

Technical University of Cluj-Napoca

Programming Techniques

Laboratory – Assignment 1

# Polynomial Calculator



**UNIVERSITATEA TEHNICĂ**  
DIN CLUJ-NAPOCA

**Laboratory Assistant:**

Ing. Ciprian Stan

**Student:**

Enoiu Diana-Cristina

## 1. Assignment objective

The objective of this assignment is to implement a Polynomial Calculator in Java using fundamental programming techniques. The calculator should be able to perform basic operations (addition, subtraction, multiplication etc.) on two given polynomials inserted by the user.

The provided polynomials have one variable and integer coefficients.

## 2. Analysis

### a) Problem analysis

By definition, a polynomial is an expression that can be built from constants and symbols called variables or indeterminates by means of addition, multiplication and exponentiation to a non-negative integer power. An example of a polynomial is:  $3x^2 - 5x + 7$ .

A polynomial with a single variable  $x$  can always be written in the form:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_0,$$

where  $a_0, \dots, a_n$  are the coefficients.

A monomial is described, in mathematics, as a polynomial with only one term. An example of a monomial is  $6x^7$  (coefficient is 6, variable is  $x$  and the degree is 7).

So, we can view a polynomial as being the sum of one or more monomials of the form  $a_i x^i$ , with  $i$  being the degree of the monomial and  $a_i$  the coefficient.

This way of representing a polynomial can be used to compute different mathematical operations (as *additions*, *multiplications*, *divisions* etc.) and will make the overall design and implementation of our application easier and more efficient.

### b) Modeling

The user will be able to use the calculator by introducing in the interface, two polynomials. The polynomials must respect some format restrictions in order, for the application, to be able to perform different mathematical operations.

Some of those format restrictions consist of these rules:

- Enter polynomials in the form:  $4x^3 - 2x^2 + x - 5 + 6x^{-2}$ .
- Do not use spaces, parenthesis or the  $*$  operator.
- Polynomials do not have to be sorted or simplified.
- For derivative and integral operations, enter the polynomial in the first field.

The user can perform any operation by pressing the corresponding button (“+” for addition, “-” for subtraction, “\*” for multiplication, “/” for division, “d/dx” for derivation and “[dx]” for integration).

At the end, the result will be displayed next to the result label.

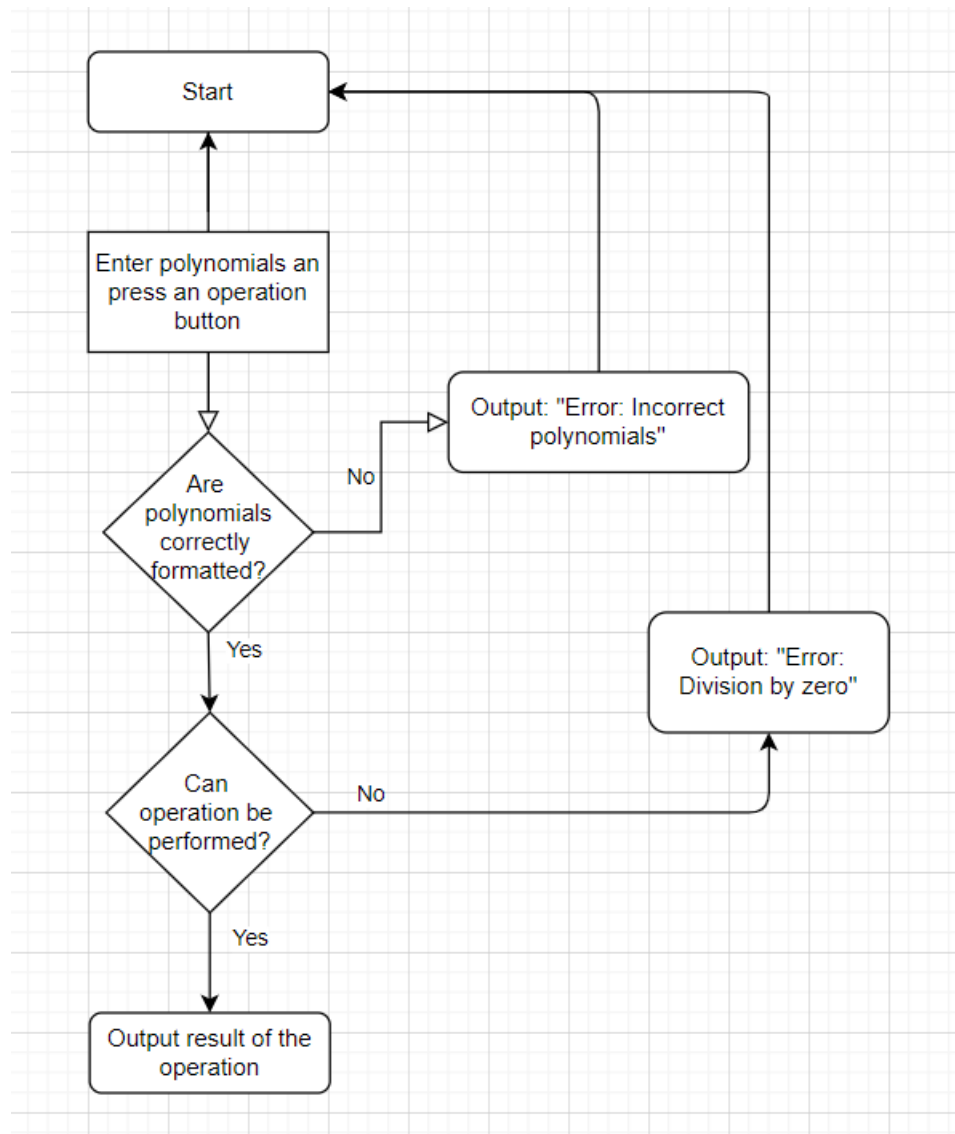
c) *Different scenarios and use cases*

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 are correlated with the steps the user makes when using the application.

The application should have a user friendly interface in order for the user to understand faster what are the steps that are needed to be performed in order to achieve the desired result.

For this application, the flow-chart from below is describing the different ways the application will behave to different user inputs.



### 3. Design and Implementation

#### a) Design decisions

This application uses an object-oriented programming design. The application has a graphical user interface made in JavaFX, with the help of Scene Builder.

For the design of this application it was used, also, an architectural pattern, the MVC pattern, (Model-View-Controller pattern), which is used to separate application's concerns.

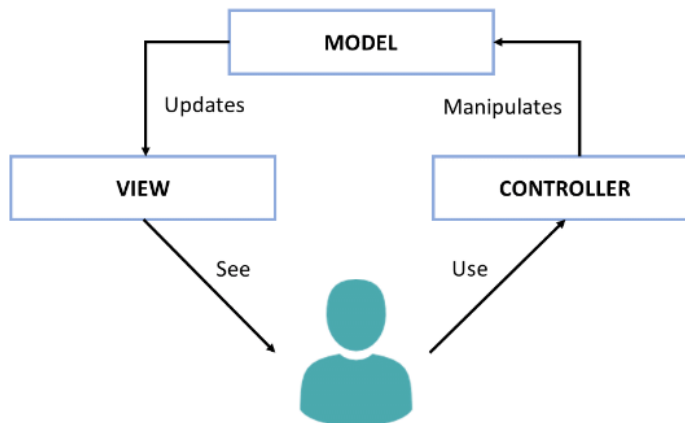
The Model-View-Controller is a software design pattern commonly used for developing user interfaces that divides the related program logic into three interconnected elements.

The *model* is responsible for managing the data of the application. It receives user input from the controller.

The *view* renders presentation of the model in a particular format.

The *controller* responds to the user input and performs interactions on the data model objects.

The controller receives the input, optionally validates it and then passes the input to the model.



This application's classes have, thus, been split into four main packages:

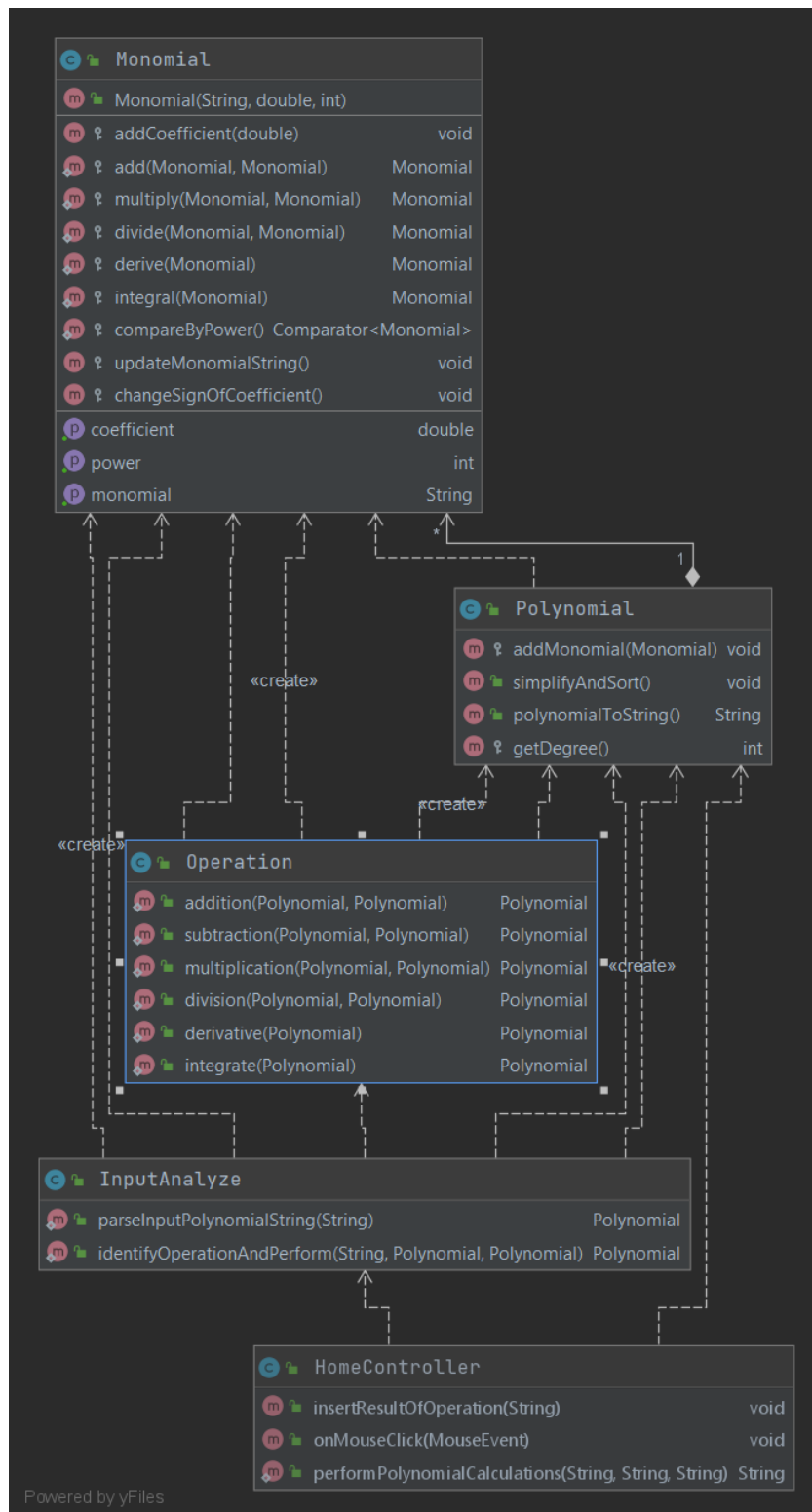
- The application package which contains the Main class;
- The model package which has the Polynomial, Monomial used in the modelling of the program, Operation class for mathematical computations and InputAnalyzer for parsing the data;
- The view package containing .fxml, .css files used for displaying the actual interface;
- The controller package, which manages the view and model packages, contains the HomeController class.

#### b) Data structures

For this application the data structures that were used are primitive data types (integers, doubles), non-primitive data types (Strings), more complex data types such as ArrayLists and new created object such as Monomial and Polynomial.

An ArrayList instead of the classic arrays implementation is more efficient from the point of view of memory management, performance and provide a faster access to their content. Also, the size of an ArrayList is not fixed and adding new elements to the list is very easy because we do not have to worry about exceeding the "length" of the list (array).

c) UML class diagram



#### d) Class design

Because of the MVC architecture, this program consists of 4 parts:

##### 1) **The Model** – contains the logic of the application

###### *Monomial Class*

A polynomial is composed by one or more terms, which are called monomials.

This class has a String monomial, an int power and a double coefficient as fields.

Constructor:

- *public Monomial(String monomial, double coefficient, int power);*  
//the constructor that initializes the monomials with the transmitted coefficient and degree

Methods:

- *protected void addCoefficient(double coefficientToAdd);*  
//add a value to the coefficient
- *protected static Monomial add(Monomial monomial1, Monomial monomial2);*
- *protected static Monomial multiply(Monomial monomial1, Monomial monomial2);*
- *protected static Monomial divide(Monomial monomial1, Monomial monomial2);*
- *protected static Monomial derive(Monomial monomial1);*
- *protected static Monomial integral(Monomial monomial1);*  
//perform basic operations on monomials
- *protected void updateMonomialString();*  
//computes the string field of the Monomial
- *private String buildTheNewMonomialString(Number coeff, boolean isDouble);*  
//method used in the updateMonomialString method
- *protected void changeSignOfCoefficient();*  
//changes sign of the coefficient

###### *Polynomial Class*

This class has only an instance variable which consists of a ArrayList<Monomial> class that is used to keep the monomials of this polynomial. The monomials are ordered in this list, from the highest to the lowest exponent of the polynomial, in the order as they were introduced.

Constructor:

- *public Polinom();* //default constructor

Methods:

- *protected void addMonomial(Monomial monomial);*  
//adds a new monomial to the polynomial
- *private void updatePolynomial();*  
//updates all the monomials strings in the polynomial
- *public void simplifyAndSort();*  
//simplifies the polynomial then sorts it
- *public String polynomialToString();*  
//computes the corresponding string of the polynomial so that it can be displayed in the corresponding text field of the GUI
- *protected int getDegree();*  
//returns the degree of the polynomial

### *Operation Class*

This class performs the required mathematical operations. It does not contain a constructor since all of its methods are static.

Methods:

- *public static Polynomial addition(Polynomial firstPoly, Polynomial secondPoly);*
- *public static Polynomial subtraction(Polynomial firstPoly, Polynomial secondPoly);*
- *public static Polynomial multiplication(Polynomial firstPoly, Polynomial secondPoly);*
- *public static Polynomial division(Polynomial dividendPoly, Polynomial divisorPoly);*
- *public static Polynomial derivative(Polynomial inputPolynomial);*
- *public static Polynomial integrate(Polynomial inputPolynomial);*
- *//all of the above methods are for mathematical operations on two polynomials*
- *private static Polynomial computeDivisionResult(Polynomial quotient, Polynomial remainder);*
- *private static Monomial computeNewQuotientTerm(Polynomial remainder, Polynomial divisorPoly);*
- *//the above methods are implied in the division method*

### *InputAnalyze Class*

This class is responsible for parsing the data received from the user, in terms of polynomial strings and required operation string. For this parsing to be fast and efficient, it was used the regex pattern matcher library.

A regular expression (shortened as regex or regexp; also referred to as rational expression) is a sequence of characters that specifies a search pattern. Usually such patterns are used by string-searching algorithms for "find" or "find and replace" operations on strings, or for input validation.

Methods:

- *public static Polynomial parseInputPolynomialString(String userInputPolynomial);*  
*//for parsing the polynomial string into monomials strings*
- *private static Monomial parseMonomial(String userInputMonomial);*  
*//further parsing of the monomials to obtain the coefficient and power values*
- *public static Polynomial identifyOperationAndPerform(String operation, Polynomial firstPoly, Polynomial secondPoly);*  
*//method responsible with identifying which button was pressed by the user, in order to perform the required operation*

## 2) **The View**– contains the graphical interface, the user interface.

This package contains the .fxml, .css files used in the displaying and the designing of the graphical user interface.

For this application, Scene Builder was used in order to be able to develop much faster an user-friendly interface, and with a nice-looking design without having advanced knowledge with .fxml files.

## 3) **The Control**–manages the model and the view packages

This is a very important class because it acts on both model and view. It controls the data flow into model object and updates the view whenever data changes.

#### *HomeController*

```
- public void onMouseClick(MouseEvent mouseEvent);  
//receives an request from the user when the mouse click on one of the operation buttons  
- public void insertResultOfOperation(String resultPoly);  
//will get the response back to the user with the result polynomial  
- public static String performPolynomialCalculations (String firstPolyString, String  
secondPolyString, String operation)  
//performs the necessary steps in order to compute the result
```

This class has no explicit constructor, it contains only the above methods which are used to run the application. The onMouseClick() method is responsible with receiving the input from the user (the two polynomial strings), further used to create the two corresponding instances of the polynomial class. Polynomials gets simplified and sorted, the operation is computed by the performPolynomialCalculations() method. The final result is being received back by the user, with the help of insertResultOfOperation() method.

- 4) The Application – contains the Main class which runs the main() method from the HomeController in order to “turn on” the application, and also set the necessary stage and scene for the Graphical-User Interface.

#### e) Packages

*Main packages:*

*Application*

*Model*

*View*

*Controller*

*Test*

#### f) Algorithms

##### ADDITION

The sum of two polynomials is obtained by adding together the coefficients sharing the same powers of variables so, for example,

$$(a_2 x^2 + a_1 x + a_0) + (b_1 x + b_0) = a_2 x^2 + (a_1 + b_1)x + (a_0 + b_0)$$

and has order less than (in the case of cancellation of leading terms) or equal to the maximum order of the original two polynomials.

##### SUBTRACTION

The subtraction of two polynomials is obtained by subtracting the coefficients sharing the same powers of variables so, for example,

$$(a_2 x^2 + a_1 x + a_0) - (b_1 x + b_0) = a_2 x^2 + (a_1 - b_1)x + (a_0 - b_0)$$

and has order less than (in the case of cancellation of leading terms) or equal to the maximum order of the original two polynomials.



## MULTIPLICATION

The product of two polynomials is obtained by multiplying term by term and combining the results, for example

$$\begin{aligned}(a_2 x^2 + a_1 x + a_0)(b_1 x + b_0) &= a_2 x^2 (b_1 x + b_0) + a_1 x (b_1 x + b_0) + a_0 (b_1 x + b_0) \\ &= a_2 b_1 x^3 + (a_2 b_0 + a_1 b_1) x^2 + (a_1 b_0 + a_0 b_1) x + a_0 b_0,\end{aligned}$$

and has order equal to the sum of the orders of the two original polynomials.

## DIVISION

To divide two polynomials P and Q, the following steps should be performed:

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.

## DIFFERENTIATION

A very simple algorithm to differentiate a polynomial which is represented by a sequence of ordered pairs was used:

Drop the ordered pair that has a zero exponent.

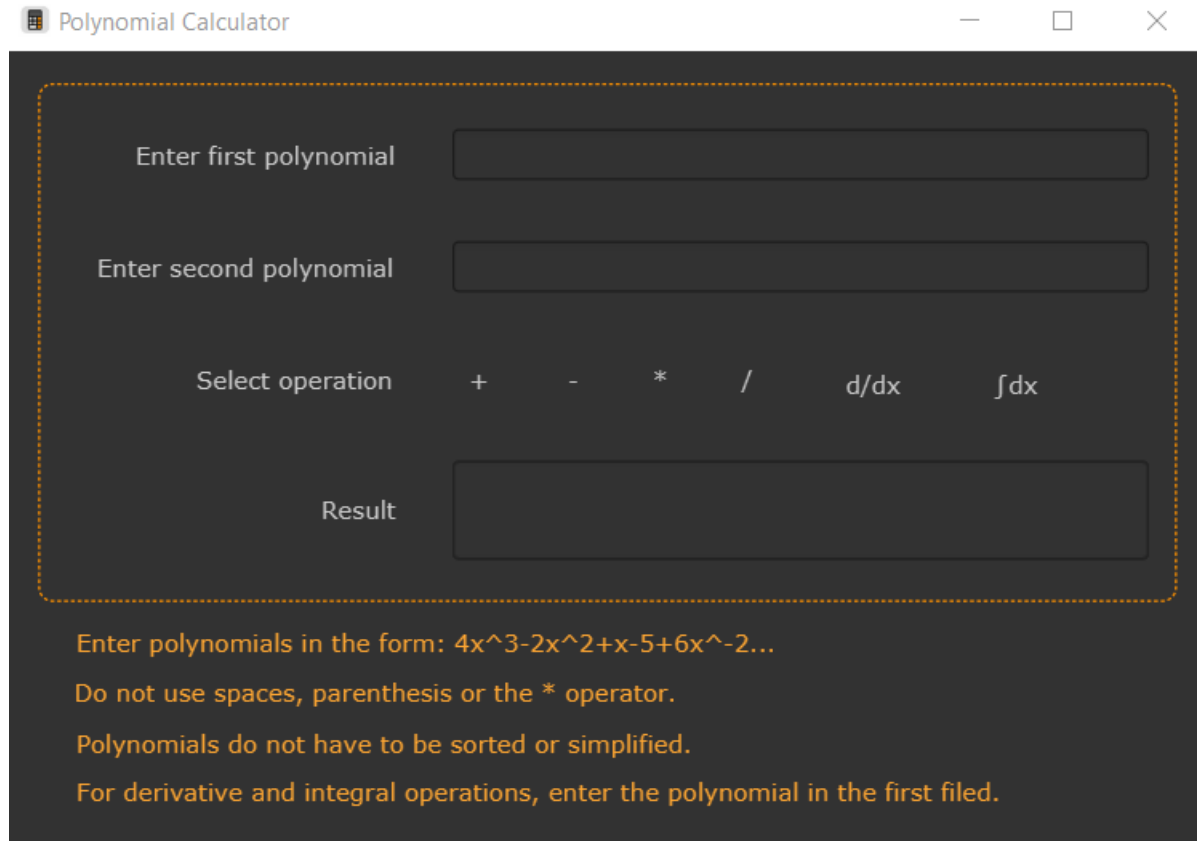
For every other ordered pair, multiply the coefficient by the exponent, and then subtract one from the exponent.

## INTEGRATION

The algorithm checks whether the exponent is -1. If so, it keeps the coefficient unchanged and prints the integration result as the natural logarithm (ln) function.

Otherwise, it multiplies the coefficient with the exponent and divides it with incremented by one exponent.

g) User interface



Polynomial Calculator

Enter first polynomial

Enter second polynomial

Select operation + - \* / d/dx ∫ dx

Result

Enter polynomials in the form:  $4x^3-2x^2+x-5+6x^{-2}$ ...

Do not use spaces, parenthesis or the \* operator.

Polynomials do not have to be sorted or simplified.

For derivative and integral operations, enter the polynomial in the first filed.

## 4. Results

Testing, performed in Junit, was done in order to identify any errors or missing requirements .

What I test	Input Data	Expected Output	The effective result	Pass / Fail
Addition	$x^3-8x^2+17x-10$ ; $x^3-6x^2+11x-6$	$2x^3-14x^2+28x-16$	$2x^3-14x^2+28x-16$	Pass
Subtraction	$x^3-8x^2+17x-10$ ; $x^3-6x^2+11x-6$	$-2x^2+6x-4$	$-2x^2+6x-4$	Pass
Multiplication	$x^3-8x^2+17x-10$ ; $x^3-6x^2+11x-6$	$x^6-14x^5+76x^4-206x^3+295x^2-212x+60$	$x^6-14x^5+76x^4-206x^3+295x^2-212x+60$	Pass
Division	$x^4+2x^2+1$ ; $3x^3+2x^2+4$	Q: $+0.33x-0.22$ R: $+2.44x^2-1.33x+1.89$	Q: $+0.33x-0.22$ R: $+2.44x^2-1.33x+1.89$	Pass
Division by zero	$x^4+2x^2+1$ ; 0	Error: Division by zero	Error: Division by zero	Pass
Differentiation	$x^4+2x^2+1$ ;	$4x^3+4x$	$4x^3+4x$	Pass
Integration	$x^4+2x^2+1$ ;	$0.2x^5+0.67x^3+x$	$0.2x^5+0.67x^3+x$	Pass

## 5. Conclusions

This assignment made me realize the importance of the parsing the data in the developing of a software application. Using the java regex package for pattern matching with regular expressions made it very simple to identify the sought pattern in a string.

This assignment also was an introduction in building a project with Gradle, and it helped me familiarize with the building of a user interface with JavaFX.

The using of the MVC architectural was also important for me to understand because it is the basic structure which most web applications, mobile apps and desktop programs are built on. In the end, testing had to be performed as well, which increases productivity and stability of program code and reduce the time for debugging. It is also very helpful if the code has to be improved as it will provide immediately the results of the test cases, instead of rerunning the application and reentering all the necessary data to tests manually, which is time consuming.

## 6. Bibliography

<https://ro.wikipedia.org/wiki/Polinom>

<https://en.wikipedia.org/wiki/Monomial>

<http://stackoverflow.com>

<https://openjfx.io/openjfx-docs/#gradle>

<https://ro.wikipedia.org/wiki/Model-view-controller>

<https://www.jetbrains.com/help/idea/create-tests.html>

<https://app.diagrams.net/>

Programming Techniques – Lectures of prof. Ioan SALOMIE