Programming Techniques

Laboratory - Assignment 1

Polynomials Calculator

Teacher: prof. Ioan Salomie

Teacher Assistant: Ciprian Stan

Student: Moșolea Victor-Andrei

Group: 30422

1) Objective

The objective of this assignment is to propose, design and implement a system used for polynomial computing. We consider the polynomials to be of only one variable and having integer coefficients.

2) Dimensions of the problem

a) Analyzing the problem

In mathematics, 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.

A polynomial in a single indeterminate x can always be written (or rewritten) as

$$\sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

where a_i, $i = \overline{0,n}$ are constants and x is a symbol with no particular value, although it can be substituted for any value whatsoever.

Every polynomial consists of a list of terms called *monomials* (e.g. $2x^3$ is a monomial having 2 as coefficient and 3 as exponent). Summing several monomials results in a *polynomial* (e.g $x^2 + x - 1$). The previously mentioned polynomial can be represented as the list ([1, 2], [1, 1] , [-1, 0]) where each item represents a monomial [i, j] where i is the coefficient and j is the exponent.

Using monomials we perform operations such as addition, subtraction, division, multiplication, differentiation, and integration much easier.

b) Modelling the problem

The user will be able to input one or two polynomials in the reserved text fields. Depending on the chosen operation the input will then be checked and parsed. If the operation only includes the first polynomial, the text in the second input field will be ignored and vice versa. The polynomial calculator is able to perform the following operations:

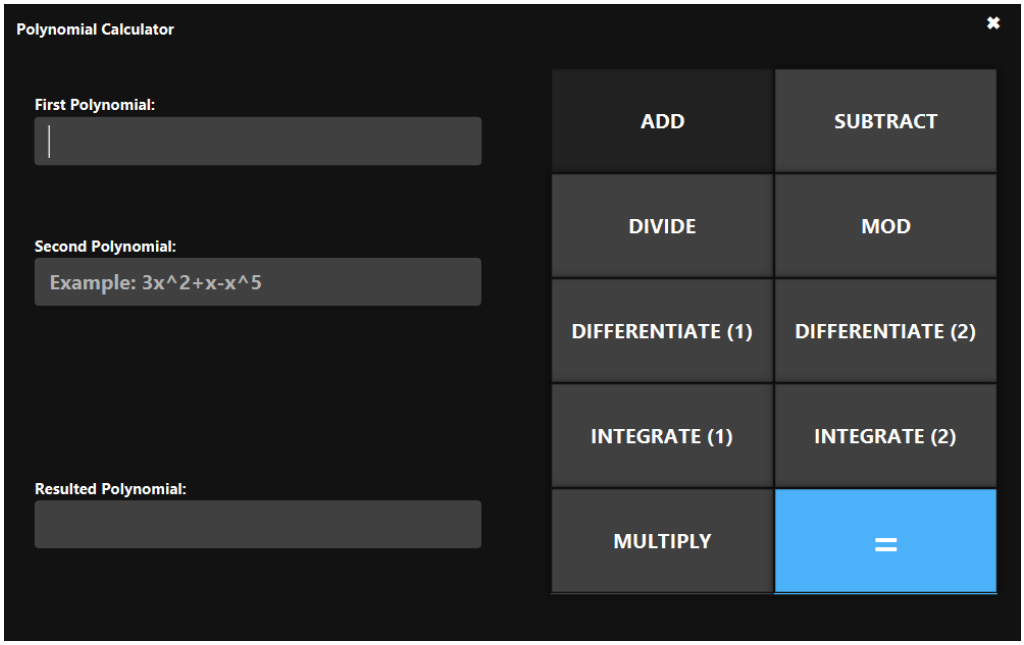
- Addition of two polynomials
- Subtraction of two polynomials
- Multiplication of two polynomials
- Division of two polynomials
- Differentiation of one of the two input polynomials
- Integration of one of the two input polynomials

After inputting the polynomials and choosing the operation, the user can then press the "=" button in order to see the result. If the input required for that specific operation is incorrect, the user will be prompted with an error message.

c) Different scenarios and use cases

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal.

The use cases and the user steps are strongly connected and it is important to design an user interface that takes this into account. This is exactly what I have tried to do and the result is the following:

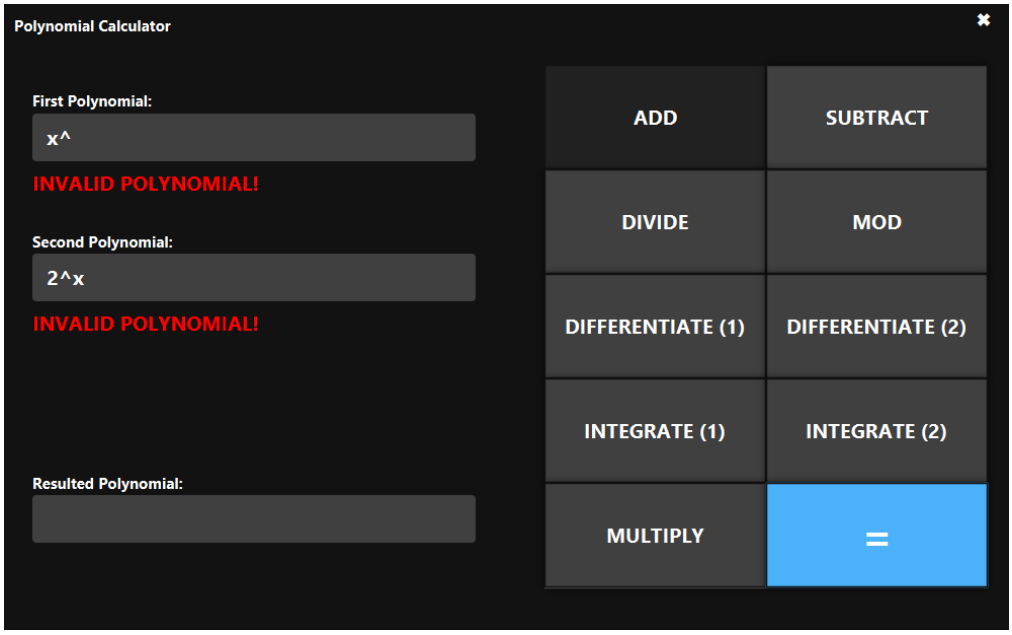


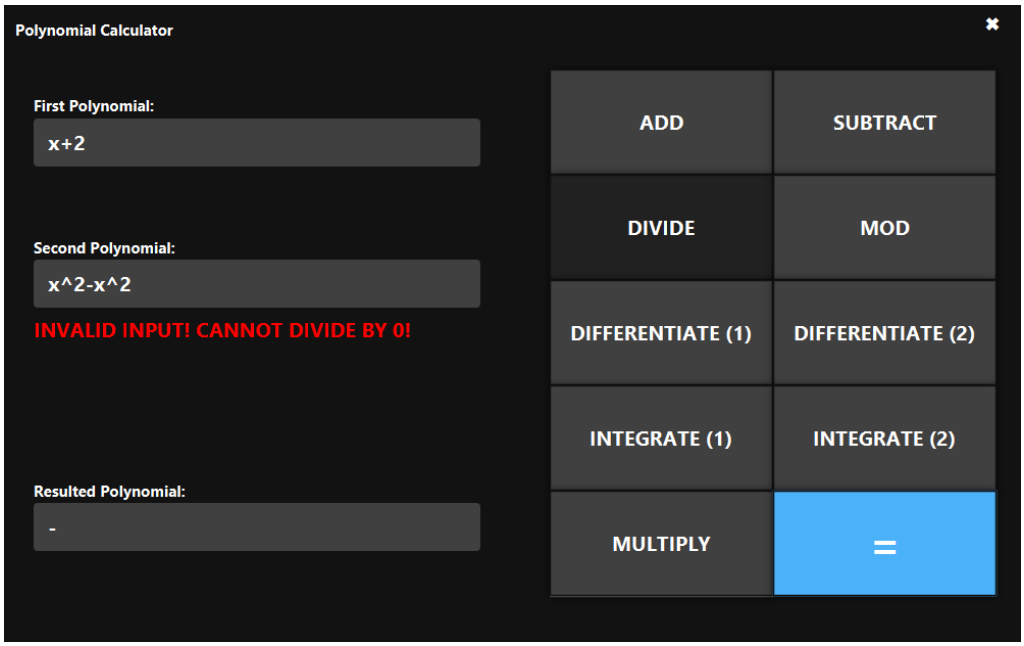
The user will introduce the two polynomials in the corresponding text fields. The input will be checked and parsed after the user chooses the operation and presses the blue button in the lower right corner. If the input is not correct an error prompt will appear under the text field that contains an incorrect input.

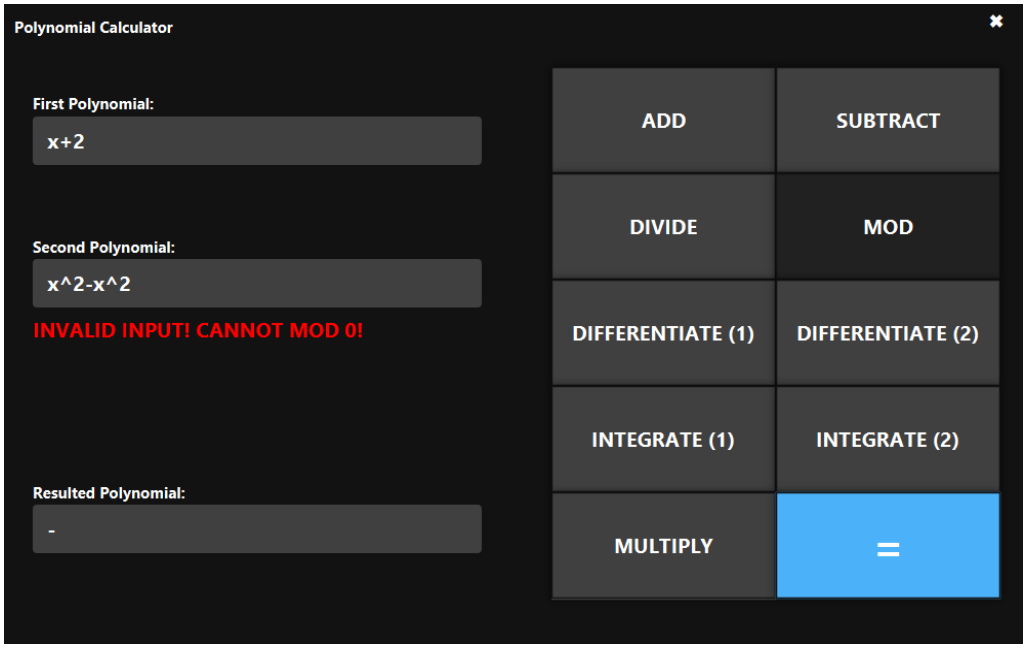
The user must pay attention to follow the format of the polynomial, which can be seen as prompt text in both input text fields.

The correctitude of the input doesn't depend only on the correctitude of each polynomial, but it also takes into account the chosen operation. For instance if the user chooses to divide or find the rest of the division between two polynomials, 0 or a polynomial which is equal to 0 such as $x^2 - x^2$ will not be accepted as an input for the second polynomial and therefore an error prompt will appear under the second input text field.

Here are some examples of wrong inputs and error prompts:



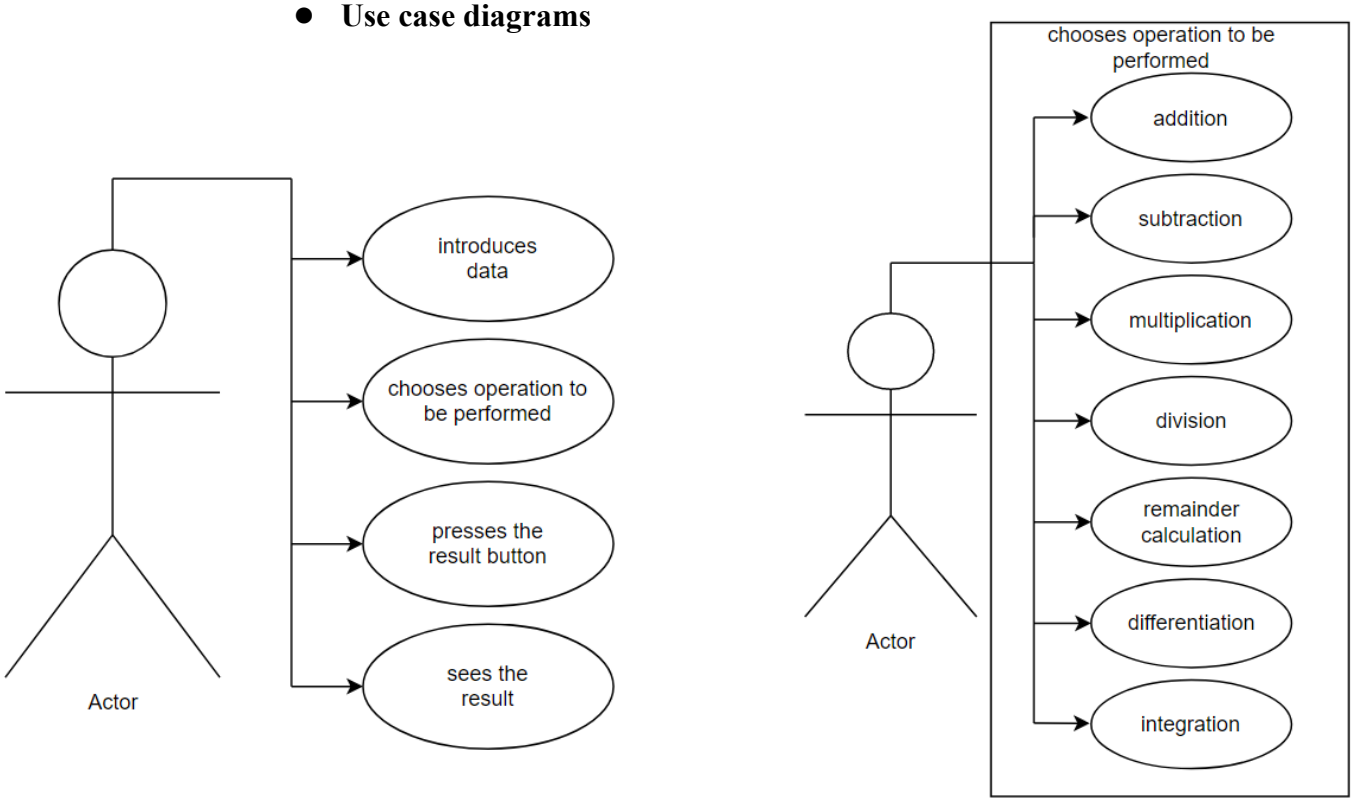

Polynomial Calculator
First Polynomial:
x+2
Second Polynomial:
x^2-x^2
INVALID INPUT! CANNOT DIVIDE BY 0!
Resulted Polynomial:
-
ADD
SUBTRACT
DIVIDE
MOD
DIFFERENTIATE (1)
DIFFERENTIATE (2)
INTEGRATE (1)
INTEGRATE (2)
MULTIPLY
=


Polynomial Calculator
First Polynomial:
x+2
Second Polynomial:
x^2-x^2
INVALID INPUT! CANNOT MOD 0!
Resulted Polynomial:
-
ADD
SUBTRACT
DIVIDE
MOD
DIFFERENTIATE (1)
DIFFERENTIATE (2)
INTEGRATE (1)
INTEGRATE (2)
MULTIPLY
=

3) Design

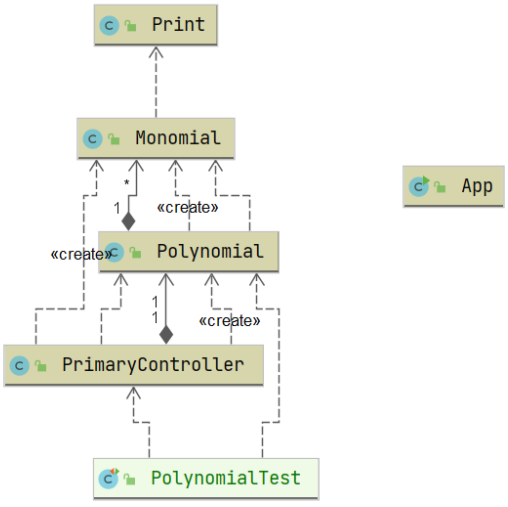
a) Diagrams

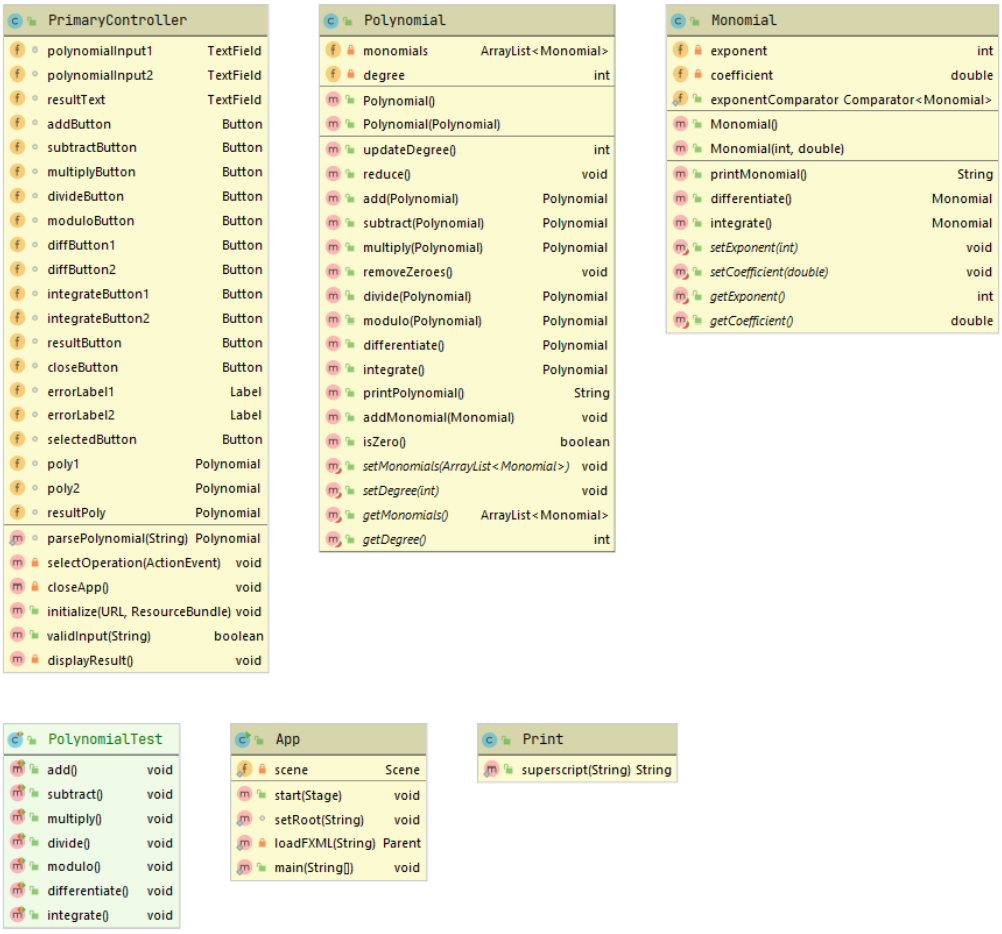
- **Use case diagrams**



The use case presents the actor which in our case is the user that is interacting with our application. He/She can perform several operations, some which require one or both polynomials read from the input. All the possible operations can be seen in the above use case diagram on the right of the page.

- **Class Diagram**





b) Data Structures

The data structures with which I have worked for this project are mostly primitives, such as integers and doubles, java objects such as String, StringBuilder, DecimalFormat, and newly created objects such as Monomials and Polynomials. I have also decided to use ArrayList instead of the classic arrays because I think that they are more efficient from the point of view of memory management and performance. Though the main advantage of ArrayLists over classic arrays is memory management as the size of a list is not fixed and depends only on the number of elements.

c) Class design

The purpose of splitting a program into several classes corresponds to splitting a bigger problem into smaller subproblems which are obviously easier to solve. Solving these small subproblems results in the solving of the general larger problem. A popular way of splitting a program into classes is the MVC (Model - View - Controller) architecture.

I. **The Model -** contains all the logic of the application

The Model is composed of the following classes: Monomial, Polynomial and PolynomialTest.

II. **The View -** is responsible for the GUI

It contains only one file called **primary.fxml** which was built using scene builder and using a stylesheet called **styles.css**

III. The Controller - is links together the model and the view

This is the most important class as it ensures the functionality of the application. It controls the data flow into the objects inside the model and updates the view whenever neccesary.

This class is also present in my program as PrimaryController.

d) Algorithms

- **addition**

The sum of two polynomials is obtained by summing the coefficients sharing the same exponent. I've chosen to do this by sorting the monomials of both polynomials by their exponent and traversing each one of them while checking if their exponents match and adding their coefficients if this is the case.

$$(ax^2 + bx^3 + cx) + (dx^2 + ex^4) = (c + 0)\,x + (a + d)\,x^2 + (b + 0)\,x^3 + (e + 0)x^4$$

- **subtraction**

The subtraction is basically equivalent with reversing the sign of all coefficients of the second polynomial and then adding it to the first one and this is exactly how I have chosen to implement this operation.

$$(ax^2 + bx^3 + cx) - (dx^2 + ex^4 + fx) = (c - f)\,x + (a - d)\,x^2 + (b - 0)\,x^3 + (e - 0)x^4$$

- **multiplication**

The multiplication of two polynomials is obtained by just multiplying term by term the monomials of the two polynomials

$$(ax^2 + b) * (cx + d) = acx^3 + adx^2 + bcx + bd$$

- **division**

For the division I have decided to implement the polynomial long division algorithm.

- **modulo**

The previously mentioned algorithm returns not only the quotient but also the remainder of the division.

- **integration**

Integration of a single monomial is done by incrementing the exponent and then dividing the coefficient by the new value of the exponent. Integration of a polynomial is done by integrating each monomial individually.

$$(ax^n)' = \frac{a}{n+1}\,x^{n+1}$$

- **differentiation**

Differentiation of a single monomial is done by multiplying the coefficient by the exponent and then decrementing the exponent. Same as for integration, we get the result of this operation by differentiating each monomial individually.

$$\int ax^n = nax^{n-1}$$

e) Graphical User Interface

The Graphical User Interface's purpose is to connect the user with our application. The user can input data (one polynomial is required for differentiating and integrating and two polynomials for every other operation) The user must keep in mind that the input must respect a few conditions. Although it is easy to respect them as they're intuitive and the prompt texts contain easy to understand examples.

4) Implementation

a) Monomial Class

As I have stated previously, a polynomial is composed of several monomials. This class contains two fields: exponent (`int`) and coefficient (`double`). It also contains a comparator that is used for sorting a list of monomials.

Constructors:

- `public Monomial()` - this is the default constructor which initializes all class fields with 0
- `public Monomial(int exponent, double coefficient)` - this constructor initializes the class fields with the method's parameters.
-

Methods:

- `public String printMonomial()` - returns a `String` which represents the printed monomial
- `public Monomial differentiate()` - returns a new Monomial which is the differentiated this monomial object.
- `public Monomial integrate()` - returns a new Monomial which is the integrated monomial object.

The class also contains setters and getters but they have been generated automatically using Lombok.

b) Polynomial Class

This class contains two fields: an `ArrayList` of Monomials and a degree (`int`).

Constructors:

- `public Polynomial()` - returns a new Polynomial
- `public Polynomial(Polynomial newPol)` - returns a new Polynomial having the same monomials as newPol

Methods:

- `public int updateDegree()` - updates the degree of the Polynomial and returns it
- `public void reduce()` - adds all monomials which have the same exponent resulting in a reduced form of the polynomial
- `public void removeZeroes()` - gets rid of all monomials having 0 as a coefficient
- `public Polynomial add(Polynomial b)` - returns the result of `this` Polynomial + b
- `public Polynomial subtract(Polynomial b)` - returns the result of `this` Polynomial - b
- `public Polynomial multiply(Polynomial b)` - returns the result of `this` Polynomial * b
- `public Polynomial divide(Polynomial b)` - returns the quotient of `this` Polynomial / b

- `public Polynomial modulo(Polynomial b)` - returns the remainder of `this` Polynomial / b
- `public Polynomial differentiate()` - returns the result of differentiating `this` Polynomial
- `public Polynomial integrate()` - returns the result of integrating `this` Polynomial
- `public boolean isZero()` - return true if polynomial is equal to 0
- `public void addMonomial(Monomial m)` - adds m to the list of monomials
- `public String printPolynomial()` - returns the String representation of `this` Polynomial

c) Print Class

This class has only one method and no fields as it is an utility class that I use for printing the exponents of monomials.

d) PrimaryController Class

This is the most important class of this project as I have stated previously. It has many fields, most of which have the **FXML** tag as they're also part of the user interface. The only fields that work in the background are Button selectedButton, Polynomial poly1, Polynomial poly2, Polynomial resultPoly.

Methods:

- `static Polynomial parsePolynomial(String input)` **-** it takes as input a String and returns the equivalent polynomial.
- `private void selectOperation(ActionEvent event)` **-** it handles the event of choosing an operation and it sets the selectedButton variable and changes the style in order for the user to know which button they have selected.
- `private void closeApp()` - it closes the app whenever the user requests it.
- `public void initialize(URL url, ResourceBundle resourceBundle)` - initializes the selectedButton when the application is opened.
- `public boolean validInput(String input)` - using regex, it checks if the input String represents a valid polynomial
- `private void displayResult()` - this method is responsible for executing the selected operation and displaying its result if the input is correct or setting the error labels otherwise.

e) App Class

This class contains the **main** method and it is basically responsible for opening the app using the **start** method which loads the **primary.fxml** file into a new scene and initializes and shows the stage

5) Results

Tested operation	Input Data	Expected Output	Actual output	Pass/Fail
Addition	6x^3+20x-10x^2+2; 10x^4+10x+15x^2+5	$10x^4+6x^3+5x^2+30x+7$	$10x^4+6x^3+5x^2+30x+7$	Pass
Subtraction	20x^6-10x+15x^4+2; -x^5+x^7-10x^6-15x^4+2	$-x^7+30x^6+x^5+30x^4-10x$	$-x^7+30x^6+x^5+30x^4-10x$	Pass
Multiplication	10x+20x^2-5-2x^3; 13x^2-20	$-26x^5+260x^4+170x^3-465x^2-200x+100$	$-26x^5+260x^4+170x^3-465x^2-200x+100$	Pass
Division	x^2+x+2; x+1	x	x	Pass
Modulo	13x^9+11x^4; x^2+1	13x+11	13x+11	Pass
Differentiation	13x^9+11x^4;	$117x^8+44x^3$	$117x^8+44x^3$	Pass
Integration	1+2x+62x^3+10x^2	$15.5x^4+3.33x^3+x^2+x$	$15.5x^4+3.33x^3+x^2+x$	Pass

.

For testing I used the JUnit Framework.

6) Conclusions

To sum up, I think that this project was a great practice for my OOP skills acquired in the first semester of this school year and, of course, improving them. I have also gained more experience using JavaFx. This project also enabled me to learn new things such as testing with JUnit, developing a project in Maven and plenty of other stuff.

Another important aspect that I have learned is the importance of time management. Every time I thought I was done with this project I found new edge cases which my application did not handle properly, and testing it proved to be truly useful.

This project also confirmed to me that the best way of learning something is by just trying to implement something right away and finding out new things about the tools you need on the way. And I also finally understood how much time planning can save.

As everything else in the world, this project is not perfect and there are many improvements that can be made. Here are some that I thought of:

- Computing more complex operations using multiple polynomials at the same time. For instance: P1*P2+P3 (where P1, P2, P3 are all polynomials).
- Plotting the graph of a polynomial
- Finding the roots (both real and complex)
- Calculating the value of a polynomial in a certain point.

7) Bibliography

- Object-Oriented Programming - Lecture Slides of prof. Marius Joldos
- Programming Techniques - Lecture Slides of prof. Ioan Salomie
- www.stackoverflow.com
- en.wikipedia.org/wiki/Polynomial_long_division
- www.draw.io