Technical University of Cluj-Napoca

Programming Techniques

Laboratory – Assignment 1

Polynomial Calculator

*Logo, company name

Description automatically generated*

Teacher: prof. Cristina Bianca Pop

Teacher Assistant: Cristina Bianca Pop

Student: Siladi Ciprian

Group: 30424\_2

# Objective

The objective of this project is to propose, design and implement a system for calculating polynomials. Considering the polynomials of one variable, integer degrees and integer/double coefficients.

# Analysis, modelling, scenarios, use cases of the problem

1. *Polynomial as a structure*

A polynomial is an [expression](https://en.wikipedia.org/wiki/Expression_(mathematics)) that can be built from [constants](https://en.wikipedia.org/wiki/Constant_(mathematics)) and symbols called indeterminates or variables by means of [addition](https://en.wikipedia.org/wiki/Addition), [multiplication](https://en.wikipedia.org/wiki/Multiplication) and [exponentiation](https://en.wikipedia.org/wiki/Exponentiation) to a [non-negative integer](https://en.wikipedia.org/wiki/Non-negative_integer) power.

A polynomial in a single indeterminate x can always be written (or rewritten) in the form

where  are constants and is the indeterminate. The word "indeterminate" means that  represents no particular value, although any value may be substituted for it.

The polynomial consists of a list of certain terms, also called monomials, for example   is a monomial. The coefficient is 3, the indeterminate is and the degree is 2. Forming a sum of several terms produces a polynomial, like the following one: , which has three terms with different exponents:  the first is degree two, the second is degree one, and the third is degree zero.

An alternative representation for polynomials consists of a sequence of ordered pairs:

Each ordered pair corresponds to the term of the polynomial. An ordered pair is composed of the coefficient of the i-th term and its index representing the exponent i.

For example, the polynomial can be represented by the sequence {(3,2), (2,1), (1,0)}. The sequence contains the (coefficient, exponent) pair of each monomial from the given polynomial.

So, we can say that polynomial is composed of one or more monomials with i in the range [0, n] where n is the degree of the polynomial.

This way of representing polynomials can be used to perform the most common operations on polynomials: addition, subtraction, multiplication, division, differentiation, and integration.

1. *Interaction with the user*

The user will be able to use the functions of the calculator by introducing in the text fields of the

interface two polynomials in the first form described above. After introducing the polynomials in the fields, the user can choose which operation to be performed next, by pressing the corresponding button.

The available operations are:

* Addition of two polynomials
* Subtraction of two polynomials
* Multiplication of two polynomials
* Division of two polynomials
* Compute the derivative of one polynomial
* Compute the integral of one polynomial

The result of the operation will be displayed below the polynomials, in the interface. In the case of computing the derivative or the integral of one polynomial, an alert will pop-up asking the user to select which one of the inserted polynomials to be computed. If one of the polynomials does not respect the correct pattern, an error will pop-up telling the user to check the polynomials.

# Implementation

1. *Diagrams*

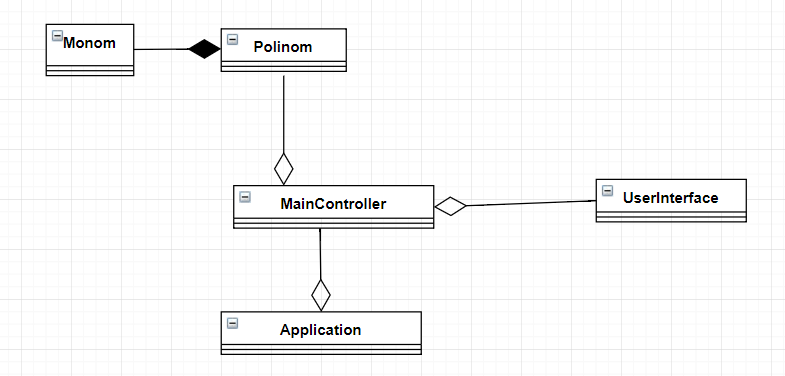
* Use Case Diagrams

Diagram

Description automatically generated

Diagram

Description automatically generated

* Class Diagram
* Class Diagrams

**Controller**

Graphical user interface

Description automatically generated with low confidence

1. *Data Structures*

The data structures with which I have been working on this project are either primitive data types, integers or doubles, I also used the Number class for the coefficients in order to have more flexibility and also ArrayList type object. The main two data structures are the ones which I created in order to perform the required operations: Monomial and Polynomial.

Regarding the implementation, monomial has a power and a coefficient, while Polynomial is an ArrayList of Monomials. I decided to use ArrayList instead of the classic arrays because I think that they are 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 predefined length of the array.

1. *Packages*

Java packages help in organizing multiple modules and group together related classes and interfaces.

In object-oriented programming development, model-view-controller (MVC) is the name of a methodology or design pattern for successfully and efficiently relating the user interface to underlying data models. The MVC pattern is widely used in program development with programming languages such as Java, C, and C++.

The MVC pattern has been heralded by many developers as a useful pattern for the reuse of object code and a pattern that allows them to significantly reduce the time it takes to develop applications with user interfaces.

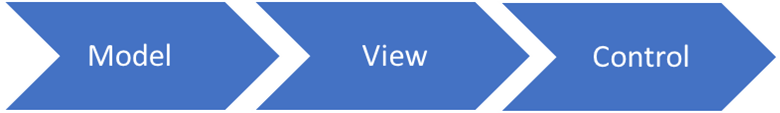
The model-view-controller pattern proposes three main components or objects to be used in software development:

- Model, which represents the underlying, logical structure of data in a software application and the high-level class associated with it. This object model does not contain any information about the user interface.

- View, which is a collection of classes representing the elements in the user interface (all of the things the user can see and respond to on the screen, such as buttons, display boxes, and so forth)

- Controller, which represents the classes connecting the model and the view, and is used to communicate between classes in the model and view.

My project is based on Model – View – Controller patter, so I divided my classes into 3 packages



* **Model** – contains the two main data structures used: Monomial and Polynomial classes
* **View** – is more of a “hidden package” because of the fact that I used JavaFX for the interface, which  
  loads a .fxml file using FXMLoader from the UserInterface class, then launches the project in main  
  So the View “package” would be composed of the UserInterface class and the calculator.fxml file
* **Control** – Controller class which connects the interface with the model

1. *Class Design*

I will shortly describe each method from each class from the 3 packages in order to get a better understanding of how I chose to implement the operations

**MODEL**

* **Monomial Class**

A polynomial is composed of one or more terms, which in mathematics are called monomials.

* + public Monomial() - default constructor (initializes power and coefficient with 0)
  + public Monomial(int power, Number coefficient) – constructor
  + public Number getCoefficient() – getter for the coefficient
  + public void setCoefficient(Number coefficient) – setter for the coefficient
  + public int getPower() – getter for the power
  + public void setPower(int power) – setter for the power
  + public void displayMonomial() – displays the monomial in the system console
  + public void getStringMonomial() – transforms the monomial into a string which can be read easily by the user. By far the most complex function in this class, because I tried my best to cover all special cases to make the output as beautiful as possible (ex: 1X^1, X^0, 0X)
* **Polynomial Class**

This class has only one variable which consists of a ArrayList<Monomial>. I keep the monomials ordered in the list from the one with the highest power to the one with the lowest.

* + public Polynomial() - default constructor
  + public ArrayList<Monomial> getPolynom() – getter for the polynomial
  + public Polynomial(String s) – the main constructor, here I use pattern matching in order to construct a polynomial from a given string
  + public void displayPolynomial() – displays the polynomial in the system console
  + public void sortByPower() – function used to sort the ArrayList by the power of the monomials
  + public void simplifyPolynomial() – function that combines the monomials with the same power
  + public String getStringPolynomial() – transforms the polynomial into a string which can be read
  + public Monomial findPower(int power) – I used this function in most operations in order to check if I already have a monomial with the same power. For example, for addition, if this returns null, then I just add the new monomial, if it returns a monomial, I add the result to that
  + public Polynomial duplicatePolynomial(Polynomial p) – function that creates a copy of a given polynomial
  + public Monomial findNotZeroCoeff() – I used this function for realizing the division operation, to check if I have monomials with coefficient != 0
  + public Polynomial add(Polynomial p) – performs addition
  + public Polynomial subtract(Polynomial p) – performs subtraction
  + public Polynomial multiply(Polynomial p) – performs multiplication
  + public ArrayList<Polynomial> divide(Polynomial p) – performs division, I use ArrayList<Polynomial> because I return both the quotient and the remainder from the division
  + public Polynomial integrate() – performs integration
  + public Polynomial derivate() – performs the derivative

**CONTROL**

* **MainController Class**

I implemented this class from the interface Initializable and I override the method and set actions for each of the buttons for the operations.

I have a static method createAlert() which I call in the constructor of the polynomial, if the pattern matcher detects other symbols other than the allowed ones. And another function which gets the strings from the textFields and calls the Polynomial constructor for them.

I also have some error checking and Alerts which I implement in this method as well

* If the user enters other characters other than x, ^, - +, and numbers

Graphical user interface, application

Description automatically generated

* If the user presses derivative or integration I kindly ask them which one they would like to perform the operation

Graphical user interface, text, application, chat or text message, email

Description automatically generated Graphical user interface, text, application

Description automatically generated

**VIEW**

Graphical user interface, application, Teams

Description automatically generated

In this class I just load the calculator.fxml file, then I launch everything.

Text

Description automatically generated

# Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| What I test | Input Data | Expected Output | The effective result | Pass / Fail |
| Addition | P1 : 4x^5-3x^4+x^2-8.5x+1  P2 : 3x^4-x^3+x^2+2x-1 | +4X^5 -X^3 +2X^2 -6.5X | +4X^5 -X^3 +2X^2 -6.5X | Pass |
| Subtraction | P1 : 4x^5-3x^4+x^2-8.5x+1  P2 : 3x^4-x^3+x^2+2x-1 | +4X^5 -6X^4 +X^3 -10.5X +2 | +4X^5 -6X^4 +X^3 -10.5X +2 | Pass |
| Multiplication | P1 : -3.5x^2-X+1  P2 : x-2 | -3.5X^3 +6X^2 +3X -2 | -3.5X^3 +6X^2 +3X -2 | Pass |
| Division | P1 : x^3-2x^2+6x-5  P2 : x^2-1 | Q: +X -2  R: +7X -7 | Q: +X -2  R: +7X -7 | Pass |
| Differentiation | 6x^5-3x^4+6x^2-7x+1 | +30X^4 -12X^3 +12X -7 | +30X^4 -12X^3 +12X -7 | Pass |
| Integration | 6x^5-3x^4+6x^2-7x+1 | +X^6 -0.6X^5 +2X^3 -3.5X^2 +X | +X^6 -0.6X^5 +2X^3 -3.5X^2 +X | Pass |

The testing scenarios have been described using use cases in the beginning of the documentation. For the automatic testing I used Junit.

# Conclusions

The application was developed and tested in InteliJ. From the point of view of the algorithms I used, they are the basic ones taken from mathematics and implemented in Java, so they should be fast enough even for heavy uses.

The application is an user friendly application and pretty helpful for checking results for different operations on polynomials, as long as the user introduces the polynomials in the established convention and is familiar with how the operations work. As the application is developed on a Java platform it is highly portable and can be run on several operating systems as long as the user has Java SDK installed.

For my personal experience, it was a very good exercise in remembering the OOP concepts learned in the first semester, but also learning new ones. I found it very useful and challenging at the beginning because it took me a lot of time just to figure out what exactly is Maven and Junit. So time management was crucial because I found myself in the last week before the project and I barely managed to set-up the proper environment.

Another important thing which I noticed was how much time is wasted because I didn’t have a clear path of implementing the classes from the beginning and I found myself changing a lot of things on the go.

Overall the project was very helpful for me in both areas of OOP and also math because the subject of polynomials was removed from the final exam in high school, so I had to learn the long division algorithm from scratch.

# Future improvements

For the moment the polynomial calculator is able to compute 6 basic operations, but this can be improved by adding new functionalities, such as:

* Compute the square of the polynomial
* Find the roots of the polynomial
* Plot the graphic of the polynomial

And many more that I can’t think of right now.

# Bibliography

* 1. Object-Oriented Programming - Lecture Slides of prof. Marius JOLDOS
  2. Programming Techniques – Lectures of prof. Cristina Bianca Pop
  3. [www.stackoverflow.com](http://www.stackoverflow.com)
  4. [www.math.stackexchange.com](http://www.math.stackexchange.com)
  5. [www.wikipedia.org](http://www.wikipedia.org)
  6. <http://whatis.techtarget.com/definition/model-view-controller-MVC>
  7. <https://www.draw.io/>