Polynomial Calculator

STUDENT NAME: Deac Denisa Bianca

CONTENTS TABLE

1. Objective

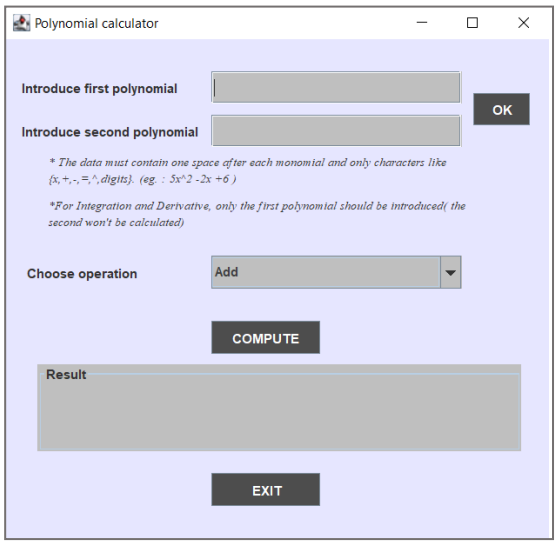
The first assignment for Fundamental Programming Techniques has the main objective of creating a Polynomial calculator that can perform several operations on 2 polynomials. The purpose is to design and implement a calculator for polynomials, represented as a list of monomials given from the user through a graphical user interface. The operations performed by the calculator are addition, subtraction, multiplication, division, integration, and derivative. All these operations will be detailed in the following sections, in chapters 2, and their implementation in 3.

The result will be displayed on the 3rd field after the polynomials are introduced and the operation is chosen (by pressing ok and compute buttons).

2. Problem analysis, modelling, scenarios, use cases

A polynomial is an expression of the form a0+ a1*x+a2*x^2+…+ an*x^n, representing a list of monomials (this is the way it is also implemented). The application has a user interface, created using Java Swing through which the user will give the inputs from the keyboard by writing them on the text fields.

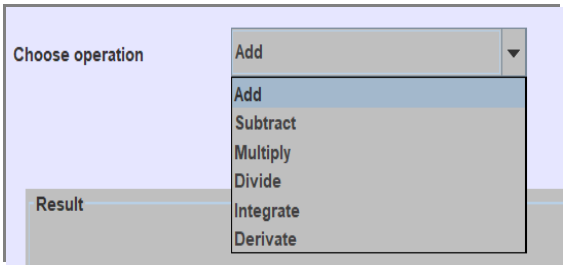
The application starts with the graphical user interface, looking like in the following figure. It should be used as follows:



- The inputs are introduced by writing them on the first and the second text fields, from the keyboard. The labels situated in the left-hand side of the text fields work as instructions for the user; other necessary instructions are displayed bellow the fields (about the format of the polynomials).
- After the input is given, by pressing the "OK" button on the right-hand side, the polynomials will be introduced and transformed from string to Polynomial type, ready for the computations.
- The field bellow represents a combo box displays the available operations from which the user can chose the desired one by pressing it. The operation should be chosen from the combo box only after the polynomials are introduced.
- The button "COMPUTE" will start the calculations and will display the result on the next field called "Result".
- The following text filed, named "Result" is used to display the result after the computation.
- To exit the application the user can press the "x" button of the window or the "EXIT" one at the bottom of the page.

The polynomials should be introduces as said in the instructions: with a space after each monomial, without '*' sign (for multiplication). If the monomial has coefficient 1, degree 0 or degree 1, it shouldn't be written (e.g. 2^x2 -x^2 +4x).

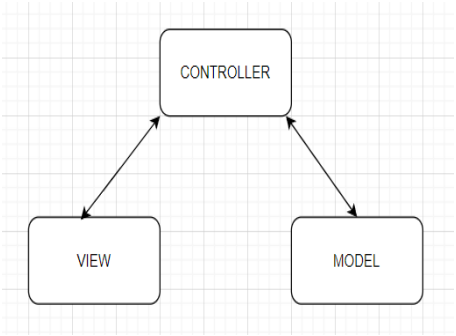
Also, is not mandatory to give the polynomial sorted by degree in descending order, the application will sort it before working with it, after it was transformed into a polynomial.

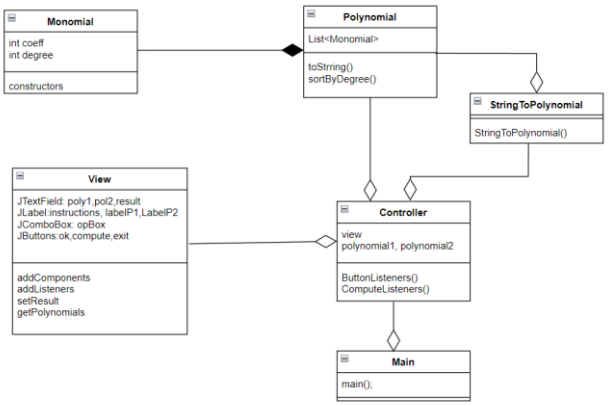


3. Design

The calculator is constructed using MVC architecture and is organized in 3 main packages: Model, View, Controller(to respect the architecture). The Model contains the polynomial and monomial classes (also the Double one used for Integration) and the "Operations" package containing all the operation classes (each class contains the method "calculate" which computes the necessary

calculation). The View represent the graphical user interface and the controller instantiates the View and the Model in order to make the connections between them.





The data structures with which I have been working in this problem are either primitive data types, especially integers and doubles, and a more complex one, such as ArrayList type object or new created object such as Monomial and Polynomial. To be able to have accurate result for integration, I created 2 classes very similar to the first 2 described, except that the coefficient of the monomial is double (named DoubleMonomial, DoublePolynomial). The details about them are found in the following chapter.

In matter of interfaces, my classes implemented ActionListener to link the buttons on the GUI with their functionality and the Comparable interface to sort polynomials.

4. Implementation

My project consists of 4 packages, 3 of them "creating" the applicaton: Model, View, Controller, since it respects the MVC architecture. All 3 packages contain multiple classes. For all the classes, the attributes are private, while the methods are public so they can be accessed from the other packages.

The Model contains information about data, in this case meaning the Polynomial, Monomial, DoublePolynomial, DoubleMonomial classes and another package, called "Operations" where there is a class for each individual operation.

The Monomial and DoubleMonomial (see *Figure 1*) classes have as attributes the coefficient and the degree of a monomial, being integer or double (for the different classes).

Polynomial and DoublePolynomial (see *Figure 2*) classes take as attribute an ArrrayList of Monomials, respectively DoubleMonomials, and have implemented, beside the constructor, getters and setters, 2 other methods: toString() (which is overridden) and sortByDegree(). The first one is used to display the result polynomial as a string. The second one mentioned is used to sort the polynomial in descending order of the degree, in case the user isn't giving the input that way. Also, it is used after multiplication to sort everything, since after multiplying element by element, the degrees may be in another order.

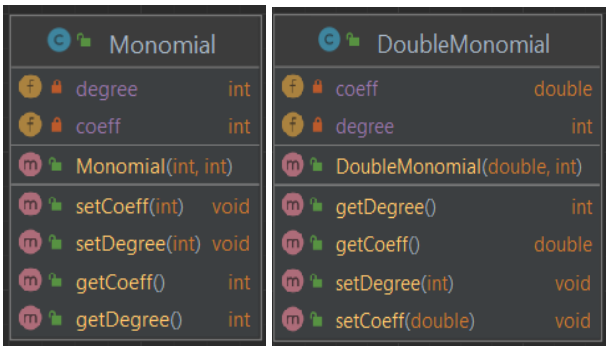


Figure 1

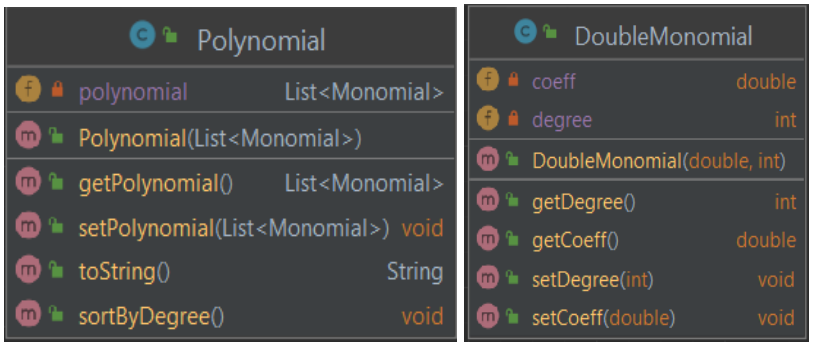


Figure 2

Inside the Model package is the Operations package. It contains a class for each operation, implementing a method called "calculate" to compute the result(see *Figure 3*).

Beside the operations, there is another "auxiliar class" called "StringToPolynomial" that implements two methods (one for Polynomial and another for DoublePolynomial) that is used to transform a string taken as input from the text field into a Polynomial, in order to be able to compute the necessary calculations. This methods technically represent the reading of the input polynomials.

It works by splitting the string by spaces and '+' sign and verifying each character and dealing with different possibilities. For example, if the degree is 0, or the coefficient is 1, if there is a negative sign and so on.

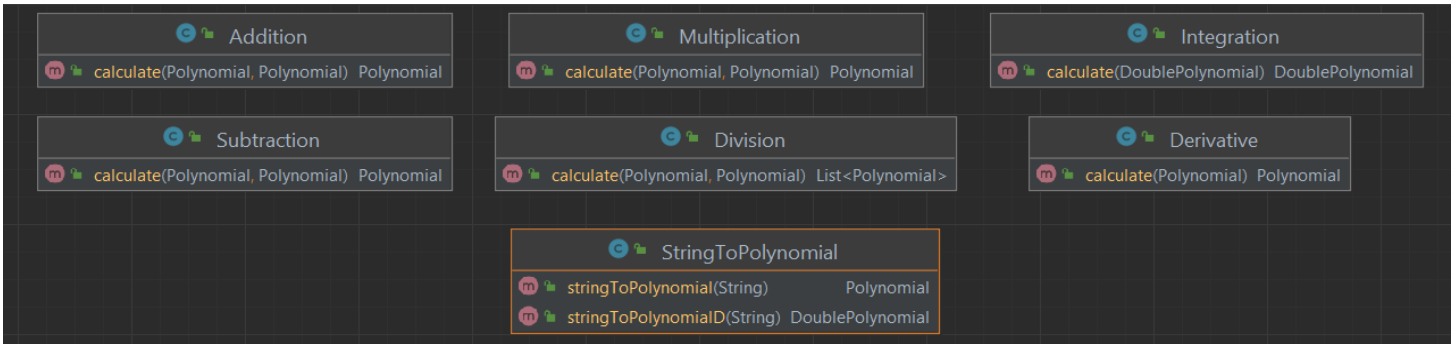


Figure 3

Figure 4

The View (see *Figure 4*) package has only one class, having the same name, where the graphical user interface is implemented by using Java Swing. The class extends JFrame and takes as attributes different components from Swing, like: JButtons, JTexteFiels, JLabels, JComBox, all being static and private. The class contains methods for each type of component, called in the constuructor to add the components (e.g. addButtons, addPolyLabels etc.). The methods that do not configurate the components are used to set ActionListeners that are added only in the Controller when the method is accessed. These are added to the buttons and the comboBox that gets the operation. The class also have some auxiliary methods that return a button, a string from the input or sets the text in a textArea (like for result).

The Controller package contains a class with the same name and the Main.

The Controller(see *Figure 5*) accesses the data from the Model and the View. It add the listeners to the components in the view and manipulates data and methods from the Model. In this class there are used the ActionListeners classes. By using them, the input is taken when the ok is pressed and its transformed into polynomials to be ready for the computations. Then the buttons are pressed the coresponding operation is called from its class in the Model.Operations and the calculate method is used.

In this class, the method checkPolynomial uses pattern matching to check if the polynomial doesn't have any unwanted characters and displayes a dialog message of type error in case one of the polynomial is wrong.

The Main class instantiates the Controller for the program to run.

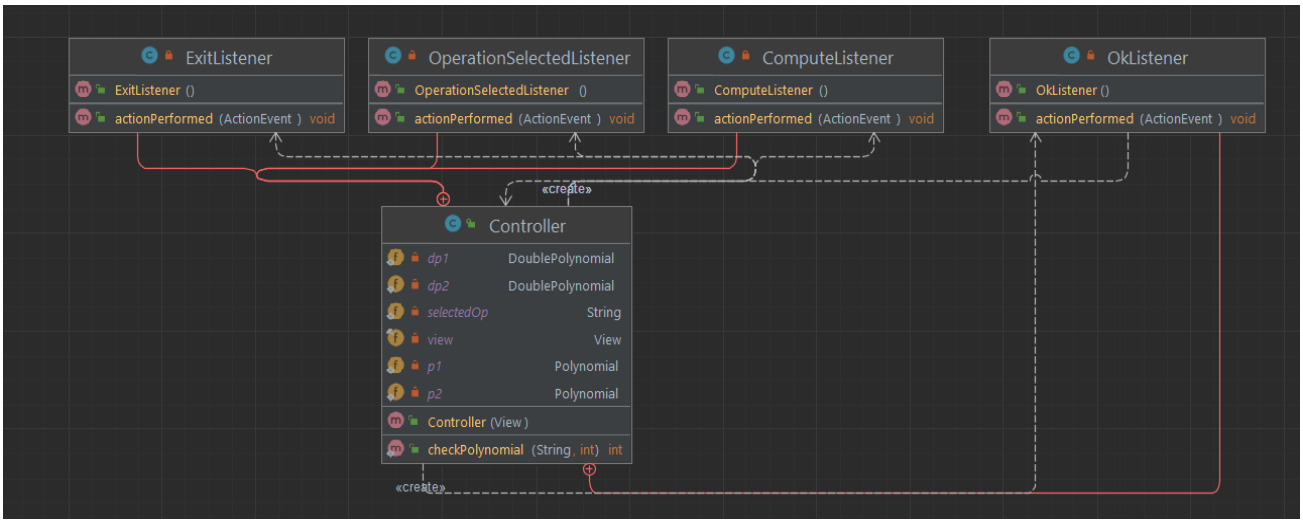
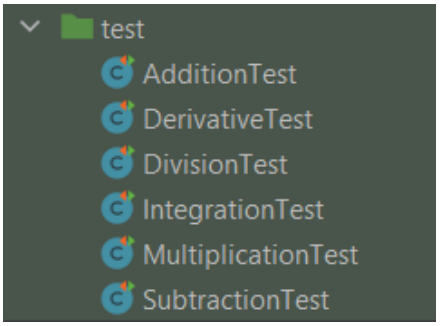


Figure 5

5. Results



The 4th package is called “test” and it has classes for each operation (see *Figure 6*). By using JUnit, the “calculate” method from each individual class is tested. Each class represent a test, has the name of the operation followed by the word “test”.

Practically, all of them are testing the calculate method for each type of operation.

The polynomials are given as string and by using StringToPolynomial it is transformed and ready for the calculations. By giving the polynomial this way, the StringToPolynomial is also tested.

In each class the string was given in such a way that is contains positive and negative coefficients, equal to 1, greater than 1, degree 0 and 1. This way all possible cases were covered.

The testing was done using assertEquals() . All tests passed successfully.

```
import static junit.framework.TestCase.assertEquals;

public class AdditionTest {
    @Test
    public void addTest(){

        Addition addOp = new Addition();
        StringToPolynomial st = new StringToPolynomial();

        //case with all positive values
        Polynomial p1 =  st.stringToPolynomial( a: "2x^2 +5x +1");
        Polynomial p2 = st.stringToPolynomial( a: "x^2 +7");
        String result = "3x^2 +5x +8";
        assertEquals(result, addOp.calculate(p1,p2).toString());

        //case with negative values and different coefficients
        p1 = st.stringToPolynomial( a: "-x^4 +2x^2 -1");
        p2 = st.stringToPolynomial(("-8x^2 -1"));
        result = "-x^4 -6x^2 -2";
        assertEquals(result, addOp.calculate(p1,p2).toString());


    }
}
```

In the next figure is shown the test for Addition operation which instantiates 2 polynomial, by calling the method StringToPolynomial. Using “assert”, the result string is compared with the result of the operation done on the two polynomials.

6. Conclusion

Creating this Polynomial calculator was a pleasant way of consolidating my knowledge on OOP and Java syntax. I learned a lot more about ActionListener, MVC architecture, working on strings and overall java language.

In my opinion this type of project is a creative and interesting way of learning programming language and also a important step in getting used to work like a programmer and think like one (a good exercise for a future job in this domain).

I came to the conclusion that even if a task may seem easy, a lot of work and attention must be put into is because it is full of details and layers of particular cases.

Even though my application is not perfect, I could see a progress by each day I worked on it.

A good way of improving my application would be to use generic classes for Monomial and Polynomial, this way the DoubleMonomial and DoublePolynomial classes would not be necessary. I think my calculator needs more dialog messages in case of error. Also, there are still some operations that could be performed on polynomials, like getting the roots of the polynomial, rising to a power

7. Bibliography

- https://stackoverflow.com/questions/8895337/how-do-i-limit-the-number-of-decimals-printed-for-a-double
- https://www.javatpoint.com/how-to-sort-arraylist-in-java
- https://www.tutorialspoint.com/display-multiple-lines-of-text-in-a-component-s-tooltip-with-java
- https://towardsdatascience.com/everything-you-need-to-know-about-mvc-architecture-3c827930b4c1#:~:text=-MVC%20is%20an%20architectural%20pattern,the%20view%20whenever%20data%20changes.
- https://ro.wikipedia.org/wiki/Model-view-controller
- https://github.com/utcn-oop-grupa1/class-materials
- https://www.youtube.com/watch?v=we3zJE3hlWE&t=21s
- https://stackoverflow.com/questions/34946528/decode-polynomial-from-string-with-pattern-and-matcher