Technical University of Cluj-Napoca

Faculty of Automation and Computer Science

Semester 2021-2022

Fundamental Programming Techniques

- Laboratory Assignment no. 1 -

Polynomial calculator

Student: Maria Gliga-Hambeț

Group: 30423

Supervising teacher: Cristina Pop

# TABLE OF CONTENTS

1. Objectives ……………………………………………………………………………………………………3
2. Problem analysis, modelling, use cases ...……………………………………………………………………3
3. Design …………………………………………………………………………………………………….….5
4. Implementation ………………………………………………………………………………………………7
5. Results ……………………………………………………………………………………………………….10
6. Conclusions ………………………………………………………………………………………………….11
7. Bibliography …………………………………………………………………………………………………11
8. **Objectives**

The main purpose of this project is to design and later on implement a polynomial calculator with a user-friendly graphical interface, which does various operations, on one or two polynomials. The user can choose to insert only the first polynomial and execute the differentiation or integration of it or insert also the second one and perform the addition, subtraction, multiplication or division. There will also be the option for the user to delete all that he inserted in any of the polynomial text fields.

At the same time, considering the type of operation being chosen, the coefficient type will be displayed as integer or float, in order to provide the data as close to reality as possible.

In order to achieve the desired result, the whole project can be divided into the following, smaller tasks:

* Analyze the problem and identify requirements
* Design the polynomial calculator
* Implement the polynomial calculator
* Test the polynomial calculator, by means of JUnit testing

1. **Problem analysis, designing, scenarios, use cases**
   1. **Problem analysis**

The polynomial calculator should be able to do various operations on the inserted polynomials. In order to do that correctly, we firstly must be aware of what a polynomial is: a mathematical sequence of monomials. A monomial contains three main parts: the coefficient, the unknown value (marked by x) and the power to which it should be risen. Considering the fact that the operations are computed on more complex input than just numbers, a good understanding of how they work on polynomials is also mandatory. In addition to that, in order to achieve clear and correct results, the application should also at any point update the polynomial in terms of its monomials (deleting the ones with coefficient zero or joining the ones which have the same power). A special case must also be taken into account here and the point where the result should be zero (a monomial with coefficient zero and power one), as opposed to an empty polynomial. In terms of the division operation, the application should pay attention to the division by zero error and announce it to the user.

To achieve the above-mentioned description and functioning of the problem, we must take into consideration the following requirements:

* **Functional requirements:** 
  + The user should be able to input one or two polynomials, depending on the desired operation
  + The user should be able to select one of the six operations
  + The application should make the user aware of any errors (for example, the division by zero)
  + The user should be able to delete each polynomial by pressing a button, rather than doing it manually
  + The application should take the input and, by using pattern matching, turn the string into a list of smaller groups, out of which the coefficient and power of each monomial can be extracted
* **Non-functional requirements:** 
  + The user interface should be intuitive for the user
  + The polynomial calculator should be easy to maintainand further develop in the future
  1. **Scenarios and use cases**

When the simulation is started, the user inputs either one or two polynomials, depending on the wanted operation. For the differentiation and integration operation, the text field of the first polynomial should be completed. From than on, if he/she presses the “delete” button, present next to both the polynomial fields, the correspondent written input will be deleted. When pressing any of the operation buttons, even one after the other, the wanted results will be displayed on the screen.

**Use case 1 description**

**Title:** Polynomial operations on one polynomial

**Primary actor:** The user

**Main success steps:**

1. The user inputs the first polynomial;

2. The user presses the “integration” or “differentiation” button;

3. The selected operation is computed on the inserted polynomial and the result is displayed on the screen;

4. The user can now push another button of the above-mentioned ones and the new result will be displayed, without the need to introduce the polynomial again;

**Alternative sequences:**

2.1 The user decides to input a different polynomial and, in order not to delete it manually, in presses the “delete” button which is next to the text field;

2.2 The text field is emptied and the user can input the wanted polynomial.

**Use case 2 description**

**Title:** Polynomial operations on two polynomials

**Primary actor:** The user

**Main success steps:**

1. The user inputs both polynomials;

2. The user presses any of the two-polynomial operation buttons (subtraction, addition, multiplication or division);

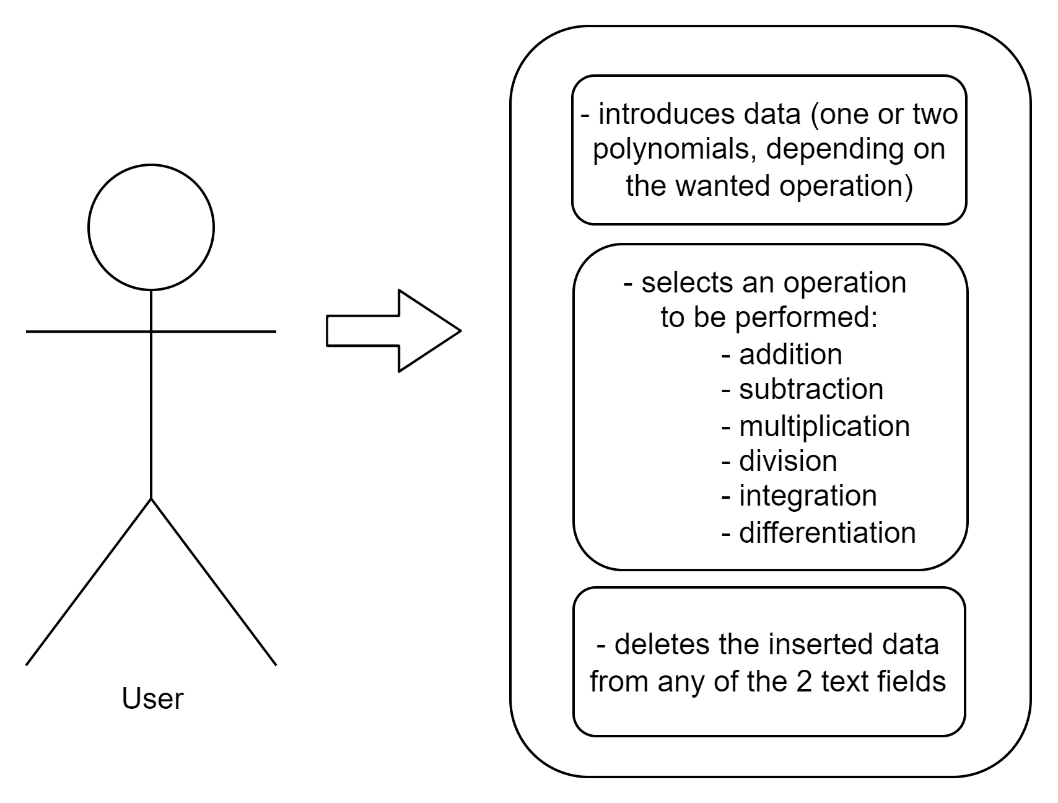
3. The selected operation is computed and the result is displayed on the screen. In doing the computation, in the case of the subtraction, the second polynomial is subtracted out of the first one. For the division, the polynomial with the greatest degree will be divided with the other one. Here, if it is the case of division by zero, an error will be displayed on the screen and the operation will not be performed;

**Alternative sequences:**

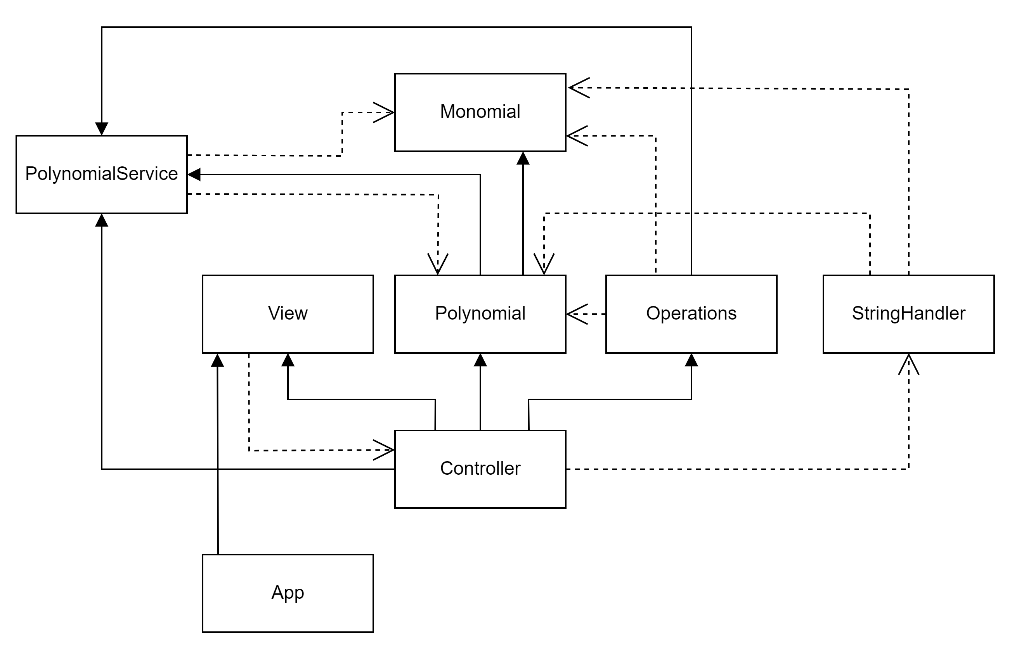
2.1.1 The user presses one of the buttons which corresponds to one-polynomial operations;

2.1.2 The use case becomes the one previously described, the fact that the second polynomial is inserted having no effect on the results;

2.2.1 The user decides to input a different polynomial and, in order not to delete it manually, in presses the “delete” button which is next to the text field;

****2.2.2 The corresponding text field is emptied and the user can input the wanted polynomial.

1. **Design**
   1. **Class diagram**

The following diagram mainly presents the relationship between classes. The individual variables and functions of each class are going to be presented as well, but later on when I will be describing the role of each class.

* 1. **Packages and relationship between them**

The architecture used in this project is a MVC (model-view-controller) one. Generally speaking, when talking about a MVC architecture, the roles of the three components are the following ones:

* The Model contains only application data and no data
* The View displays to our user the data given by the model. It, however, does not know what the data actually means or how the user can manipulate it
* The Controller is the one tying together the Model and the View. It is signaled by the view of certain events and executes certain actions, manipulating the data given by the Model. These actions are, in turn, displayed by the View.

In my particular case, the 3 packages consist of the following classes:

* The Model: classes Monomial and Polynomial. Since the polynomials and, of course, the monomials (out of which the polynomials are actually formed) are the main “actors” of the simulation, it seemed natural for their data to be the foundation of everything that I was designing.
* The View: class View. It is a class that extends JFrame and holds all the components of my graphical interphase. As it can be seen from above, it communicates with the Controller class, through the actions performed on the operation buttons.
* The Controller: here I have placed all the classes that manipulate data from the previously mentioned packages. It consists of: Operations, PolynomialService, StringHandler the Controller class.

In addition to the 3 packages mentioned above, I also added a App class which is used simply for running the simulation.

* 1. **Data Structures**
* An **ArrayList** is a resizable array. What makes it different from an array is that it does not have a given size which has to be altered at a certain moment. Internally, it uses a dynamic array to store the elements. I have chosen this data structures when working with data that did not need to be synchronized – the initially generated clients, for instance. I made use of the ArrayList when storing the monomials of a polynomial, as it made it easy to manipulate the data stored in the list (accessing elements, deleting multiple elements at once, checking to see whether or not and element is present).
  1. **Theoretical knowledge**

In order to achieve correct results, a good understanding of the way polynomials are built and the way operations are performed on them is essential. Therefore, the following things must be taken into account:

A polynomial P in an indeterminate X is formally defined as:

𝑃(𝑋) = 𝑎𝑛 ∗ 𝑋 ^ 𝑛 + 𝑎𝑛−1 ∗ 𝑋 ^ (𝑛−1) + ⋯ + 𝑎1 ∗ 𝑋 + 𝑎0, where 𝑎𝑛, 𝑎𝑛−1, …, 𝑎1, 𝑎0 represent the polynomial’s coefficients and n represents the polynomial degree.

A monomial is a special type of polynomial with only one term.

The addition and subtraction on polynomial should be done by grouping monomials with the same power and computing the result with the help of their coefficients. The multiplication implies that each term of the first polynomial will be multiplied with each term of the second one. This way, their coefficients will be multiplied and their powers summed up. The division is the most difficult operation and it follows the next algorithm (which shows the division between polynomials P and Q):

* **Step 1** - Order the monomials of the two polynomials P and Q in descending order according to their degree.
* **Step 2** - Divide the polynomial with the highest degree to the other polynomial having a lower degree (let’s consider that P has the highest degree)
* **Step 3** – Divide the first monomial of P to the first monomial of Q and obtain the first term of the quotient
* **Step 4** – Multiply the quotient with Q and subtract the result of the multiplication from P obtaining the remainder of the division
* **Step 5** – Repeat the procedure from **step 2** considering the remainder as the new dividend of the division, until the degree of the remainder is lower than Q.

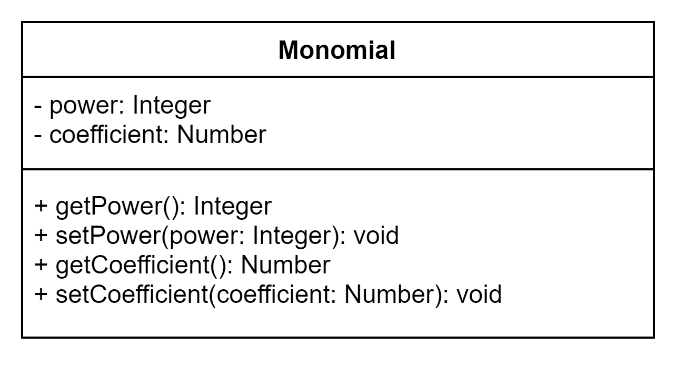
The differentiation and integration rules are the same as generally considered, with the polynomial being divided into its list of monomials.

1. **Implementation**

In this section, I will be discussing each individual class, its way of working and purpose in the whole application.

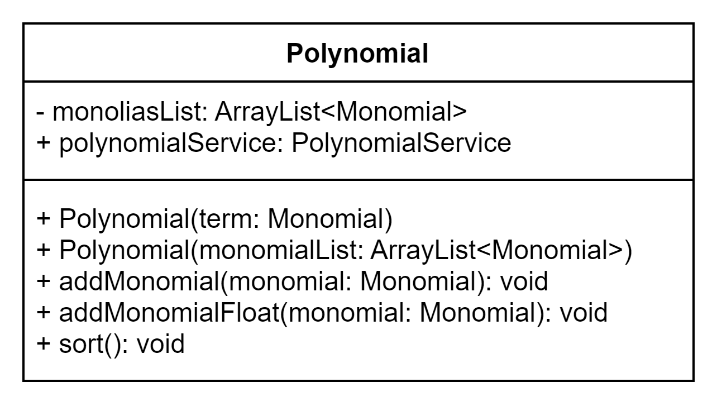
* 1. **Monomial Class** (part of Model):

The Monomial Class is, practically, one of the most important classes of the whole project as it holds the unit of data on which everything else is based upon (the polynomial, with its list of monomials, and the operations implemented for it). The attributes of the class are Coefficient and Power. Coefficient is of type Number in order to use it either as a float or as an integer, depending on the wanted case. Since for the power there are no cases in which it becomes a float, it is describe as integer. For both of the attributes, there are getters and setters.

****

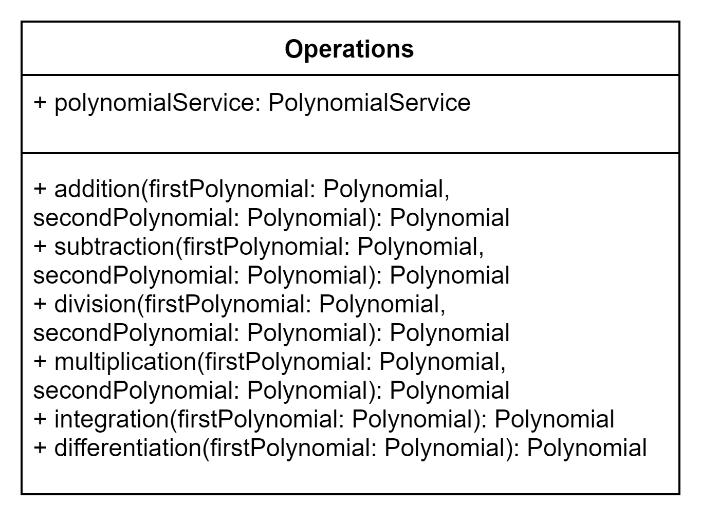
* 1. **Polynomial Class** (part of Model):

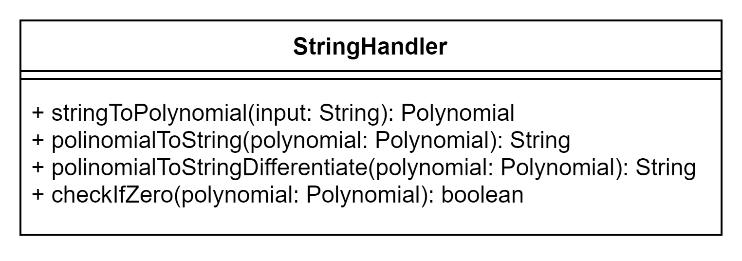
The Polynomial Class holds a role as important as the one of the Monomial Class. The attribute describing each polynomial is the list of monomials. There are 2 constructors for the polynomials, one through which it is given a list of monomials and one with only one monomial. The methods described in the class have the purpose of modifying the polynomial: adding new terms (with coefficient of type integer or float) and sorting the terms based on their power. It also has a getter for the list of monomials.



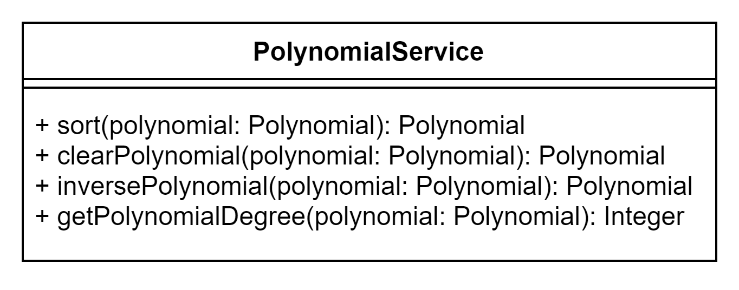
* 1. **Operations Class** (part of Controller):

The operations Class contains a method for each operation which the application implements.

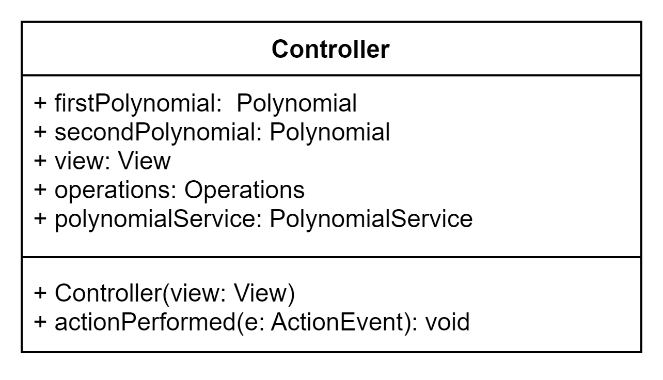
* **The addition operation**: firstly, inside the method there is a decision to see which of the polynomials to be added to the other one, in order to perform as few operations as possible. After that, using the addMonomial() the addition is performed. The polynomial is sorted once again and cleared in order to avoid terms with coefficients which are equal to zero. The final verification check whether at the end of the computation the polynomial is null or it. If it is, the result is given the value 0.
* **The subtraction operation**: the subtraction makes use of a function described in the polynomialService class, which returns a polynomial with all of its coefficients multiplied by -1. After that, the method calls the addition, basically adding the inverse of the second polynomial to the first one.
* **The multiplication operation**: the method multiplies each term of the first polynomial with each term of the second one. By doing so, the powers of every two monomials are summed and their coefficients are multiplied. The result is cleared and sorted and afterwards returned.
* **The division operation**: the method follows the algorithm described above, with the help of the subtraction and multiplications operations.
* **The differentiation operation**: the method goes through all the monomials of the polynomial and creates a new monomial (with the value of the coefficient = the value of the initial monomial’s coefficient multiplied with the value of the power; with the value of the power = the previous power - 1), which is added to the result.
* **The integration operation**: it goes the same method as the differentiation operation, but using the rules that polynomial integration follows.
  1. **StringHandler Class** (part of Controller):

The StringHandler Class’ purpose is to transform an inserted string into a polynomial and the other way around. The string to polynomial function uses pattern matching in order to extract groups of monomials out of the input. After that, the smaller extracted strings are gone through in order to find out the corresponding coefficient and power of the monomials. The polynomial to string method is implemented twice, depending on the wanted type of the coefficients. There is also a function which check to see if a polynomial is equal to 0.

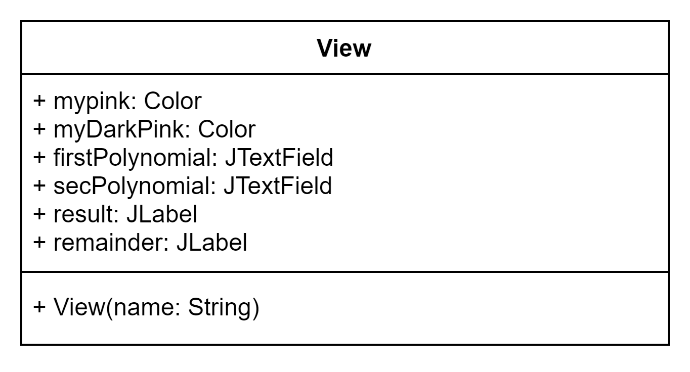
* 1. **PolynomialService Class** (part of Controller):

The PolynomialService Class handles the polynomial and does various operations on it. It consists of 3 methods: clear – which goes through the polynomial and deletes the monomials with coefficient 0, inversePolynomial – the method used in subtraction, in which all the coefficients of the polynomial are multiplied with -1 and getPolynomialDegree which returns the degree of the polynomial if the polynomial is not null and -1 in case it is.

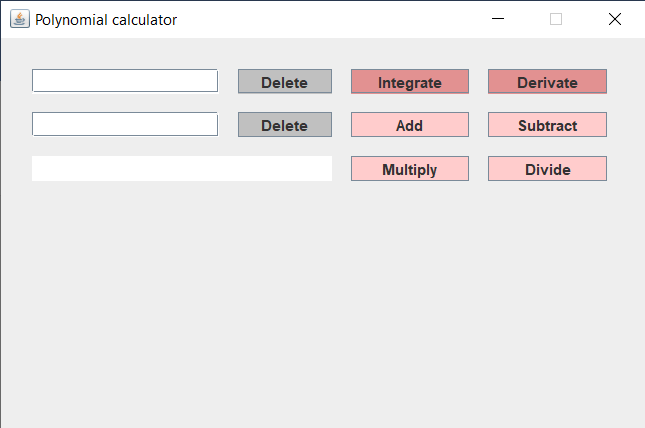
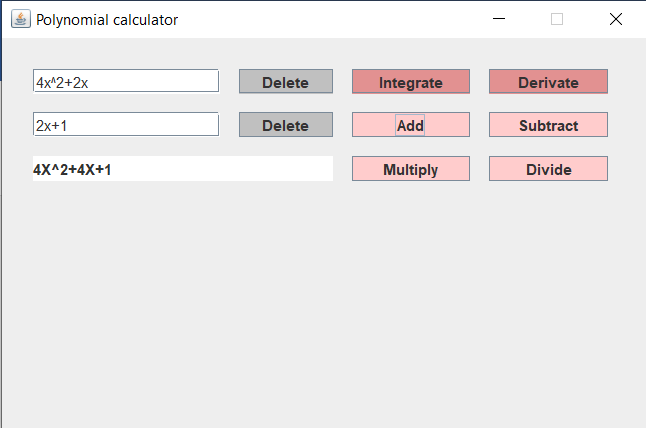
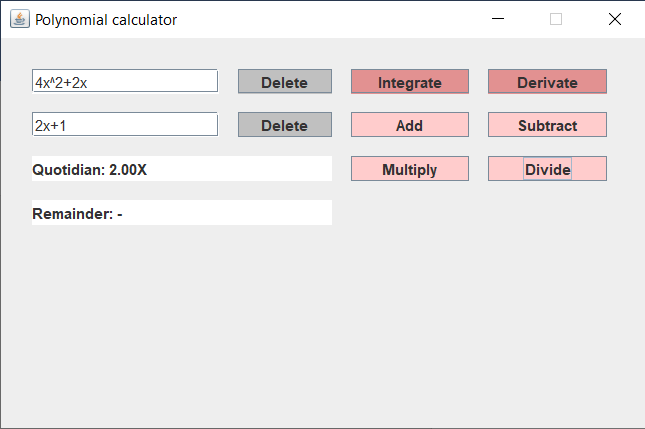
* 1. **Controller** **Class** (part of Controller):

****The Controller Class implements ActionListener and contains the steps that should be done at the push of each button: computing the right operation and displaying the result.

* 1. **View** (part of View):

****As previously said, the View Class holds all the components of the graphical user interface and, basically, describes it. It communicates with the Controller through the actions performed on the buttons.

1. **Results**

In order to test that the application was implementing the operation properly, I used the Junit framework. I added some screenshots of how the graphical user interface looks. The second picture shows the addition of the two polynomials, while the third one shows their division.

1. **Conclusions**

The project was a good opportunity to further improve the basic concepts of OOP. In the same time, it introduced me to new features such as JUnit testing and pattern matching. By using a model-view-controller architecture, my abilities of working in a cleaner and more grouped way were improved.

Future developments for the project:

* add the possibility to work with a greater number of polynomials
* improve the way the input is read and examined

1. **Bibliography**

* <https://app.diagrams.net/>
* <https://dsrl.eu/courses/pt/>
* <https://stackoverflow.com>
* <https://en.wikipedia.org/wiki/Polynomial_long_division>