

# Programming Techniques

## Homework 1

### Operations with Polynomials

Student:

Gombos Kriszta

Group 30424

# **CONTENT**

1. Objectives
2. Analysis of the problem, modeling, scenarios, use cases
3. Design, UML diagram, classes used, interface, relations, algorithms, interface for the user
4. Implementing and testing
5. Results
6. Conclusions, what have I learned from this homework, further developments
7. Bibliography

# 1. The Objective

Requested:

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

The objective of this homework is to implement in Java programming language a program with Graphical User Interface for operations applied on polynomials. The mandatory operations are: addition, subtraction, multiplication, division, derivation and integration. We were requested to use integer coefficient polynomials but after some operations, namely integration and division we may get also real coefficients. This fact must be taken into account.

In order to make the work of the user much easier, I implemented a friendly Graphical User Interface.

## 2. Analysis of the problem, modeling, scenarios, use cases

The problem requests to implement operations on polynomials, operations which are used in mathematics and we can apply that knowledge to implement this program. A polynomial is composed of several monomials (or one) monomials. The structure of a monomial is as follows: it has a coefficient which in our case can be either integer value or real value and an exponent which we will consider to be integer. The general form of a polynomial is:

$$P(x) = a[n] * x^n + a[n-1] * x^{(n-1)} + \dots + a[1] * x + a[0], \quad \text{where } a[i] \text{ is the coefficient of the } i \text{ monomial and } n \text{ is the power of the monomial } i.$$

One knows that the operations on polynomials have some properties, that is: the sum of two polynomials is a polynomial, the difference of two polynomials is a polynomial, the multiplication of two polynomials is a polynomial, the division of two polynomials is a

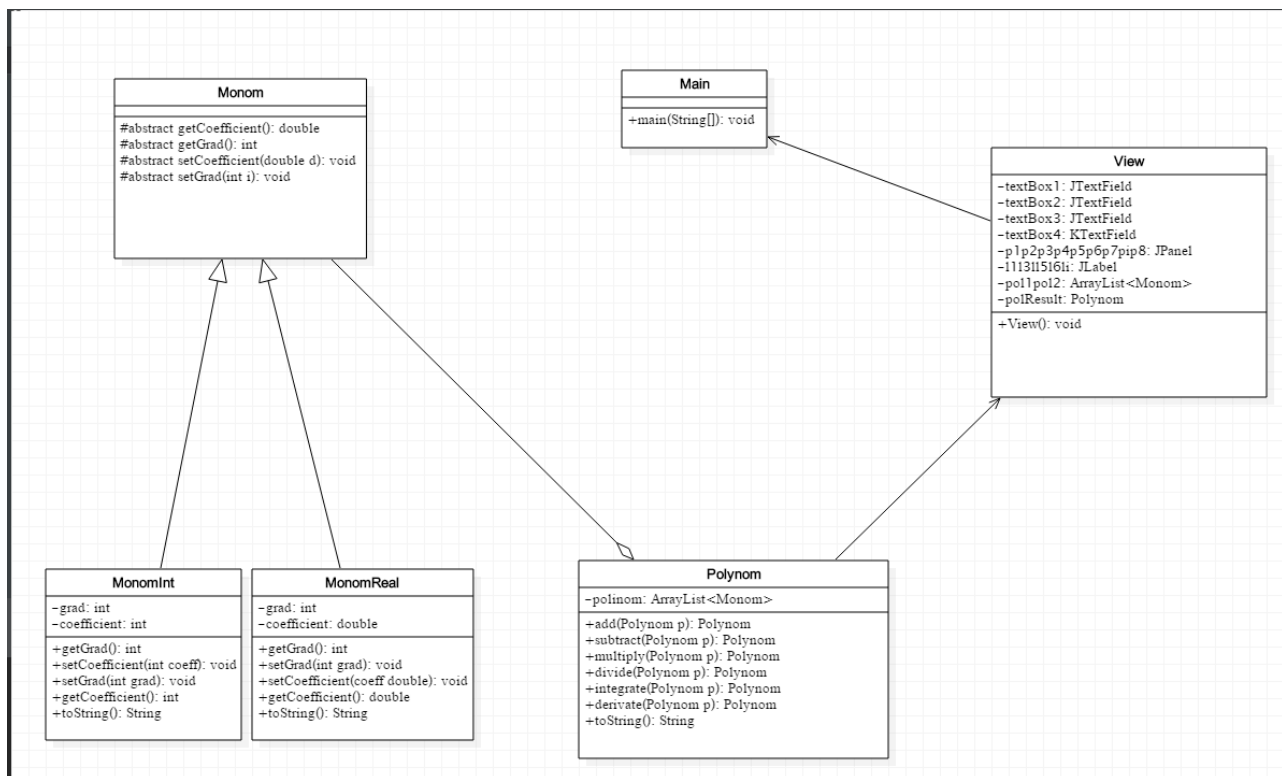
polynomial, the derivative of a polynomial is a polynomial, the polynomial integrated gives a polynomial.

Analyzing the problem one can think of more than one solution. First of all the representation of the polynomial can be done by using a vector holding the coefficients of the polynomial and the index to be used as the power of  $x$  related to the coefficient it is holding. There is also another way of representing a polynomial and that is by using an ArrayList of monomials. In the first case we do not use object oriented programming concepts as we would write functions that take two polynomials and return one. As Java is an OOP language I think is better to use methods that could be called later like `p1.add(p2)`. In the same time the reading of the data would be harder using the first option.

To implement the requested operations on polynomials we need to identify first the classes that will be used.

### 3.Design

I chose to divide my program into seven classes, for each subtask I one class to be easier to follow. The classes `MonomInt` and `MonomReal` inherit from the abstract class `Monom`, and the class `Polynom` has an ArrayList of `Monoms` (as a polynomial is composed of one or several monomials). Beside these classes I have a `Main` where an object of the class `View` is instantiated; the class `View` implements all what is needed for the Graphical User Interface. There is also a class called `JUnitTest` where I tested every operation that I implemented on polynomials. The diagram can be observed below:



The relationships between the classes are the following:

- between `Main` and `View` is an association because in the `Main` there is instantiated an object of type `View`

- between `Polynom` and `Monom` there is a relation of aggregation as a polynomial is composed of one or several monomials

- as a monomial can have integer or real coefficients between `Monom` and `MonomInt` and between `Monom` and `MonomReal` there is a relation of inheritance as `MonomInt` and `MonomReal` inherit from the abstract class `Monom`

Below I will present every class in particular:

## 1. Monom Class

It is an abstract class that has four methods which are not implemented. It is very important as it acts like an interface between the class `Polynom` and the other two classes, namely `MonomInt` and `MonomReal`. The abstract methods will be implemented in the classes that inherit from the `Monom` class.

## 2. MonomInt Class

This class has two attributes, namely an integer coefficient and an integer grad. I defined a constructor to set their values that looks like this: `public MonomInt(int coeff, int grad)`. I also implemented getters and setters for the coefficient and the grad as these two attributes are private, only seen in the interior of this class. There is another very important method named `toString()`, that constructs a string representation of the integer coefficient monom. For example the monomial with coefficient = 5 and grad = 3 will be represented like “5x^3”.

## 3. MonomReal Class

Has the same instance variables as `MonomInt`, the difference is that the coefficient this time is of type double. Use `MonomReal` objects when we have to integrate or divide two polynomials, as there might happen that the coefficient becomes a double value. As `MonomInt`, `MonomReal` has also a `toString()` method that generates a string representation of the monomial. Beside this method it has a constructor, getters and setters for the coefficient and the grad.

## 4. Polynom Class

It has an instance variable named `polinom` which is an `ArrayList` of monomials. This is the way I chose to represent polynomials. The constructor of this class has one parameter and that is the `ArrayList` of monomials. In this class I implemented all the required operations on polynomials, namely addition, subtraction, multiplication, division, integration and derivation. There is also a method named `toString()` that actually overrides the `toString()` method and generates a string representation of the polynomial so that further it can be printed out on the Graphical User Interface.

In the addition method I used two iterators to go through the both `ArrayLists` of the polynomials that must be added and I verified whether the grads are equal. If they are equal, then we should add the coefficients of the two monomials. In the end we have to be careful as there may be some monomials that were left out from one polynomial that has monomials with different grads as the other one. I used the `addAll()` method to add them too to the result. After that I have to sort the array of monomials as there might happen not to be in the right position and we want to print the resulted polynomial with the grads in descending order.

The subtraction method resembles the addition method. They are almost equivalent, but in the subtraction method we need to do an additional step. We have to multiply each coefficient of the second polynomial with minus one and then add them together.

In the multiplication method we have to multiply each element of the first polynomial with the second one's elements. I did this by using two iterators through the polynomials and then I called the `sort()` method, as in the previous cases.

Regarding the division method I split the problem into the following cases: if the two polynomials are composed of single monomials then all I have to do is divide the coefficients and subtract the powers of them; in the second case if the first polynomial has smaller coefficient

than the second one with which we want to divide than the result will be zero and the rest will be the first polynomial; in the third case the first polynomial has greater grad than the second one and firstly we have to divide the maximal grad monomials and multiply the result with the polynomial which we divide by, than subtract this polynomial from the first polynomial we are dividing. Repeat this process until the grad of the polynomial to be divided will be smaller than the second polynomial's grad. The result will be returned as an ArrayList of two lists, the first one is the resulted polynomial and the second one is the rest.

In the derivation method all I had to do was to go through the polynomial that was chosen to be derivated and multiply each monomial's coefficient with the grad and reduce with one the magnitude of the grad. The result will be used to construct a new polynomial that will be returned.

In the integration method the grad of each monomial will be increased by one and the actual coefficient will be divided with the value  $\text{grad} + 1$ . In this case we can easily get a double coefficient, that is why I used MonomReal elements to construct the resulted polynomial that will be returned by the method.

## 5. View Class

In this class I implemented the Graphical User Interface of the program using the javax.swing library. Here I implemented the window (frame) of the application, the labels to be printed on the screen, the textFields in which the user should introduce the data and the buttons to add the entered polynomials and to choose the desired operation to be performed on them. In order to place every label, field and button in an elegant way I find using BoxLayout helpful and easier to use. The BoxLayout helps placing elements vertically and horizontally. I used the labels "Polynomial 1" and "Polynomial 2" to show where the data should be inserted, I also gave instruction at the beginning in which form must the data be conveyed. On the buttons I placed the symbols of the operations and the result and rest for division are also marked by labels.

## 6. Main Class

This is the main class of the program and here is instantiated an object of type View, the dimensions of the window are set and the visibility is set to true so that it will be visible to the user.

# 4. Implementing and testing

The Graphical User Interface of the application is easy to use. The user has to introduce two polynomials. At the top of the window I wrote some instructions regarding the way in which the input must be conveyed. After introducing a polynomial the button "Apply" should be pressed. After that the user can choose what operation wants to perform and the result will be placed in

the textField that is after the “Result” label or eventually in the textField placed after the label “Rest”. After this, if the user wants some further operations, he/ she must press the button “Reset”. There is also a button called “Close” which closes the program in case the user wants to quit.

I also implemented a class called JUnitTest which I mentioned earlier but I did not describe. This class is used in order to test every method that implements operations so that we can check that the result is correct. Use for this the `assert.Equals()` method.

## Tests

### 1. Addition

The screenshot shows a Java Swing window titled "Procesarea polinoamelor". The window has a light orange border and a title bar with standard Windows controls. Inside the window, the text "Introduce the polynomials in the form:  $a_1x^n + a_2x^{(n-1)} + \dots + a_{(n-1)}x^1 + a_n$ " is displayed. Below this, there are two sections for polynomial input. The first section, labeled "Polynomial 1", has a text field containing  $3x^2 + 1x^1 + 1x^0$  and an "Apply" button. The second section, labeled "Polynomial 2", has a text field containing  $5x^1 + 1x^0$  and an "Apply" button. Between these two sections are two buttons: "Derivate" and "Integrate". Below the "Polynomial 2" section are four buttons for arithmetic operations: "+", "-", "\*", and "/". Below these buttons is a "Result:" label followed by a text field containing  $3x^2 + 6x^1 + 2$ . Below the result field is a "Rest (for division):" label followed by an empty text field. At the bottom of the window are two buttons: "Reset" and "Close".



## 2. Subtraction

Procesarea polinoamelor

Introduce the polynomials in the form:  $a_1x^n+a_2x^{(n-1)}+....+a_{(n-1)}x^1+a_n$

Polynomial 1

Polynomial 2

Result:

Rest (for division):

### 3. Multiplication

Procesarea polinoamelor

Introduce the polynomials in the form:  $a_1x^n+a_2x^{(n-1)}+...+a_{(n-1)}x^1+a_n$

Polynomial 1

Polynomial 2

Result:

Rest (for division):

## 4. Division

Procesarea polinoamelor

Introduce the polynomials in the form:  $a_1x^n+a_2x^{(n-1)}+....+a_{(n-1)}x^1+a_n$

Polynomial 1

Apply

Derivate

Integrate


Polynomial 2

Apply

+

-

\*



Result:

Rest (for division):

Reset

Close

## 5. Integration

Procesarea polinoamelor

Introduce the polynomials in the form:  $a_1x^n+a_2x^{(n-1)}+....+a_{(n-1)}x^1+a_n$

Polynomial 1

Polynomial 2

Result:

Rest (for division):

## 6. Derivation

Procesarea polinoamelor

Introduce the polynomials in the form:  $a_1x^n+a_2x^{(n-1)}+....+a_{(n-1)}x^1+a_n$

Polynomial 1

Polynomial 2

Result:

Rest (for division):

## Results

The implemented program performs six operations on polynomials. The Graphical User Interface is a friendly one, easy to use. I put also instructions on the top of the window to explain in which form dose the program expect the input. I also used labels to make the usage of the application easier.

## **Conclusion, what I have learned from this assignment**

During the work on this assignment I remembered the concepts of object oriented programming and I understood better the relationships that I used, namely the association, aggregation and inheritance. I also understood better the idea of encapsulation and how to work with ArrayLists. But most importantly I learnt how to make a graphical user interface and how to use ActionListeners and how to handle events done by the user.

Regarding the improvements that can be done on this project are:

- use polynomials with multiple variables
- implement other operations like evaluation of the polynomial in a given point, integration on a given interval  $[a, b]$ , and what would be the most interesting, implementation of the graphic of the polynomial
- implementation of an interface in which the user do not have to reset all the time, but introduce two polynomials and perform any operation he/ she wants

## **Bibliography**

- <http://stackoverflow.com/>

- <http://docs.oracle.com/>

-many youtube videos that show how to make a graphical user interface