DOCUMENTATION

ASSIGNMENT *1*

Polynomial Calculator

STUDENT: VĂLCĂUAN LAURA-MARIA
GROUP: 30422

TABLE OF CONTENTS

1.The purpose of the paper 2

2.Problem analysis, modeling, scenarios, use case 3

3.Design 3

4.Implementation 5

5.Conclusions 7

6.Bibliography 7

1. The purpose of the paper

The paper presents the design and the implementation of a polynomial calculator using a graphical interface which will help the user to introduce the polynomials on whom to apply different operations (addition, subtraction, multiplication, addition).

Secondary purposes:

·The implementation of ompital algorithms for each operation (addition, subtraction, etc.)
·The parsing of the string polynomial input into the polynomial that will be used to make the desired operation.
·Generating the correct result and in the case of division, the correct result and remainder.
·The input from the user through the graphical interface and also displaying the output on the graphical interface.

2. Problem analysis, modeling, scenarios, use case

A polynomial may be represented using array or structure. A structure may be defined such that it contains two parts – one is the coefficient and second is the corresponding exponent.

The polynomial calculator will receive through input from the text fields two polynomials which will be modified such that the compiler can use them. The polynomial result will be displayed on the results text field and in the case of the division the remainder will be displayed on the remainder text field. The calculator allows the user to introduce the polynomial in which order he/ she prefers.

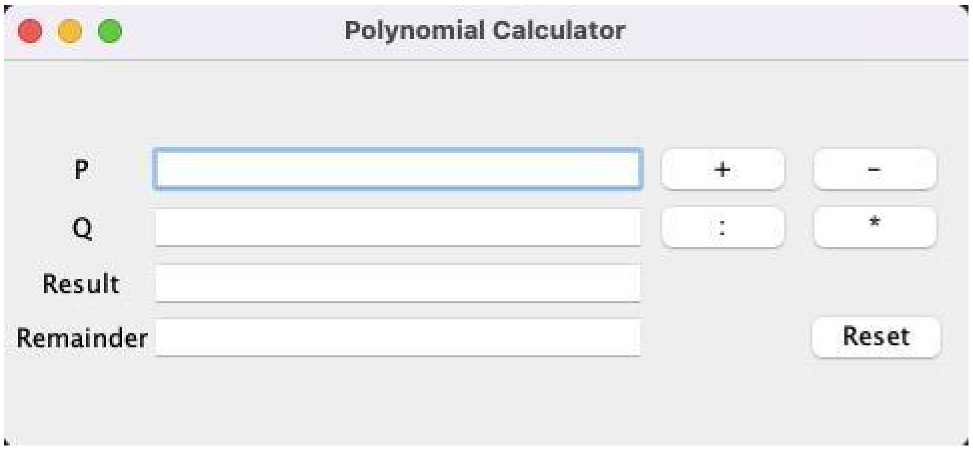

Figure 1 - The interface

3. Design

The design of the application is based on the Model-View-Controller model.

Model-View-Controller (MVC) software design pattern commonly used for developing user interfaces that divide the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted from the user.

Traditionally used for desktop graphical user interface (GUIs), this pattern became popular for designing web applications. Popular programming languages have MVC frameworks that facilitate implementation of the pattern.

The Model package contains everything related to the polynomial operation and also the polynomial parsing, in few words here is the “mathematical” soul of the program and it’s heart, the polynomial converting from string to polynomial. The Model packages contain the classes Monom, Polinom and PolynomialParser which are responsible for the polynomial parsing. And it also contains an Operation class, where are implemented all the operations available for the user to use, the “mathematical” soul from a few lines earlier. The operations which the user can use are addition, subtraction, multiplication and division. The data structures used are represented through classes Monom, Polinom and Operation. In the case of a polynomial is represented as a list of monoms. A monomer is represented through it’s coefficient and exponent. In the Operation class the method division returns a hashmap that contains two polynomials, the quotient and the reminder.

The View package contains the View class that is used to design the graphical user interface. That means that in the View class is the styling part of this program, where every text field, label, button is defined and each of them is placed in the right position so the interface looks more friendly for the user. For the graphical user interface is used Java Swing. For the organization of the main panel is used GridBagLayout which leads to GridBagConstraints, which was used for the text fields, labels and buttons positioning on the panel. For each button is added an ActionListener which leads to the Controller class (therefore to the Controller package) where the ActionPerformed is implemented.

As the name suggests the Controller package (and Controller class) represents the manipulation part of this program more precisely the manipulation of the button. For that is used an override method actionPerformed that through a switch case specifies to the compiler what to do if a certain button was pressed.

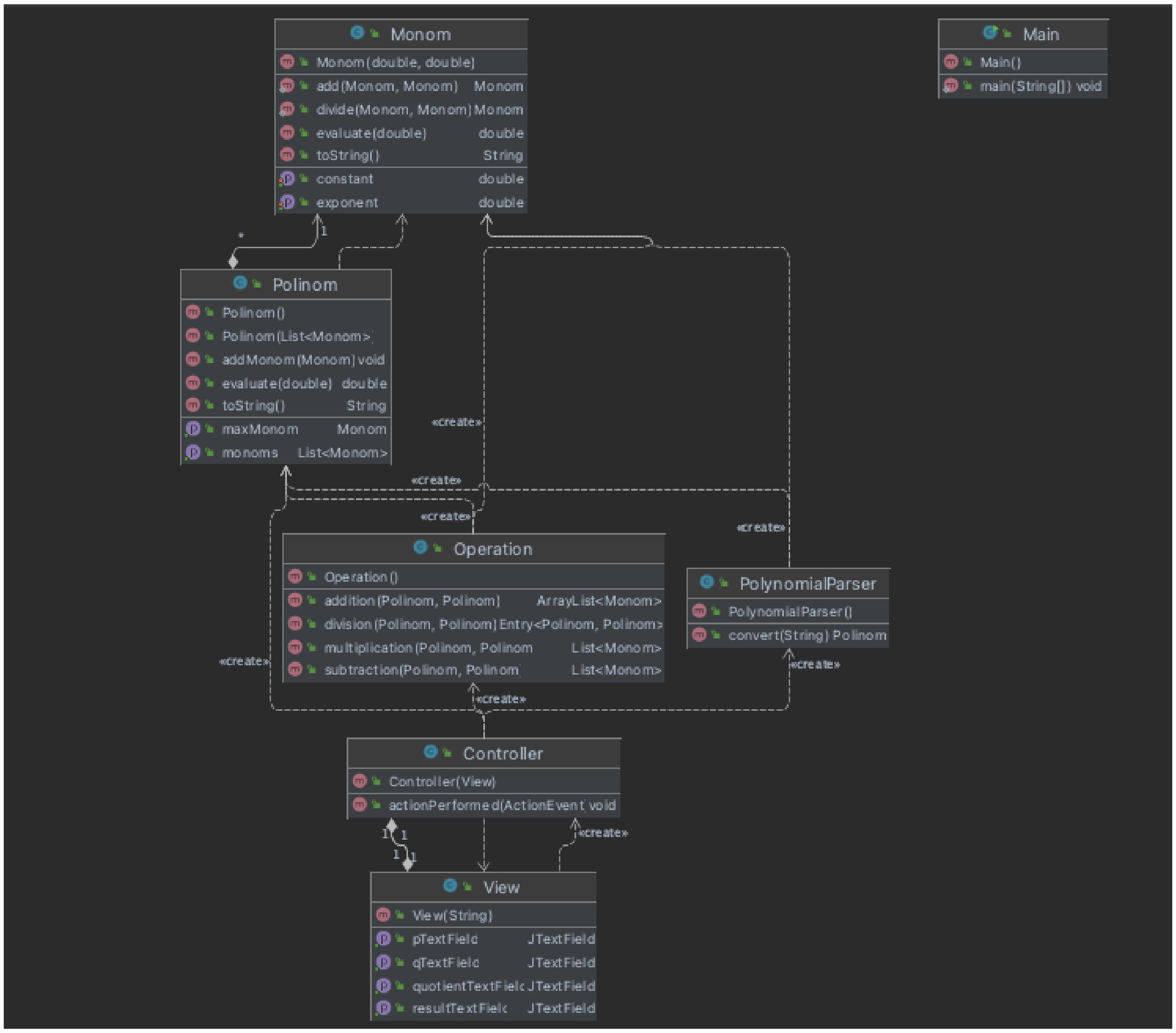

Figure 2 - The UML diagram of the classes

4. Implementation

The Monom class contains as attributes a coefficient and an exponent of double type. The coefficient and the exponent are double type because it is necessary for the division operation and moreover is more precise for each of the operations. This class contains 4 methods: evaluate, add, divide and toString. The evaluate method was more used as a tool for the programmer which helped to determine if the polynomial and the parsing was a success. A monomial is an algebraic expression that has only one term. The basic building block of a polynomial is a monomial. A monomial is one term and can be a number, a variable, or the product of a number and variables with an exponent. The number part of the term is called the coefficient.The add method is used for the division operation; somewhere in the algorithm is needed to sum up two monomers and this also applies to the divide method, which divides two monomers and the result is used in the division algorithm. The toString method helps the compiler to make a difference from the string "X^" and to work only with it's exponent and coefficient. In this method is also treated the case where the constant is 1 and the exponent is zero, it helps the user instead of writing "1X^2" he/ she can easily write "X^"; also in the exponent case the user does not need to write "X^0" he/ she can easily write the constant or leave the polynomial in the desired form.

The Operation class contains as it was mentioned earlier the mathematical part of the program, the operations that the user can apply. In this class is a method for each operation and each of them takes as arguments two polynomials. The addition method algorithm consists in two traversals, one in which we verify the monomers with the same exponent and then we add them; the second traversal verifies if there are left terms and we add them to the sum. The subtraction method follows the same principle as the addition but instead of adding the terms we subtract them. The multiplication method travers the polynomial and multiplies each constant (coefficient) while adding each exponent of the monomers. The division method firstly finds the maximum monomers from both polynomials and then divides them, after this subtract the reminder with the dividend and we repeat the process until the exponent of the divider is bigger or equal with the exponent of the dividend.

The Polinom class contains lists of monomers and three methods: evaluate, getMaxMonom and toString. The evaluate method is used as an insurance for the programmer that the work is done well. The getMaxMonom method as the name suggests returns the maximum monomer from a polynomial and this method is called for the division. The toString method is a part of parsing, where the monomers are converted into strings.

The PolynomialParser class as the name suggests converts the strings into polynomials so that the compiler can work with the operations implemented. And that was about the Model package.

The Controller class contains as attributes an instance of the View class and one of the Operation class. The Controller class implements the ActionListener interface and overrides the ActionPerformed method in which controls the actions on the buttons when they are pressed.

The View class contains as attributes 5 buttons, for addition, subtraction, divisio, multiplication and reset. It also contains two labels for the polynomials and two labels for the result and for the remainder (in case of division); and four text fields after each label where can be input/ displayed the polynomials.

5. Conclusions

By working on this project I have learned so many things and I hope I will be able to optimize and to accomplish a better understanding of them. For the beginning I did not even hear of Model View Controller model and it sure made my life easier on the organization part of the project. I have learned to parse a string into a polynomial, to use regeX especially. Also, and I am a little ashamed of this because it was not my first time with Swing, I did not know about GridBagLayout together with GridBagContstraints and all I can say after working with it is that it is a life savior.

I have also learned that every case is important and needs to be treated because it can really damage your work if you do not consider every scenario. I know it does not have to do with the project meaning code implementation and so one but it was a part of it which is the Git. I am so glad that I finally know how to use it.

But the main thing I have learned is that I will never leave work in the last week :) .

6. Bibliography

·https://www.khanacademy.org/math/algebra2/x2ec2f6f830c9fb89:poly-div/x2ec2f6f830c9fb89:quad-div-by-linear/v/polynomial-division
·https://stackoverflow.com/questions/420791/what-is-a-good-use-case-for-static-import-of-methods
·https://stackoverflow.com/questions/9103226/getting-user-input-by-jtextfield-in-java
·https://docs.oracle.com/javase/tutorial/uiswing/components/textfield.html
·https://perso.ensta-paris.fr/~diam/java/online/notes-java/GUI/components/30textfield/11textfield.html
https://stackoverflow.com/questions/13415573/how-to-extract-polynomial-coefficients-in-java
·https://www.orekit.org/site-orekit-10.0/jacoco/org.orekit.data/PolynomialParser.java.html
https://www.dreamincode.net/forums/topic/144180-parsing-a-polynomial/
·Java swing GUI tutorial #20: GridBagLayout - YouTube
·https://docs.oracle.com/javase/tutorial/uiswing/layout/gridbag.html#:~:text=java%20.,necessarily%20have%20the%20same%20height.