Technical University of Cluj-Napoca

Fundamental Programming Techniques

Laboratory – Assignment 1

Polynomials Calculator

Logo, company name

Description automatically generated

Teacher: prof. Ioan Salomie

Teacher Assistant: Ciprian Adrian Stan

Student: Itu Anca

Group: 30422

1. Assignment objective

Design and implement a polynomial calculator with a dedicated graphical interface through which the user can insert polynomials, select the mathematical operation (addition, subtraction, multiplication, division, derivative, integration) to be performed and view the result.

1. Problem analysis, modeling, scenarios, use cases

Problem analysis

A polynomial is an expression involving a sum of powers in one or more variables multiplied by coefficients. It involves the operations of addition, subtraction, multiplication, division, derivation and integration. A polynomial with one variable (a univariate polynomial) with constant coefficients is given by



Each one of the terms of this summation that represent the polynomial are called monomials. The highest power in a univariate polynomial is called the degree of the polynomial. In the monomial ,  is the coefficient, is the variable and is the exponent.

Modelling

In the interface there are two text fields intended for the polynomials inserted by the user, 6 buttons for the operations and another text field which is designated for the obtained result after selecting the desired operation to be performed on the input.

The six operations are divided into 2 categories: 4 of them perform on the content of both the input text fields (addition, subtraction, multiplication, division) and 2 of them perform on the input introduced in the text field corresponding to the first polynomial (derivation, integration).

If the user fill only one or even none of the 2 input text fields in case one of the buttons corresponding to binary operations is pressed, an alert box containing an error message will appear on the screen. Also, in case the selected operation is to derive or to integrate the polynomial, an error will appear in case the first text field is empty.

After introducing the data correctly according to the operation which is selected afterwards, the application will perform some operations on it in order to generate internally the polynomials in their correct form, then the selected operation by pressing the specific button is performed and the result will be shown in the “Result” text field. Then, if the user wishes to modify one of the polynomials he will be able to do this and to select any operation. Consequently, the result in the “Result” text field will be modified. Also, the user has the possibility to let the input unchanged, but select another operation and the corresponding result will appear.

Use cases

A use case is a definition of a specific objective that the system needs to accomplish. The two types of use-cases can be described at an abstract level (business use-case) or at an implementation-specific level (system use-case).

The **business use-cases** in my project are “insert polynomials”, “select operation”, “see the result” referring to the high-level business process being described, and the user, who is the primary actor and represents the external entity that takes part in the process. The business needs to perform a sequence of actions, which are defined by the business use-case, in order to generate a meaningful result to the external entity.

Diagram

Description automatically generated

The **system use-cases** are written at a lower level of detail than the business use case and refer to the specific processes that will be carried out by different parts of the system. In my project, such use-cases are “add polynomials”, “subtract polynomials”, “multiply polynomials”, “divide polynomials”, “differentiate polynomial” and “integrate polynomial”.

Diagram

Description automatically generated

Scenarios

The main success scenario would be the following one: the user inserts the 2 polynomials (or the polynomial, depending on the operation), then he selects the operation and the result is computed and displayed.

Another possible scenario would be in case the input is not inserted correctly according to the operation selected to be performed on it. An error occurs, and the user has to insert a valid input.

1. Design (design decisions, UML diagrams, data structures, class design, interfaces, relationships, packages, algorithms, user interfaces)

My idea was that, after the user inserts the input and presses a button corresponding to an operation, one or both strings are processed. A string is parsed into multiple strings, using a regular expression. Then, each such obtained string is parsed and converted into a monomial by obtaining both a coefficient and a power from that string. So, in my case, each monomial is characterized by its coefficient and the exponent of the variable x. Then, the selected operation is performed on the obtained list(s) of monomials and the obtained result is converted into a string and displayed.

As data structures, I used classical arrays and arrays of type ArrayList. Arrays are collections containing elements of similar type and has contiguous memory location. The disadvantage of this data structure is that it can store only a fixed set of elements and it doesn’t grow its size at the runtime. To solve this problem, I preferred to use ArrayList, which is a resizable-array implementation of the List interface.

As data types, I used both primitive data types, like **int** and **boolean**, and non-primitive data types, such as String, Integer, Arrays and Classes.

I chose to use the interactive system pattern **Model View Controller.**

* **Model** is the central component of the pattern and it is the application’s dynamic data structure, independent of the user interface and receives user input from the controller.
* **View** is used for the UI logic of the application.
* **Controller** receives the input, inserted by the user and converts it to commands for the **Model** and **View**. It is like an interface between the other two components.

**Controller** updates the **Model** based on the events happening as a consequence of the inserted input in the **View** part.

Diagram

Description automatically generated

My project contains classes divided into 4 packages:

* application – contains the “Main” class;
* model – contains any business logic-related data. This package contains the following classes: “Polynomial”, “Monomial”, “Addition”, “Subtraction”, “Multiplication”, “Derivation”, “Integration” and the interfaces “BinaryOperation” and “UnaryOperation”;
* view - contains the class “CalculatorInterface” and the FXML file used to create the interface;
* controller – contains the classes “CalculatorController” and “Reversed”.

application:

* Main – Here, I create the object controller of the class CalculatorController and I call the start() method from the CalculatorController.

model: - which contains the classes:

* Polynomial
* Monomial
* Addition
* Subtraction
* Multiplication
* Division
* Derivation
* Integration

-and the interfaces:

* BinaryOperation
* UnaryOperation

view:

* CalculatorInterface
* the FXML file sample.fxml

controller:

* CalculatorController
* Reversed

A picture containing text

Description automatically generated

The **sum** of two polynomials is obtained by adding together the [coefficients](https://mathworld.wolfram.com/Coefficient.html) sharing the same powers of [variables](https://mathworld.wolfram.com/Variable.html) (i.e., the same [terms](https://mathworld.wolfram.com/Term.html)) so, for example,

.

and has order less than or equal to the maximum degree of the original two polynomials.

The **product** of two polynomials is obtained by multiplying term by term, for example



The coefficients are multiplied and the exponents of the variables which are multiplied are summed. The resulted polynomial has the degree equal to the sum of the degrees of the two polynomials.

The **division** of two polynomials is made by following the long division polynomial algorithm:

1) the order of the monomials in the polynomials must be descending according to their degree;

2) the polynomial with the higher degree must be the dividend and the one with the smaller degree must be the divisor;

3) the first monomial of the dividend is divided to the first monomial of the divisor;

4) the quotient is multiplied with the divisor and then the result is subtracted from the dividend, obtaining the remainder of the division;

5)the steps from the second one have to be repeated by considering the remainder as the new dividend until the degree of the remainder is lower than the degree of the divisor.

* (X3 - 2\*X2 + 6\*X – 5) : (X2 – 1) = X – 2
* -X3 + X
* - 2\*X2 + 7\*X – 5
* 2\*X2 – 2

7\*X – 7

The **derivative** of a power function has the next formula:  .

Each monomial of a polynomial can be derived respecting this formula, except for the case when the monomial represents a constant, in which case the derivative is equal to zero. The derivative of a polynomial represents the sum of the derivatives of the monomials which compose it.

The **integral** of a polynomial is the sum of the integrals of its terms and it is the inverse of the derivative:

, .

1. Implementation

* Model
* Polynomial

An object of the class Polynomial is characterized by an ArrayList of type Monomial, containing the monomials which form the polynomial.

The class Polynomial contains the following methods:

Constructors:

public Polynomial(ArrayList<Monomial> monomials)

Getters and setters:

public ArrayList<Monomial> getMonomials()

public void setMonomials(ArrayList<Monomial> monomials)

Other methods:

public void createMonomialsList(String stringPolynomial, ArrayList<String> stringMonomials, ArrayList<Monomial> monomialsList)

* This method calls a method from the CalculatorController in order to parse the input into separate strings which are converted into monomials, by calling the function createMonomialFromString(String stringMonomial). These monomials will populate the ArrayList containing the monomials of the object polynomial.
* Monomial

The attributes of an object of this class are coefficient(double) and power(int). Each monomial has a variable(x) a coefficient and an exponent.

Constructors:

public Monomial(double coefficient, int power)

public Monomial()

Getters and setters:

public double getCoefficient()  
public void setCoefficient(double coefficient)   
public int getPower()  
public void setPower(int power)

Other methods:

public String createStringFromMonomial()

* This method is called after the operation selected by the user is performed and the result has to be displayed. It takes care of two possible situations: when the coefficient has the decimal part equals to zero and the opposite case. In case the decimal part is equal to zero, only the whole number part is displayed. In the other case, it has to be displayed the decimal part of the number, also.
* BinaryOperation
* It contains the abstract method performOperation(ArrayList<Monomial> monomialsListFirstPolynomial, ArrayList<Monomial> monomialsListSecondPolynomial, ArrayList<Monomial> monomialsListResult)
* This interface is implemented by the Addition, Subtraction, Multiplication and Division classes because they perform operations on two operands. The abstract method is inherited by these classes. I decided to use an interface because all the 4 operations which perform on two operands have a similar structure: they take as arguments both the ArrayList representing the list of monomials of the first polynomial and the ArrayList representing the list of monomials of the second polynomial and the ArrayList representing the list of monomials of the result. The operations are performed on the elements of the first two arrays and the resulted monomials are added to the list of monomials which will represent the result.
* UnaryOperation
* It contains the abstract method performOperation(ArrayList<Monomial> monomialsListFirstPolynomial, ArrayList<Monomial> monomialsListResult)
* This interface is implemented by the Derivation and Integration classes because they perform operations on one single operand. The abstract method is inherited by these classes. I decided to use an interface because all the 2 operations which perform on one operand have a similar structure: they take as arguments the ArrayList representing the list of monomials of the polynomial and the ArrayList representing the list of monomials of the result. The operations are performed on the elements of the array and the resulted monomials are added to the list of monomials which will represent the result.
* Addition – it performs the addition of two polynomials
* Subtraction – it performs the subtraction of two polynomials
* Multiplication - it performs the multiplication of two polynomials
* Division - it performs the division of two polynomials
* Derivation - it differentiates one polynomial
* Integration - it integrates one polynomial

All the operations are performed on the list of monomials corresponding to the polynomials.

* View
* CalculatorInterface
* Here, an object of type URL is created in order to realize the connection with the fxml file which contains the declaration and definition of all the elements that appear on the user interface. Inside the main stage is inserted a scene which represents the content displayed inside a window. When the JavaFX application starts up, it creates a root Stage object which is passed to the start method of the root class of the application.
* As a layout I used an AnchorPane and as UI controls I used 3 labels 3 text fields and 6 buttons.

Graphical user interface

Description automatically generated

* Controller
* CalculatorController
* In this class I declared as public the text fields and the buttons, 2 arrays of type String, 2 arrays of type Monomial and 2 polynomials. It has:
  + 6 methods, each one of them being used to set an actionEvent on a button;
  + 2 methods used to get the inserted input from the 2 text fields;
  + One method to validate the input for a binary operation (it verifies if in both text fields was inserted something) and another one to validate the input for unary operation (it verifies if something was inserted in the first text field):

*private void validateInputBinaryOperation()*

*private void validateInputUnaryOperation()*

* + One method to separate the input into multiple strings:

*public static void separateInputIntoMonomialsString(String stringPolynomial, ArrayList<String> stringMonomials)*

* + One method to create monomial from string:

*public static Monomial createMonomialFromString(String stringMonomial)*

* + One method to add the terms having the same exponent in case the user inserts 2 or more such monomials when writing a polynomial in one of the text fields

*private static void addTermsHavingTheSameExponent(ArrayList<Monomial> monomialsList)*

* + 2 methods used to manage the polynomials(there are called other methods which are performed on the inserted string(s)): one for the binary operations and one for the unary ones

*public void managePolynomials()*

*public void managePolynomial()*

* + *public void sortMonomialsList(ArrayList<Monomial> monomialsList)*
  + *public String convertResultIntoString(ArrayList<Monomial> monomialsList)*
  + One method used to rewrite the initial attributes in case the user wants to perform more than one operation or if he wants to change the input and then to perform an operation on it

*public void reset()*

* + - Reversed

-This class is dedicated for reversing the order of monomials because, in my project, the monomials list of the result contains the monomials in the increasing order of their degrees and I need to order them reversely when I want to display them.

1. Results

In this application the used mathematical algorithms are implemented in Java, aiming to perform some primary operations on one or 2 polynomials inserted by the user using the interface of the application.

1. Conclusions

The implemented design is a polynomial calculator which is intuitive and easy to use by the user.

1. Bibliography

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

https://www.tutorialspoint.com/mvc\_framework/mvc\_framework\_introduction.html

https://www.guru99.com/mvc-tutorial.html

https://www.javatpoint.com/array-in-java#:~:text=Java%20array%20is%20an%20object,of%20a%20similar%20data%20type.&text=In%20Java%2C%20array%20is%20an,in%20an%20array%20in%20Java.

https://www.w3schools.com/java/java\_arraylist.asp

https://www.w3schools.com/java/java\_classes.asp