**DOCUMENTATION HOMEWORK 2**

**QUEUES SIMULATOR**

**Chindea Tudor Eugen**

**Grupa 30423**

**Profesor Laborator Assist Antal Marcel**

Contents

[1. Functional Requirements 3](#_Toc67007389)

[2. Objectives 3](#_Toc67007390)

[2.1. Principal Objective: 3](#_Toc67007391)

[2.2. Secondary Objectives: 3](#_Toc67007392)

[3. Problem Analysis And Use Cases 4](#_Toc67007393)

[3.1 Analysis 4](#_Toc67007394)

[3.2 Use Cases 4](#_Toc67007395)

[4. Design 5](#_Toc67007396)

[4.1. Data structures 5](#_Toc67007397)

[4.2. Class diagram 6](#_Toc67007398)

[4.3. Algorithms 6](#_Toc67007399)

[5. Implementation 7](#_Toc67007400)

[6. Testing 8](#_Toc67007401)

[7. Conclusions and Future Development 8](#_Toc67007402)

[8. Bibliography 8](#_Toc67007403)

# Functional Requirements

Build a queue simulator which can compute the following operations:

* Reading number of clients, Queues, minimum and maximum Arrival and service times.
* Simulate Q queues that run in parallel and perform the tasks necessary for each client.

# Objectives

## Principal Objective:

Propose, build and test a system that is capable of doing basic operations of polynomials with one variable. The polynomials will have integer coefficients and positive integers at the degree. The polynomials will be written in descending order in order to be computed correctly.

## Secondary Objectives:

|  |  |  |
| --- | --- | --- |
| **Secondary Objectives** | **Description** | **Chapter** |
| Development of use cases and scenarios | The application should be able to deal with many types of input and many scenarios. The input should be easy and natural for the user to type. | 3 |
| Choosing the data structures | The right data structures enable an easy manipulation and computation of our resulting polinom. For example the coefficient of each term of the polinom was stored as a double to enable the computation of the division, integration and derivation. | 4 |
| Division into classes | Dividing the methods into separate classes makes the code easier to understand | 4 |
| Algorithm development | The algorithm implementation was simple once the data was stored properly | 4 |
| Solution implementation | A method was written for each arithmetic operation | 5 |
| Testing | For each operation, one or more tests were done to show that the algorithms are functional. The output format needs to be known before comparing the expected and computed results. | 6 |

# Problem Analysis And Use Cases

## 3.1 Analysis

The fundamentals of our problem are the monomial and the polynomial. In mathematics, a monomial is, roughly speaking, a polynomial which has only one term. A monomial, also called power product, is a product of powers of variables with nonnegative integer exponents, or, in other words, a product of variables, possibly with repetitions. For example, {\displaystyle x^{2}yz^{3}=xxyzzz}{\displaystyle x^{2}yz^{3}=xxyzzz} is a monomial. The constant 1 is a monomial, being equal to the empty product and to x0 for any variable x. If only a single variable x is considered, this means that a monomial is either 1 or a power xn of x, with n a positive integer. If several variables are considered, say, {\displaystyle x,y,z,}x,y,z, then each can be given an exponent, so that any monomial is of the form {\displaystyle x^{a}y^{b}z^{c}}x^{a}y^{b}z^{c} with {\displaystyle a,b,c}a,b,c non-negative integers. In our case, only monomials of one variable will be used.

In mathematics, a polynomial is an expression consisting of variables (also called indeterminates) and coefficients, that involves only the operations of addition, subtraction, multiplication, and non-negative integer exponentiation of variables. An example of a polynomial of a single indeterminate x is x2 − 4x + 7.

The operations are applied on polynomials of one variable. The sum is done by merging the two initial polynomials so that the output will already be a sorted array depending on the degree of each term. The substraction uses the method of addition but the second polynomial will first be multiplied by -1. Integration and Derivation are only done on one of the two polynomials that can be inserted. The multiplication is done by multiplying each term of the first equation with all other terms in the second equation. The resulting polynomial will be sorted in descending order after the degree of each monom. Polynomial long division is an algorithm that implements the Euclidean division of polynomials, which starting from two polynomials A (the dividend) and B (the divisor) produces, if B is not zero, a quotient Q and a remainder R such that A = BQ + R, and either R = 0 or the degree of R is lower than the degree of B. These conditions uniquely define Q and R, which means that Q and R do not depend on the method used to compute them. The result R = 0 occurs if and only if the polynomial A has B as a factor.

## 3.2 Use Cases

Use Case Name: Sum of polynomials.

Actors: Person using the application.

Triggers: The user presses the button that adds two polynomials that were already typed.

Preconditions: The user has typed the polynomials in the designated space. The user must also press the button Save Polinomials in order to store the polinoamials that he typed in.

Post Conditions: The data will be split in designated data structures and the chosen arithmetic function will be applied. After the computation of the result a string representing the sum of polynomials will appear on screen.

Normal Flow:

1. The user types in the first textbox the first polynomial on which the operation will be done.
2. The user types in the second textbox the second polynomial on which the same operation will be done.
3. The user will confirm that the wanted input was typed by pressing the button Save Polynomials.
4. If the data that was typed by the user does not match the structure for a polynomial that is supported by the program, a warning will appear on screen. The data will not be stored until the input is typed in correctly.
5. The user presses the button for addition and the output will be displayed.

Use Case Name: Integration of polynomial.

Actors: Person using the application.

Triggers: The user presses the button that one polynomial that was already typed.

Preconditions: The user has typed the polynomial in the designated space. The user must also press the button Save Polinomials in order to store the polinoamial that he typed in. The polynomial must be typed in the first textbox (for the f(x) function. Integration function only works on the input from the first textbox.

Post Conditions: The data will be split in designated data structures and the chosen arithmetic function will be applied. After the computation of the result a string representing the integrated polynomial will appear on screen.

Normal Flow:

1. The user types in the first textbox the first polynomial on which the operation will be done.
2. The user will confirm that the wanted input was typed by pressing the button Save Polynomials.
3. If the data that was typed by the user does not match the structure for a polynomial that is supported by the program, a warning will appear on screen. The data will not be stored until the input is typed in correctly.
4. The user presses the button for integration and the output will be displayed.

# Design

## Data structures

The data will be stored in a form such that it would be easy to interpret it as a polynomial and also to be ease to perform mathematical operations on each term. The input String will be turned into a polynomial. The polynomial is an ArrayList of monomials and all methods are done one polynomials. Each monom has a degree stored as an integer but the data will be represented only by natural numbers. The coefficient of a monomial is stored as a double and no monomial will have a coefficient 0.

## Class diagram

Diagram

Description automatically generated

## Algorithms

Addition is done by merging the two polynomials. We iterate through the two polynomials simultaneously in one loop and the term with the biggest degree of one polynomial will be added to the sum, the iterator of that polynomial will increase. If for both polynomials the current monomials have the same degree, both elements will be added to the sum. The first loop ends when we have iterated through all monomials of at least one of the polynomials. If we didn’t get to the end of the other polynomial, we iterate through the remaining elements and add them to the sum. The sum will already be sorted in descending order by the degree since that is how we iterated through our equations.

The substraction is done by multiplying the coefficients of all monomials from the second polynomial with -1. We use the addition method on the polynomials

Multiplying the polynomials will be done with two for loops embedded. We multiply each monomial of the first polynomial with the monomials of the second polynomial. We sort the resulting polynomial in descending order.

The division of the polynomials will result in a quotient and a rest. The rest will only have value 0 if the second polynomial is a common divisor of the first. We can use the division of we only divide by a non 0 polynomial. The rest will the take the value of the dividend. The quotient will be zero. While the rest is not 0 and the degree of the rest is greater than the one of the dividend we do the following operations: First we divide the leading term of the rest with the one in the divisor an store the result. Second we take the result and add it in the quotient. Third step is to remove from the rest our term obtained in the first step multiplied with the divisor. At the end we will have the quotient and the rest.

Integration is one of the operations where the coefficient of each monomial might get a value different from an integer. We take one monomial at a time and increase the degree by one and divide the coefficient by the obtained degree.

To derivate a polynomial we iterate through the monomials and decrease the degree if it is greater than zero and multiply the coefficient with our initial degree. The monomial with degree 0 will be deleted.

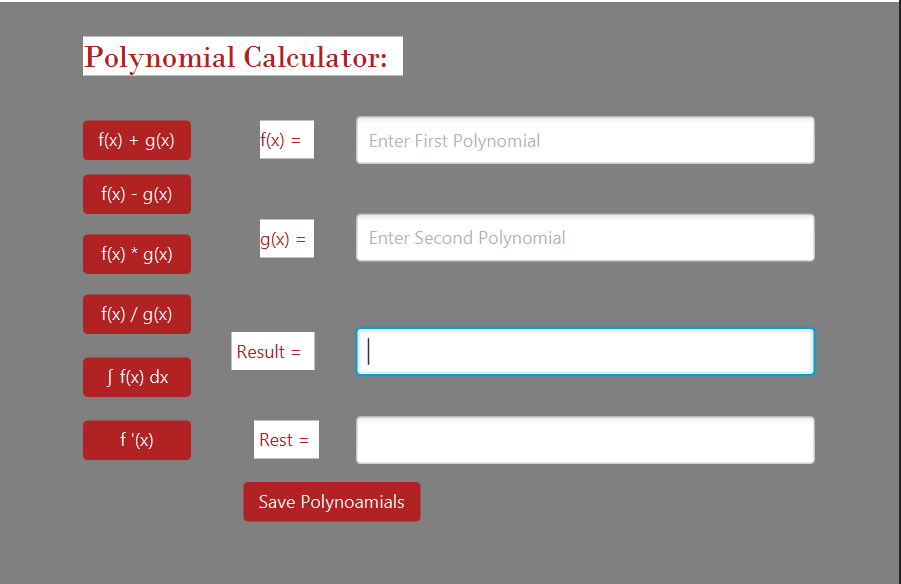
# Implementation

The Monom class is our lowest level class. It stores each element of each polynomial in its structure. A method compareTo is overwritten in the monom class, this method compares the degree of the this. Monom with one given as a parameter and returns the difference between the two, this method is used for sorting when doing the multiplication.

The Polinom class has most of the methods used in the program. The addition, substraction, division and integration work as described in the algorithms above. For the multiplication we use the method sort from Collections with an overwriden method in the class Monom. In the division method, inside the loop mentioned in the algorithm, we use at the end the method deleteZero on our rest. The deleteZero method removes all elements with coefficient 0 from the polynomial given as parameter. Method makeString is used to to convert an object of type Polinom into a string so that it can be used for testing and as output. This method works by iterating through a polinomyal and adding the coefficient in a degree to a String with the characters “x^” in between and a “+” sign before each coefficient smaller than 0. If the first element of the resulting string is a +, we remove that particular character. The method strToPoli converts a String to an object of type Polinom. We replace every ‘-‘ sign with a “+-“ sign in order to use a tokenizer after the “+” sign. After parsing regarding the “+” sign, we put each element in a monomial. This method is used after validating the input after the regex.

The validation uses pattern matching and a regex to tell us if the user has typed in the correct input. This is done in the Controller class. Also in the Controller class we have the actions of all the buttons and all the calls for the methods from Polinom.

The user interface is very intuitive. There are two textboxes where the user types in the polynomials in the form of “-x^4+3x+3” or”x^-2” or others. If the input does not match this format, the input will not be stored and the user will be warned. After typing a valid input, the user presses the button Save Polynomials and he will be able to select from the other buttons, what operation he wants to do. If the user chooses integration or derivation, this will be done only on the first equation. If the user wants to do a new computation, he has to press the save polynomials button again.



# Testing

For each operation one or more test were done. The results were compared as strings so after each operation a conversion had to be done. All six operations pass various tests.

# Conclusions and Future Development

This application is useful for any person that needs to do basic operations on polynomials. The UI is simple with the use of buttons and the keyboard.

For future development several functionalities might be added to the program. Plotting the exuations might be one of those. A variable number of polynomials might be added for some operations, not just two.

# Bibliography

1. Laboratory and Course material
2. <https://en.wikipedia.org/wiki/Polynomial_long_division>
3. <https://beginnersbook.com/2013/12/java-arraylist-of-object-sort-example-comparable-and-comparator/>
4. <http://tynerblain.com/blog/2007/04/09/sample-use-case-example/>
5. <https://topic.alibabacloud.com/a/use-a-regular-expression-font-colorredregexfont-to-match-polynomials-polynomial-regexpolynomial_1_31_32668196.html>