

# Polynomial Calculator

Miu Daria

# 1. Assignment Objective

The purpose of this assignment is to design and implement a Java application for calculating polynomial operations, such as addition, subtraction, division and also derivation or integration.

The application should come with a dedicated graphical user interface.

## 2. Analysis of the problem

A polynomial is an expression of addition, subtraction and multiplication consisting of variables and coefficients. Exponentiation of variables can only consist in non-negative integers. An example of a polynomial of a single variable  $x$  is  $x^2 - 3x + 4$ .

Performing operations on polynomials, such as division, multiplication, integration or derivation or even subtraction and addition, by hand, can be time consuming and small mistakes can occur easily. This is the reason why an application like this, a polynomial calculator, turns out to be extremely useful.

### ➤ Modeling the problem

To reach its purpose, such an application should meet certain requirements like:

- It should be intuitive and easy to use by the user.
- It should solve the problem fast.
- It should give the user warnings if something went wrong in the process.
- It should allow users to enter polynomials.
- It should let the user choose the desired operation.
- It should perform correctly all sorts of operations on polynomials.

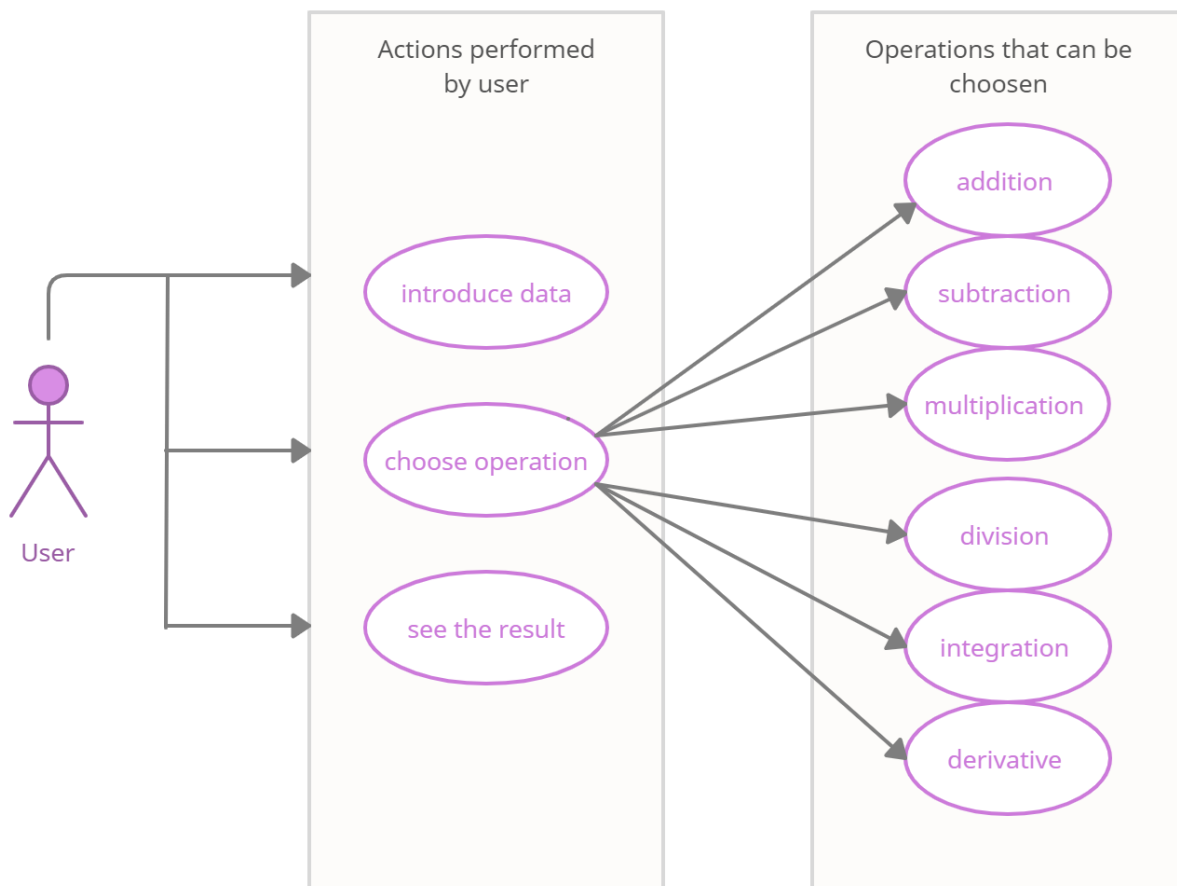
The solving of the problem is modeled in the following way: the application provides a user-friendly and intuitive interface where the users introduce polynomials in a form like " $x^2+x-7$ ", '^' character preceding the degree of that term, then choose the desired operation by clicking one of the buttons. The program solves the calculations for them and outputs the correct result. The interface has 2 sections, one for operations that require 2 polynomials, and 1 for operations on 1 polynomial.

In order to show that the application meets the functional and non-functional requirements specified, in the next section, will be presented some use-cases.

## ➤ Use Case diagram, use cases and scenarios

### ➤ The use case diagram

The use case diagram is the simple form of presenting the way a software acts. It does not show the details about the use cases, only summarizes some of the relationships between the user and the application, and it does not show the order in which steps are performed to reach the goal of each use case.



### ➤ Use cases and scenarios

Use cases represent the expected behavior of an application, but not the exact method of making it happen. They represent the way the application is seen from the end user's perspective. Use case modeling is an important tool because it helps the developer design a system from the user's point of view.

The division operation will be taken as an example for uses cases and the same logic can be applied to all the other operations.

Successful scenario

Step 1: The user inserts 2 polynomials in the dedicated text fields.

Step 2: The user clicks on the division button.

Step 3: The application checks if the input is valid.

Step 4: The application generates internally the polynomials and performs the division.

Step 5: The result is displayed in the user interface.

Unsuccessful scenario

User introduces wrong input, wrong meaning other characters than the ones allowed in writing a polynomial, or tries to introduce polynomials with real coefficients, or tries to introduce a polynomial with smaller degree as a numerator than the denominator for division. So the process stops at step 3 and generates an exception which will show a message to the user announcing what the issue is.

## 3. Design&Implementation

The application was developed according to the Model-View-Controller pattern. The model is independent of the user interface, it manages the logic and the background of the application. The view represents the part that generates the graphical user interface. The controller creates the connection between model and view, it takes inputs and converts them into commands for the model or view.

To represent clearly this pattern, the application is divided in multiple packages.

### ➤ Packages

The Model Package: contains the **Polynomial**, **Monomial**, **Operation** classes, the fundamental objects of the app;

The Controller Package: contains the class **Controller**, which intermediates the relationship between view and model;

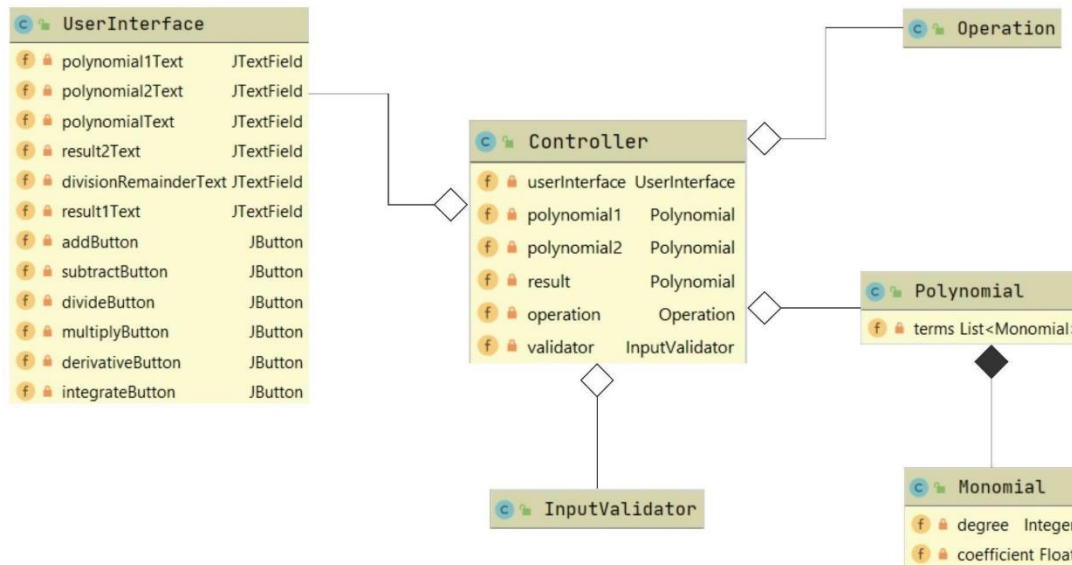
The View Package: contains the class **UserInterface**, where the entire aspect of user interface is created;

The Validator Package: contains one class **InputValidator**, used in the Controller to check if the data introduced by user is valid to be used in the application;

The App Package: contains only the **Main** class, which is run to start the application.

## ➤ UML Diagram

The relationships between classes are represented in the following diagram.























## ➤ Data structures

Data Structure represents a way in which the data can be stored and organized so that it can be used efficiently and easily. The data structures used in this project are Lists, more specifically ArrayLists. The choice for this data structure was made based on the fact that it comes with multiple helpful features and methods that make the process of writing code simpler and faster. Specific methods such as `add`, `remove` or `isEmpty` were used. For iterating through lists the `iterator` and `list iterator` were used and also the `foreach` instead of the simple `for`.

## ➤ Class design and implementation

Everything in Java is associated with classes and objects with methods and attributes. A class is the prototype from which an object is created. Taking into consideration this aspect and the MVC(Model-View-Controller) pattern, the following classes were designed and implemented in the application.

### ➤ The Monomial Class

		<b>Monomial</b>
		degree Integer
		coefficient Float
		Monomial(Integer, float)
		Monomial()
		getDegree() Integer
		setDegree(Integer) void
		getCoefficient() float
		setCoefficient(float) void
		changeSign() void

Attributes: degree(integer), coefficient(float);

Constructors: 2 constructors, the default one, one with parameters;

Methods

Getters and setters for the attributes;

changeSign() – is used in the subtract operation to make the code easier to understand and more readable.

## ➤ The Polynomial Class

Polynomial		
f	terms	List<Monomial>
m	addMonomial(Monomial)	void
m	getTerms()	List<Monomial>
m	setTerms(List<Monomial>)	void
m	addCoefficient(StringBuilder, Float)	void
m	polynomialToString()	String
m	stringToPolynomial(String)	Polynomial
m	addPolynomial(List<Monomial>)	void
m	equalsPolynomial(Polynomial)	boolean

### Attributes:

This class has only one attribute, an ArrayList of monomials(Monomial).

### Methods:

Getter and setter for the attribute;

addMonomial(Monomial)- method used to add monomials in the terms list;

addPolynomial(List<Monomial>)- method used to add more monomials at once in the polynomial;

equalsPolynomial(Polynomial)- checks if the polynomial given as parameter is equal to the current polynomial;'

polynomialToString()- transforms the current polynomial into a string, used for the display of the polynomial on the screen

stringToPolynomial(String)- transforms the input string into a polynomial such that the application can perform operations on it

addCoefficient(StringBuilder,Float)- adds coefficients to terms when transforms from polynomial to string. The methods purpose is to make a difference between float coefficient and integer coefficient. Since the coefficients are presented as Float, even is the number is an integer, it will be displayed with the .0 decimal. This problem is resolved with this method. Also, this method truncates float coefficients after the second decimal for a better output.

## ➤ The Operation Class

Operation		
m	sortPolynomial(Polynomial)	void
m	add(Polynomial, Polynomial)	Polynomial
m	subtract(Polynomial, Polynomial)	Polynomial
m	derivative(Polynomial)	Polynomial
m	integrate(Polynomial)	Polynomial
m	reduceTerms(Polynomial)	Polynomial
m	multiply(Polynomial, Polynomial)	Polynomial
m	multiplyPolyMonom(Polynomial, Monomial)	Polynomial
m	division(Polynomial, Polynomial)	Polynomial[]

### Methods

sortPolynomial(Polynomial)- for sorting the given polynomial based on the degrees of each monomial in descending order;

add(Polynomial,Polynomial)- performs the addition operation between polynomial1 and polynomiala2;

subtract(Polynomial, Polynomial)- performs the subtract operation between polynomial1 and polynomiala2;

multiply(Polynomial,Polynomial)- performs the multiplicationoperation between polynomial1 and polynomiala2;

multiplyPolyMonom(Polynomial, Polynomial)- performs multiplication of a polynomial with a monomial, used in the division algorithm, for making the method more readable and easy to follow;

division (Polynomial,Polynomial)- performs the division operation between polynomial1 and polynomiala2;

integrate(Polynomial)- integrates the given polynomial;

derivative(Polynomial)- derivates the given polynomial;



## ➤ The Controller Class

Controller		
f	userInterface	UserInterface
f	polynomial1	Polynomial
f	polynomial2	Polynomial
f	result	Polynomial
f	operation	Operation
f	validator	InputValidator
m	Controller()	
m	setButtons()	void

Attributes:        userInterface(UserInterface),    polynomial1,    polynomial2,    result(Polynomial),  
operation(Operation), validator(InputValidator);

Constructor: the default constructor in which the attributes are initiated, and the method to initiate the buttons setButtons() is called;

Methods:

setButtons()- which sets the action listeners for all the buttons in the user interface, sends the written input to the InputValidator, calls the methods for conversions (polynomial to string and string to polynomial), calls the method corresponding to the desired operation and sets the result. More precisely, this method sets the connection between the view and the model.

## ➤ The InputValidator Class

InputValidator		
m	validate(String)	void
m	validateDivision(Polynomial, Polynomial)	void

Methods:

validate(String)- method in which the input introduced by the user is validated. A valid input is an input that does not contain other characters than “x+ - ^” and digits and does not contain mistakes such in this string “x^^+x2+6^x” (even if the string contains only allowed characters, some patterns in the string mean nothing to a polynomial representation ). Also a valid input means an

input without real coefficients, so the method also checks for patterns like “digit.digit” or “digit,digit”;

validateDivision(Polynomial, Polynomial)- takes the 2 polynomials from the input and checks if they are suitable for division, the condition being: degree of the first polynomial to be greater or equal to the degree of the second polynomial and second polynomial different from 0.

### ➤ The UserInterface Class

UserInterface		
f	polynomial1Text	TextField
f	polynomial2Text	TextField
f	polynomialText	TextField
f	result2Text	TextField
f	divisionRemainderText	TextField
f	result1Text	TextField
f	addButton	Button
f	subtractButton	Button
f	divideButton	Button
f	multiplyButton	Button
f	derivativeButton	Button
f	integrateButton	Button
m	UserInterface()	
m	initializePanel(JPanel)	void
m	addButtonActionListener(ActionListener)	void
m	subtractButtonActionListener(ActionListener)	void
m	multiplyButtonActionListener(ActionListener)	void
m	divideButtonActionListener(ActionListener)	void
m	derivativeButtonActionListener(ActionListener)	void
m	integrateButtonActionListener(ActionListener)	void
m	setResult1(String)	void
m	setResult2(String)	void
m	setDivisionRemainderText(String)	void
m	getPolynomial1()	String
m	getPolynomial2()	String
m	getPolynomial()	String
m	displayErrorMessage(Exception)	void

### Attributes:

- 6 buttons, one for each operation (add, subtract, multiply, divide, integrate, derivative);
- 3 text fields for introducing polynomials, 2 for the operations on 2 polynomials and 1 for the operations on 1 polynomial;
- 3 text fields for displaying the results, one for the operations on 2 polynomial, one for displaying the result of operations on 1 polynomial, and an additional one that is used when the selected operation is division, for the division remainder.

Constructor: the default constructor in which the attributes of the JFrame are set, such as dimension, color, title, close operation, defining a JPanel on the frame;

### Methods:

initializePanel(JPanel)- method used to add all the components to the panel which is added to the frame, for all the components, the size and position are set.;

Methods for action listeners for all the buttons, which will be called in the controller;

setResult1(String)- sets on the screen the result for the operations on 1 polynomial, used in the controller;

setResult2(String)-sets on the screen the result for the operations on 2 polynomials, used in the controller;

setDivisionRemainder()- sets on the screen the remainder when the operation chosen is division, used in controller;

getPolynomial()- gets the string introduces by the user in the TextFiled used for operations on 1 polynomial;

getPolynomial1()- gets the string introduces by the user in the TextFiled for the first polynomial used for operations on 2 polynomials;

getPolynomial2()- gets the string introduces by the user in the TextFiled for the second polynomial used for operations on 1 polynomial;

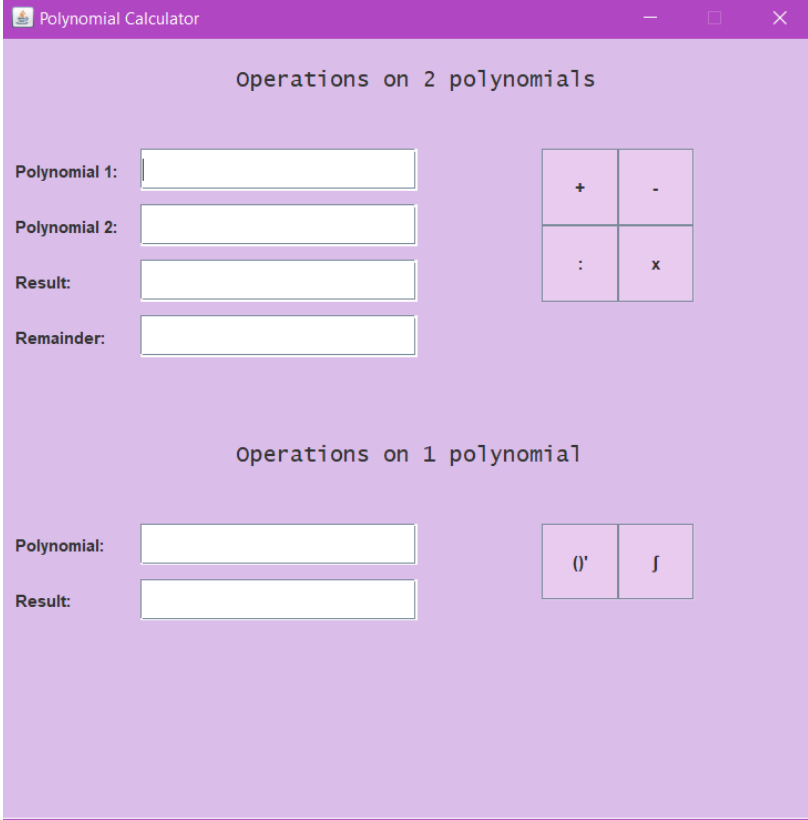
displayErrorMessage(Exception)- shows a message dialog when an exception is thrown by the application in the proces of input data validation.

### ➤ The main class

Contains just an instance of the controller which practically start the application.

## ➤ The Graphical User Interface

This is an image of how the application is presented to the user.



The image shows a window titled "Polynomial Calculator" with a purple header bar. The main area has a light purple background and is divided into two sections. The top section, titled "Operations on 2 polynomials", contains four text input fields on the left labeled "Polynomial 1:", "Polynomial 2:", "Result:", and "Remainder:". To the right of these fields is a 2x2 grid of buttons with the operators "+", "-", ":", and "x". The bottom section, titled "Operations on 1 polynomial", contains two text input fields on the left labeled "Polynomial:" and "Result:". To the right of these fields is a 1x2 grid of buttons with the operators "()" and "J".

The users can introduce data, either for operation on 1 (text filed Polynomial) or 2 polynomials (text field Polynomial 1 and Polynomial 2), then click 1 of the 6 buttons, representing the wanted operation, and the result is displayed on the result filed corresponding to the section from which the user has selected the operation.

Below, are presented some ways in which the user interface will look like after different operations.

Polynomial Calculator

Operations on 2 polynomials

Polynomial 1:

Polynomial 2:

Result:

Remainder:

+	-
:	x

Operations on 1 polynomial

Polynomial:

Result:

0'	∫
----	---

The interface after a successful addition operation.

Polynomial Calculator

Operations on 2 polynomials

Polynomial 1:

Polynomial 2:

Result:

Remainder:

+	-
:	x

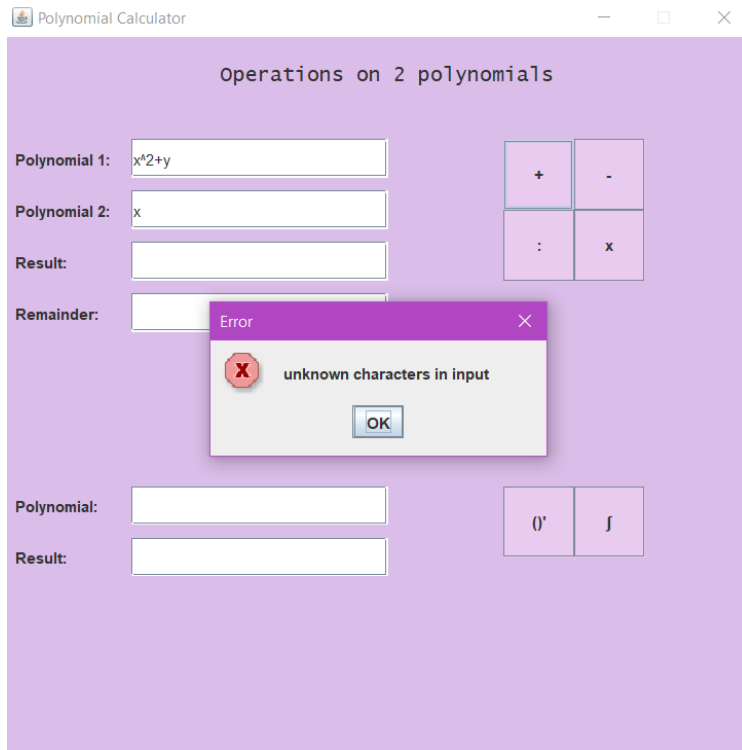
Operations on 1 polynomial

Polynomial:

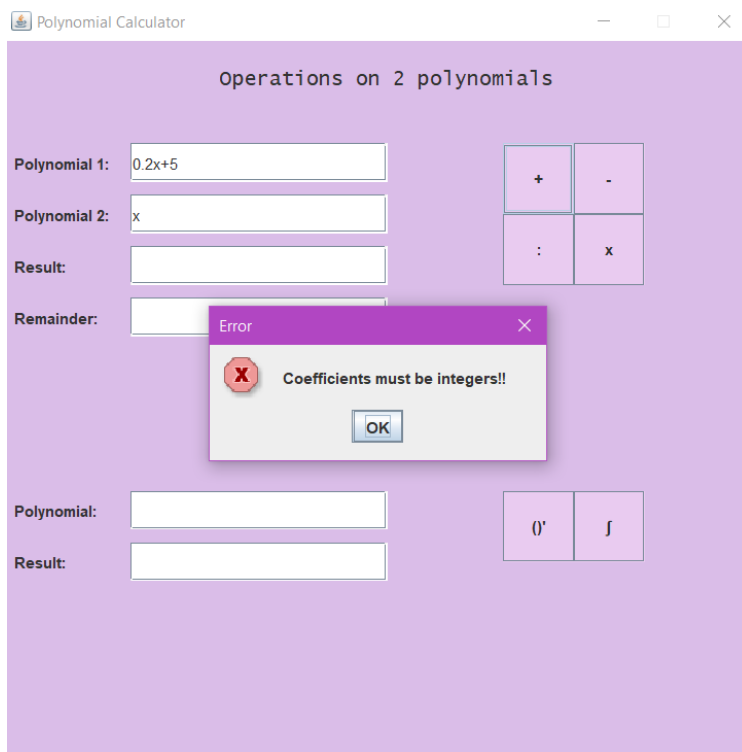
Result:

0'	∫
----	---

The interface after a successful integration operation.



The error when a not permitted character is given in the input. (here y)



The error when the user inserts non integer coefficients.

## 4. Testing&Results

The application was tested using Junit Testing. The application contains 2 testing classes, one for operations and one for the conversion from polynomial to string and string to polynomial. The results of the tests will be presented in 2 tables.

The OperationTest class contains 6 test methods, one for each operation.

Operation	Polynomial 1	Polynomial 2	resultOk	resultTest	Test passed
add	$3x^2+x+2$	$x^3+x+4$	$x^3+3x^2+2x+6$	$x^3+3x^2+2x+6$	✓
subtract	$3x^2+x+2$	$x^3+x+4$	$-x^3+3x^2-2$	$-x^3+3x^2-2$	✓
multiply	$x^2+x+1$	$x-1$	$x^3-1$	$x^3-1$	✓
divide	$3x^2+5x+2$	$2x+1$	$1.5x+0.75; (Q)$ $0.25 (R)$	$1.5x+0.75; (Q)$ $0.25 (R)$	✓
integrate	$x^2+x+1$	-	$0.(3)x^3+0.5x^2+x$	$0.(3)x^3+0.5x^2+x$	✓
derivate	$x^2+x+1$	-	$2x+1$	$2x+1$	✓

The ConversionTest class contains 2 methods, one for each conversion.

Conversion	Input	Result	Test passed
String to polynomial	$x^3+3x^2+2x+6$	$x^3+3x^2+2x+6$	✓
Polynomial to string	$x^3+3x^2+2x+6$	$x^3+3x^2+2x+6$	✓

## 5. Conclusions

The task of implementing a polynomial calculator was successfully completed, since it can be seen that the application meets all the functional and non-functional requirements.

In conclusion, the polynomial calculator application is a useful tool that eliminate the issues that could appear when solving polynomial operations on paper.

The modeling, design and implementation of this software are based on programming techniques and Object-Oriented-Programming principles, so the creation of this app was a great learning opportunity. I've learned about the Junit testing environment, since I've never used it before. Also, the regular expressions and the pattern matching were some Java features new to me. I think that I have also improved my skills in writing OOP code. Moreover, facing and fixing all sorts of errors and bugs were an important step in learning. I would like to add the fact that, as this is my second

complex project, as a “programmer”, the feeling of seeing the application actually working, and the feeling of learning new concepts are amazing.

## 6. Further possible developments

Some improvements that could make the software better would be:

- letting the user introduce polynomials of any variable, not only  $x$ ;
- implementing an operation for simple fraction decomposition, considering that the polynomials are the numerator and denominator of a fraction;
- implementing a system of displaying the graphical representation of a polynomial;
- implementing an operation which returns the roots of the polynomial.

## Bibliography

ASSIGNMENT\_1\_SUPPORT\_PRESENTATION.PPT

<https://stackoverflow.com/>

<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>

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

<https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>

[https://www.tutorialspoint.com/design\\_pattern/mvc\\_pattern.htm](https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm)

<https://regex101.com/r/zH8mE4/4>

<https://www.geeksforgeeks.org/regular-expressions-in-java/>

<https://www.baeldung.com/junit-5>