helps you develop more ways to work with known methods on your project. Also I have learnt how to better use ArrayList Class and how to create new object classes and not always rely only on the already implemented classes.

As for further developments:

* An easier way for the user to input the data ; a not so constrained format
* A text file where to hold all the previous results
* A more entertaining interface
* More operations available
* Make visible only the buttons with operations that can be performed on the current input data
* Make it easier for the user to add terms to its’ current polynomial

1. References

* <http://www.purplemath.com/modules/polydiv2.htm>
* http://stackoverflow.com/questions/28859919/java-regex-separate-degree-coeff-of-polynomial