

Dan Mitrea
Group: 30421

Technical University of Cluj-Napoca

Programming techniques

-Assignment 1-

Contents

Assignment Objective	3
Problem Analysis	3
Modelling	3
Scenarios	4
Use Cases	4
Design	5
Class Diagrams	5
Packages	6
Monomial Classes	6
Polynomial Class	7
Graphic Design Class	9
Main Class	9
User Interface.	10
Testing and Implementation.	10
Results	11
Conclusions	11
What has been learned	11
Further improvement possibilities.	12
Bibliography.	12

Assignment Objective

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

Problem Analysis

In mathematics, a **polynomial** is an expression consisting of variables (or indeterminates) and coefficients, that involves only the operations of addition, subtraction, multiplication, and non-negative integer exponents. An example of a polynomial of a single indeterminate x is $x^2 - 4x + 7$. An example in three variables is $x^3 + 2xyz^2 - yz + 1$.

Polynomials appear in a wide variety of areas of mathematics and science. For example, they are used to form polynomial equations, which encode a wide range of problems, from elementary word problems to complicated problems in the sciences; they are used to define **polynomial functions**, which appear in settings ranging from basic chemistry and physics to economics and social science. Because of this, applications that easily model these expressions and that implement operations on them are of a high importance for the scientists.

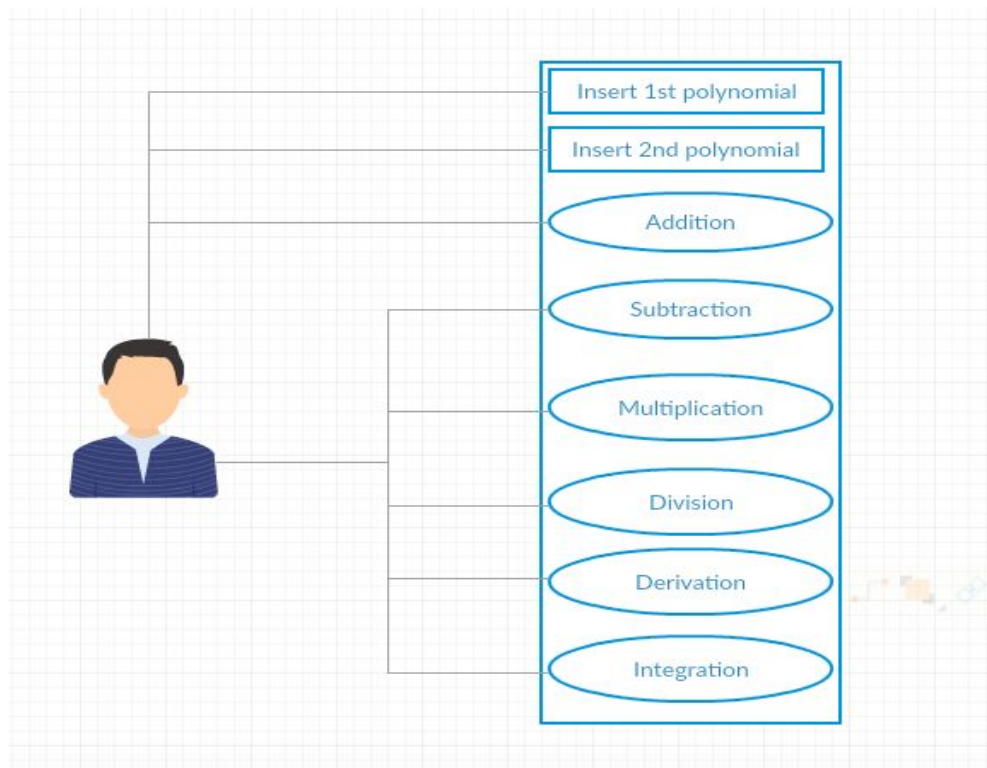
In order to process and work with more complex computations, first of all we must tackle the simpler operations, such as: addition of two polynomials, subtraction of two polynomials, multiplication of two polynomials, division of two polynomials, derivation of one polynomial, integration of one polynomial.

Modelling

Scientific modelling is an activity with the aim of making a particular part or feature of the world easier to understand, define, quantify, visualize, or simulate by referencing it to existing and usually commonly accepted knowledge. It requires selecting and identifying relevant aspects of a situation in the real world and then using different types of models for different aims, such as conceptual models to better understand, operational models to operationalize, mathematical models to quantify, and graphical models to visualize the subject.

In our case, the best way to model our problem is by looking at the polynomials as a collection of monomials. A polynomial is composed of a sum of individual monomials, or, more simple, a monomial is a polynomial with only one term. As a result, our problem is easier to tackle if we make the computations directly on monomials and only after that, we use these methods in our polynomial operations.

Scenarios



In our application, the user has the possibility to do one of the above mentioned operations: insert the coefficients of two polynomials in ascending order of the degree, and then perform the basic operations on them, or on only one of them, depending on the case.

Use case : addition

I will illustrate how the program works on one of the available operations (sadly, the application does not yet support the operation of division) . Even though I will only illustrate it on one scenario, the logic is very simple, the interface is also extremely intuitive, so all the other operations will be done in an almost identical way .

- the user introduces the coefficients of the first polynomial , with how many white spaces between them as he pleases .
- the user introduces the coefficients of the second polynomial , with how many white spaces between them as he pleases .
- if the user wants to perform an operation that requires two polynomials, and he types in only one input, a message will be displayed saying that the operation requires more operands.
- the program takes the user's input as a string and converts it to a polynomial, on which later on it will perform the operations .
- then, the user has a list of buttons with the available operations, and by clicking on one of them, on the Result text field, the result of the operation will be displayed.
- after performing the computation, the application converts the resulting polynomial object to a string .

```

public Monom add (Monom m) {
    if(((double)m.getCoeff().intValue() != m.getCoeff().doubleValue())) {
        return m.add(this);
    } else {
        Monom m1 = new MInt(m.getGrade(), this.getCoeff().intValue() +
m.getCoeff().intValue());
        return m1;
    }
}

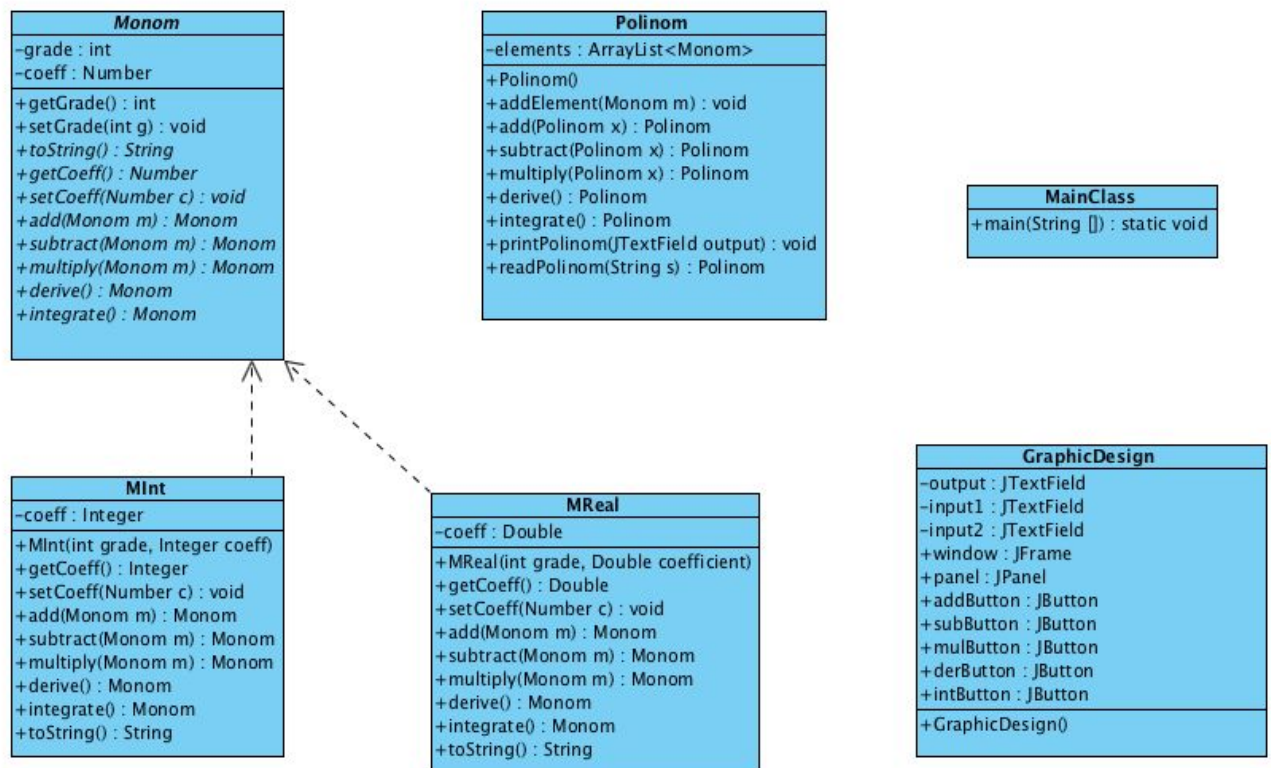
```

Unfortunate sequence of events :

- the input is designed to support only a string containing numbers and white spaces, so if the user inserts other characters, the application will generate an error.
- if the user wants to do the operation of division on the polynomials, the application currently does not support it yet.

Design

Class Diagrams



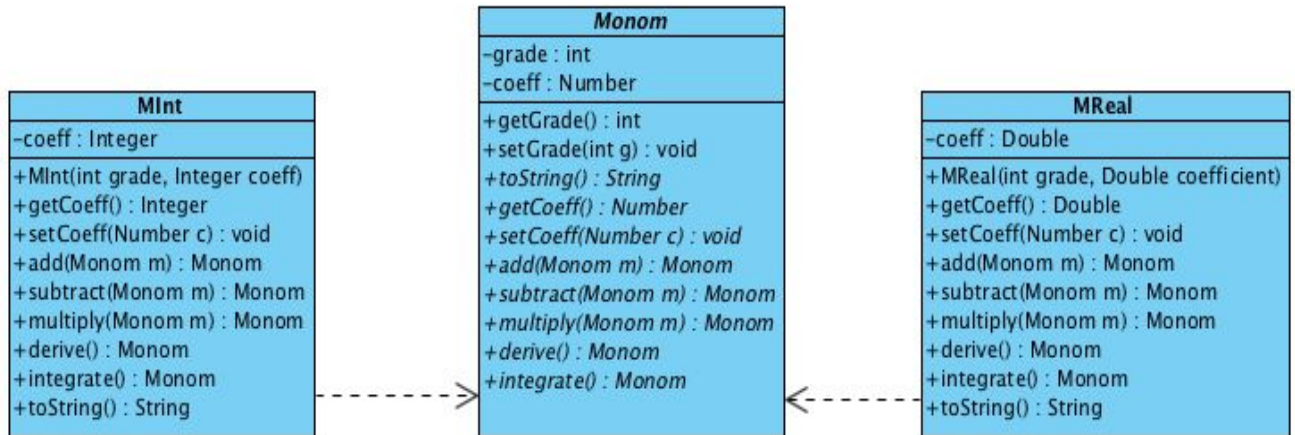
As it can be seen from the above diagram, the logic behind the implementation is pretty obvious. After some operations, the coefficients of the resulting polynomials can be of type Double. So, to make our application more efficient, the abstract parent class Monom has two subclasses, one for the case of integer coefficients (MInt) and one for the case of real coefficients (MReal). I will further explain the implementation in the next lines.

Packages

I have used the standard MVC (Model - View - Controller) design pattern. So, the project will have three packages, as follows:

- model: containing monomial classes and the polynomial one
- view : containing the class GraphicDesign which implements the graphical user interface
- controller : containing MainClass, where we create our environment

Monom Class



The monomial class and its children have the purpose to make the program more practical and more efficient. Operations on polynomials, as stated before, can be reduced to operations on monomials. Due to the fact that some operations may alter the type of coefficients, I have created two subclasses, for each case : real coefficients and integer coefficients. The Monom class does not have a constructor, it contains getters and setters for the grade attribute, because it will remain the same in both subclasses. Furthermore, the coeff attribute is defined as Number so that in both subclasses it may be redefined accordingly, with the correct type(Double or Integer).

All the other methods are defined as abstract in the parent class and written using the correct data types in the children classes. The methods that implement all the operations on these monomials are written clearly, with suggestive variable names and method names, so that anyone who reads the code can easily understand everything. Each one verifies if the values of the coefficients are the correct ones (int values for MInt and double values for MReal) and, if not, it returns the value of the method from the other subclass of the same parent, Monom.

I have overridden the toString() method so that each monomial will be converted to a string, making it easier to be displayed on the screen.

Polinom Class

Polinom
-elements : ArrayList<Monom>
+Polinom() +addElement(Monom m) : void +add(Polinom x) : Polinom +subtract(Polinom x) : Polinom +multiply(Polinom x) : Polinom +derive() : Polinom +integrate() : Polinom +printPolinom(JTextField output) : void +readPolinom(String s) : Polinom

The polynomial class is the actual model of a polynomial. It contains an array list of monomials, representing the monomials which compose the actual polynomial. I have created a method that adds an element of type Monom to the list so that, after performing operations on monomials, it will be easier to add the resulting monomial to the resulting polynomial. I have methods that take care of the implemented operations : addition, subtraction, multiplication, derivation, integration. They take all the elements of the polynomials(the monomials), do the operations on them, and after that, add the resulting monomials to the resulting polynomial.

The method for printing a polynomial does the following : it has as a parameter the text field from the user interface where the displaying will be made. For each monomial, by using the previously overridden method toString(), converts them and adds them to a string which will be displayed for the user.

The method for reading a polynomial does the following : it has as a parameter a string, which contains the polynomial's coefficients, in ascending order regarding the degree. It splits the string into an array of strings, and by parsing them to integer type, we create monomials accordingly and add them to the polynomial, resulting in the desired polynomial.

```
public Polinom readPolinom(String s) {
    String pS = s;
    Polinom polRead = new Polinom();
    String terms = pS.split(",");
    for (int i = 0; i < terms.length; i++) {
        Monom m = new MInt(i, Integer.parseInt(terms[i]));
        polRead.addElement(m);
    }
}
```

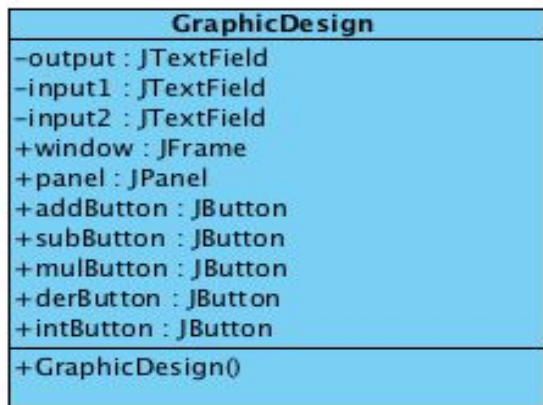


```

    }
    return polRead;
}

```

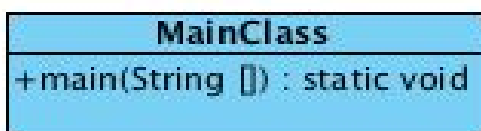
GraphicDesign class



This class has the sole purpose to create the graphical user interface and to create methods for the user to interact with it, by using buttons. When a button is pressed, a specific method for that button is run, which will do the required operation the user intends to do. These methods also display the resulting polynomial on the screen so that the user can see the result.

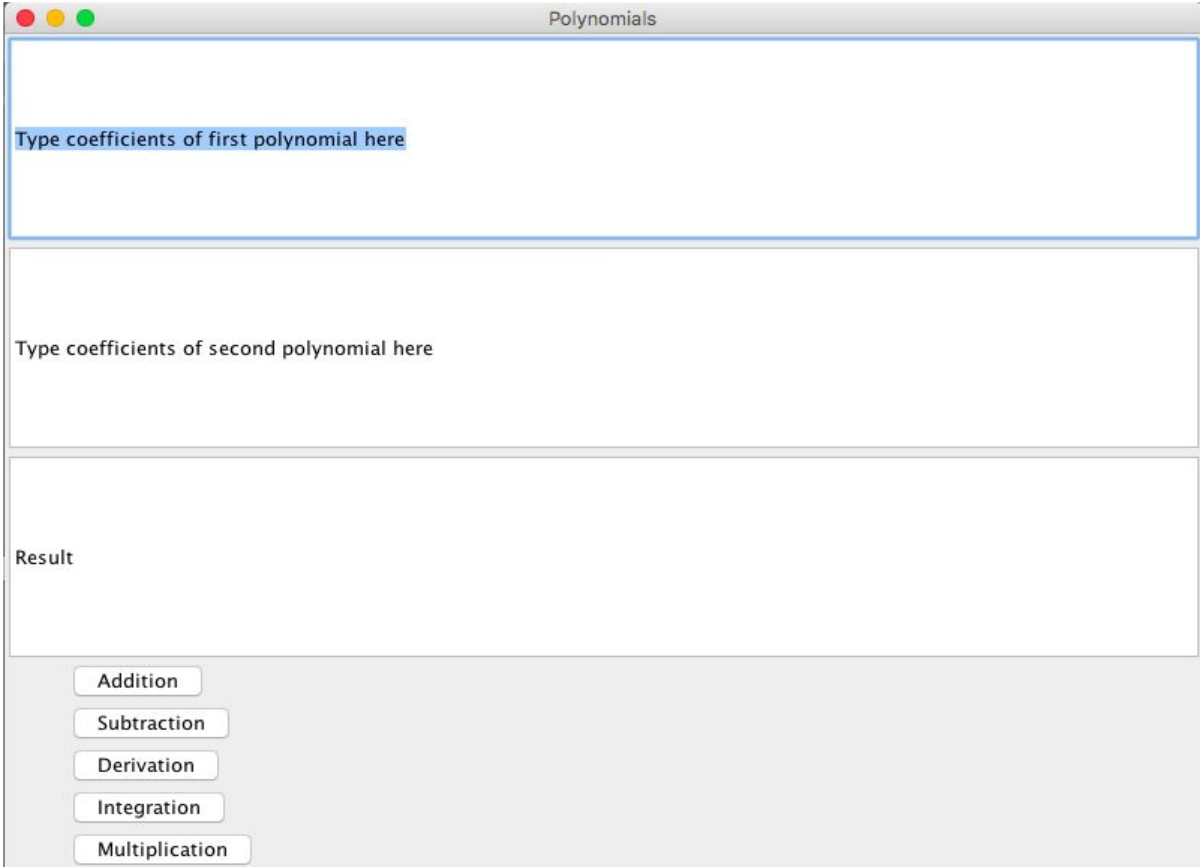
Even though the user introduces the data in a text field as a string, the application has methods (invisible to the user) that convert the input data to polynomials , do the required operations on them, and then convert the resulting polynomial back to a string. This way, the application is user friendly and makes working with it very easy and convenient for the user .

MainClass



The main class, situated in the “ controller ” package, is a very simple yet extremely important class. Without it, the whole application would not work at all. It has a main method where the `GraphicDesign()` method is called which takes care of the interface and also implements the required operations.

User Interface



The screenshot shows a window titled "Polynomials" with a standard macOS-style title bar (red, yellow, green buttons). The window contains three main text input areas and a set of buttons at the bottom. The first input area is labeled "Type coefficients of first polynomial here" and has a blue selection highlight. The second input area is labeled "Type coefficients of second polynomial here". The third input area is labeled "Result". At the bottom, there are five buttons: "Addition", "Subtraction", "Derivation", "Integration", and "Multiplication", arranged vertically.

The user interface for this application is neither a very good looking one, nor a very complex one. Instead, by being the interface of a simple problem, it should also be simple and get the job done.

A text field tell the user where to enter the data and also what data he/she should enter (coefficients of the first / second polynomial). Below, there is a third text field where the result would be displayed. On the bottom of the frame, there is a button for each operation. If the user wants to do an operation that requires two polynomials and he/she forgets to enter one of them, the application tell him/her that he/she should enter one more polynomial in order to do the desired operation.

Implementation and Testing

First of all, I have created the methods for each operation in the Monom classes and also in the Polinom class (Polinom class uses the methods from Monom classes and just iterates through the monomials contained in the polynomials, doing the operations directly on them) . I have verified the corectness of each method by simply creating in the main class two polynomials (each as a list of monomials) and checked if the result displayed in the console was the correct one . After seeing that the methods work as they should, I was confident that I can proceed to other aspects of the implementation, such as the user interface and more methods that would optimize the code. Also, to further verify the corectness of the methods specified for the monomials, I have implemented JUnit tests that verify each method by comparing the result introduced by me, knowing what it should be, and the result of my method, and I have seen that each method works exactly as it should.

So, as I have previously said, after making sure that the methods I had created work as they should, I proceeded with the user interface. This step was rather difficult for me, as I was not used to creating user interfaces before. Firstly, I created all the components that my user interface would have and made sure the design and appearance was how I wanted it to be. Secondly, I implemented all the methods that deal with the pressing of the buttons. Each button, recognisable by its name (the same as the operation it does), has to check if the inputs are the right ones, to do the desired operation on the given input, and then display it.

Results

The result is, after many hours of coding, testing, and making sure everything is working as it should (almost as it should in some cases) , a simple yet effective application that implements a set of operations on polynomials, both of integer and double type coefficients. The application has a user interface that makes it very simple for the user to do what operation he / she desires. It is neither the most efficient one, nor the most good looking, but it does the job it is intended to do, in a simple way that makes it easy to use.

Conclusions

What I learned

Even though this assignment did not contain any new knowledge of the OOP paradigm, or regarding the syntax of the programming language I have used, I can safely say it is one of the most complex Java applications I have ever done. I consolidated my knowledge in the field and realised how much there is still to be learned and to be improved, both in my code, and in my efficiency in writing it. Encountering errors and getting stuck may be frustrating, but on the long run, each difficulty makes us stronger, and more knowledgeable. Also, I have once again realised that time management is extremely important and that it is one of the key factors in delivering a project on time.

Further possibilities for improvement

I am well aware that my application has some flaws that could be fixed in order for a better user experience, or for a better functionality. First of all, it does not yet implement the division operation on two polynomials, so the first improvement would be to design this operation and make it available for the user. Secondly, it could have more test cases that handle wrong inputs from the user. These verifications would tell the user what mistake has been made and how can he/she correct it. Thirdly, the user interface could use some improvements in the design, in order to be more pleasing and maybe even more easy to use and understand. What is more, we can even add more operations to the set of available ones, like finding the root of a polynomial or rising it to a certain power.

However, in my opinion, simplicity is very important in making the application easy to use and understand. Every application should strive to be as simple as possible implementing all the required methods, and I believe that, more or less, I have accomplished this objective.

Bibliography

1. https://ro.wikipedia.org/wiki/Pagina_principală
2. https://creately.com/app/?templID=gc7qvpsj1&login_type=demo#
3. <http://stackoverflow.com>
4. <https://wordcounter.net>