

**POLYNOMIAL CALCULATOR**

Student: Francesca Dițulescu

Teacher: Viorica Chifu

Table of contents

Assignment objective

Main objective

Sub-objectives

Problem analysis, modeling, scenarios, use cases

Problem analysis

Solution

Requirements

Use cases

Design

Overall System Design

Division into packages

Division into classes

UML Class Diagram

GUI Design

Implementation

Conclusions

Bibliography

ASSIGNMENT OBJECTIVE

MAIN OBJECTIVE

The main objective of this project ist o design and implement a polynomial calculator with a dedicated graphical user interface through which the user can perform different operations (addition, subtraction, multiplication, division, derivation and integration) on polynomials.

SUB-OBJECTIVES

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

PROBLEM ANALYSIS

The purpose of this project is to solve the following problem: performing polynomial operations on paper is difficult and time consuming when you have to deal with big polynomials and big coefficients.

SOLUTION

As a solution, this project provides a faster and more interactive application that can help the user to perform operations on polynomials easier.

REQUIREMENTS

Functional requirements:

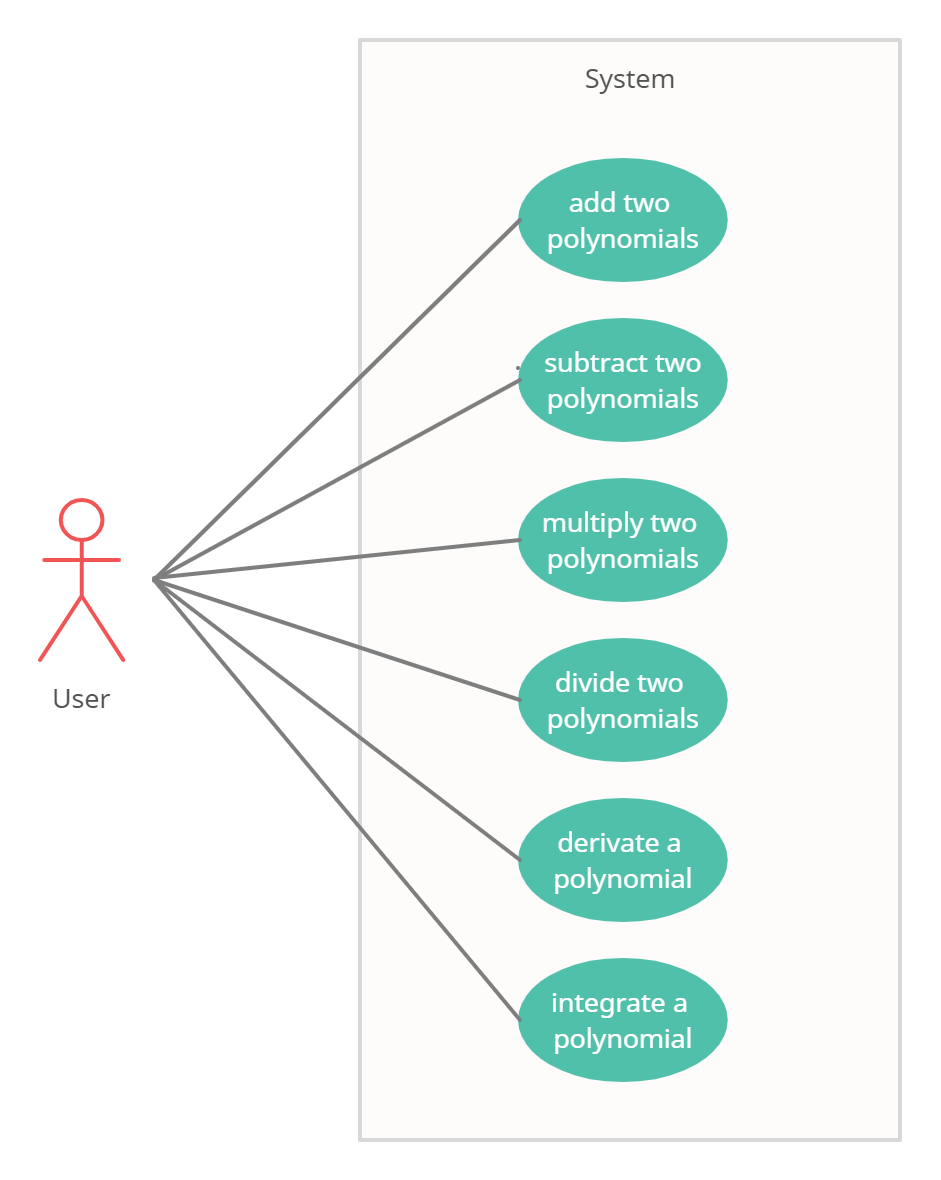
* The polynomial calculator should allow the user to insert data.
* The polynomial calculator should allow the user to select which operation he/she wants to perform.
* The polynomial calculator should be able to read a polynomial written in its algebric form.
* The polynomial calculator should be able to add two polynomials.
* The polynomial calculator should be able to subtract two polynomials.
* The polynomial calculator should be able to multiply two polynomials.
* The polynomial calculator should be able to divide two polynomials.
* The polynomial calculator should be able to derivate a polynomial.
* The polynomial calculator should be able to integrate a polynomial.

Non-functional requirements

* The polynomial calculator should be intuitive and easy to use.
* The polynomial calculator should have a nice and pleasant graphical user interface.

USE CASES

The use case diagram:



1. Add two polynomials

Primary actor: User

Main success scenario:

1. The user inserts 2 polynomials in the graphical user interface
2. The user selects the “+” (addition) button
3. The polynomial calculator performs the addition of the two polynomials and displays the result

Alternative sequence:

1. The user inserts an invalid input (incorrect polynomial form)
2. The scenario returns to step 1
3. The user inserts a polynomial with non-integer coefficients
4. The scenario returns to step 1

2. Subtract two polynomials

Primary actor: User

Main success scenario:

1. The user inserts 2 polynomials in the graphical user interface
2. The user selects the “-” (subtraction) button
3. The polynomial calculator performs the subtraction of the two polynomials and displays the result

Alternative sequence:

1. The user inserts an invalid input (incorrect polynomial form)
2. The scenario returns to step 1
3. The user inserts a polynomial with non-integer coefficients
4. The scenario returns to step 1

3. Multiply two polynomials

Primary actor: User

Main success scenario:

1. The user inserts 2 polynomials in the graphical user interface
2. The user selects the “\*” (multiplication) button
3. The polynomial calculator performs the multiplication of the two polynomials and displays the result

Alternative sequence:

1. The user inserts an invalid input (incorrect polynomial form)
2. The scenario returns to step 1
3. The user inserts a polynomial with non-integer coefficients
4. The scenario returns to step 1

4. Divide two polynomials

Primary actor: User

Main success scenario:

1. The user inserts 2 polynomials in the graphical user interface
2. The user selects the “/” (division) button
3. The polynomial calculator performs the division of the two polynomials and displays the result

Alternative sequence:

1. The user inserts an invalid input (incorrect polynomial form)
2. The scenario returns to step 1
3. The user inserts a polynomial with non-integer coefficients
4. The scenario returns to step 1

5. Derivate a polynomial

Primary actor: User

Main success scenario:

1. The user inserts one polynomial in the graphical user interface
2. The user selects the “ ‘ “ (derivate) button
3. The polynomial calculator performs the derivation of the polynomial and displays the result

Alternative sequence:

1. The user inserts an invalid input (incorrect polynomial form)
2. The scenario returns to step 1
3. The user inserts a polynomial with non-integer coefficients
4. The scenario returns to step 1

6. Integrate a polynomial

Primary actor: User

Main success scenario:

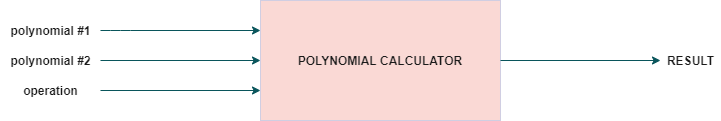
1. The user inserts one polynomial in the graphical user interface
2. The user selects the “ ∫ “ (integrate) button
3. The polynomial calculator performs the integral of the polynomial and displays the result

Alternative sequence:

1. The user inserts an invalid input (incorrect polynomial form)
2. The scenario returns to step 1
3. The user inserts a polynomial with non-integer coefficients
4. The scenario returns to step 1

DESIGN

Level 1: Overall System Design



Level 2: Division into sub-systems/packages

MODEL-VIEW-CONTROLLER ARCHITECTURE

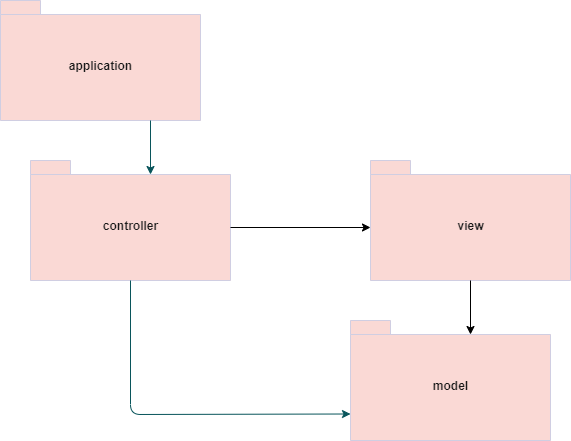
The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data.

The View component is used for all the UI logic of the application.

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output.

My package diagram is the following:



Level 3: Division into classes

a) APPLICATION PACKAGE – contains the Main class with the main method which starts the application

b) MODEL PACKAGE – contains the classes which model the application:

- Monomial: an element from a polynomial which is extracted from the introduced string in the graphical user interface

- Polynomial: a list of monomials that has to be used as a termn when performing the operations

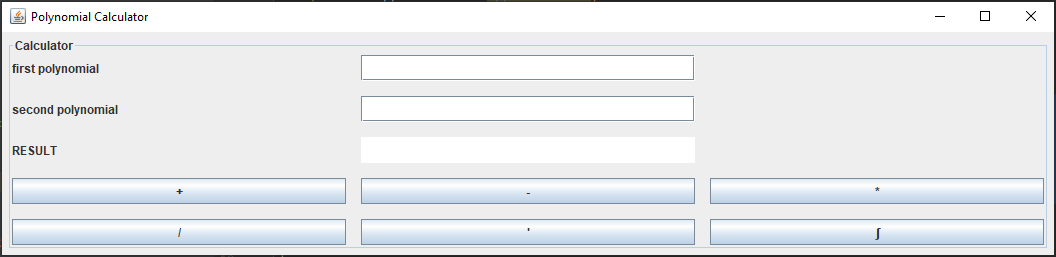
c) CONTROLLER PACKAGE – deals with the business logic of the application and initializes the buttons from the graphical user interface

d) VIEW PACKAGE – contains the class which creates the graphical user interface (the visual part of the application)

UML CLASS DIAGRAM

GUI DESIGN

The main frame contains three labels, two text fields in which the user can introduce data, one test area where the result of the operations selected is printed for the user to see. Moreover, the interface also provides six buttons, each representing an operation (addition +, subtraction -, multiply \*, divide /, derivate ‘, integrate ∫).



IMPLEMENTATION

In order to explain all the operations and how I thought of implementing, we will consider two polynomials:

*P(x) = anxn + an-1xn-1 + … + a1x1 + a0*

*Q(x) = bnxn + bn-1xn-1 + … + b1x1 + b0*

1. addition

When adding two polynomials, for each

2. subtraction

3. multiplication

4. division

5. derivation

6. integration

RESULTS

CONCLUSIONS