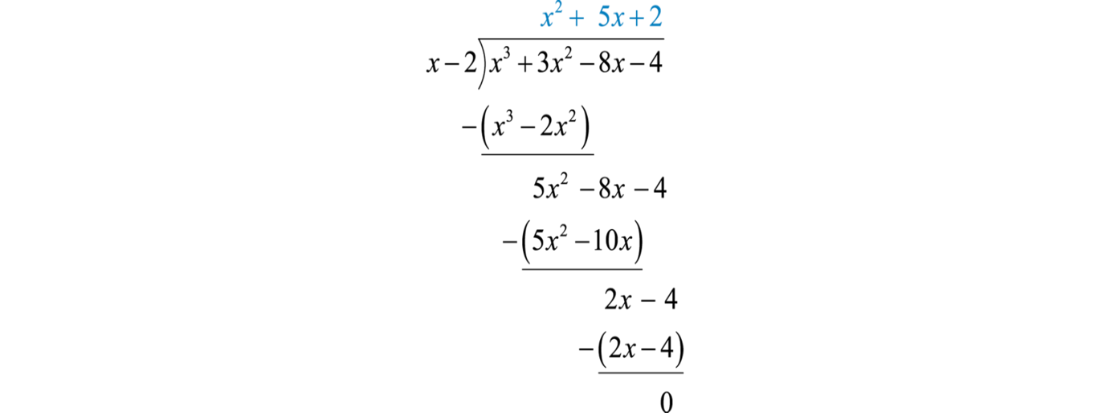
Programming Techniques

-Polynomial calculator -



Student : Giurgiu Razvan

Group : 30422

1. **Assignment objective**

The objective of this assignment is to implement a system designed for polynomial processing.Polynomials are of one variable and have only integer coefficients.

1. **Problem Analysis**

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. An example in three variables is x3 + 2xyz2 − yz + 1.

Polynomials appear in many areas of mathematics and science. For example, they are used to form polynomial equations, which encode a wide range of problems, from elementary word problems to complicated scientific problems; they are used to define polynomial functions, which appear in settings ranging from basic chemistry and physics to economics and social science; they are used in calculus and numerical analysis to approximate other functions. In advanced mathematics, polynomials are used to construct polynomial rings and algebraic varieties, which are central concepts in algebra and algebraic geometry.

The polynomials we use will only have integer coefficients. For instance a polynomial accepted as input for the application described in this document would look like this: x 2 -9x-10.

As polynomials are used in a wide variety of domains, a tool that computes different operations on this mathematical structure can be very useful.

1. **Modelling**

The process of modelling is defined as the process of modularising a big problem into smaller ones, that are easier to understand and debug. This also helps making an abstract idea more clear. In software development, modelling is essential in order to build an application that has a strong background structure. Our problem, as stated above in the “Problem Statement” paragraph, includes polynomials that we look at as a series of monomials, that is why the basic unit of our polynomial processing system will be the monomial.

1. **Use cases**

The software is a basic one and we have only basic polynomial operations as Addition, Division,Multiplication, Differentiation and Integration. Each of these opeartions can be selected from one of the buttons displayed in the GUI.

* 1. **Scenarios**

Most important thing is that the end user is able to sucessfully launch the application by pressing the run button inside the compiler ( ex. IntelliJ, Eclipse, etc. ) or by running the compiled executable inside the folder of the project.

Notes :

Data must be inserted as Coef1x^Deg1 + Coef2x^Deg2 + Coef3x^Deg3 + ..., Ceofi is the ith coefficent of the monomial and Degi is the ith degree fof the ith monomial. If these conditons are not respected the application will stop.

**Steps for the Addition Example:**

1. The app is launched.

2. The user introduces the first polynomial in the firstPolynomial text box (see the rules for introducing the input data in the program above).

3. The user introduces the second polynomial in the secondPolynomial text box (see the rules for introducing the input data in the program above).

4. The user clicks on the first button of the user interface, the one that has the word “Addition” written on it.

5. The application reads the first string given by the user.

6. The application converts the first string in an array of integer numbers.

7. The application converts each pair of numbers in a monomial, using the first number as the coefficient of the monomial and the second one as the degree of the monomial.

8. The application adds all these monomials into the first Polynomial

9. The application reads the second string given by the user.

10.The application converts the second string in an array of integer numbers.

11.The application converts each pair of numbers in a monomial, using the first number as the coefficient of the monomial and the second one as the degree of the monomial.

12.The application adds all these monomials into the second Polynomial.

13.The application computes the sum of the 2 Polynomials.

14.The application outputs the result in label below the text boxes.

15.The user closes the window.

**Steps for the Differentiation Example:**

1. The app is launched.

2. The user introduces the polynomial in the left text box (see the rules forintroducing the input data in the program above).

3. The user clicks on the first button of the user interface, the one that has theword “Differentiation” written on it.

4. The application reads the string given by the user.

5. The application converts the string in an array of integer numbers.

6. The application converts each pair of numbers in a monomial, using the first number as the coefficient of the monomial and the second one as the degree of the monomial.

7. The application adds all these monomials into the first Polynomial.

8. The application differentiates the previously built Polynomial.

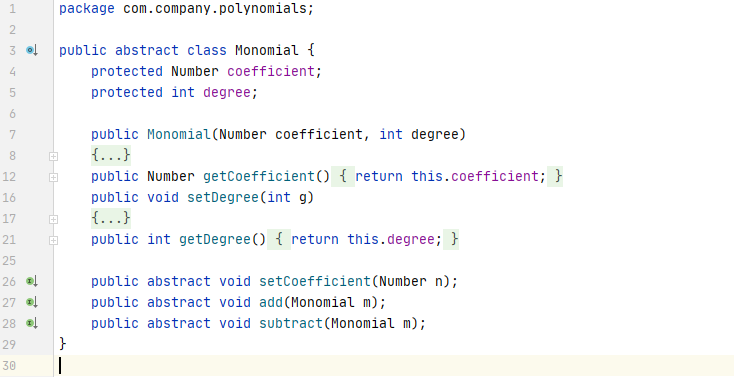
9. The application outputs the result in the resultPolynomial text box

10. The user closes the window.

11. The app stop.

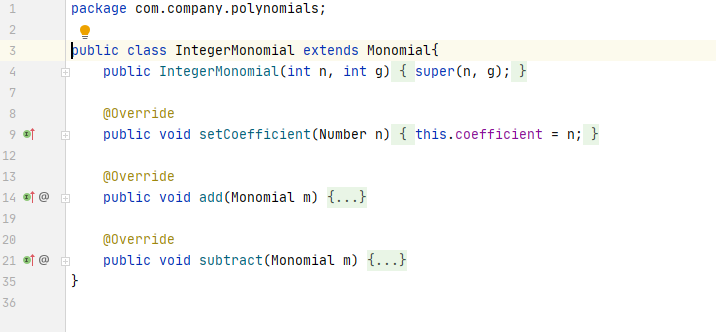
1. **Design**

**The Monomial class**



Monomial class is the main class of the whole project, it has setters and getters for the coefficent and degree of the each monomial. It also contains 2 abstract methods for adding and subtracting Monomials from the current monomial.

**The IntegerMonomial class**

****

We have here the abstract methods defined in the Monomial class.

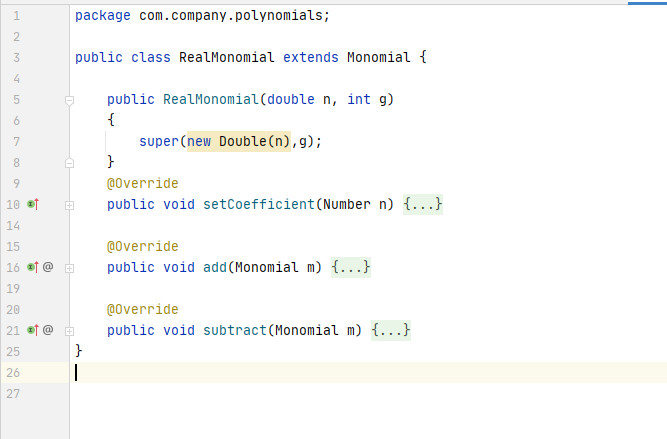
• The constructor gets two integers and set them as the coefficent and the dregree of the monomial in that way we can easily create new monomials if we know theese two values.

• The setCoefficient method takes a number as an argument and sets the coefficient of the cureent monomial with that value.

• The add method gets a Monomial as an argument and checks if this argumentis an IntegerMonomial or a RealMonomial. If it is an IntegerMonomial it adds it to the current instance of the the IntegerMonomial class. Otherwise, it calls the add function int the RealMonomial class, to make the sum as real numbers.

• The subtract method gets a Monomial as an argument and checks whether it it is or it isn’t a RealMonomial. If it is real, the function calls the subtract method in the RealMonomial class, to process the operation for real numbers. On the other hand, if the Monomial is an integer one, it multiplies its coefficient by -1 and calls the add function of the same class, as the difference between two monomials m1 and m2 equals the sum between m1 and m3, where m3 has the same degree as m2 but the coefficient is the one of m2 multiplied by -1.

**The RealMonomial class**



It implements the abstract methods defined in the Monomial class.

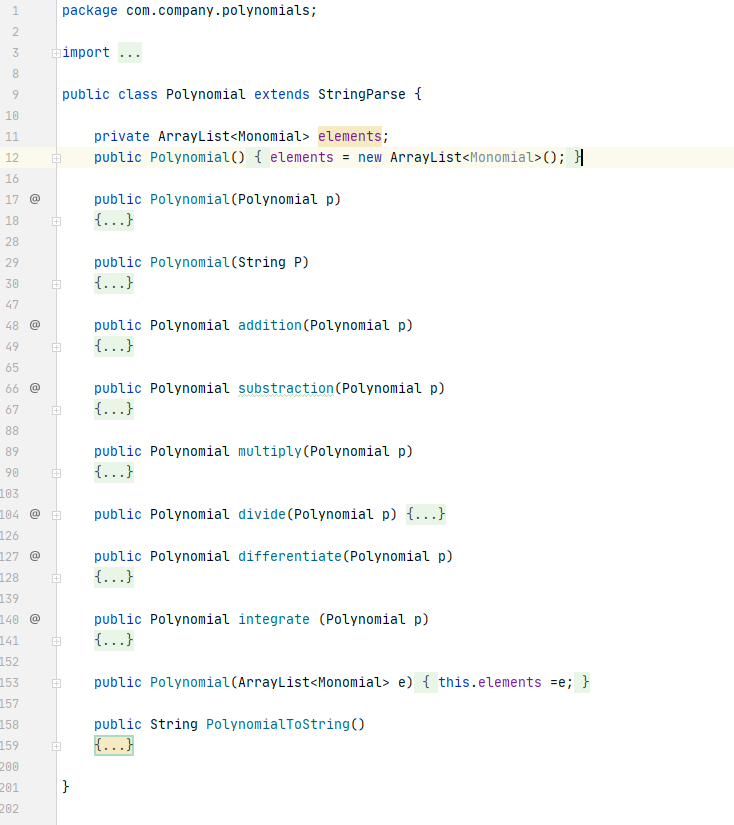
• The constructor gets a double as an argument and an integer, The double value is the coefficent of the monomial that we are constructing and the integer G is the degree of the Monomial that we are working on.

• The setCoefficent method uses a Number n as an argument and sets the coefficent of the current monomial to have a value of Number n.

• The add method gets a monomial as an argument and adds it to the current instance of the RealMonomial class.

• The substract method gets a Monomial as an argument and it multiplies its coefficient by -1 and calls the add function of the same class, as the difference between two monomials m1 and m2 equals the sum between m1 and m3, where m3 has the same degree as m2 but the coefficient is the one of m2 multiplied by -1.

**The polynomial class**

****

• Whole functionality of the project is implemented in this class by some constructors and some methods.

• „elements” is an arrayList of type Monomial. Multiple monomials form a polynomial which is stored inside the elements list that we have just declared. An ArrayList is like a List but with the implementation of a resizable array. We have some useful list operations but also some array operations that we can execute like sorting the elements in ascending / descending order or by using a sorting function. The get, set, isEmpty, size, etc operations run in constant time, all other operations run in O(n) time.

• The constructor that takes no arguments, instantiates the elements Array List with a new Array List, which is empty. (elements = new ArrayList <Monomial>();)

• The second constructor takes another Polynomial as an argument and instantiates the Polynomial being built with copies of the elements in the received Polynomial.

• The third constructor takes the string P as an argument. The string must respect the rules above ( See Notes section ) otherwise the application will stop. The constructor split the string by using a regex pattern and a matcher and we have 2 methods declared in the StringParse class which we use to get the coefficent from each matcher group by constructing the coefficient and the degree. We use the IntegerMonomial constructor to build the Monomial using the coefficient and the degree that we have just extracted using the matcher and we add that Monomial in the „elements” ArrayList, in that way we form our polynomial by repeatedly parsing each string.

• The addition method gets another Polynomial as an argument and basically computes the sum of this and that polynomial. It iterates through the elements of the polynom to be added and for each Monomial in that Polynomial, it looks for a Monomial of the same degree in the this Polynomial. If there is one the method sums up the two Monomials using the methods in the IntegerMonomial and IntegerMonomial classes. If there is no Monomial of the same degree, it adds the Monomial to the list of elements if this. The result is returned and it can also be found in this. The method does not return a sorted Polynomial.

• The subtraction method gets a Polynomial as an argument and basically computes the difference of this and that polynomial. It iterates through the elements of the polynom to be subtracted from this and for each Monomial in that Polynomial, it looks for a Monomial of the same degree in the this Polynomial. If there is one the method subtracts the Monomial from the received Polynomial from the found Monomial in this using the methods in the IntegerMonomial and IntegerMonomial classes. If there is no Monomial of the same degree, it adds the Monomial to the list of elements if this. The result is returned and it can also be found in this. The method does not return a sorted Polynomial.

• The multiply method receives a Polynomial as an argument and returns the product of this and the received Polynomial. The method creates a new Monomial for each pair of monomials, taking one from this and the other one from the received Polynomial. It then creates a new Polynomial p1 containing only the Monomial we just computed. Finally it computes the sum between the prod Polynomial (the result Polynomial) and p1. This assures us that there will be no duplicate degree after the multiplication process.

• The integrate method returns the integrated version of this instance of Polynomial. It takes each Monomial of the Polynomial and creates a new Monomial with the degree increased by 1 and the coefficient divided by the previous degree. It adds the new Monomial in the res Polynomial, which will be returned.

• The differentiate method returns the differentiated version of this instance of Polynomial. It takes each Monomial of the Polynomial and creates a new Monomial with the degree decreased by 1 and the coefficient multiplied by the previous degree. It adds the new Monomial in the res Polynomial, which will be returned.

**The GUI class**



The GUI class is the class that contains the frame that we will instantiate in the main

• The interface was designed using swingUI designer built inside IntelliJ and everything is displayed on the Jpanel „panel1”.

• We have the following buttons : ADDButton, SUBSTRACTButton, DIVIDEButton, MULTIPLYButton, INTEGRATEButton, DERIVATEButton and each of them are displayed with on the screen with the text ADD, SUBTRACT, DIVIDE, MULTIPLY, INTEGRATE, DERIVATE so the end user can choose what operation he needs to perform accordingly to its needs.

• Constructor of the GUI class initialize the window with a text that we will use as an argument inside the main function and sets the size for the windows to be displayed ( 450 x 500 ) and also position it in the middle of the screen and the setVisible is set to TRUE.

• Each button action listener reads and initialize the two polynomials polynomial1 and polynoimial2 using the text in the firstPolynomial and secondPolynomial textBox and realize the opperation that we need. After the result is computed, it is displayed in the resultPolynomial textbox.

1. **Results**

• The result of the development of his application, that lasted for two weeks together with writing the documentation, is a software easy to use that makes it easy for the user to perform a couple of basic computations on polynomials. This tool can be, of course improved in several ways that will be described in the next section.

**7.Further development**

• A couple features can be modified or added in order to improve the functionality of the application:

- A better graphical user interface

- A simpler way of introducing the data

- Computations on more than 2 polynomials

- More operations

**8.Conclusions**

• During the development of this project i learned and improved of several skills:

- A better use of intelliJ

- A better use of the Java coding language

- A better understanding and usage of the object oriented programming paradigms

- More organized and clear code

**9. Bibliography**

1. Polynomial division tutorial http://www.purplemath.com/modules/polydiv2.htm

2. Java general documentation https://docs.oracle.com/javase/7/docs/api/

3. Lecture and laboratory guidelines http://www.coned.utcluj.ro/~salomie/PT\_Lic/3\_Lab/