

Assignment 1 Documentation

Polynomial Calculator

Grama Mălina Bianca

Group 30423

Teaching Assistant: Antal Marcel

Table of Contents

1. Assignment Objectives	3
a. Primary Objective.....	3
b. Secondary Objectives	3
2. Problem Analysis	4
3. Design	5
4. Implementation	6
5. Testing and Results	9
6. Conclusions and Further Development	9
7. Bibliography	10

1. Assignment Objectives

The objective of the polynomial calculator is to do several different computations (addition, subtraction, multiplication, division, derivation and integration) on one or two mathematical polynomials and display the correct result. Because performing computations on paper is fairly difficult and time-consuming, implementing a calculator that takes as an input one or two strings of characters representing polynomials, and computes the selected operation, will make solving problems involving polynomials much faster and easier.

Therefore, the assignment objectives are to implement a polynomial calculator with a dedicated graphical interface through which a user can insert polynomials of the form $2x^3+3x^2-5x+1$, select the desired mathematical operation, and view the result. The assignment should be made using an object-oriented programming design. Some requirements to be considered are the Java naming conventions, architectural pattern for the assignment, using regular expressions and pattern matching for manipulating the input, Junit for testing, and so on.

a. Primary Objective

The primary objective of this first assignment is to propose, design and implement a system that is able to process and compute the mentioned operations with polynomials that have a single variable and integer coefficients.

b. Secondary Objectives

The secondary objectives of the assignment that I followed while implementing my solution would be the following:

- **Development of use cases and scenarios** – we need to think about how the user will interact with the application and what scenarios could arise. This will be described in detail in the **problem analysis** section of this documentation.
- **Choosing the data structures** – we need to think about how the user input will be stored and managed by our application, so that it will generate the required result. This will be briefly mentioned in the **design** section of the documentation.
- **Division of the solution into classes** – we need to think about how we will model the real-life “object” that is a polynomial into an object-oriented programming object. This will be mentioned in the **design** section of this documentation and detailed in the **implementation** section.
- **Declaration of variables and methods** into aforementioned classes – after we figure out the classes that we need, we also need to think about the variables and methods that will go into these classes. Most of the times we start from the project specification, looking for nouns, which become possible candidate classes, and verbs that could play the role of class methods. This will be detailed in the **implementation** section.
- **Development of the algorithms** necessary for the mathematical computations – we need to think about how we will implement the required operations with polynomials. For this, we need a strong mathematical background, so that we can model the real-life steps of a certain operation into the Java programming language. This objective will be detailed in the **implementation** section of the documentation.
- **Implementation of the solution** – we will need to describe more specifically now each class used in the project. Also, we will describe the implementation of the user interface. This will be detailed in the **implementation** section of this documentation.
- **Testing** – detailed in the **testing and results** section.
- **Debugging.**
- **More testing.**

2. Problem Analysis

By analyzing the problem, we refer to a first abstract set of operations and properties through which we try to detect possible features and behaviors of unknown processes. Object-oriented programming offers us a clear advantage here, precisely because it allows us to tackle the problem from a higher level, without being constrained, to such an extent, by the technical characteristics.

Below we will exemplify a use case for the operation of addition performed by the user on two polynomials:

Use Case Name: Add Polynomials

Primary Actor: User

Triggers: The user indicated that he wants to perform an addition operation by clicking the “Addition” button on the user interface.

Preconditions: The user has typed the two polynomials to be computed in the designated text fields.

Normal Flow:

- The user inserts the two polynomials in the graphical user interface.
- The user selects the “Addition” operation by pressing on the specific button on the graphical interface.
- The Polynomial Calculator performs the addition of the two polynomials and displays the correct result.

Alternate Flows:

- The user inserts the two polynomials, but with syntax errors (either by mistake or because the user is not aware of the syntax required for the two polynomials).
- The user selects the “Addition” operation by pressing on the specific button on the graphical interface.
- The Polynomial calculator checks the two expressions and displays an error message, without displaying anything in the result text field.
- The scenario returns to the first step.

We need to define a set of functional and non-functional requirements, to ensure that we cover all of the cases and provide the functionality of the application.

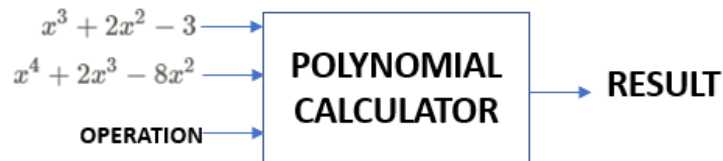
Some of the functional requirements could be:

- The polynomial calculator should allow users to insert polynomials (both through the keys displayed on the graphical interface representing digits from 0-9, +, -, ^ and x, and through the computer keyboard).
- The polynomial calculator should allow users to select the desired operation they want to make on the polynomials.
- The polynomial calculator should check if the input of the user has the correct syntax of a polynomial, displaying an error message in the eventuality that the user typed illegal inputs.
- The polynomial calculator should perform correctly the following operations: the addition of two polynomials, the subtraction of two polynomials, the multiplication of two polynomials, the division of two polynomials, the derivation of a polynomial and the integration of a polynomial.
- The polynomial calculator should display the correct results on the user interface in the form of a correct polynomial (the degree of the variables in the correct decreasing order of the power, non repeating degrees).

Some of the non-functional requirements of the application:

- The Polynomial Calculator should be easy to use and intuitive, and the correct syntax of a polynomial should be clearly stated to help the user to type correctly the input.
- The result should be clearly stated so that it is easily understood by the user.

3. Design



The “black box” of the application

We try to think of the solution in a bottom-up manner, dividing the components of the problem into smaller pieces. As we advance to lower levels, the complexity of the problem increases.

In our example, the bigger component is represented by a polynomial. A polynomial is an expression consisting of variables (also called indeterminates) and coefficients, that involves only the operations of addition, subtraction, multiplication, and non-negative integer exponentiation of variables. We can think of a polynomial like a sum of smaller components that we will call monomials. Monomials consist of a single variable, a coefficient and a degree (power) for the variable.

After this analysis, we decide that we are going to use two classes to handle the Polynomials typed by the user: a class named **Monomial**, that has as variables a coefficient and a degree (power), and a class named **Polynomial** that will consist of an `ArrayList` of elements in the **Monomial** class.

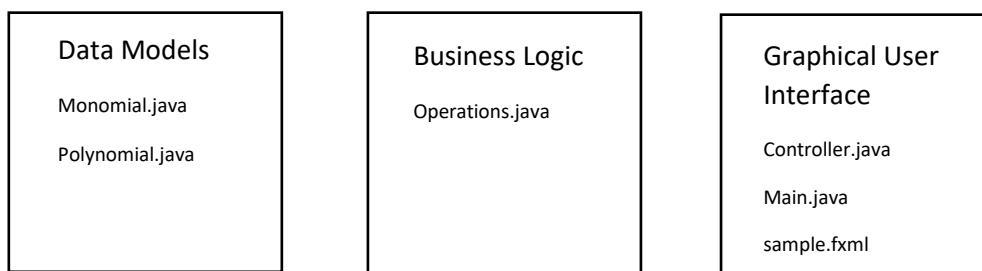
After this brief analysis, it is clear that we will implement the following Java Classes:

- A class called **Monomial** with the variables: `private double coefficient` and `private int power`, representing the degree of the variable. The coefficient is declared as a double number, because the division and integration operations will display polynomials that will contain real numbers as coefficients.
- A class called **Polynomial** with the variable: `private final List<Monomial> monomials`.

Using an architectural pattern, we know that we need three types of classes in our design: **Data Models** that contains the classes modeling the application data, **Business Logic** that contains the classes implementing the functionality of the application, and finally, the **Graphical User Interface** that contains the classes and files implementing the graphical user interface.

Following this pattern, we know that we need another class, called **Operations**, that will handle the arithmetical operations required by the Polynomial Calculator. This class will not contain any variables, only methods implementing the said operations. We will place this class in the Business Logic of our project.

Regarding the Graphical User Interface, we will use JavaFX to implement the Polynomial Calculator. Therefore, we will need a **Controller** class that will handle the functionality of the elements placed in the user interface, a **Main** class that will be the driver of our application and will hold the main method, and an `FXML` file that will contain information about our GUI and will link the GUI with the Controller class.



4. Implementation

Now that we have decided what classes are needed for our project, let's talk briefly about the main methods of these Java Classes.

We will start with the core of our application, mainly the **Monomial** class. We know that we need variables for the coefficient and power of the monomial. The first step is to define a method that gets as a parameter a string representing a monomial with the right syntax, and transforms this parameter to the variables mentioned before. This method is called `public boolean validate(String)`, and returns a boolean true/false depending on the validity of the String. If the String does not have the right form, the method returns a false, otherwise it returns a true and sets the variables coefficient and power to the right values, using regular expressions and pattern matching. Another method that we will need in this class would be a method that transforms the variables to their corresponding String expression. In other words, we need the inverse of the previous method. This method is called `public String monToString()`. It has no parameters, because we work with the parameters of the class. Now, depending on whether the coefficient of the monomial is an integer or a real number, the method will transform this number in a string, appending "x^" and the string transformation of the power variable. This method will be used for displaying the results of the computations on the GUI.

After we talked about the methods of the Monomial class, it is time to list some methods in the **Polynomial** class. `public void addMonomial(Monomial monom)` will add a monomial to the ArrayList. The inverse of this method is `public List<Monomial> getMonomials()`, that gets the monomial ArrayList and is useful for manipulating this list in other classes. We will implement some helper methods that will be useful in doing the operations required, like:

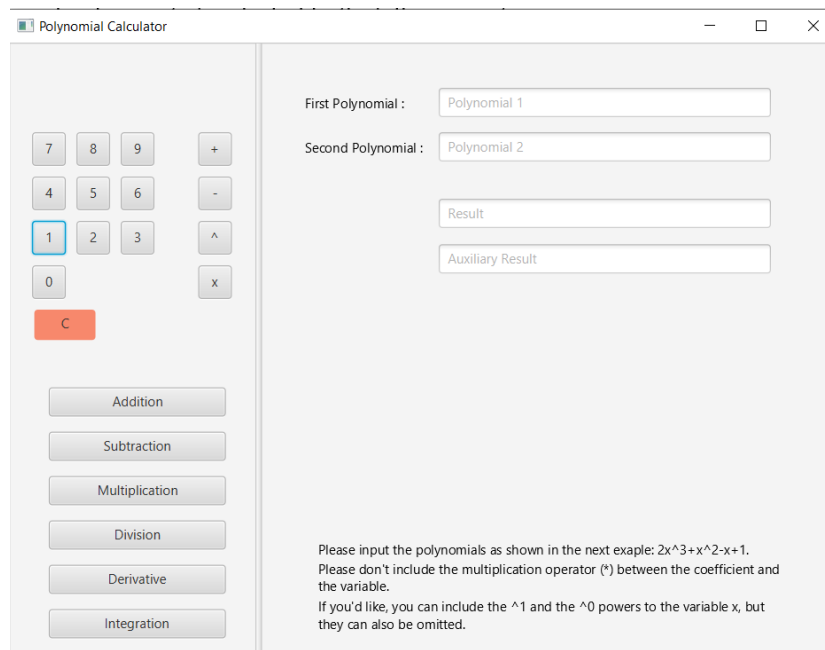
- `private void sort()` : sorts the ArrayList based on the power of the monomials;
- `public void group()` : checks if there are monomials with the same power in the list, and adds their coefficients so we don't have duplicates in the final results;
- `public void check()` : traverses the list to see if there are monomials that have the coefficient 0, and discards them;
- `public boolean isZero()` : checks if the polynomial is zero (empty);
- `public int degree()` : returns the degree of the polynomial;
- `public double getHighestCoefficient()` : returns the coefficient of the monomial with the highest degree;
- `public double getCoefficient(int degree)` : returns the coefficient of the monomial with the degree passed;
- `public void negate()` : multiplies the polynomial with -1.

Two important methods of the class Polynomial are the method that takes a string and forms the list of monomials that represent the polynomial, and the inverse method that takes the list and converts it into a string representing the polynomial. These methods are `public boolean validateAndSet(String exp)` and `public String polToString()`.

Now we will list the methods of the **Operations** class. These methods mirror the operations required by the problem specification, and are as following:

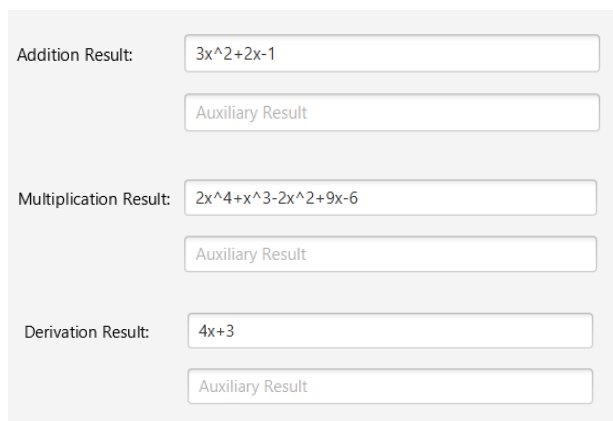
- `private Polynomial appendPolynomials(Polynomial p, Polynomial q)` : appends the two polynomials; helper function used in the addition and subtraction methods;
- `public Polynomial addition(Polynomial p, Polynomial q)`;
- `public Polynomial subtraction(Polynomial p, Polynomial q)`;
- `public Polynomial multiplication(Polynomial p, Polynomial q)`;
- `public Polynomial[] division(Polynomial p, Polynomial q)`;
- `public Polynomial derivate(Polynomial p)`;
- `public Polynomial integrate(Polynomial p)`.

The graphical user interface looks like the following image:

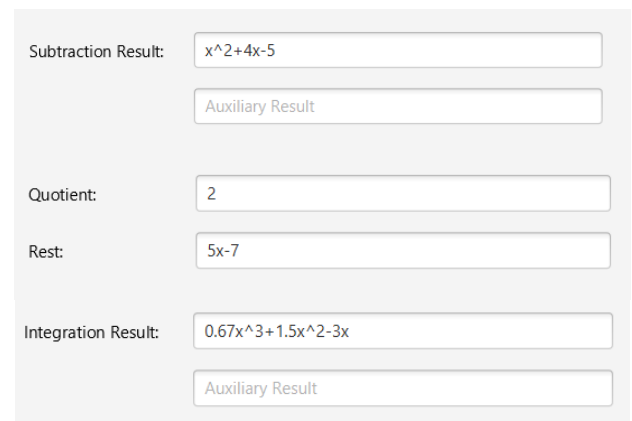


The screenshot shows a window titled "Polynomial Calculator". On the left is a numeric keypad with buttons for digits 0-9, a decimal point, and operators +, -, ^, and x. Below the keypad are buttons for "Addition", "Subtraction", "Multiplication", "Division", "Derivative", and "Integration". On the right, there are two input fields labeled "First Polynomial:" and "Second Polynomial:", each with a placeholder text "Polynomial 1" and "Polynomial 2" respectively. Below these are two output fields labeled "Result" and "Auxiliary Result". At the bottom right, there is instructional text: "Please input the polynomials as shown in the next example: $2x^3+x^2-x+1$. Please don't include the multiplication operator (*) between the coefficient and the variable. If you'd like, you can include the ^1 and the ^0 powers to the variable x, but they can also be omitted."

On the left side of the window we have an AnchorPane containing the Buttons of the application, and on the right side of the screen we have an AnchorPane with the input and output values. The buttons representing 0-9 digits and the symbols +, -, ^ and x can be used to input the two polynomials, only **after** the user clicks on the TextField representing the desired polynomial. For example, if the user wants to input the first polynomial, he will click on the TextField that reads "Polynomial 1", then he will click on the buttons representing the keys of the calculator to type the syntax of the polynomial. If the user wants to input the second polynomial, he will click on the TextField below that reads "Polynomial 2", then he will press the keys of the calculator. It should be noted that the user is not constrained to use the keys of the calculator on the screen. He can easily type the two polynomials from the computer's keyboard, making sure to follow the instructions displayed at the bottom of the screen. After the two polynomials have been typed correctly, the user will press the button with the desired computation. After this, the result will be displayed in the TextFields that read "Result", and, if the computation selected was "Division", the remainder will be listed in the field that reads "Auxiliary Result". Before these text fields will appear the texts:



This screenshot shows the results of three operations. For "Addition Result:", the main field contains $3x^2+2x-1$ and the "Auxiliary Result" field is empty. For "Multiplication Result:", the main field contains $2x^4+x^3-2x^2+9x-6$ and the "Auxiliary Result" field is empty. For "Derivation Result:", the main field contains $4x+3$ and the "Auxiliary Result" field is empty.



This screenshot shows the results of four operations. For "Subtraction Result:", the main field contains x^2+4x-5 and the "Auxiliary Result" field is empty. For "Quotient:", the main field contains 2. For "Rest:", the main field contains $5x-7$. For "Integration Result:", the main field contains $0.67x^3+1.5x^2-3x$ and the "Auxiliary Result" field is empty.

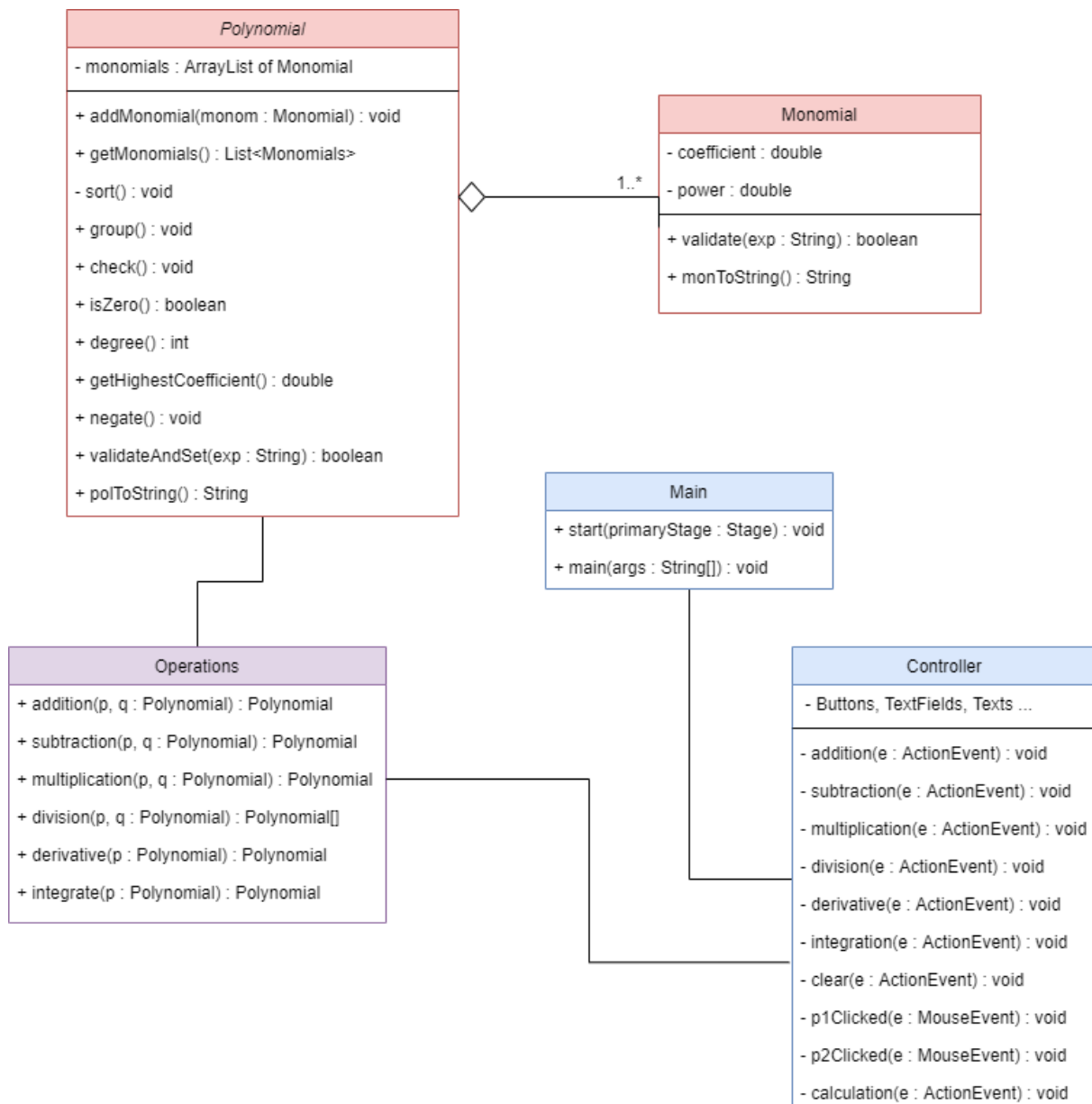
If the user inputs a polynomial with the wrong syntax in any of the two fields, the GUI will display a warning message:

First Polynomial :

Second Polynomial :

Incorrect polynomial syntax. Try again!

Bellow, you will find the UML diagram of the application. Each class is colored corresponding to the architectural pattern (pink for Data Model, purple for Business Model and blue for GUI). The variables and the main methods are listed.



UML Diagram of the project

5. Testing and Results

For testing, I used the Junit framework to write tests for all of the operations. I tested all of the operations on the polynomials $p1=x^5+7x^3+5$ and $p2=x^2-13$. I computed all of the operations by hand on a piece of paper and came to the following results:

Addition: $x^5+7x^3+x^2-8$

Subtraction: $x^5+7x^3-x^2+18$

Multiplication: $x^7-6x^5-91x^3+5x^2-65$

Division: Quotient: x^3+20x

Rest: $260x+5$

Derivative: $5x^4+21x^2$

Integral: $0.17x^6+1.75x^4+5x$

I wrote the code for all the testing functions and all of the six tests have passed. Bellow I will attach screenshots of two test methods, namely the test for the multiplication and division method:

```
@Test
public void mulTest() {
    String exp1 = "x^5+7x^3+5";
    String exp2 = "x^2-13";
    Polynomial p1 = new Polynomial();
    Polynomial p2 = new Polynomial();
    p1.validateAndSet(exp1);
    p2.validateAndSet(exp2);

    Operations op = new Operations();
    String result = "x^7-6x^5-91x^3+5x^2-65";
    Polynomial res = op.multiplication(p1, p2);

    assertEquals(result, res.polToString(), message: "The result of multiplying x^5+7x^3+5 and x^2-13 is x^7-6x^5-91x^3+5x^2-65");
}

@Test
public void divisionTest() {
    String exp1 = "x^5+7x^3+5";
    String exp2 = "x^2-13";
    Polynomial p1 = new Polynomial();
    Polynomial p2 = new Polynomial();
    p1.validateAndSet(exp1);
    p2.validateAndSet(exp2);

    Operations op = new Operations();
    String quotient = "x^3+20x";
    String rest = "260x+5";
    Polynomial[] result = op.division(p1, p2);

    assertEquals(quotient, result[0].polToString(), message: "The quotient of dividing x^5+7x^3+5 and x^2-13 is x^3+20x");
    assertEquals(rest, result[1].polToString(), message: "The rest of dividing x^5+7x^3+5 and x^2-13 is 260x+5");
}
```

6. Conclusions and Further Development

In conclusion, I found the assignment a little bit challenging, but that is just because I did not have a lot of practice with the Java language beforehand. It could be the case that the code I wrote may not be very advanced or it could be a little disorganized in some places, but I tried my best to solve the tasks with the knowledge that I have.

A thing I have learned from this first task is the importance of dividing the objectives at the very beginning of the problem. A lack of rigor in the planning phase can set back the implementation, so the best approach is to “divide and conquer” the big problem into smaller tasks and implementing, testing and debugging them before going forward.

As for further development of the Polynomial Calculator, some additions could be to display the real coefficients of the results of division and integration in a fraction form, like we do on paper when we compute different expressions. Another development would be to output the resulting polynomials in a nicer form, like “ $2x^3-x^2+x-2$ ”, instead of “ $2x^3-x^2+x-2$ ”. Also, some additional operations could be added to the calculator, like computing the roots of the polynomials, drawing the graphs of the polynomials and so on.



7. Bibliography

- <https://gist.github.com/derlin/40545e447fffe7599d26d0a435d9b113> - for the division method in the class Operations (also mentioned in the comments of the code).
- https://code-it.ro/calculator-de-polinoame-in-java/#_Toc445825166 – for the validate method in the class Monomial (also mentioned in the comments of the code).
- <https://en.wikipedia.org/wiki/Polynomial>.
- Class Materials provided by the professor and the teaching assistant.
- draw.io – for the UML diagram
- <http://tynerblain.com/blog/2007/04/09/sample-use-case-example/>