Fundamental Programming Techniques

Assignment no.1

Polynomial Calculator

Husman Andrei,

Group 30421

**1.Objectives.**

**I. Main Objective:**

The main objective of this assignment is to implement a polynomial calculator which is capable of computing the following operations: addition, subtraction, division, multiplication, integration and differentiation.

**II. Secondary Objectives:** The main objective can be divided into secondary objectives that can be achieved on their own, but put together they represent the final solution for the assignment. These secondary steps are the following:

1. **Creating Data Structures (4.Implementation)**

The operands to be stored in the data structures (polynomials) were decomposed into monomials and each polynomial was actually a list of monomials. This helped with performing the operations.

1. **Formatting and loading the information into the data structures (4.Implementation)**

The input data was a introduced as strings and it was needed to format it in order to introduce it into monomials.

**3. Performing the operations**

**a. Addition/ Subtraction (4.Implementation)**

The addition operation was implemented in such a way that it takes every monomial in the first polynomial and checks if in the second polynomial there are monomials with the same degree. If it finds 2 monomials with the same degree it adds their coefficient and stores it. If It does not, it stores the monomial. Then, it takes the second polynomial and does the same thing comparing the monomials in it with the monomials in the first polynomial. After that we call the method “filterZero” to get rid of the monomials with coefficient 0.

The subtraction operation was implemented in such a way that it takes every monomial in the first polynomial and checks if in the second polynomial there are monomials with the same degree. If it finds 2 monomials with the same degree it subtracts their coefficient and stores it. If It does not, it stores the monomial. Then, it takes the second polynomial and does the same thing comparing the monomials in it with the monomials in the first polynomial, but the difference is, if it does not find a monomial with the same degree in the first polynomial, it stores it with a different sign (“-“ if it was “+” and vice versa) . After that we call the method “filterZero” to get rid of the monomials with coefficient 0.

**b. Multiplication (4.Implementation)**

The multiplication operation was implemented so that each element from the first polynomial will be multiplied with each element of the second polynomial. All these multiplications will be added then into a new, result polynomial. In order to facilitate the implementation of this operation, we will take one element from first polynomial, multiply all elements from second polynomial, and add this partial result to the final result, using the addition operation implemented earlier. Then, apply “filterZero”.

**c. Division (4.Implementation)**

The division was implemented using the algorithm to divide two polynomials taught in highschool. We make use of the subtraction operation like in the previous case, at multiplication, to make the implementation of division faster and easier.

**d. Differentiation (4.Implementation)**

The differentiation was implemented by differentiating each monomial, by multiplying it’s coefficient with it’s degree and reducing it’s degree by one. The differentiation allows us to save some memory, because we perform the operation on the same polynomial, so unlike the previous four operations, we won’t need a new polynomial in order to store the result.

**e. Integration (4.Implementation)**

The integration was implemented similarly with the differentiation. We perform the operation on the same polynomial in order to save memory. The operation is done by integrating each monomial by incrementing it’s degree with one and dividing it’s coefficient with it’s newly incremented degree.

1. **Implementing a GUI to facilitate the communication with the user (4.Implementation).**

The purpose of the GUI is to make the application easier to use for the common user, because it is a visual interpretation of the application. It contains labels, text boxes in which the user can insert a polynomial, read the result of an operation and it has buttons which once pressed can perform the operations mentioned above. It is mandatory that the GUI has to be intuitive, so that the it will be easy to use, even if the user doesn’t have any programming skills.

**2. Problem analysis**

The problem can be divided in smaller steps which will help us understand how the application works. First, using the GUI, we need to introduce the polynomials. After it is introduced, it is important to know that we introduced it as a string, so the application will parse this string and extract the coefficients and degrees of each monomial. After the polynomials are introduced, we can start doing the operations we want. The result of operations addition, subtraction, multiplication, integration and differentiation will appear in the field “Result”, while the results of division will be displayed in the fields “Result” and “Remainder”. In order for the application to work correctly, we assume that the inputs introduced are correct, each monomial respects the format “(coefficient)x^(degree)” and between each polynomial we can find “+” or “-“. However, if we forget to introduce the ^degree part, the application will still work, but it will consider that the monomial has degree 1. As we can deduce, the presence of the coefficient is mandatory. If we forget to introduce the coefficient, we will get an error.

How to do operations:

Addition: Introduce polynomials -> Press button “+”

Subtraction: Introduce polynomials -> Press button “-”

Multiplication: Introduce polynomials -> Press button “\*”

Division: Introduce polynomials -> Press button “/”

Integration: Introduce polynomial -> Press button “integr”

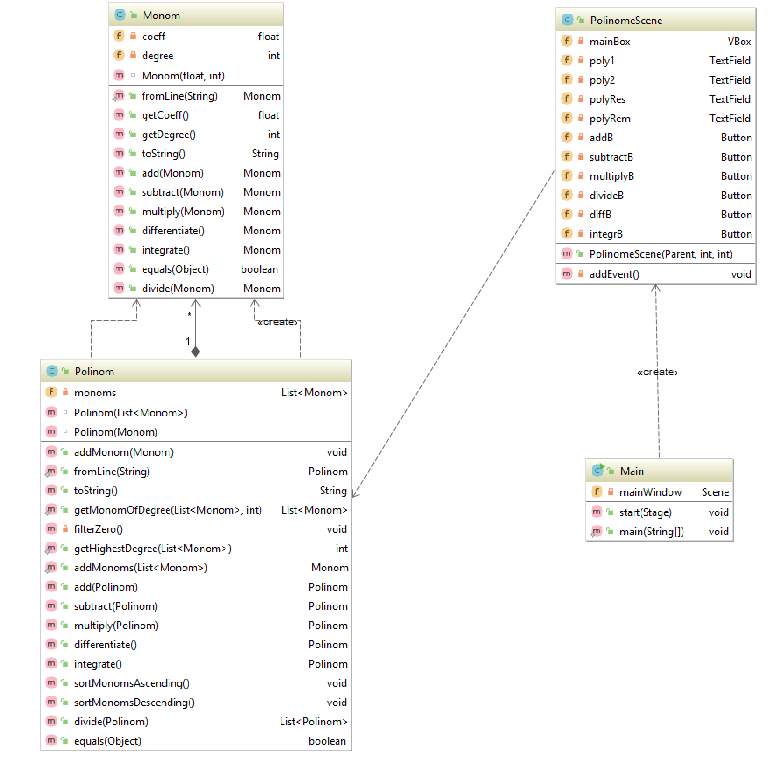
Differentiation: Introduce polynomial -> Press button “d/dx”

1. **Design**

In order for our application to have an OOP design, the application is structured in 4 major classes, which fulfill the following roles: data structures, GUI implementation, and the main class, which controls and instantiates all the other classes. The data structure classes are the Monom and Polinom classes. The Monom class implements a monomial which is the smallest independent unit of a polynomial and it has a coefficient and a degree. The Polinom is implemented as a list of monomials. It was implemented this way, because it is much easier to perform the operations and to access every separate element of it.

The GUI class is the PolinomScene, which implement the GUI.

The GUI class and the Main class is located in the UserInterface package, the Polinom and Monom classes in the Polinom package.



As we see in the UML diagram, we have 2 types of relationships in our application: aggregation relationship between:

PolinomScene and Polinom

Polinom and Monom

Main and PolinomScene

and Composition relationships between:

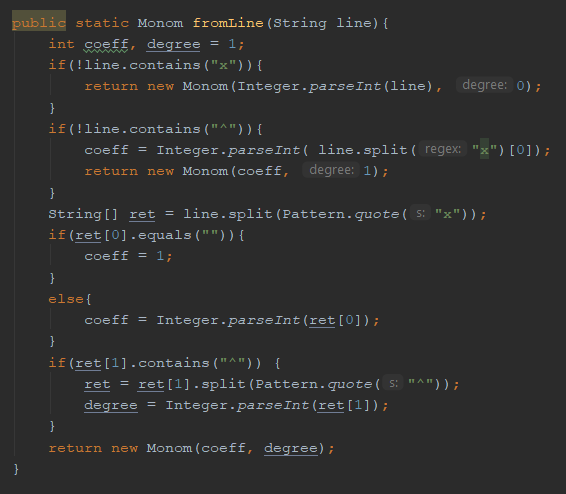
Polinom and Monom

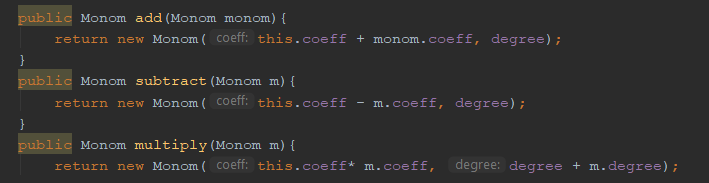
This means that the class Polinom is made of Monom objects.

**4.Implementation**

**Describing each class and their most important methods and fields.**

**I.Monom**

****

In this picture we can see the implementation of the constructor of the Monom class. It gets as parameter the string Line, and then it extracts from it the coefficient and the degree of one monomial. The first three if clauses format the expression so that inputs like “1”,”2x” or “x^2” would be accepted. The instructions that follow separate the coefficient and the degree part.

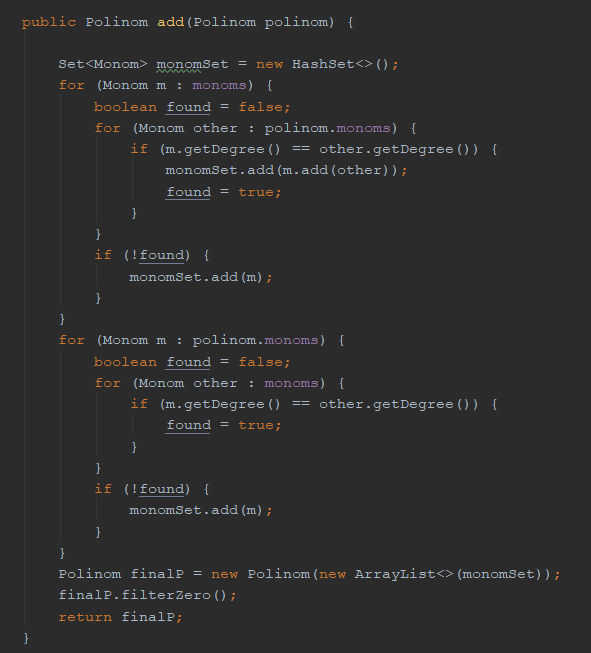
The following three methods facilitate the implementation of the addition, subtraction and multiplication operations in the class Polinom .The monom add method returns the result of the addition of two monomials, the monom subtract method returns the result of the subtraction of two monomials while the monom multiply method returns the result of the multiplication of two monomials.

**II.Polinom**

The Polinom class is made of a list of Monoms.

Presenting the 6 operations that constitute de capabilities of the Polynomial Calculator.

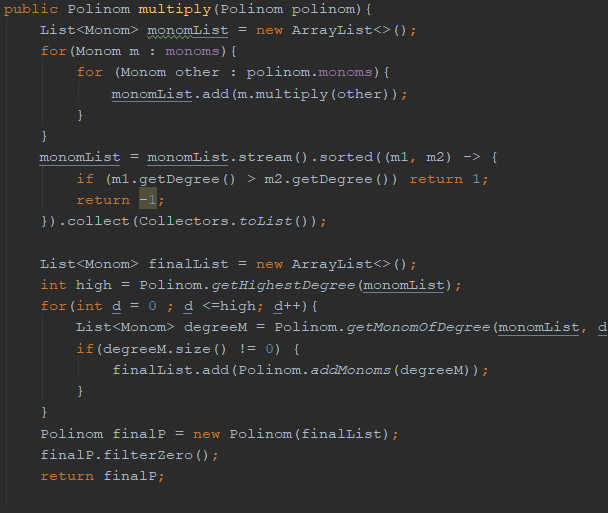
The first operation is the add operation which takes every monomial in the first polynomial and checks if in the second polynomial there are monomials with the same degree. If it finds 2 monomials with the same degree it adds their coefficient and stores it. If It does not, it stores the monomial. Then, it takes the second polynomial and does the same thing comparing the monomials in it with the monomials in the first polynomial. After that we call the method “filterZero” to get rid of the monomials with coefficient 0.



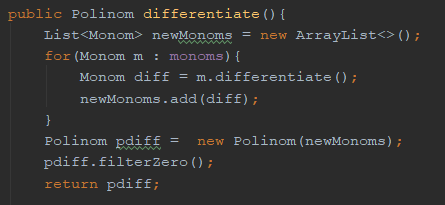
The second operation is the subtraction operation which was implemented in such a way that it takes every monomial in the first polynomial and checks if in the second polynomial there are monomials with the same degree. If it finds 2 monomials with the same degree it subtracts their coefficient and stores it. If It does not, it stores the monomial. Then, it takes the second polynomial and does the same thing comparing the monomials in it with the monomials in the first polynomial, but the difference is, if it does not find a monomial with the same degree in the first polynomial, it stores it with a different sign (“-“ if it was “+” and vice versa) . After that we call the method “filterZero” to get rid of the monomials with coefficient 0.



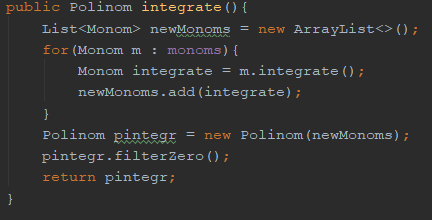
The third operation is the Multiplication operation. The multiplication operation was implemented so that each element from the first polynomial will be multiplied with each element of the second polynomial. All these multiplications will be added then into a new, result polynomial. In order to facilitate the implementation of this operation, we will take one element from first polynomial, multiply all elements from second polynomial, and add this partial result to the final result, using the addition operation implemented earlier. Then, apply “filterZero”



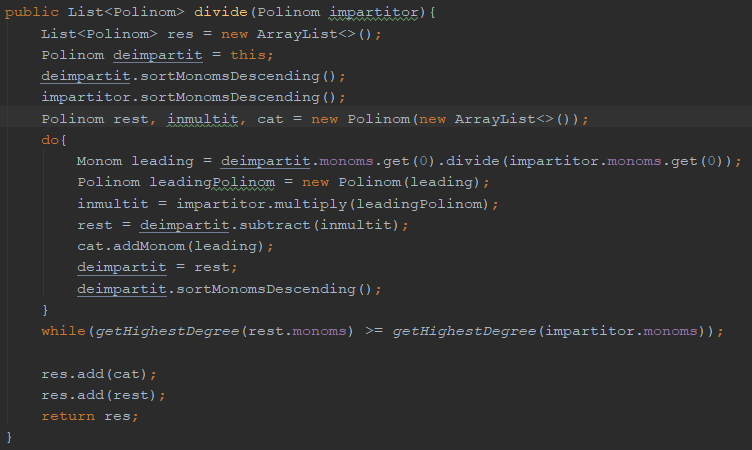
The fourth operation is the differentiate operation. The differentiation was implemented by differentiating each monomial, by multiplying it’s coefficient with it’s degree and reducing it’s degree by one. The differentiation allows us to save some memory, because we perform the operation on the same polynomial, so unlike the previous four operations, we won’t need a new polynomial in order to store the result.



The fifth operation is the integration operation. The integration was implemented similarly with the differentiation. We perform the operation on the same polynomial in order to save memory. The operation is done by integrating each monomial by incrementing it’s degree with one and dividing it’s coefficient with it’s newly incremented degree.

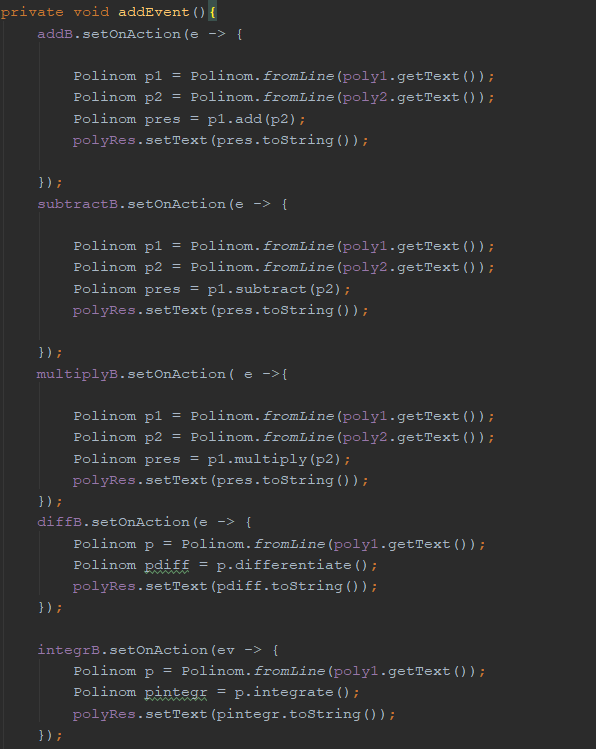


And the last one is the division operation. The algorithm used in HighSchool was implemented.



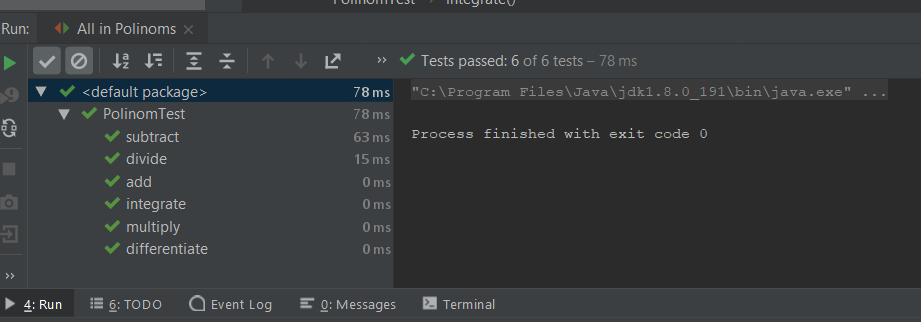
**III.GUI**

In the class PolinomeScene are placed the view parts, which means that this is the class which contains the visual elements, like labels, buttons, and text fields. Besides these, it contains also the getters and setters which manipulate the text fields and through which the information is passed to the controller. In the PolinomeScene class happens the action behind the view. It’s constructor contain two polynomial objects, which constitute the model part, and the view, through which the information is passed. Here, in the class, every event is handled, and the corresponding instructions are executed, depending on which button is pressed.



**5.JUnit Tests.**

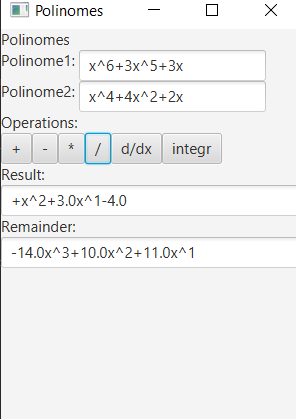
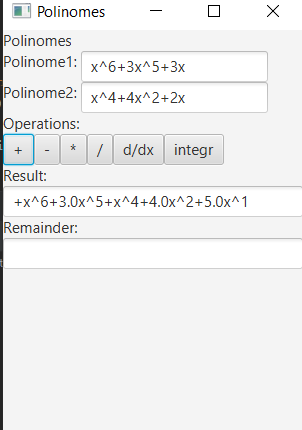
The test methods are the six operation methods. Junit Plug-in from Intellij IDE was used for testing.

****

It can be observed that the application is successful in every situation, and every operation is implemented correctly.

**6. Results**

We can see some results of operations performed with this application.



**7. Conclusions**

In conclusion, the secondary objectives that were established at the beginning of the development of the application were helpful to succeed and create a functional polynomial calculator. From the development process we can see that the decisions we took helped us. For example, the idea to implement the polynomials as lists of monomials facilitated the implementation of the operations and thanks to that the time necessary for developing the application decreased. By using the JUnit testing Plug-in the application is sure to be a correct one, which performs the desired tasks. For further development, I have the following ideas: -implementation of an input introducing way which accepts every correct case and detects the wrong inputs -making the GUI more user-friendly

**8. Bibliography**

<https://stackoverflow.com/questions/36490757/regex-for-polynomial-expression>

for the regular expression which was used at parsing the polynomial. <https://www.tutorialspoint.com/>

for Java syntax and java method related questions.

<http://www.anidescoala.ro/educatie/matematica/formule-matematice-algebra-liceu-generala/impartirea-polinoamelor/> -for implementing the division algorithm learned in highschool.