

Technical University of Cluj-Napoca
Faculty of Automation and Computer Science

Polynomial Processing

Diaconu Călin
Group: 30422

1. **Task**

Propose, design and implement a system for polynomial processing. Consider the polynomials of one variable and integer coefficients.

- Implement polynomial operations
 - Implement the Monomial class
 - Implement the Polynomial class
 - Implement the actual operations
 - Implement addition
 - Implement subtraction
 - Implement multiplication
 - Implement division
 - Implement derivative
 - Implement antiderivative
- Implement user interface
 - Implement actual interface
 - Implement input parsing

2. **Problem analysis**

The project aims the design and implementation of the problem of polynomial processing. What polynomial processing refers to is a multiple step process: input polynomial and use them in arithmetical operations. The operations considered are: addition, subtraction, multiplication, division, derivative and antiderivative. As for the structure of the polynomials, the input must only contain integer coefficients, but the variable can have any one-letter name, while the output will contain float coefficients and the variable will be marked with the letter 'x'.

Definition:

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 exponents of variables.

The individual summands with the coefficients included are called monomials. However, the term “monomial” is sometimes used to mean polynomial summands without their coefficients. The highest power in an univariate polynomial is called its order or, sometimes, its degree.

The sum of two polynomials is obtained by adding together the coefficients sharing the same powers of variables. The result is a polynomial having the degree the maximum degree of the two polynomials.

The subtraction of two polynomials is obtained by adding together the minuend and the opposite values of the monomials of the polynomial representing the subtrahend.

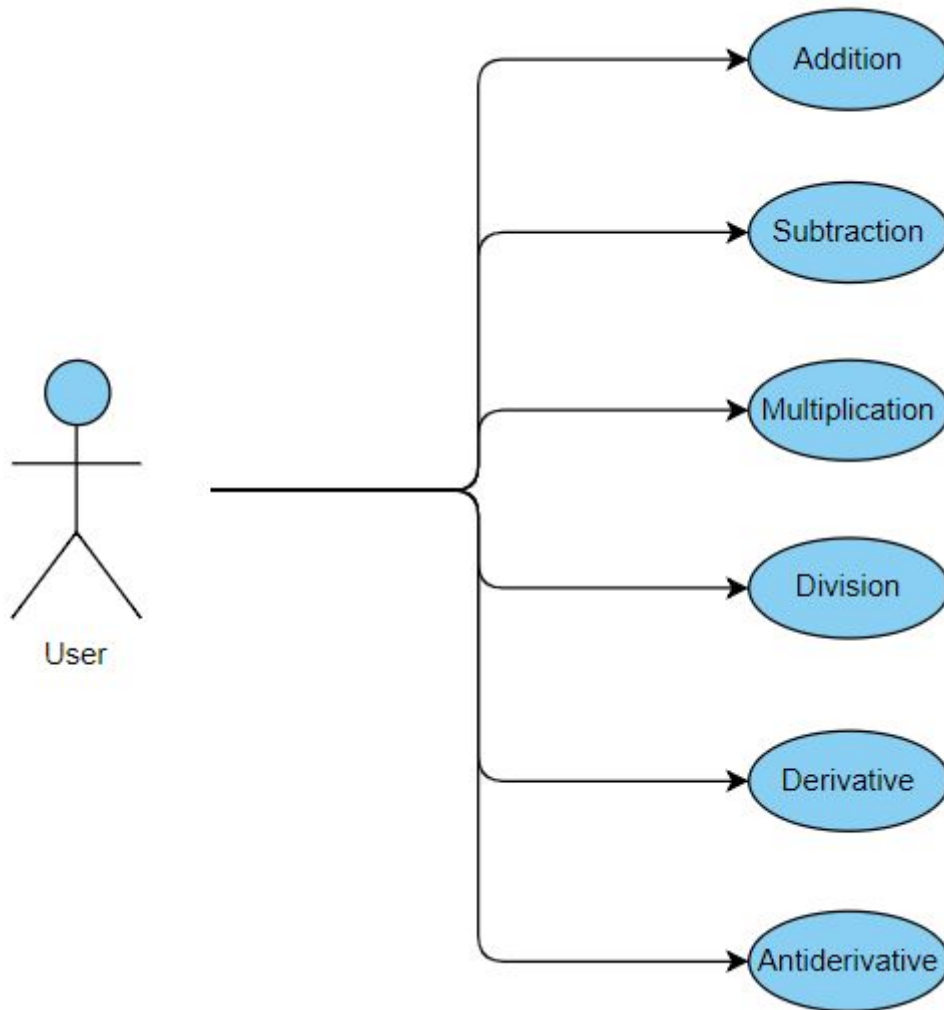
The product of two polynomials is obtained by multiplying term by term and combining the result. The result will have the degree the sum of the degrees of the two initial polynomials.

The division of two polynomials will have as a result two polynomials: the quotient and the remainder. This operation uses the following algorithm: the dividend is divided to the divisor as long as it has a greater degree; then, the result is multiplied by the divisor and the number is subtracted from the dividend.

The derivative of a polynomial is the formula for finding the slope of a curve and it is done by differentiating each member of the polynomial.

The antiderivative of a polynomial is the sum of the antiderivatives of the monomials and is the reverse operation of the derivative.

Use-case diagram:



The user is provided with two text fields which can be used to input the two polynomials. The input can contain both positive and negative monomials and the variable can have any one-letter name, but it doesn't support non integer coefficients. If there is an unexpected input, the user will be faced with an error window.

Once the input has been given, in order to apply the needed operation, the user should press the corresponding button: "Addition", "Subtraction", "Multiplication" and "Division" for the operations applied on both polynomials, "DerivativeP1" and "AntiderivativeP1" to apply these operations on the polynomial given by the user in the text field labeled "Polynomial 1:" and "DerivativeP2" and "AntiderivativeP2" for "Polynomial 2:" respectively.

To stop the execution of the program, the user must simply press the "Close window" button (positioned differently according to the OS, but usually symbolized by a red "X" button).

3. Design

The program contains four packages: backEnd, frontEnd, junit and main.

The BackEnd package contains two classes: Monomial and Polynomial.

The FrontEnd package only has the MainFrame class, which represents the Graphical User Interface.

The JUnit package contains the tests for the Monomial and Polynomial classes.

The Main package only contains the Main class.

The Monomial class implements the Comparable interface and contains a float coefficient, representing the coefficient, and an integer, representing the exponent. The methods included are a constructor that takes values for both of the variables, the needed getters and setters and the compareTo needed by the implemented interface.

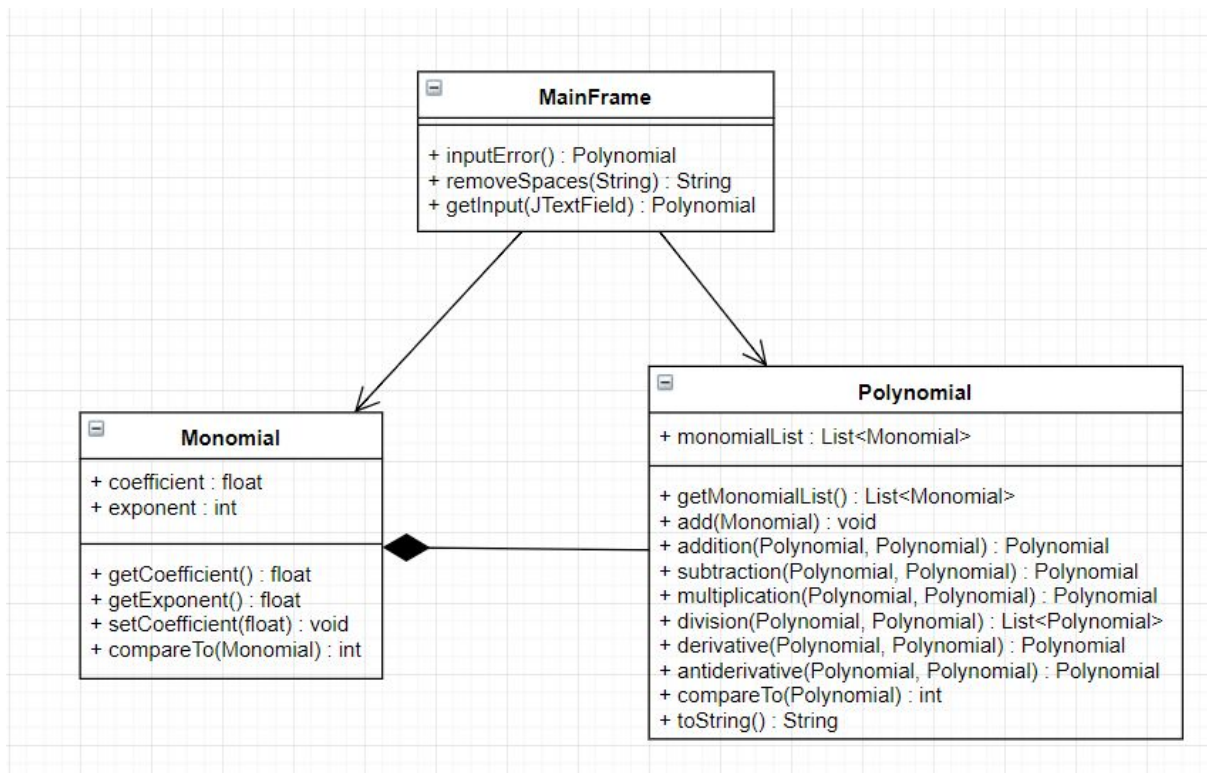
The Polynomial class implements the Comparable interface and contains a list of Monomials. The methods included are a constructor that creates a Polynomial with an empty monomialList, a getter, an "add" functions that is used for the addition between the initial Polynomial and a given Monomial, static functions for the required operations: addition, subtraction, multiplication, division, derivative and antiderivative, the compareTo needed by the implemented interface and an overridden toString method.

The MainFrame extends the JFrame class and has a constructor, a method that gives the user the "Input Error" message, a method for removing blank spaces at the beginning of a String and also the function to obtain, parse and interpret the input given by the user in the text fields.

There are two classes for testing the methods implemented in the Monomial and Polynomial classes.

The Main class only creates a MainFrame object in order to display the graphical user interface.

Uml diagram:



4. Implementation

Class:

Monomial:

Fields:

- coefficient: Float variable that contains the coefficient of the monomial
- exponent: Integer variable that contains the exponent of the monomial

Methods:

- Constructor: The only implemented constructor is the one which requires both the coefficient and the exponent, which is enough for the further usage of the program.
- compareTo: The class implements the Comparable interface, useful for different operations implemented in Polynomial. This method is required by the Comparable interface. This method returns only the value of the method compareTo applied between the exponents. Although a complete method would also compare the coefficients, this is enough for our purposes.

Polynomial:

Fields:

- monomialList: The list of the Monomials contained by the Polynomial.

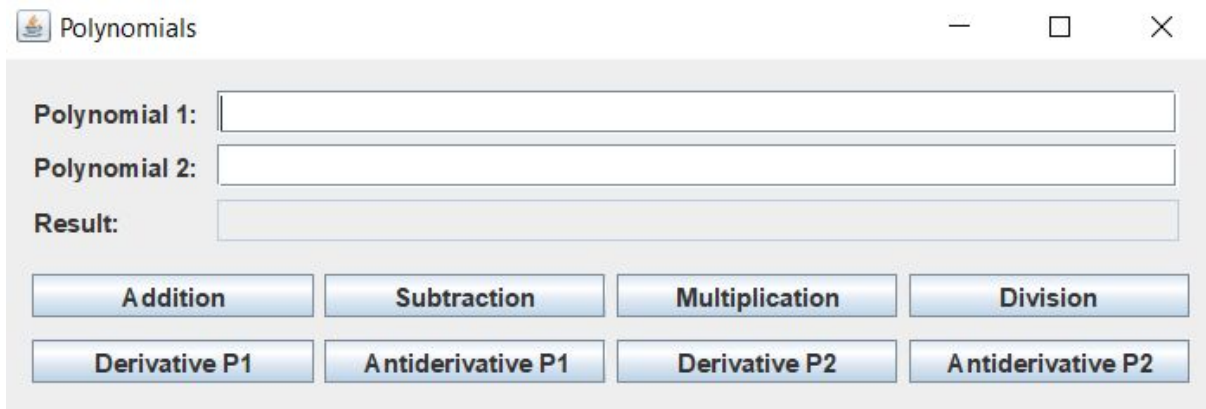
Methods:

- constructor: The only constructor is the one without parameters, which will generate a Polynomial with an empty monomList.
- add: The method through which a Monomial is added to the Polynomial. The method doesn't return anything, but it needs a Monomial parameter which represents the Monomial to be added to the Polynomial. The method passes through all the Monomials that are already in the Polynomial to look for the place where the new Monomial should be added. It stops when it finds another Monomial with the same exponent as the Monomial that needs to be added, making the sum of the coefficients of the two Monomials, or when it passes over the place where the Monomial to be inserted should be placed (we can stop this passage earlier because all the Polynomials obtained from different operations will be sorted). After this passage, if no other Monomial with an exponent equal to the exponent of the Monomial to be added has been found, the new Monomial will be added to the monomialList. In the end, we sort the monomialList in ascending order of the exponents of the Monomials in the list.
- addition: The operation of addition between two Polynomials. It is a static method which returns another Polynomial that represents the result of the operation. Because of the way the add method works, it is enough to use the add method with every Monomial from both Polynomials as parameters on a third, initially empty, Polynomial, which will represent the result of the operation.
- division: The operation of division between two Polynomials. It is a static method which returns another Polynomial that represents the result of the operation. Because of the way the add method works, it is enough to use the add method with every Monomial from the Polynomial that represents the minuend (first Polynomial) and the opposite of every Monomial from the Polynomial that represents the subtrahend (second Polynomial) on a third, initially empty, Polynomial that will represent the result of the operation.
- multiplication: The operation of multiplication between two Polynomials. It is a static method which returns another Polynomial that represents the result of the operation. The method passes through all the Monomials of the two given Polynomials and it adds a third Monomial (using the add method) to a third, initially empty, Polynomial, according to the rules of the polynomial multiplication (each Monomial from the first Polynomial is multiplied with every Monomial from the second Polynomial, obtaining with every Monomial multiplication a new Monomial with the coefficient equal to the product of the

coefficients of the two initial Monomials and the exponent equal to the sum of the exponents of the two initial Monomials).

- division: The operation of division between two Polynomials. It is a static method which return a list with two other Polynomials which will represent the quotient and the remainder (in this order) of the operation. The method runs while the remainder, a Polynomial initially equal to the numerator, is greater or equal to the denominator. At every passing through the loop, the algorithm applies the rules of the division between two polynomials: it obtains the Monomials with the highest exponent from the Polynomials representing the denominator and the remainder and then it computes the Monomial that needs to be added to the quotient. In the end, it will use the subtraction and multiplication methods to compute the new remainder. Outside the loop, it builds the list of Polynomials that will be returned.
- derivative: The operation of differentiation applied to a Polynomial. It is a static method which returns a Polynomial which will represent the result of the operation. The method passes through all the Monomials of the initial Polynomial and it adds the Monomials obtained with the differentiation rules (the new coefficient will be the product between the initial coefficient and the initial exponent and the new exponent will be one unit smaller than the initial exponent) to a new, initially empty, Polynomial.
- antiderivative: The operation of integration applied to a Polynomial. It is a static method which returns a Polynomial which will represent the result of the operation. The method passes through all the Monomials of the initial Polynomial and it adds (using the add method) all the Monomials obtained using the integration rule (the new coefficient will be the quotient of the division between the initial coefficient and the new exponent and the new exponent will be one unit higher than the initial exponent) in a new, initially empty, Polynomial.
- compareTo: The class implements the Comparable interface, useful for different operations. This interface requires for a compareTo method to be described. The method initially checks if any of the two Polynomials is empty (doesn't contain any Monomial), in which case the empty one will be considered the smallest one. If both Polynomials contain no Monomials, the two will be considered equal. The method then compares the Monomials with the highest exponent in each Polynomial. The exponents of the two Monomials will be compared. In the case of equality, the coefficients will then be compared.
- toString: The initial toString method has been overridden to obtain a conventional format for a Polynomial (from the Monomial with the highest exponent to the one with the smallest exponent, with the conventional signs of addition and subtraction and the "^" character to mark the exponent of each Monomial).

Graphical User Interface:



The GUI contains three text fields. Two of them are used for the input of the two polynomials necessary for the operations and the third, which doesn't allow the change of the text by the user, will be used as an output for the result of the operation. The content of each text field is marked by a label positioned to the left of each text field.

To apply the required operation, the user must click the corresponding button: four of these correspond to the operations applied on both polynomials: Addition, Subtraction, Multiplication, Division, while other four buttons correspond to the operations applied on only one Polynomial: Derivative P1 and Antiderivative P1, applied to the Polynomial given by the user in the first (the topmost) text field, and Derivative P2 and Antiderivative P2, applied to the Polynomial given by the user in the second text field.

In the case of a wrong input (one that doesn't correspond to the expected format or when division by 0 is attempted), the program will show an error message.

In the MainFrame class, the method for parsing the input can also be found. This will use the regex package to divide the input in Monomials and then the method will pass through each character to check if the expected format is respected and to build the Polynomial according to the input.

5. Results

For testing, the junit framework has been used. There are exceptional cases tested, like addition and subtraction with 0 or the division with a result below 0, but also random tests by using the Math package to generate Monomials with random coefficients and exponents.

6. Conclusions

Java provides a collection of features large enough to make the programmer's work easier, faster and more efficient, by reducing the necessity of implementing different algorithms (managing a list, parsing a text, etc.).

For further developments, the parsing of the input could be improved: allowing parenthesis, non integer coefficients and checking for more errors in the input format.

7. Bibliography

- <https://www.mkyong.com/tutorials/junit-tutorials/>
- <https://stackoverflow.com>
- <http://www.anidescoala.ro/educatie/matematica/formule-matematice-algebra-liceu-general/impartirea-polinoamelor/>
- <https://wikipedia.org/>