Polynomial Calculator

By Bogdan Tudor-Alexandru

**1) Assignment Objectives**

Design and implement a polynomial calculator with a dedicated graphical interface through which the user can insert polynomials, select the mathematical operation to be performed and view the result.

Secondary objectives:

* Analyze the problem and identify the requirements
* Desigh the polynomial calculator
* Implement the polynomial calculator
* Test the polynomial calculator

**2) Analyze the problem and identify the requirements**

Considering that operating with big polynomials on paper is time consuming and it is easy to make mistakes, we want to design and implement a polynomial calculator which will allow the user to input the polynomials he has to operate with and select the operation to reduce the time to make the operation and to reduce eliminate the risk of making mistakes. By analyzing this we can define the functional and non-functional requirements.

Functional Requirements:

* The Polynomial calculator should allow the user to easily insert polynomials;
* The Polynomial calculator should allow the user to select which mathematical operation should be performed on the inputted polynomials;
* The Polynomial calculator should add two polynomials;
* The Polynomial calculator should subtract two polynomials;
* The Polynomial calculator should multiply two polynomials;
* The Polynomial calculator should divide two polynomials;
* The Polynomial calculator should integrate a polynomial;
* The Polynomial calculator should differentiate a polynomial.

Non-Functional Requirements:

* + The Polynomial calculator should be intuitive to use;
  + The Polynomial calculator should be easy to use.

A) **Use case**: add polynomials

**Primary Actor:** user

**Main Success Scenario:**

* The user inserts 2 polynomials in the graphical user interface.
* The user clicks on the “Add” button.
* The Result will be printed in place of the first polynomial for further use.

**Alternative Scenario:**

* In case of bad input a message will be shown and nothing will be done.
* The user will check the correctness of the polynomials and try again

B) **Use case**: subtract polynomials

**Primary Actor:** user

**Main Success Scenario:**

* The user inserts 2 polynomials in the graphical user interface.
* The user clicks on the “Substract” button.
* The Result will be printed in place of the first polynomial for further use.

**Alternative Scenario:**

* In case of bad input a message will be shown and nothing will be done.
* The user will check the correctness of the polynomials and try again

C) **Use case**: multiply polynomials

**Primary Actor:** user

**Main Success Scenario:**

* The user inserts 2 polynomials in the graphical user interface.
* The user clicks on the “Multiply” button.
* The Result will be printed in place of the first polynomial for further use.

**Alternative Scenario:**

* In case of bad input a message will be shown and nothing will be done.
* The user will check the correctness of the polynomials and try again

D) **Use case**: divide polynomials

**Primary Actor:** user

**Main Success Scenario:**

* The user inserts 2 polynomials in the graphical user interface.
* The user clicks on the “Divide” button.
* The Result will be printed in place of the first polynomial for further use.

**Alternative Scenario:**

* In case of bad input a message will be shown and nothing will be done.
* The user will check the correctness of the polynomials and try again

E) **Use case**: Integrate polynomials

**Primary Actor:** user

**Main Success Scenario:**

* The user inserts 1 polynomial in the graphical user interface in “Polynom 1”
* The user clicks on the “Integrate” button.
* The Result will be printed in place of the first polynomial for further use.

**Alternative Scenario:**

* In case of bad input a message will be shown and nothing will be done.
* The user will check the correctness of the polynomials and try again

F) **Use case**: Differentiate polynomials

**Primary Actor:** user

**Main Success Scenario:**

* The user inserts 1 polynomial in the graphical user interface in “Polynom 1”
* The user clicks on the “Differentiate” button.
* The Result will be printed in place of the first polynomial for further use.

**Alternative Scenario:**

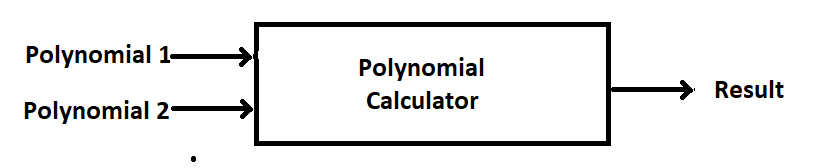
* In case of bad input a message will be shown and nothing will be done.
* The user will check the correctness of the polynomials and try again

**3) Design the polynomial calculator**

The Design will be implemented in 5 different levels.

**A) Level 1**

In the overall design the user will input 1 or 2 Polynomials into the Calculator which will print the result base on which operations the user selects (by pressing the button corresponding to the respective operation)

**B) Level 2**

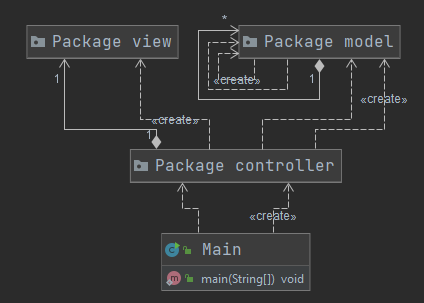
In the second level or design we will split our classes into different packages by using the Model-View-Controller architectural Pattern:

* + **Model components** - encapsulates the core data and functionality (polynomial and monomial classed)
  + **View components** - display information to the user and obtains data it displays from the model
  + **Controller** – This component will deal with receiving input and with the events from mouse (inserting the polynomials and selecting the operations)

**C) Level 3**

Each package will be split into classes. Base on the Mode-View-Controller Architectural pattern we will have:

* **Model Package**: Monomial and Polynomial classes.
* **View Package**: the Graphic User Interface class (GUI)
* **Controller**: the Controller class.



**D) Level 4**

Furthermore, each class will be divided into routines. Each class will have methods for getting or setting data (getters and setters) and constructors. The Specialized methods for each class are:

**Monomial**:

* addMonomial
* substractMonomial
* compareTo (needed for sorting the Polynoms based on coefficients)

**Polynomial:**

* addMonomialToPolynom
* substractMonomialFromPolynom
* substractMonomialFromPolynomWithoutRemove
* addPolynom
* substractPolynom
* substractPolynomWithoutRemove
* multiplyPolynom
* derivatePolynom
* integratePolynom
* divideByPolynomial
* copyPolynomial
* clearPolynomial
* polynomValidation
* generatePolynomToParse
* SortDescending

**Controller**:

* Constructor

**GUI**:

* + Constructor

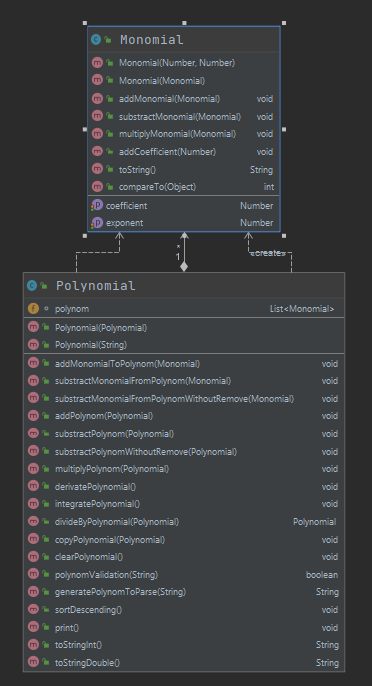
**E) Level 5**

For the Internal routine design (the implementation) please check the code for more details.

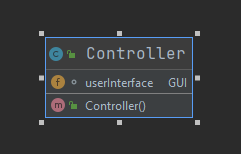
**Monomial**:

* **addMonomial** – adds a monomial over the give one by adding their coefficients
* **substractMonomial** – subtracts a monomial from the give one by subtracting their coefficients
* **compareTo** (needed for sorting the Polynoms based on coefficients) - compares 2 Monomials based on their exponents to determine the order for sorting.

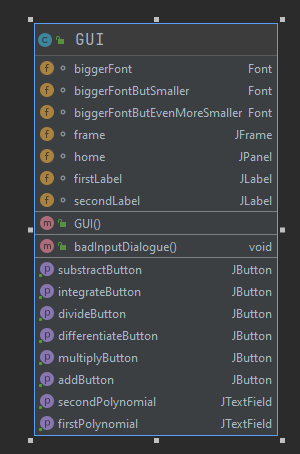
**Polynomial:**

* **addMonomialToPolynom** – it adds a Monomial to the given Polynomial. If a Monomial with the same exponent is found, the first Monomial is added to the found one and if the coefficient reaches 0 it is removed from the Polynomial. If a Monomial with the same exponent is not found we will add the Monomial to the Polynomials (insert in the list).
* **substractMonomialFromPolynom** - it subtracts a Monomial from the given Polynomial. If a Monomial with the same exponent is found, the first Monomial is subtracted to the found one and if the coefficient reaches 0 it is removed from the Polynomial. If a Monomial with the same exponent is not found we will add the Monomial to the Polynomials (insert in the list) with a minus Coefficient.
* **substractMonomialFromPolynomWithoutRemove** - it subtracts a Monomial from the given Polynomial. If a Monomial with the same exponent is found, the first Monomial is subtracted to the found one. If a Monomial with the same exponent is not found we will add the Monomial to the Polynomials (insert in the list) with a minus Coefficient.
* **addPolynom** – we add each of the Monomials from the second Polynomial to the given Polynomial (this).
* **substractPolynom** - we subtract each of the Monomials from the second Polynomial from the given Polynomial (this).
* **substractPolynomWithoutRemove** - we subtract each of the Monomials from the second Polynomial from the given Polynomial (this) using the substractMonomialFromPolynomWithoutRemove method.
* **multiplyPolynom** – we multiply each of the Monomials from the first Polynomial with each of the Monomials from the second Polynomials and add them to an auxiliary Polynomial result, which will be returned by the method at the end of it’s execution
* **derivatePolynom** – we multiply each monomials coefficient by it’s exponent, decrease the exponent by 1 or remove the Monomials if the exponent is 0 before decreasing
* **integratePolynom** – we divide each Monomial’s coefficient by it’s exponent + 1 and increse the exponent by 1
* **divideByPolynomial** – we will multiply the second polynomial with a Monomial such that by subtracting it from the first Polynomial we will reduce the most significant term. The Monomial that we multiplied with will be added to the Quotient Polynomial which will be put instead of the first Polynomial at the end of the method. We will continue reducing the most significant term until the grad of the first polynomial is small ther the second’s one. The Polynomial that we get in the end in the first Polynomial will be the Remainder and will be returned by the method.
* **copyPolynomial** – we copy the Monomials from second Polynomial (given as parameter) to the first one (this).
* **clearPolynomial** – we empty the Monomial list of the given Polynomial (this)
* **polynomValidation** – we check if the Polynomials given by the user and generated by our function generatePolynomialToParse can be a Polynomial and if yes the constructing of the Polynomial will begin in the Constructor. If not an exception is thrown.
* **generatePolynomToParse** – we modify the strings obtained from the Polynomials inputted by the user by removing all “\*” and “^” and whitespaces and by adding a “+” at the beginning of the Polynomial if the first term is not negative. Like this it will be easier to split the String intro groups and capturing them using Regular Expressions.
* **SortDescending** – We sort the Polynomial Descending based on the exponent’s of the Monomials
* 

**Controller**:

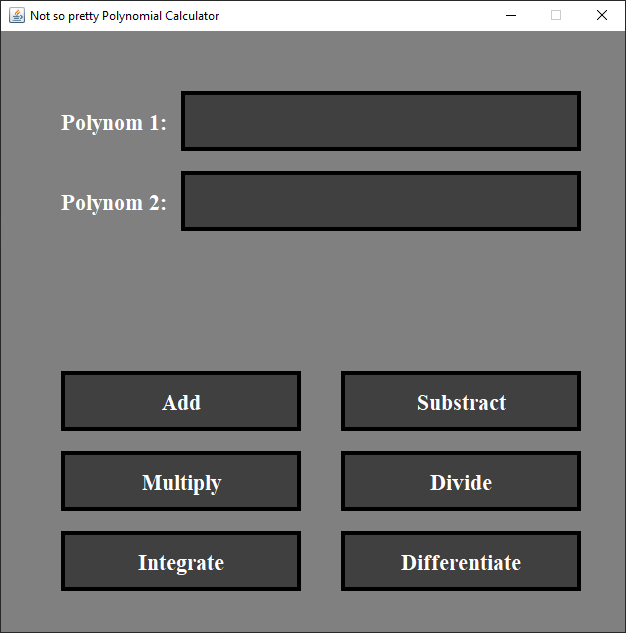
* **Constructor** – Here the Functionality of the Interface is implemented: we retrieve the Polynomials from the Textboxes, try to construct the Polynomials from the retrieved strings and try to perform the operations. In case of bad input (the impossibility of creating the Polynomial) the badInputDialog method is called and a pop-up will warn the user of the mistake he made.
* 

**GUI**:

* + **Constructor** – The Application will have 2 Text Boxes and 6 buttons in a 640 x 640 pixels window. In the Textboxes the user will insert the Polynomials as indicated by the Labels after which he will press on the button corresponding to the operation he wants to be performed on the Polynomials.
  + **badInputDialog** – this method will pop-up a dialog to warn the user of bad input in case it happens.
  + 

**4) Implementation**

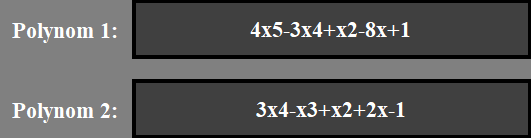
Since the description and inplementation of the classes has been done in the previous part I will just present the Graphical user interface and the Reguler Epxression used for validation and breaking the strings into Monomials here:

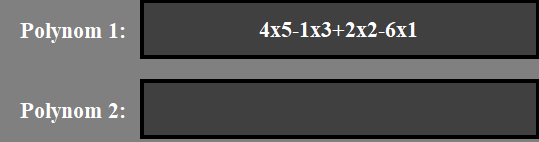
 The Main Frame has a dimension of 640 x 640 pixels on which a panel with the same dimension is laid. The panel and the frame can be considered split into squares of 10x10 pixels (we will have 2 variables divH and divW which will hold this information) and which we will use to make a better organization and a prettier layout in the frame. Also, on the panel we will have 2 labels to tell the user which Textbox is for which Polynomial, the 2 Textboxes for which the Polynomials are made and the 6 buttons to select the operation to be performed by the application.

Regarding the Regex part, I used Regex for validating the Polynomial and for splitting the Polynomial string into subgroups that fits into Monomials. Using the patter "[+-](?<coefficient>\\d+)?(?<var>[x])(?<exponent>\\d+)?" I first check if my String is formed only from groups that start with a sign (+ or -), if they have a number (that si going to be the coefficient, it might not be present which means I will make it equal to 1), THEY MUST HAVE VARIABLE X (for the free term has been appended at the end the combination “x0”, which mean that the free term must be the last one in the inputted Polynomials), and the last one is the exponent which is also a number (and might also not be present which means it is equal to 1). Also the signs “\*” are “^” optional in the Polynomial description as string since they will be removed first. This pattern will check that this combination of sign + coefficient + X + exponent repeats and nothing else is present. If the string matches this pattern it is a good polynomial. For capturing the Monomials we will use the same one,but without the “?” in the end so tat we can match 1 Monomial at a time and add it into the Polynomial.

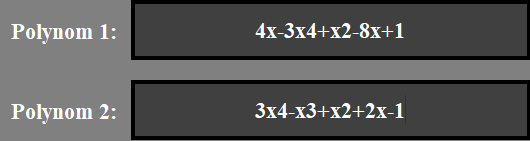
**5) Results**

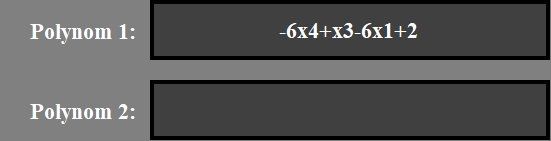
A)Add



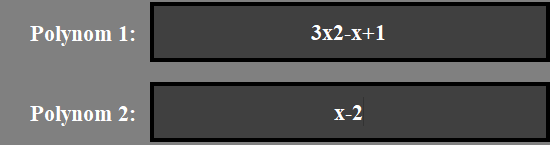


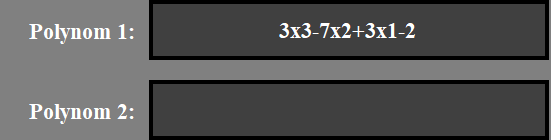
B) Subtract



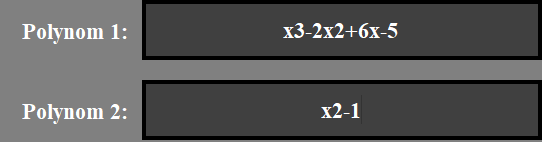


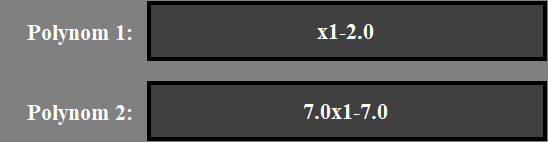
C)Multiply

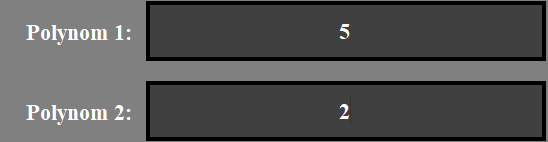


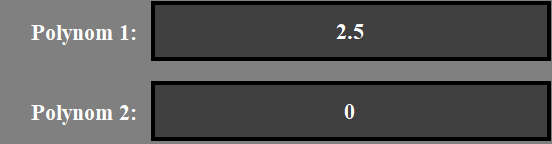


D) Divide

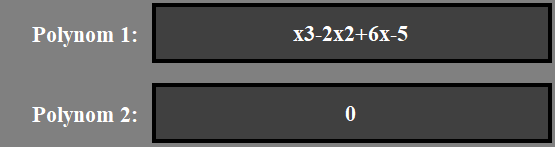
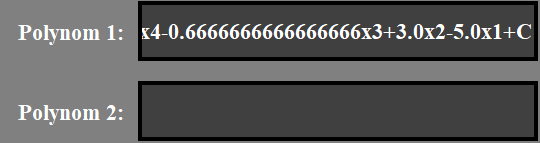




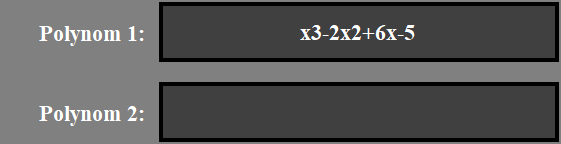


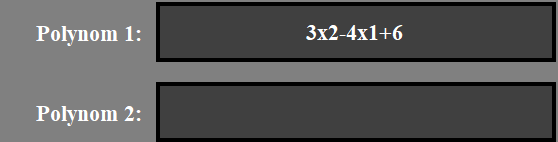


E) Integrate

F) Differentiate





**6) Conclusions**

This application was an interesting one and I got the opportunity to learn more about OOP and to deepen my knowledge in this field. I also got the opportunity to use the knowledge that I got in the first semester and to implement a project based on what I learned. I think that polynomials are an important part in our society and they are used in many fields, which gives me the opportunity to further develop this project for using it into other applications. I also got the opportunity to deepen my knowledge in the mathematical field.

**7) Bibliography**

The Assignment Support Presentation

<https://www.youtube.com/watch?v=Kmgo00avvEw>

<https://www.youtube.com/watch?v=0yQG0mNV4hY&t=386s>

Code developed in the first semester

https://regex101.com/

https://docs.oracle.com/javase/tutorial/uiswing/

https://docs.oracle.com/javase/tutorial/uiswing/

https://docs.oracle.com/javase/tutorial/essential/regex/index.html

<http://tutorials.jenkov.com/java-regex/matcher.html>

https://docs.oracle.com/javase/tutorial/essential/regex/groups.html