

**System for polynomial processing**

* Project #1 -

**Student:** Dariana Lupea

**Teacher:** Delia Balaj

**Group:** 30425

**Deadline:** March 11, 2016

**Table of contents**

1**. Introduction**

1.1 Problem specification . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1.2 Personal interpretation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

2. **General aspects**

2.1 Problem analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

2.2 Modeling . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

2.3 Scenarios and use cases . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

3. **Projection**

3.1 UML diagrams . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

3.1.1 Use-case diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . .5

3.1.2 Class diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . 6

3.2 Data structures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

3.3 Class projections . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

3.4 Packages . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7

3.4.1 Model package . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7

3.4.2 GUI package . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

3.4.3 View package . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

3.5 Algorithms . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .10

3.6 Interface . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

4. **Implementation** **and testing** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15

5. **Results .** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

6. **Conclusions** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

7. **References**  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

1. **Introduction**
   1. **Problem specification**

The entire project is based on the requirements of the proposed homework, which are: “Propose, design and implement a system for polynomial processing. Consider the polynomials of one variable and integer coefficients.”

* 1. **Personal interpretation**

I consider this application as being a great way of combining Mathematics and coding knowledge. In what follows, I will briefly present the polynomial operations that I took into account in my implementation. There are several operations that can be done using polynomials, but I tried to emphasize the most important and used ones.

Firstly, there are basic operations such as: addition, subtraction, multiplication and division (they use two polynomials – binary operations). The others, which were considered to be a bit more difficult, are done on a certain polynomial: differentiation, computing an indefinite integral or a definite one, evaluating the polynomial at a given value.

Considering the fact that a polynomial in a single indeterminate *x* can be written in the form:

a_n x^n + a_{n-1}x^{n-1} + \dotsb + a_2 x^2 + a_1 x + a_0,

there are many methods of solving the given problem, for example : using an ArrayList of **monomials** (each monomial being composed of two terms: coefficient and power of x) or using an **array of coefficients** (the method that I used and which will be explained later). I have also considered the following properties of polynomials:

* The sum of two polynomials is a polynomial
* The product of two polynomials is a polynomial
* The derivative of a polynomial is a polynomial
* The antiderivative of a polynomial is a polynomial

1. **General aspects**

**2.1 Problem analysis**

In order to obtain a good model, it is essential to start by understanding the basic concepts implied by the given problem and to try to find the best solution to it. Therefore, I studied polynomials, how are they represented, which is the method of performing a certain operation on them, etc. I will shortly present some theoretical aspects that helped me out.

A polynomial is an [expression](https://en.wikipedia.org/wiki/Expression_(mathematics)) consisting of [variables](https://en.wikipedia.org/wiki/Variable_(mathematics)) and [coefficients](https://en.wikipedia.org/wiki/Coefficient) which only employs the operations of [addition](https://en.wikipedia.org/wiki/Addition), [subtraction](https://en.wikipedia.org/wiki/Subtraction), [multiplication](https://en.wikipedia.org/wiki/Multiplication), and non-negative [integer](https://en.wikipedia.org/wiki/Integer) [exponents](https://en.wikipedia.org/wiki/Exponentiation).

An example of a polynomial of a single variable x is x2 − 4x + 7.

Two such expressions that may be transformed, one to the other, by applying the usual properties of [commutativity](https://en.wikipedia.org/wiki/Commutative_property), [associativity](https://en.wikipedia.org/wiki/Associative_property) and [distributivity](https://en.wikipedia.org/wiki/Distributive_property) of addition and multiplication are considered as defining the same polynomial.

* 1. **Modeling**

After I received the problem specifications, I tried to draw the class diagram, to see more or less what classes I need to have, also swhat kind of attributes and methods. Of course, the core of this process was represented by the choice of managing the resources I had: a **polynomial** concept. After trying several times, I decided the best solution, from the point of view of the implementation, would be representing the polynomial as an **array of coefficients** (for example, on the first position would be the *coefficient* corresponding to the greatest degree and so on) and a *degree* given by the length of the array – 1;

* 1. **Scenarios and use cases**

Before getting to the final version, it is obvious that I tried different scenarios and my program passed through different stages. The most important changes were due to the fact that I started the code implementation without having a POJO class, which, at that moment, I did not realize how important was.

After asking my teacher for some advice, I noticed that using such a class would make my work many times easier.

Considering the use-cases, this application is really helpful for those who need an easy and efficient way to perform polynomial operations, for example: students, teachers or other people interested in this mathematical area. The graphical user interface offers an explicit environment which allows user to perform several

operations: addition, subtraction, multiplication, division, differentiation, computing an indefinite integral or a definite one, evaluating the polynomial at a given value, multiply the polynomial with a scalar, etc.

**3. Projection**

**3.1. UML Diagrams**

**3.1.1 Use – case diagram**

A **use case diagram** at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different [use cases](https://en.wikipedia.org/wiki/Use_case) in which the user is involved.

Mainly, the user of this application would be a student, a teacher or a person which wants to perform operations on polynomials. This program represents a user-friendly application, it is easy to use and it offers a great functionality.

 In this case, the use-case diagram looks like this:

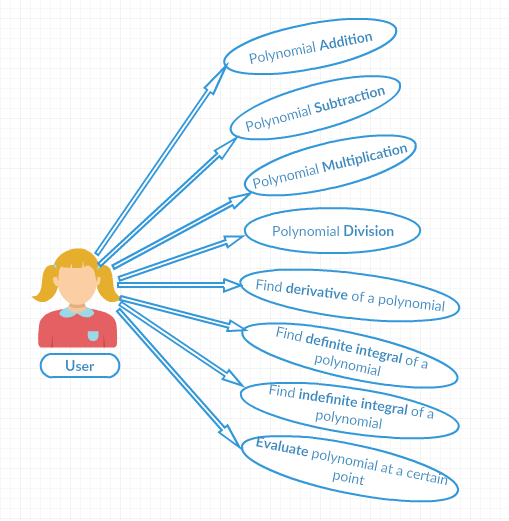
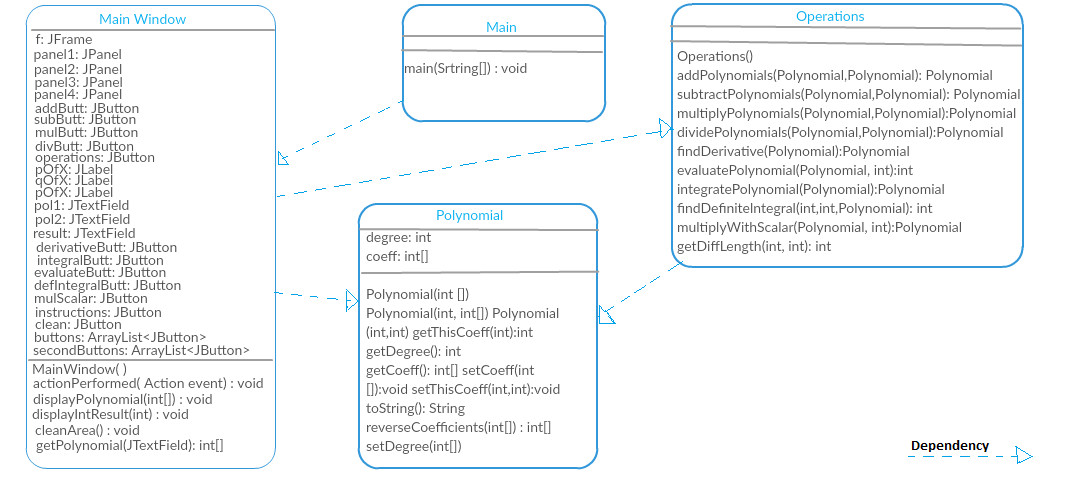


Fig. 1: Use-case diagram

**3.1.2. Class diagram**

****

**3.2 Data Structures**

The data structure that my program is based on is the **array**. I used arrays containing integer types, representing the polynomial coefficients. For example, the polynomial 5x^3 + 2x^2 - 4x + 1 would be represented as an array [5, 2, -4, 1], which has a dimension of 4 elements, so the degree of the polynomial would be equal to 3.

**3.3 Class Projections**

In the project design implementation I tried to use the **MVC** pattern. After several changes, I obtained the final version of the system’s structure. It contains 3 packages, namely: **GUI**, **View** and **Model**. Each of these consists of other classes, which perform specific tasks. I will present them below:

* ***Model***package:
* **Polynomial** class – used for modeling the entity
* **Operations** class – here are the polynomial operations implemented
* **GUI** package:
* **MainWindow** class – creates the user interface
* **View** package:
* **Main** class – which allows data visualization

**3.4 Packages**

**3.4.1 Model** package

It contains 2 classes, which capture the behavior of the application in terms of the [problem domain](https://en.wikipedia.org/wiki/Problem_domain): *Polynomial* and *Operations*.

1. **Polynomial class**

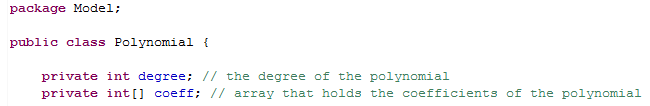
It models an entity of type polynomial. I decided to work with polynomials by considering them as having two attributes:

* **private int[] coeff:** an array of **integer** coefficients, which

contains all the coefficients of the polynomial

* **private int degree**: the degree of the polynomial (positive

integer), given by the length of the coefficients array -1



* it also contains a **contructor** and some **getters** and **setters**

1. **Operations class**

This class provides the functionality of the operations performed on polynomials. I will shortly describe each of them here but, by the end of the paper, I will have fully explained some of the operations.

* **public** Polynomial **addPolynomials** (Polynomial p1, Polynomial p2);
* **public** Polynomial **subtractPolynomials** (Polynomial p1, Polynomial p2);
* **public** Polynomial **multiplyPolynomials** (Polynomial p1, Polynomial p2);
* **public** Polynomial **dividePolynomials** (Polynomial p1, Polynomial p2);
* **public** Polynomial **findDerivative** (Polynomial p1);
* **public** Polynomial **integratePolynomial** (Polynomial p);
* **public** Polynomial **findDerivative** (Polynomial p);
* **public** int **findDefiniteIntegral**(int a, int b, Polynomial p);
* **public** int **evaluatePolynomial** (Polynomial p, int x);
* **public** Polynomial **multiplyWithScalar** (Polynomial p1, int scalar);

**3.4.2 GUI package**

It contains a class responsible with creating the user interface. I used **Swing**, which is a GUI widget toolkit for **Java:**

* **a top level container:** the class *MainWindow* extends **JFrame** and I have created an object of this class called *f*
* **JComponents: JPanels, JLabels:**
* **private JPanel** panel1, panel2, panel3, panel4;

These are the panels where I put the buttons, each one having a certain Layout, in order to be able to arrange them correspondingly

* private **JLabel pOfX, qOfX, rOfX;**

I attached to these labels some images

* **JButtons:**

I used a lot of buttons, one for each operation.

* **private JButton** addButt, subButt, mulButt, divButt, operations;
* **private JButton** derivativeButt, integralButt, evaluateButt, defIntegralButt, mulScalarButt;
* plus some others
* **Text Components**

In order to get the input from the user, my choice was **JTextField**, so I have 3 of them: for the first polynomial, for the second polynomial and one for the result.

* **private JTextField** pol1, pol2, result;

I set **actionListeners** to each button, and in this class I also have the method *actionPerformed*, which allows the connection between each swing component and the operations, which means that the program behaves following the user’s commands.

For example, if the user presses the *addButton* (represented on the interface by a JButton with a symbol “+”), the addition is performed and the output is displayed on the **JTextField** corresponding to the result.

**3.4.3 View package**

This package contains one class:

* **Main –** this class is explicitly used for “starting” the application

**3.5 Algorithms**

Each operation uses its own algorithm, all of them are equally important, but I will present only some of them here.

1. Addition

Polynomials can be added using the [associative law](https://en.wikipedia.org/wiki/Associative_law) of addition, possibly followed by reordering, and combining of like terms.  For example, if

\begin{align}
 P &= 3x^2 - 2x + 5xy - 2 \\
 Q &= -3x^2 + 3x + 4y^2 + 8
\end{align}

then

P + Q = 3x^2 - 2x + 5xy - 2 - 3x^2 + 3x + 4y^2 + 8 

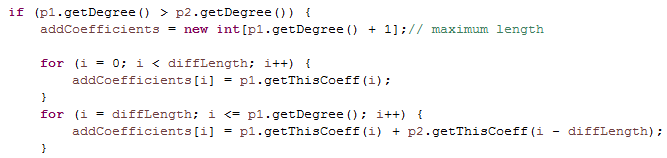
which can be simplified to

P + Q = x + 5xy + 4y^2 + 6 

The above algorithm was implemented in the following way:

C:\Users\Dariana Lupea\Desktop\Untitled.png

* the *addPolynomials* method receives as parameters two Polynomial objects and also return a Polynomial object
* first, the degrees of the polynomials are compared
* the result’s degree will be equal to the highest degree of the terms
* the addition is done considering the order of the terms, for example, the case in which the degree of the first Polynomial is less than the degree of the second one:



1. Subtraction

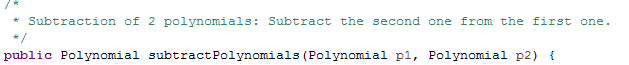
Subtracting polynomials is quite similar to adding polynomials, but you have that pesky minus sign to deal with. Here are some examples, done both horizontally and vertically:

* Simplify (*x*3 + 3*x*2 + 5*x* – 4) – (3*x*3 – 8*x*2 – 5*x* + 6)

change signs

The algorithm I chose is also similar to the one of adding polynomials, the only difference is that when subtracting a highest order polynomial from a lower order one, the higher order terms of the second one will be negated.

For example: (3*x*2 + 5*x* – 4) – (3*x*3 – 8*x*2 – 5*x* + 6) = – 3*x*3 + 11 *x*2 + 10x-10

****

1. Multiplication

* The general rule is that **each term in the first factor has to multiply each term in the other factor**
* The number of products you get has to be the number of terms in the first factor times the number of terms in the second factor. For example, a binomial times a binomial gives four products, while a binomial times a trinomial gives six products.
* Example: Product of a binomial and a trinomial

(*x* + 2)(*x*2- 2*x* + 3)

|  |  |
| --- | --- |
| You just add up the like terms that are conveniently stacked above one another: | http://www.jamesbrennan.org/algebra/polynomials/multiplication_of_polynomials_files/image016.gif |

C:\Users\Dariana Lupea\Desktop\mul.png

1. Division

I have implemented the [Euclidean division](https://en.wikipedia.org/wiki/Euclidean_division), which gives rise to a complete division algorithm using additions, subtractions, and comparisons.

1. Differentiation

Differentiating a polynomial function can help track the change of its slope. To differentiate a polynomial function, all you have to do is multiply the coefficients of each variable by their corresponding exponents, lower each exponent by one degree, and remove any constants. The algorithm that I have used here is the following:

1. Identify the variable terms and constant terms in the equation.

2. Multiply the coefficients of each variable term by their respective exponents.

3. Lower each exponent by one degree

4. Replace the old coefficients and old exponents with their new counterparts

5. Find the value of the new equation with a given "x" value.

The method *findDerivative* receives a Polynomial object and returns also a Polynomial object, the result of differentiation.

C:\Users\Dariana Lupea\Desktop\derr.png

1. Evaluate polynomial at a given value

"Evaluation" mostly means "simplifying an expression down to a single numerical value". To evaluate, you take the polynomial and plug in a value for *x*.

Evaluate ***x*4 + 3*x*3 – *x*2 + 6 for *x* = –3.** The result is : **–3.**

C:\Users\Dariana Lupea\Desktop\eval.png

1. Indefinite Integration

An indefinite integral of a [function](https://en.wikipedia.org/wiki/Function_(mathematics)) *f*  is a differentiable function *F* whose [derivative](https://en.wikipedia.org/wiki/Derivative) is equal to the original function *f*.

For example, the function *F*(*x*) = *x*3/3 is an antiderivative of *f*(*x*) = *x*2.

Consider the real-valued indefinite integral,

$$\int \left( 4x^5 - 2x^3 + x + 4 \right) \: dx$$

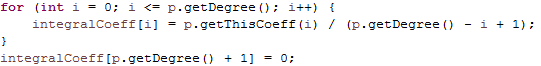
The integrand is a polynomial, and the analytic solution is

$$\frac{2}{3}x^6 - \frac{1}{2}x^4 + \frac{1}{2}x^2 + 4x + k$$

where $k$ is the constant of integration.

So, in my program, I used the following algorithm:

* create a polynomial of degree higher with an order than the integrand
* each term of the resulting polynomial will be the coefficient of the integrand divided by the term’s order
* the last element of the result will equal 0, in order to preserve the convenient position in the array of coefficients



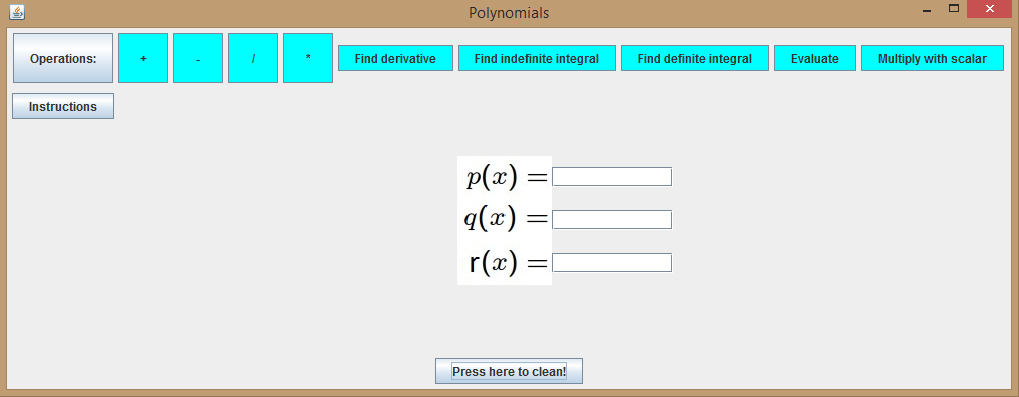
1. Definite Integration

This operations is based on the previous one, combined with the *evaluatePolynomial* method.

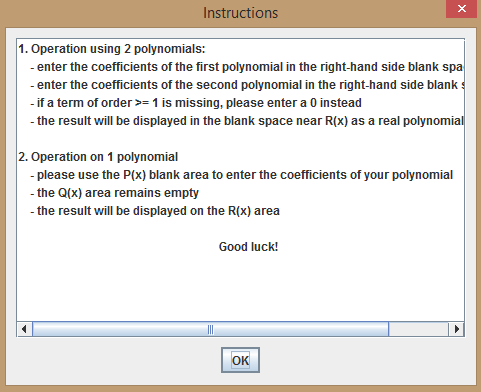
**3.6 Interface**

For the interface, I used many JButtons, JLabels, and JTextFields.

This application GUI looks like in the following image:



* The **user** should:
* Enter the coefficients of the first polynomial
* Enter the coefficients of the second polynomial
* Press the button with the desired operation
* the result will be displayed in the third field
* in order to perform a new operation, the clean button should be used
* user can find more details by pressing the “Instructions” button, which will reveal this:



1. **Implementation and testing**

Regarding the implementation process, I used the **Eclipse IDE**. During the program development steps, many changes were made and as I started to have more and more methods I realized the importance of a good structure. I refer here to organization of the code in packages and classes. I consider that the code I wrote is understandable and reusable. The algorithms I used are relatively easy, based on the well-known mathematical algorithms.

I have also tested each method, in the following way:

-> I used the console at first, for displaying the result

-> I initialized data with some appropriate values

-> I checked if the obtained result was the expected one

-> if YES, I continued the implementation process

-> if NO, I tried to figure out where is the error / problem coming from

and solve it

**5. Results**

The results can be analyzed from different points of view. From my point of view, as developer of this application, I can say that this program fulfills all the requirements of the “client”. I tried to implement the best version I could, considering the limited time and experience resources. From the user point of view, which has access only to the interface, I could assume that he/she would be satisfied by the functionalities offered by the polynomial application.

And I can’t wait to find out what is my teacher opinion concerning my project!

**6. Conclusion**

To conclude, I can say that this project meant hard work, a lot of new things learned, focusing, development and creativity. Even if I encountered a lot of problems, I was able to fix them after all, by searching on the internet or asking a colleague for advice. I think that my application satisfies the requirements and the users will have at their disposal all its functionalities.

**7. References:**

* <http://stackoverflow.com>
* <http://www.oracle.com/technetwork/articles/java/index-137868.html>
* <http://www.tutorialspoint.com/java/java_data_structures.htm>
* <https://en.wikipedia.org/wiki/Polynomial>