

# DOCUMENTATION

## ASSIGNMENT 1

### Polynomials Calculator



**TECHNICAL  
UNIVERSITY**  
OF CLUJ-NAPOCA  
ROMANIA

STUDENT NAME: Seserman Victor  
GROUP: 30422

# CONTENTS

1. Assignment Objective .....	3
2. Problem Analysis, Modeling, Scenarios, Use Cases.....	3
3. Design .....	5
4. Implementation .....	7
5. Results.....	9
6. Conclusions.....	9
7. Bibliography .....	9

## 1. Assignment Objective

Design and implement a system for polynomial calculator.

To achieve the objective it is needed to have a dedicated graphical interface through which the user can insert polynomials, select the mathematical operation to be performed and view the result, also it is needed the implementation of the class polynomial and the implementation of the mathematical operations performed on the polynomials.

NOTE: Consider the polynomials of one variable and integer coefficients.

## 2. Problem Analysis, Modeling, Scenarios, Use Cases

### a) Analyzing the problem

A polynomial is an expression that can be built from constants and symbols called indeterminates or variables by means of addition, multiplication and exponentiation to a non-negative integer power. For solving the problem it is considered that the polynomials used are of a singled indeterminate “x”. EX:  $x^2 - 4x + 7$ .

Polynomials consist of a collection of monomials.

Monomials consist of this indeterminate “x”, a coefficient and the power of the variable “x”.

The power of the monomial will always be a positive natural number and the coefficient can be any real number, but to achieve the polynomial calculator, only rational numbers will be used as coefficients.

### b) Modeling the problem

The user will be able to use the functions of the calculator by introducing two polynomials in the interface. They will have to write the polynomials correctly and then choose the operation they want to be performed, these operations are:

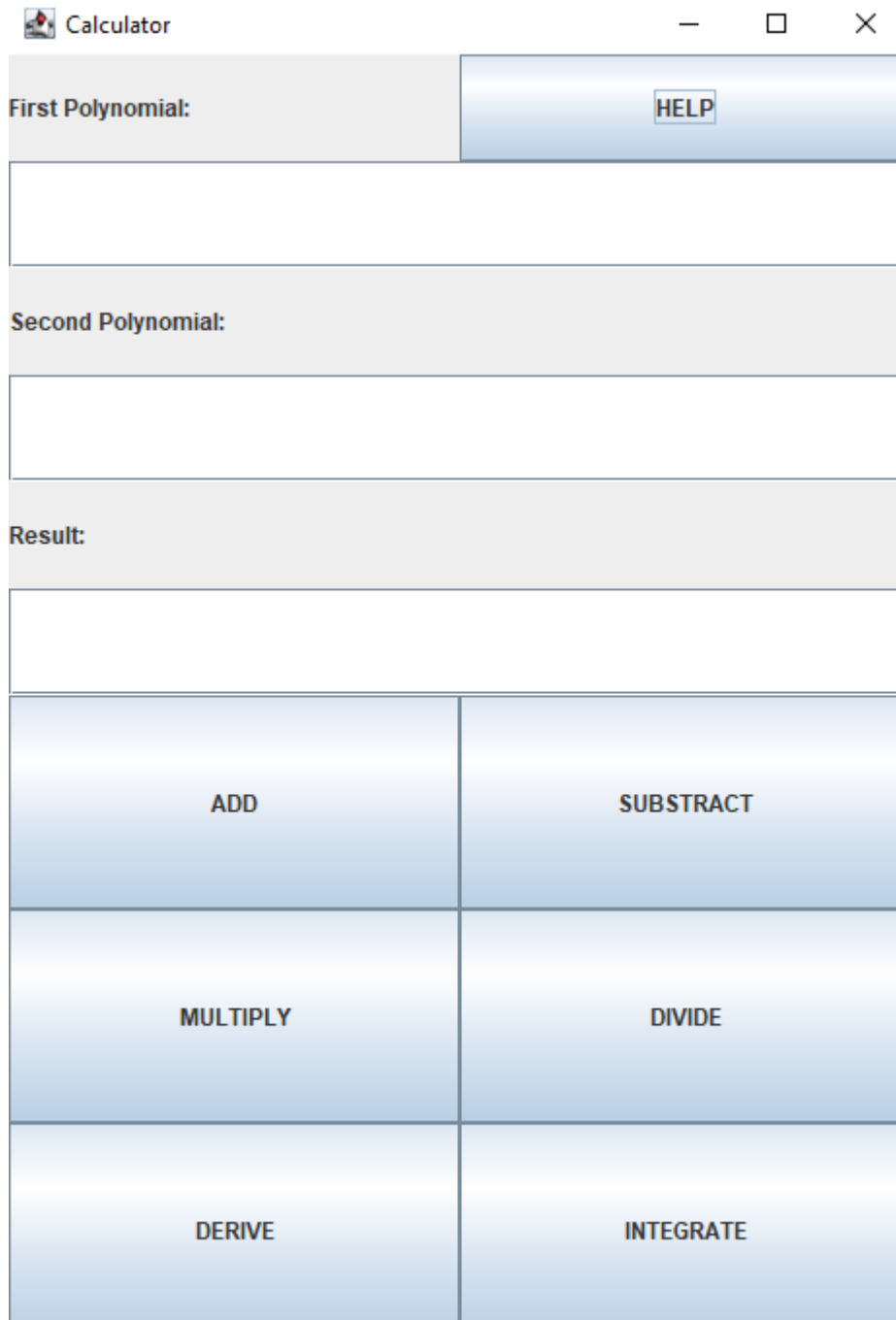
- Addition of two polynomials;
- Subtraction of two polynomials;
- Multiplication of two polynomials;
- Division of two polynomials;
- Differentiation of a polynomial(the first polynomial);
- Integration of a polynomial(the first polynomial).

The result of the chosen operation will be displayed in the interface.

### c) 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 strongly connected with the user steps. That is the reason why the design of the interface is very friendly and below is the result.



Calculator

First Polynomial:

HELP

Second Polynomial:

Result:

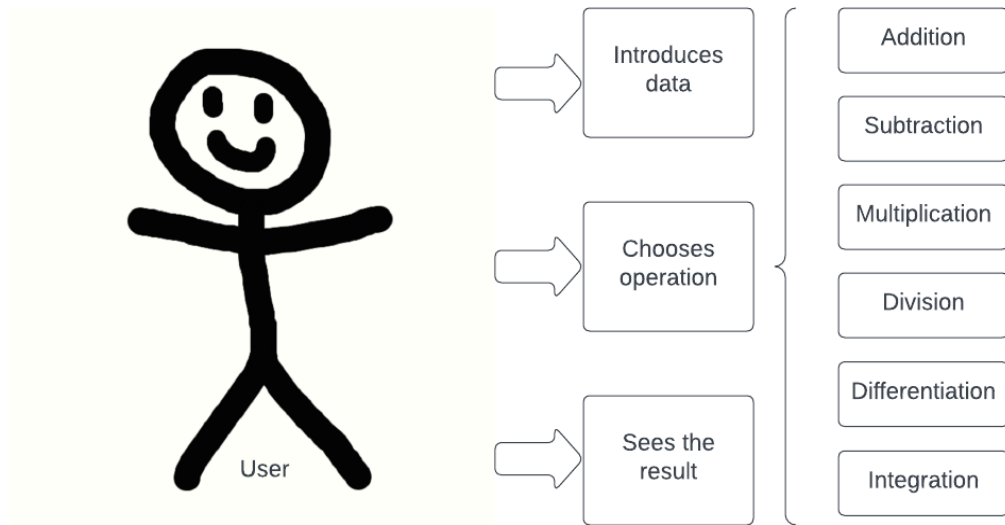
ADD	SUBTRACT
MULTIPLY	DIVIDE
DERIVE	INTEGRATE

The user will introduce the two polynomials in the corresponding textFields, and then choose which operation they want to perform. **For the calculator to work the polynomials have to be written correctly!** For the addition, subtraction, multiplication and division the result follows the order of the polynomials when performing the operations. As for the integration and differentiation, the operation will be performed on the first polynomial. **Polynomials have to be written as follows: [coefficient]\*x[power] [+/-] [coefficient]\*x[power] etc.** The last monomial can be written as only coefficient without specifying  $x^0$ .

### 3. Design

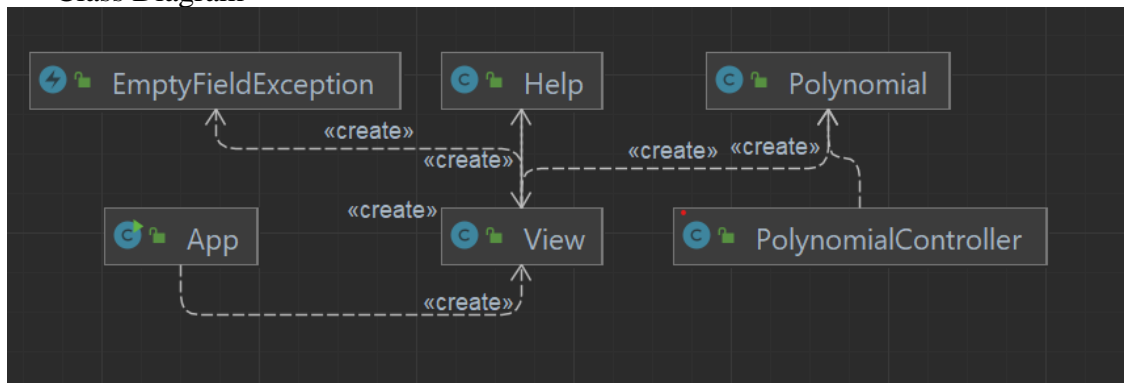
#### a) Diagrams

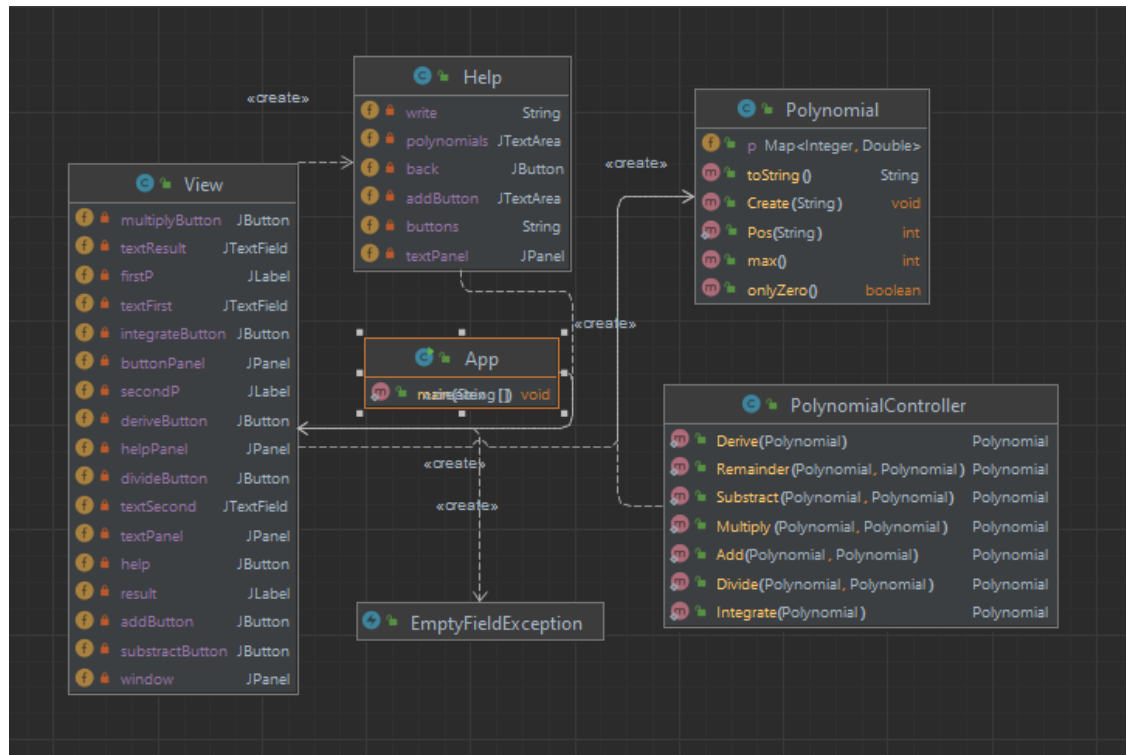
Use cases diagram:



The use cases present the user which can perform different actions mentioned above. They can perform several actions on the application and they can also interact with the application in a multitude of ways.

Class Diagram





## b) Data Structures

The data structures used in this projects are of two types: primitive, such as Integer, Double and String, and more complex, the only complex data structure used is the Map Interface.

The Map interface is great for storing vast amounts of values into a “table”, more specifically, in this project the HashMap extension of the Map interface was used.

HashMap is great for getting values out of the data structure since its search complexity is nearing  $O(1)$ , if not actually  $O(1)$ .

The HashMap is used for storing the polynomials.

## c) Packages

Java packages help in organizing multiple classes into a group such that similar classes can be found in the same place.

In OOP, model-view-controller, or MVC, is the main method of designing projects in a successful and efficient manner. The MVC pattern is widely used in program development with programming languages such as Java, C, C++ etc.

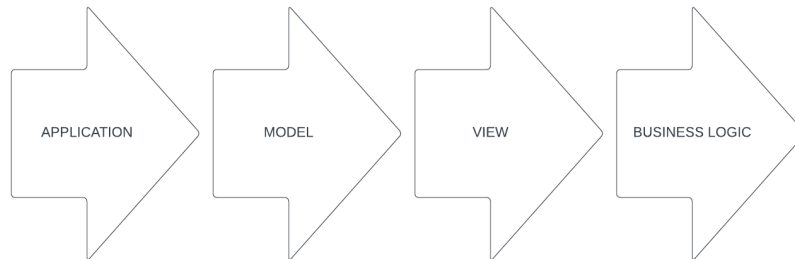
To organize the this project it was used a derived version of the MVC pattern, where the model and the view remain the same, but the controller part is slightly changed, turning it into a package called Business Logic.

Here are the benefits and explanations of each package:

-Model, this represents the logical structure of the project and the high-level class associated with it;

-View, this represents all the user interactions, such as introducing the polynomials, viewing the result or choosing the operations, also here the user can see a tutorial on how to correctly write the polynomial;

-Business Logic: in this package all the algorithms, used for calculating the result of the operations on the polynomials, can be found.



Application – not a package, but it can be consider as a imaginary package, here the main can be found, this is the starting point of the application.

Model – in this package the class “Polynomial” can be found, this is the class which models the whole problem.

View – this is the package which contains the GUI, 2 classes can be found here, a class which is interactive, where all the operations happen and where the user can input polynomials and view the result, and a GUI where the user can find some explanations.

Business Logic – this package contains only one class which has inside of it all the methods needed for the arithmetical operations performed on the polynomials.

## 4. Implementation

### Classes:

#### 1. Polynomial

This class can be found in the model package and it contains only one field: Map <Integer, Double> p. The constructor is empty, but when it is called the Map p becomes a HashMap.

The class has 5 methods and they are the following:

- -public static int Pos(String string) – this method determines where the x is in every monomial of the polynomial which is written as a String. Ex: for the string 3451\*x234 it will return the value 5, if no x is found it will return -1, and that is how the program knows that the coefficient is for  $x^0$ ;
- -public void Create(String text) – this method takes a string and from that string it makes the polynomial, it is basically a converter from string to polynomial;

- `public String toString()` – this method is an override of the basic `toString` method, it was changed in such a way that it returns the polynomial in a String format but prettier;
- `public int max()` – this method returns the order of the polynomial, the highest power that exists inside the polynomial;
- `public Boolean onlyZero()` – this method verifies if inside a polynomial there are only zeros, this method is needed for checking if a remainder exists when dividing two polynomials.

## 2. PolynomialController

This class consists of all the methods that are used for the operations on the polynomials, all functions return a polynomial:

- `public static Polynomial Add();`
- `public static Polynomial Subtract();`
- `public static Polynomial Multiply();`
- `public static Polynomial Divide();`
- `public static Polynomial Remainder();`
- `public static Polynomial Derive();`
- `public static Polynomial Integrate();`

All these methods will be explained later, when all the algorithms will be explained

## 3. App

This is where the App begins, only field that can be found here is the main.

## 4. View

This is the GUI class, apart from the TextFields, labels, buttons, etc. here the initialization of the frame can be found, this is done via a constructor with no parameters. The result of the class can be found on page 3.

# Algorithms

## 1. Addition

The addition is very straight forward, the coefficients of both polynomials will be added where the power corresponds. The order of the resulting polynomial will be either the same or lower then the maximum between the original polynomials.

## 2. Subtraction

The subtraction is very straight forward, the coefficients of the first polynomial will be added, in case of the first polynomial, and subtracted, in case of the second polynomial, from the result. The order of the resulting polynomial will be either the same or lower then the maximum between the original polynomials.

## 3. Multiplication

For the multiplication, term by term the powers will be added and the coefficients will be multiplied with each other.

## 4. Division

For the division the polynomial long division algorithm was used.



## 5. Differentiation

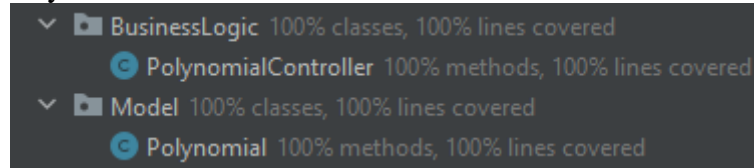
For the differentiation algorithm, only the first polynomial will be used. For every term of the polynomial the power will decrease by one and the coefficient will be multiplied by the power, except for the coefficient with  $x^0$ , that coefficient will disappear. This means that every coefficient that is multiplied will be shifted to the left once.

## 6. Integration

For the integration algorithm, only the first polynomial will be used. Every term will have their power increase by one and the coefficient will be divided by the power. This means that every coefficient that is divided will be shifted to the right once.

## 5. Results

The tests performed were done using Junit. A multitude of tests were performed such that the Model and the BusinessLogic were covered completely. The following statistics is provided by the IDE:



## 6. Conclusions

Personally, I learned a lot while doing this project, from the simple stuff that was forgotten to the more complex algorithms that needed to be efficient to respect the rules of the assignments. It was a nice challenge, not very difficult, but definitely tricky. For future developments there are a lot of features that can be added that were not specified, such as writing the polynomials, other operations can be added, a good feature might be to perform operations on multiple polynomials, or on polynomials with more than one variable.

## 7. Bibliography

- [Wikipedia](#)
- [JUnit testing tutorial](#)
- [Programming Techniques Lectures and Laboratories](#)