DOCUMENTATION

ASSIGNMENT *1*

STUDENT NAME: BOTA ALIN ALEXANDRU

GROUP: 30423

# CONTENTS

[1. Assignment Objective 2](#_Toc128043139)

[2. Problem Analysis, Modeling, Scenarios, Use Cases 2](#_Toc128043140)

[3. Design 4](#_Toc128043141)

[4. Implementation 6](#_Toc128043142)

[5. Results 12](#_Toc128043143)

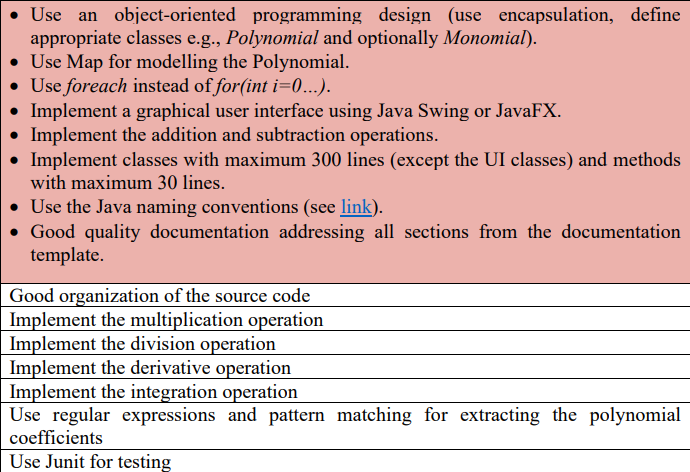
[6. Conclusions 13](#_Toc128043144)

[7. Bibliography 13](#_Toc128043145)

# Assignment Objective

The objective of the assignment is to design and implement a polynomial calculator with a dedicated graphical interface through which the user can insert polynomials, select the mathematical operation (i.e., addition, subtraction, multiplication, division, derivative, integration) to be performed and view the result.

The sub-objectives of the polynomial calculator assignment are presented in the table bellow:



# Problem Analysis, Modeling, Scenarios, Use Cases

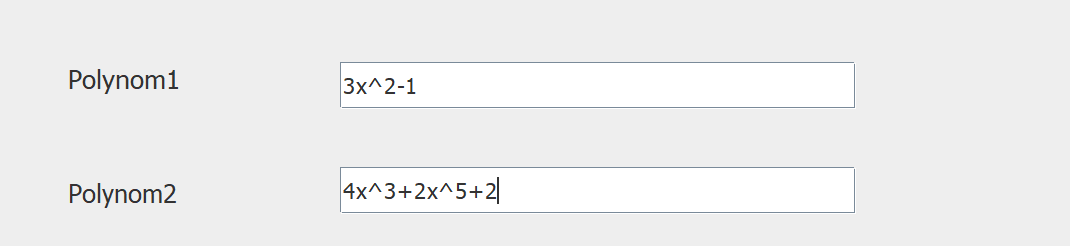
By problem analysis we refer to the process of identifying, defining, and understanding a problem in order to find a solution or a set of solutions. We clearly have an advantage because we’re using an object oriented language. It allows us to work on a superior, conceptual level, without being constrained by the technical work.

This strategy is also known by the bottom-up design. It has a big advantage because we can easily find structures that are connected in a way with the real world(objects). Clearly this concept doesn’t have only advantages, it also comes with a high level of complexity.

The application is going to be used by different kinds of people, which tells us that the start point should be the implementation of an interface that is easy to use and convenient.

We know that the Polynomial Calculator needs to implement basic algebraic operations which determines that minimum two different fields are necessary for each polynomial. The one who uses the calculator can introduce simple strings from the keyboard.

The one who uses it should use the first to labels to introduce the polynomials in the form : (coefficient x^ power). It’s very important that the monomials should not be separated by a space from their sign. An example can be: 3x^2+2x^1-2.



Also the user must know that the program has a problem when the coefficient is equal to 1 and is written in the form: “x” instead of “1x”.

The order that the polynomials are introduced should not matter because the application will rearrange the polynomials in descending order by power.

# Design

In the first picture we have a representation of the fields and dependencies of the classes. All variables are private.

Diagram

Description automatically generated

The **Calculator View** Class -> With the help of this class we create the user interface for the whole application. With the tools: Java Swing which is a specific library, and JFrame which is an interface, the user can easily use the program without needing to know coding or technical work.

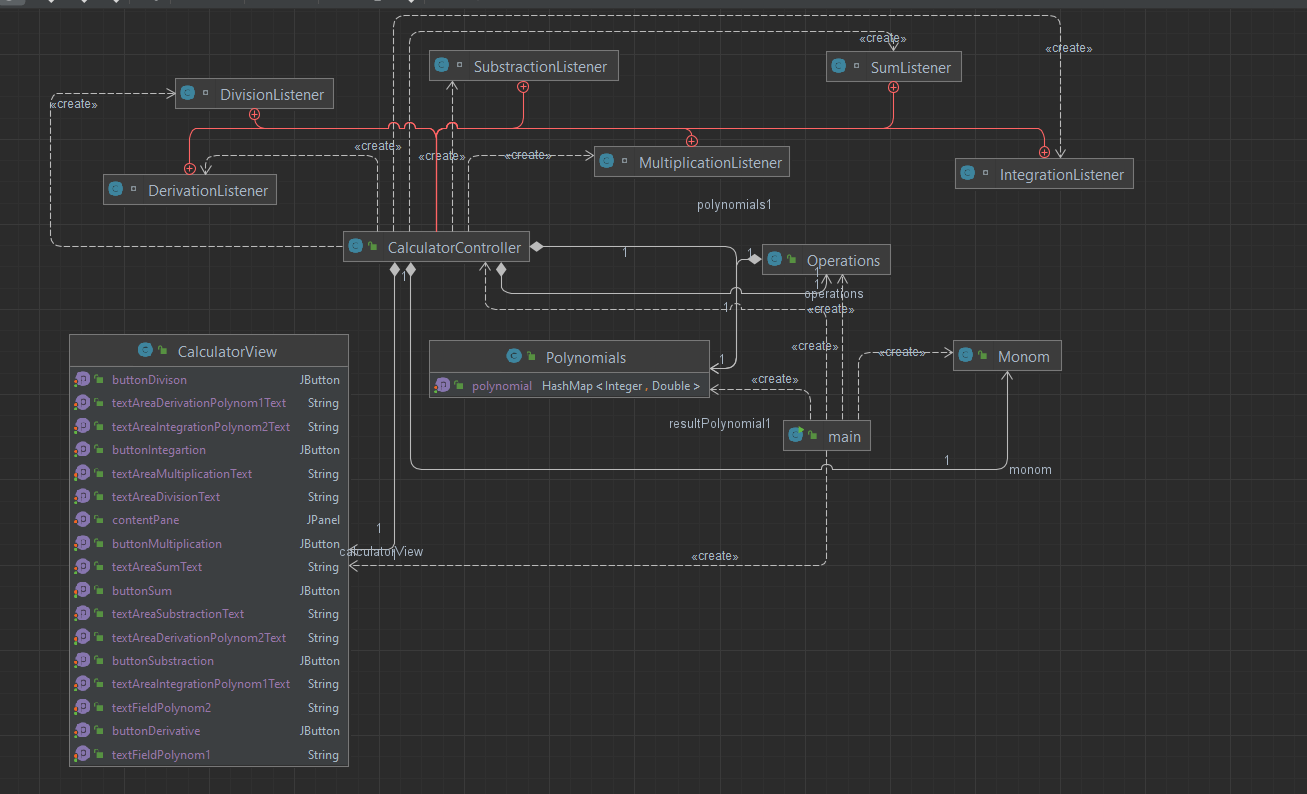
The **Polynomial** Class -> Is mainly used to memorize the polynomials that were introduced with the help of the **Calculator View** class. It is also responsible with a method that displays the output after the inputs are modified.

The **Calculator Controller** Class -> Is responsible for the events happening when the user provides an input or interacts with the application. Also provides a method that splits the string Inputs into separate monomials.

The **Monom** Class -> Simply a class that provides the method to convert monomials from the **Calculator Controller** and then memorize the coefficients and powers in the **Polynomial** class.

The **Operations** Class -> Takes the main responsibility of providing methods that make algebraic operations such as addition, subtraction, multiplication etc.

The constructors of the objects and methods:



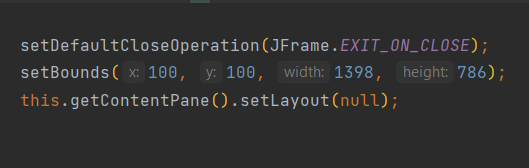
Properties and inner classes : Graphical user interface

Description automatically generated

# Implementation

* 1. *Calculator View Class*

Provides a JFrame which builds the entire interface of the application.



All the buttons, text Areas, labels etc.... are private and all the code that builds the interface is implemented in the constructor of the **Calculator View** Class. The constructor Is then called in the main class to provide visibility and functionability. Also I created setters and getters for each variable to get easy access in other classes when I need to provide output or get information from the inputs.



To make the interface functional we need to implement listeners to the buttons :

Text

Description automatically generated

* 1. *Calculator Controller Class*

Is responsible for the functionality of the buttons. When an event in the application is happening(the user interacted), this class will be responsible to do the expected action. The constructor of this action is conncted to the Calculator View Class.

Text

Description automatically generated

Also provides the read method -> takes the input that was put in the application by the user and splits it into strings that represent monomials.

Text

Description automatically generated

* 1. *Monom Class*

This class is only used for the transform method which takes the Input read from the **Calculator Controller**, converts the coefficient into Double and the power into Integer and then puts the converted results in a Hash Map provided by the **Polynomials** class.

* 1. *Polynomials Class*

This class is the representation of a polynomial, represented in a Hash Map, with the key of type Integer which corresponds to the power and value of type Double which is the coefficient.

Firstly, it provides the method: addinHashMap which takes the coefficients, and the powers as parameters, and puts them in the HashMap. If the polynomial has many coefficients of the same power the method will add them automatically.

*Text

Description automatically generated*

Secondly it provides the method, that displays the output. The parameter called type is necessary to be specified because will determine in which text Area the output will be displayed. Each operation has a different output field. The method also uses transformToString, which converts all the elements from the Hash Maps into Strings and concatenates them with the signs that correspond to their coefficients.

Text

Description automatically generated

* 1. *The Operation Class*

This class is specifically used to store algebraic operations methods. In this case the methods that were used are: addition, subtraction, multiplication, division, derivative and integration.

We’ll start with the addition method:

Text

Description automatically generated

Firstly, I used a for each instruction, to iterate through all the elements from the first polynomial’s Hash Map. If by this iteration I find that the power of an entry is equal to the power of another element from the second polynomial, I add the two key values and save the result in a resultPolynomial variable. If there’s no matching power, the other elements are put in the variable without being added. Also, I made a separate iteration for the second polynomial and put all the entries that weren’t added in the first iteration.

The subtraction method is almost identical to the addition one. The only difference is that we substitute the plus with a minus when were found two elements with the same power.

Text

Description automatically generated

The multiplication method is very straight forward. We iterate the first polynomial and multiply the current entry with all the entries from the second polynomial. The answer is then saved in a resultPolynomial variable.

Text

Description automatically generated

The integration method is used for each polynomial individually. We iterate the polynomial, and if the power is greater or equal to 0, the integration of the entry is performed. When the integration is finished the power increases with 1 and the coefficient is then divided with the power + 1.

Text

Description automatically generated

The derivation method is pretty similar to the integration. The difference is that when the derivation is performed the power is decreased with one and the coefficient is multiplied with the power.

Text

Description automatically generated

The last on the list and the most complex one is the division method. The implementation used for this algorithm is the Long Division. The code performs the following steps : find the highest degree term of the polynomial1(dividend) and polynomial2(divisor), calculate the coefficient and power of the next term in the quotient Hash Map, add the next term to the quotient, multiply the divisor by the next term in the quotient, subtract the result from the previous step from the polynomial1 to get a new reminder, and finally remove any terms with zero coefficients from the remainder Hash Map.

Text

Description automatically generated

# Results

The testing was made through the interface that was implemented in the main class and by Junit testing.

Graphical user interface

Description automatically generated

This is an example of all the outputs that can be provided through these inputs.

The Junit was used only for addition and subtraction cases which are tested in an almost identical way. We declare an expected Hash Map and create two variables of type Polynomial who store some values. The use of the assert Equals method, and the addition one provided by the Operation class will confirm if the test was successful.

Text

Description automatically generated

It is important to note that some inputs may not be the best ones for the applications. Some criterias that I found useful to the well being of the application by testing:

-“3x^2-2x” not “3x ^ 2 - 2x”. The spacing between the elements may harm the program.

-use “1x” in the case of coefficient 1 not only “x”.

***Without some clear exceptions the program is fully functional and ready to be used.***

# Conclusions

The part that challenged me the most in the process of creating the application was the conversion from the String input -> splitting the String by the specific sign and then memorize the converted results in a Hash Map. In this way I learned some tricks in using the Split method. The same goes for converting again in String(for the purpose of displaying) where I found what String Builder is and how to use Decimal Format. I also learned the Long Division algorithm and how to implement it in IntelliJ.

In future developments I could improve the UI of the application and the input system. Also, a switch button to modify the input order of the polynomials sounds interesting.

# Bibliography

* <https://stackoverflow.com/>
* <https://www.geeksforgeeks.org/>
* <https://www.youtube.com/watch?v=H62Jfv1DJlU>
* https://www.youtube.com/watch?v=\_Ph9WsCm4Uk&t=1710s