# FUNDAMENTAL PROGRAMMING

# TECHNIQUES

Assignment 1 Polynomial Calculator

## Name: Leonties Dan Alexandru

Group: 30423

Contents:

1. Objective
2. Analysis
3. Design
4. Implementation
   1. Monomial
   2. Polynomial and the operations inside it
   3. Operations
      1. Addition and Subtraction
      2. Multiplication
      3. Division
      4. Derivation and Integration
   4. GUI
      1. The GUI
      2. The Interpreter
      3. The Main
5. Results
6. Conclusions
7. Objective

The objective of this project is to design and implement a polynomial calculator with a dedicated graphical interface through which the user can enter polynomials, select the operation to be performed (in this case we have addition, subtraction, multiplication, division, derivation, integration) and display the result.

The secondary objectives are:

* Classes with maximum 300 lines (except the UI classes)
* Methods with maximum 30 lines
* Java naming conventions
* Object-oriented programming design(encapsulation, Polynomial and Monomial classes)
* Lists instead of arrays
* foreach instead of for(int i=0...)
* Implementation of the addition and subtraction operation
* Implementation of the multiplication operation
* Implementation of the division operation
* Implementation of the derivative and integration operation

The above objectives will be detailed in the implementation section

* Regular expressions and pattern matching for extracting the polynomial coefficients
* Graphical user interface (using Java Swing framework; design according to the Model View Controller architectural pattern)

This will be detailed in the Analysis and Implementation sections

* Junit for testing, which will be detailed in the result section

# Analysis

Firstly, the project will require a person to first introduce one or two Polynomials, which will perform (at the user’s will) at a time, the following operations : Addition, Substraction, Multiplication and Division.

Secondly, the project will require (if the user wants) to introduce one Polynomial which will perform one of the following operations : Integration and Division.

The Graphical User Interface (or GUI for short), requires to be easy to use for any user. In the first part, there will be 2 text inputs, for filling the first 2 Polynomials, used to perform the 4 basic operations, 2 buttons which will describe to the user that in each text prompt, only one Polynomial is allowed to be introduced. There will be a text prompt below the 4 buttons to show the output of the chosen operation. In the second part, there will 1 text input prompt, for introducing another Polynomial to be used for Integration or Division.

The user will be able to access the outputs, in case they want them to be used for further operations.

We will also need a way of confirming that the 2 inputs provided by the user are actually correctly defined Polynomials. For this we will use Regular expressions for pattern matching.

One requirement that the user should always remember is the way the input works!

Let’s say we want to introduce a Polynomial, P:

At the input P should be written like this : 3x^5 + 2x^3 + -x^2 + 5

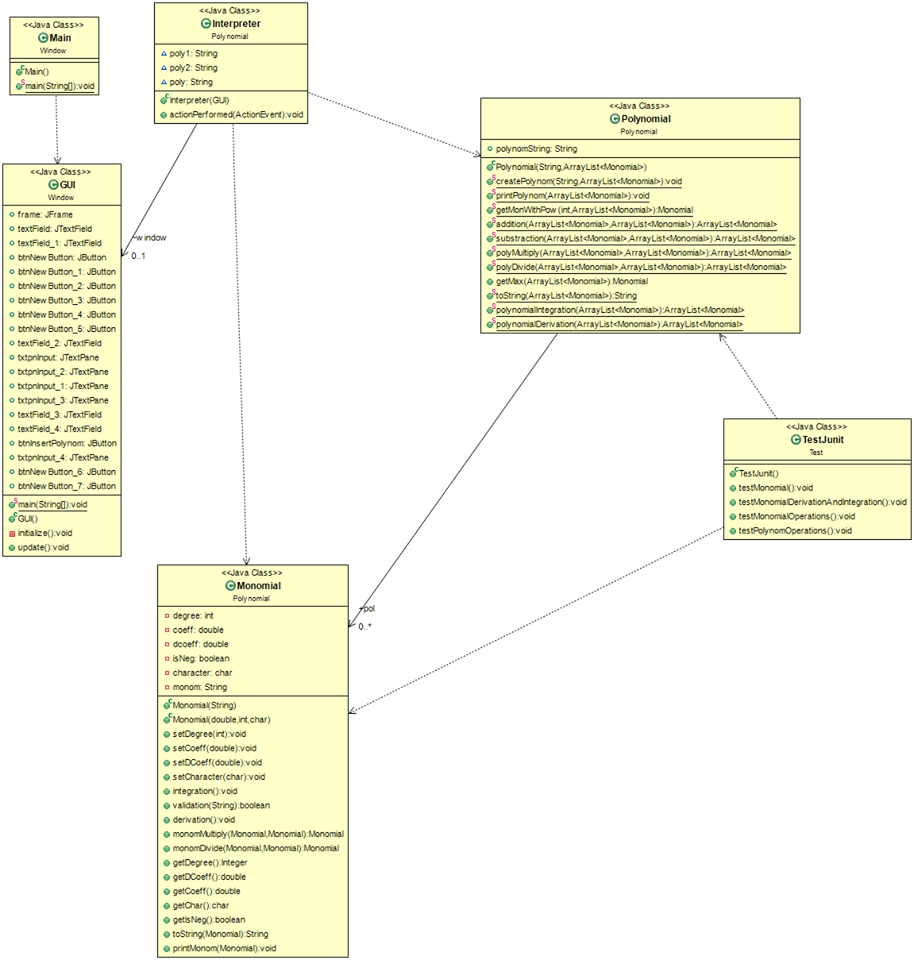
The coefficient can be absent, meaning it will be treated as the integer 1, the power can be absent as well, but the user needs to remember that if he wants to introduce a negative Polynomial, he should describe it as the summation of it with another Polynomial, and the coefficient should be negative. That’s why the user will write, for example “x^2 + -1”.

Also, the project works only by describing a Polynomial as a P(x), and the “x” should always be lowercase.

It is very important that the user leaves 1 blank space between each Polynomial and “+” sign!

# Design

The UML diagram of the whole project can be seen here, and next will be described each part of it:



For the design of the project, 3 packages were created :

Polynomial – which contains the classes required to handle the construction of Polynomials and the operations performed;

Window – which contains the classes used for designing the GUI;

Test – which tests the project using JUnit 4.13.

The Polynomial package contains 3 classes : Monomial, Polynomial and Interpreter.

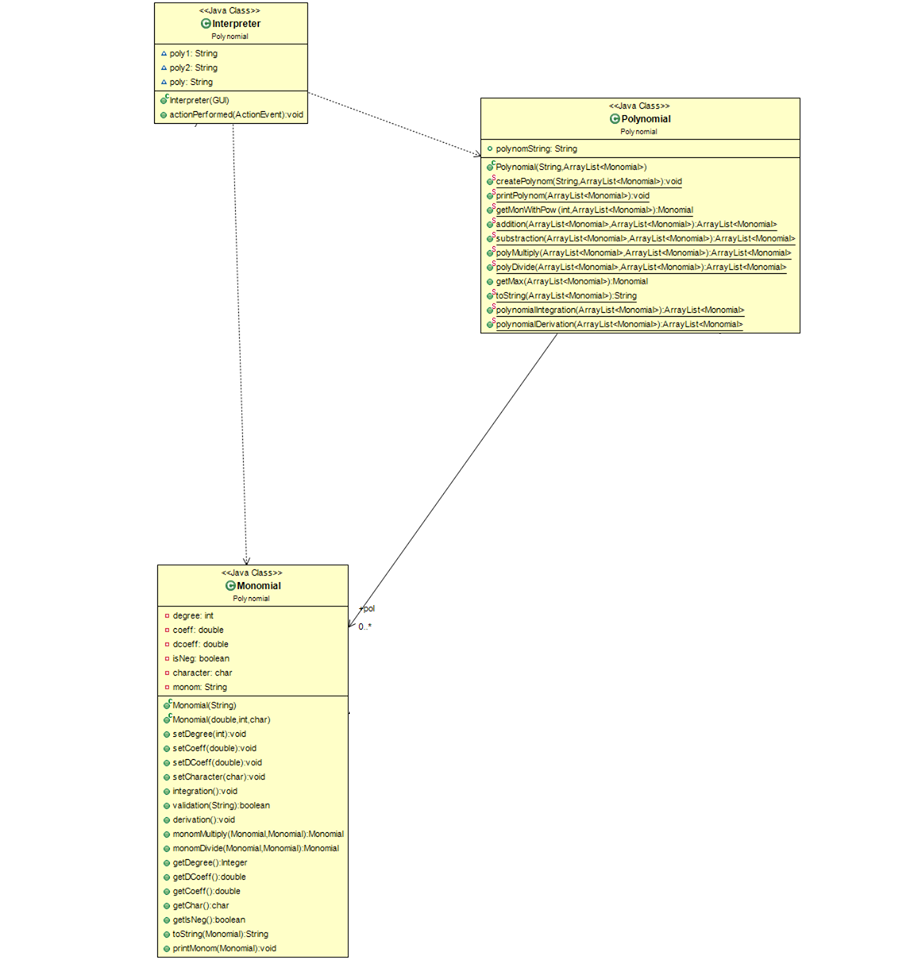
The Monomial class contains the power, coefficient, and character (which will always be ‘x’). It also contains constructors and needed methods, which I will describe in the next section.

The Polynomial class is the class that is used to store the polynomials. It contains an ArrayList

of monomials, stored in descending order of their power. It also contains the needed methods,

which will be described in the next section.

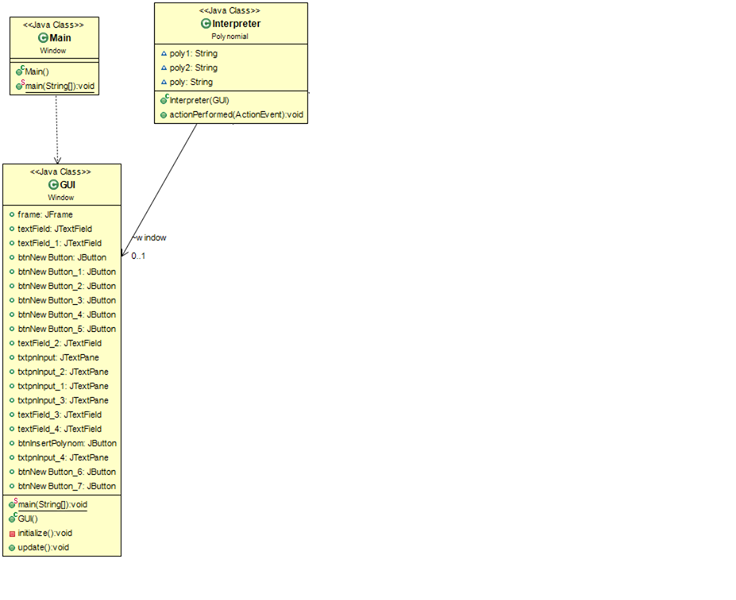
The UML diagram of this Package can be seen here:



The GUI part of the project is based on the Model – View – Controller (or MVC for short), the model was described before this.

The GUI component is the Class that has in it all the GUI parts. In this class, their existence is defined and update for the GUI is used. The Interpreter class is the one that is implementing the Action Listener interface, which is used to give the buttons functionality.

Here is the UML diagram of this part of the project:

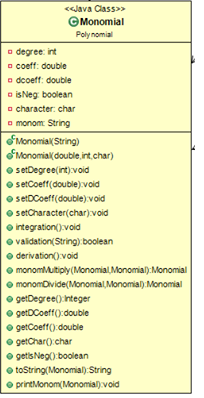


Lastly, we have the Test class which only contains tests used with JUnit.

Finishing with the Design part of the project, the algorithms that are used, aren’t complicated ones, most of them are used for doing the operations on Polynomials. Overall they are simple, and will be described in the following section of this documentation.

# Implementation

## Monomial class

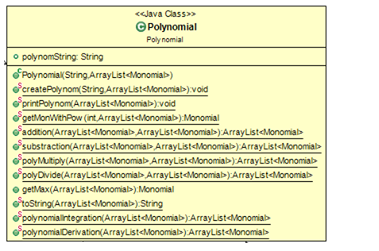


The monomial class is composed of 6 attributes, power, coefficient, Double coefficient, isNegative Boolean, character, and the Monomial string. In terms of constructors, the monomial can be created by specifying it in String form or by specifying its coefficient, power and the character ‘x’. The first constructor is a bit complicated, since it uses Regular expressions to guarantee that the String the user provided is indeed a Monomial. The second constructor is a simpler implementation, since it only uses 3 attributes, the coefficient, power and character ‘x’.

This class also contains getters and setters for the attributes, implementation of the Multiplication, Division, Integration and Derivation operations. It also contains 2 ways of printing the, printMonom which prints in the console the Monomial and to toString which transforms the given Monomial in a String.

## Polynomial class and operations inside it

The Polynomial class is the most important part of the project.



As attributes it contains a String which represents the introduced Polynomial and an ArrayList of the monomials which are inside the Polynomial.

The Polynomial class contains 1 constructor, which uses a String and the mentioned ArrayList to create the Polynomial. The string contains the Polynomial which the user will type in.

The “createPolynom” method is used to create the ArrayList which the constructor uses, in order to be used in Polynomial operations.

Other methods in this class, besides getters and setters, are as follows. The printPolynom is used for printing the Polynomial using the mentioned ArrayList, the getMonWithPow is used for getting a Monomial with a certain power, in case the order of the ArrayList has been modified, and used for Addition and Substraction later. The addition, substraction, polyMultiply, polyDivide, polynomialIntegration and polynomialDerivation are used to perform the 6 operations required (Addition, Substraction , …). getMax is used for finding the Monomial with the highest degree from the ArrayList – this method will be used for the division operation. And the last method, toString, it returns a String, which is the Polynomial from the ArrayList.

## Operations

The following are the operations used inside the Polynomial class.

## Addition and Subtraction

## The Addition and Substraction are done in a similar manner, meaning that both methods have

## 2 ArrayLists as Parameters, we iterate through the first one, using a foreach, and find a Monomial

## the second one which has the same power as the one we are currently on. If it finds one, the

## addition / substraction is performed, the second Monomial gets removed from the second ArrayList,

## and the new Monomial is inserted into the new ArrayList. If not we insert the first Monomial

## in a new ArrayList. Both methods return an ArrayList.

## Multiplication

The method contains 2 parameters, the 2 ArrayLists we want to multiply. The multiplication is done quite simple, a basic foreach in a foreach. We iterate for 1 Monomial from the first ArrayList, through all the Monomials in the second ArrayList. On each iteration we multiply the Monomials, and insert them into the result ArrayList. At the end the new ArrayList is returned.

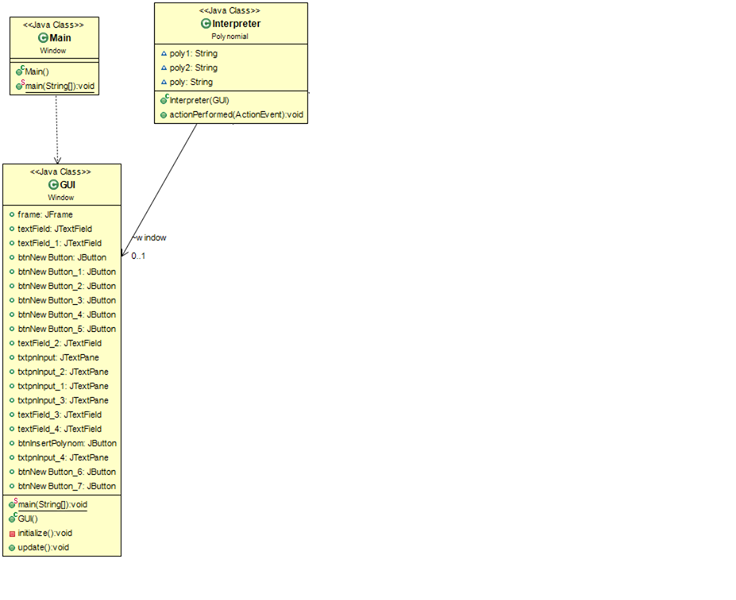
## Division

This method contains 2 parameters as well. The division is implemented according to the long division algorithm. We use a “a rest” ArrayList which is initialized with the value of the dividend. We continue to divide until the Monomial with the highest power from it, has the power lower than the divisor. When this is achieved, the Polynomial is added to the result. The method returns an ArrayList with the result.

## Integration and derivation

These two methods are almost the same, the difference only being that one of them uses the Monomial method Integration, and the other the method Derivation. We iterate through the ArrayList, and if we find a Monomial with the coefficient 0, we add to the result 0, and if not we integrate / derivate the found Monomial.

## GUI



The GUI is made out of 3 components. GUI, Interpreter and Main.

## The GUI

This class contains the code describing what and where buttons and text are placed on the screen, and an update function for updating the GUI after using the Interpreter.

The chosen layout is a mig layout. The layout is as follows : Four columns and 12 rows. First column contains messages to guide the user, like “Input” or “Output”, second column contains the input text forms, output text forms and the buttons used for operations, and the fourth column some buttons to guide the user again where to put the inputs. The third column is used for spacing, and some of the rows are used for spacing as well, so the project looks more user – friendly.

## The Interpreter

This class is the one which implements the Action Listener interface and is used to determine which button was pressed and what operation to do. After determining the button, the corresponding operation method will be called in GUI, in the update method, and the result will be displayed on one of the result text boxes.

## The Main

This class is used for starting the whole project.

# Results

In terms of verifying the correctness of the algorithms of each operation and the structures of the Monomials and Polynomials introduced, the JUnit Framework (version 4.13) was used. For each, a test class was created. In each test class there were tested many scenarios, controlling most of the input cases.

Firstly for the Monomials :

The first test class that was created was used to verify if the Monomial was correctly constructed (i.e. -3x^3). The second class was used to test the Derivation and Integration on the Monomials, coefficients and powers were selected in a way that the resulting coefficient after one of the two operations, was and integer (this was made such that the testing would be easier to see, the program perform the integration and derivation to its full potential). The last test class that was created, was for the 4 required operations : Addition, Substraction, …  
I used 2 Monomials for testing, again, simpler Monomials for easier understanding of the results.

Now for the Polynomials :

For the Addition, Substraction and Multiplication, I have used 2 Polynomials that try cover all of the possible cases, in this example being Polynomials that miss one or more powers or negative coefficients (i.e. “8x^5 + -1x^2 + 20”).

For the Division, since it is the most complicated operation as in verifying the results, I’ve used 2 equal Polynomials, since it will be easier to see the result (again used for easier understanding of the test, the program being able to perform the division to its full potential)

The last class was used for performing the Integration and Derivation.

No problems were found when all the tests.

# Conclusions

The created application is a very easy to use, fast and correct polynomial calculator.

In my opinion the most important updates that can be brought onto this projects are the addition of more Quality of Life features, such as a button that instantly copies the result polynomial into one of the input prompts, or implementing another way of inputting the desired polynomials, making the application usable on Android or iOS, which some users may find really useful.

This project helped me learn how to create a Graphical User Interface using Java and the Swing framework. It also helped me understand concepts of structuring both the project as a whole and the smaller, more organized, mini - projects, the uses of encapsulation, and I also learned how to organize my code, so that when I come back to improve this project, I won’t have a hard time understanding what I did in each part.