**Technical University of Cluj-Napoca**

**Faculty of Automation and Computer Science**

**Computer Science Department**

**Programming Techniques**

**First Homework: Operation with Polynomials**

**Name: Mihoc Nicolae**

**Group: 30425**

**Documentation**

**Specification:**

Building of an application able to process polynomials of one variable with integer coefficients and different operations between them. The operations that the application can perform are: addition, subtraction, division, multiplication, integration, and derivation.

**Objectives:**

This project wants to represent an application that is very good in processing polynomials of one variable with integer coefficients. It’s utility consists in realizing different operations between polynomials that will ease the user’s work if he needs to perform operations that need a lot of computation.

Another objective is familiarizing the beginners in this field, working with polynomials, and help him for a better understanding of how operations between polynomials has to be done.

**Problem analyze and study of the test case:**

In mathematics, a **polynomial** is an expression of finite length constructed from variables (also known as indeterminates) and constants, using only the operations of addition, subtraction, multiplication, and non-negative integer exponents. For example, *x*2 − 4*x* + 7 is a polynomial, but *x*2 − 4/*x* + 7*x*3/2 is not, because its second term involves division by the variable *x* (4/x) and because its third term contains an exponent that is not a whole number (3/2). The term "polynomial" can also be used as an adjective, for quantities that can be expressed as a polynomial of some parameter, as in "polynomial time" which is used in computational complexity theory.

Polynomials appear in a wide variety of 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 problems in the sciences; 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, a central concept in abstract algebra and algebraic geometry.

A polynomial of a single variable are called univariate polynomial. All the polynomials of one variable are equivalent with a polynomial of the form:

**+ . . . . . +**

This general form is considered the general form of the polynomials of a single variable. The degree of a polynomial is equal to the maximum of the exponents of the monomials that constitutes it.

With these polynomials, using the application we can realize different operations:

* With two polynomials: adding, subtracting, multiplication, division.
* With a single polynomial: derivation, integration.

Because the division by is not possible if we need to compute the division of two polynomials we have to avoid that the second polynomial to be 0.

The application receives as input data two polynomials given as sum of monomials written in the form: coefficient (this can miss if the coefficient is 1)+ ‘x’ ^ exponent (can miss if the exponent is 0 or 1). An example of such a polynomial is: 3x^4+3x^3+-2x+1. Notice that if the coefficient is a negative integer we still have to put the “+” operator in order for the polynomial to be processed correctly. The output data will be in the same form as the input data. The only difference is that it will be displayed in a different text area.

Another aspect to be considered is that the polynomial is limited to the maximum degree of 100. You should limit your polynomials to a maximum of 15-20 non-zero coefficients in order to display the entire result.

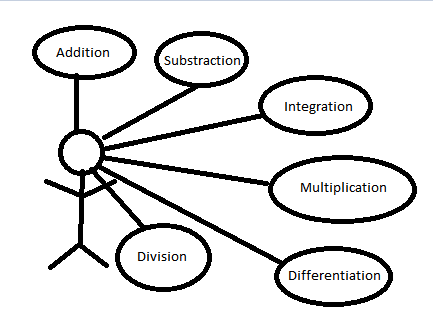
If the polynomial does not fit the expectation an error message will be generated “Invalid input” and no operations will be done. In case that the polynomial fits the expectation a validating message will be sent to the status text area that will ask you to continue with your operations by introducing another polynomial. The message that will be set will be “Give me another polynomial”.

For example, if we have in the result polynomial the polynomial: 5x^6+4x^3+2x+2 and at the input polynomial the polynomial: 6x^3+3x+2 we get for the different operations different results:

* Add: 5x^6+10x^3+5x+4;
* Subtract: 5x^6+-2x^3+-1x;
* Multiplication: 30x^9+15x^7+34x^6+24x^4+20x^3+6x^2+10x+4;
* Division: quotient = 0.83x^3+-0.42x^1+0.39x^0 remainder = 1.25x^2+1.67x^1+1.22x^0;
* Derivation: 18x^2+3;
* Integration: 1.5x^4+1.5x^2+2.0x^1.

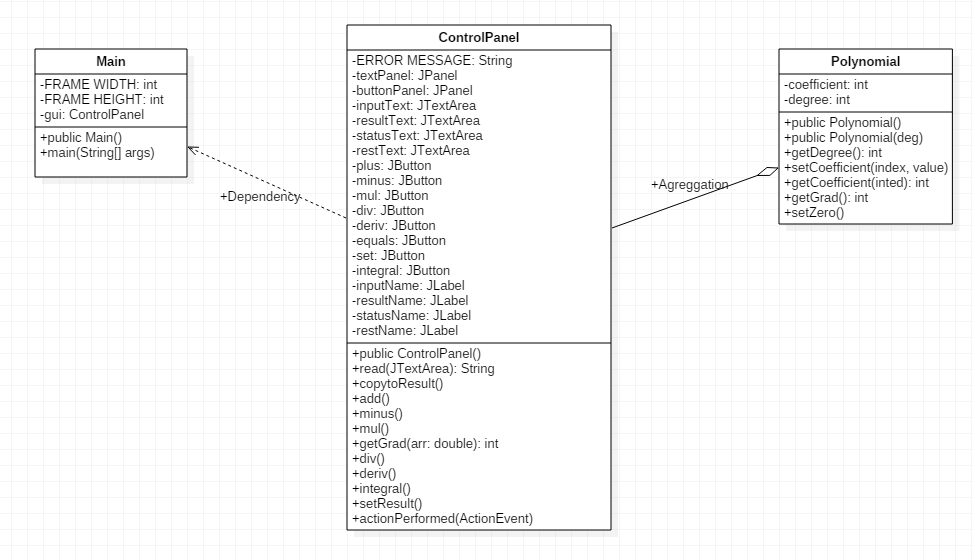
**Design and implementation**

a)Use Case Diagram



This diagram shows the user and what kind of operations can be performed using the application developed by us. The user can be any person with a little bit of knowledge about polynomial processing.

b)UML Diagram



I considered a relation of dependency between ControlPanel and Main because it cannot be visualized without the existence of the Main class and a relation of Agreggation between ControlPanel and Polynomial because the GUI exists, the read of the polynomial can be done and the operations implemented but without the class Polynomial the values cannot be added in the right place to for a polynom.

I chose for this project a separable –model architecture due to the fact that operations must be done when an event is captured (the user presses a button) so it was easier to invoke those methods in the same class.

The main class of this project is the ControlPanel class which contains the graphical interface and performs operations between polynomials. This class extends JPanel, so it will be used for storing different components, and responds to user inputs so it implements ActionListener. This class deals with the input by copying it into a string and by decomposing it into smaller strings that refer only to one term of the polynomial. We identify for each such substring the coefficients and the exponents that will characterize the polynomial. This information will be stored in an object of the class Polynomial in order to be later used.

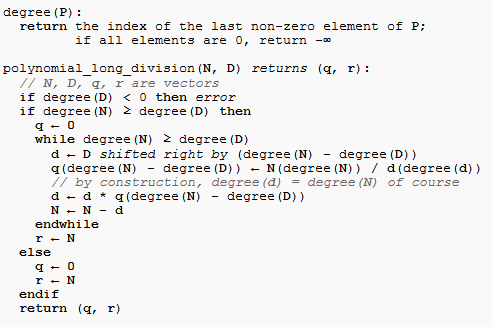
The class Polynomial characterize a certain polynomial by defining their coefficients related to their exponent, so I choose to store those information in an array and by some operations that can be done on single polynomials.

The class Main is used mainly for controlling the start of the application and for setting the environment. The Main class extends JFrame so all the other panels will just be added to this frame.

**Data structures, algorithms used**

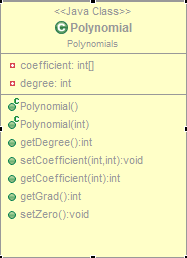
The main data structure which is implemented in this program in order to model the polynomial is an array if elements of type integer, representing the coefficients of the polynomial related to their exponents. Because integration of a polynomial and the division of two polynomials can cause the result not to have integer values for the coefficients we used local arrays in order to store the result.

For example I used the algorithm of division of two polynomials which is presented in the following pseudo code:



Derivation and integration are based on the derivation and integration of a simple power function, coupled with the linearity of the two operations with regards to addition.

**The Polynomial class**



This class characterizes a certain polynomial by defining its coefficients related to their corresponding exponent and defining the degree of the polynomial, that is the maximum exponent for which its corresponding coefficient differs from 0.

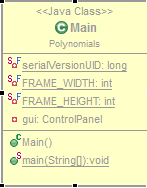
This class contains two constructors:

* one having no parameter, so the array of the coefficients will be initialized with a size of 200
* one having a parameter that contains the degree of the polynomial so the array of the coefficients will be initialized with a size having the value of the parameter.

This class also contains five methods for retrieving certain information relating to the polynomial or for setting certain parameters of the polynomial:

* getDegree: This method is used for returning the degree of the polynomial.
* setCoefficient: This method sets the coefficient for the exponent given as a parameter with a value given also as a parameter.
* getCoefficient: This method returns the coefficient for the exponent given as a parameter.
* getGrad:This method is used for returning the degree of the polynomial if no information about the polynomial is set. It is very useful if the polynomial is modified within a loop and we need to repeatedly actualize the information about its degree.
* setZero: This method is used for setting all the coefficients of a polynomial to 0. It is useful if the polynomial to be used was set previously and we need to reset it

**The Main class**



This class is used for defining the frame that will used for adding all the components of the interface and for triggering the start of the application.

Main class has 3 attributes:

* FRAME\_WIDTH and FRAME\_HEIGHT which are constants so they are defined as final static. These constants were used for easier possibility of resizing the window. If a later update of the application requires the modification of the window size, this can be done easier just by modifying these numbers.
* gui which is an object of the class ControlPanel that will be used for adding an instance of the ControlPanel class, which is a panel to the frame consisting of the main class.

This class contains only one constructor, one with no parameters, in which the panel consisting of an instance of ControlPanel is added and where the frame is set. The operation that is done on the frame is adding the panel, setting the frame to close when the application is closed, setting the visibility of the frame to *true*so the frame will be visible, setting the size of the frame according to the two attributes that were presented previously and setting the frame to be fixed by using the method inherited from the class JFrame,setResizable having the parameter set to *false.*

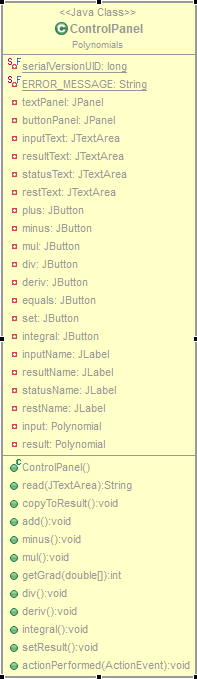
This class also contains a method: the void main method that will be invoked whenever the application is started. This method contains an instance of the Main class. This command will effectively initialize the window.

**The ControlPanel class**

This class extends JPanel, so it will be used for storing different components, and responds to user inputs so it implements ActionListener. This class deals with the input by copying it into a string and by decomposing it into smaller strings that refer only to one term of the polynomial. We identify for each such substring the coefficients and the exponents that will characterize the polynomial. This information will be stored in an object of the class Polynomial in order to be later used. When a button is pressed the method called actionPerformed handles the event according to what button was pressed.

The attributes for this class are:

* ERROR\_MESSAGE: define a constant string that will be displayed on the status code;
* textPanel: the panel used for storing the upper components(text areas);
* buttonPanel: the panel used for storing the lower components(buttons);
* inputText: the text area used for introducing the input polynomial;
* resultText: the text area used for displaying the result of the operation;
* statusText: the text area used for displaying different messages to the user;
* restText: the text area used for displaying the rest in case we have a division operation;
* plus: button for activating the operation of addition of two polynomials;
* minus: button for activating the operation of subtracting the two polynomials;
* mul: button for activating the operation of multiplication of two polynomials;
* div: button for activating the division of the two polynomials;
* deriv: button for activating the operation of derivation of the input polynomial;
* equals: button for setting in the result text area the information stored in the result polynomial;
* set: button for setting in the result text area the polynomial stored in the input text area;
* integral: button for setting in the result text the result of the integration of the polynomial stored in the input text area;
* inputName: label just for the interface, letting the user to know which is the input text area ;
* resultName: label just for the interface, letting the user to know which is the result text area;
* statusName: label just for the interface letting the user to know which is the status text area;
* restName: label just for the interface letting the user to know which is the rest text area;
* input: object of the polynomial class that will be used for storing the input polynomial;
* result: object of the polynomial class that will be used for storing the result polynomial.



This class has only one constructor, one with no parameters, in which the panel defined by the ControlPanel class is described. The main operations that are done inside the body of this constructor:

* setting the layouts to null in order for the position of the components to be set manually;
* setting for each component the bounds in order not to take different positions if the frame is resized;
* setting different colors for the text areas in order for the user to easily distinguish them;
* add the components that were previously configured;
* add action listeners to each button.

This class can perform different operations that will be accomplished by invoking one of its methods:

* read: is the method for processing the input polynomial. First, it takes the string contained in the input text area, decompose it by tokens with the delimiter being   
  “+”. For this operation an object of the class StringTokenizer is used., having the delimiters “+” and white space. Each token will be processed, the result being the identification of the coefficient and exponent that that token points to. On each token we will verify it the first character is “-“ indicating that the sign of the coefficient of the term contained by the token is “-“.Next, we verify if the character is a digit. If not an error message is returned and the execution of the method is stopped because the content of the input text area does not fulfill the specifications. After the coefficient is processed we verify if the next character is “x” or no more characters are left in the string. If there is no such case an error message will be returned. If the execution was not stopped until this step it means that the input corresponds to the expectations until now. So, we check if the next character is “^” or no character is left .If none of this case is fulfilled an error message will be returned. Now, if the previous character was “^” we still need to read the corresponding exponent. So we check if we still have characters left in the token and if they represent a digit. If these characters left does not fulfill the expectation an error message will be returned. This method memorize in the input polynomial the polynomial contained in the input text area and returns a message telling if the polynomial was read successfully or if it generated an error;
* copyToResult: copies the information stored in the input polynomial to the result polynomial;
* add: this method is used for adding the two polynomials. The coefficients having the same exponents will be added. For optimization we check if the coefficients are different from 0 in order to be added.
* minus: this method is used for subtracting the two polynomials. The coefficients having the same exponents will be subtracted. For optimization we check if the coefficients are different from 0 in order to be subtracted.
* mul: this method is used for multiplying the two polynomials. An auxiliary vector is used for storing temporarily the result. This vector will take values such that for each combination of coefficients: one from input polynomial and one from result polynomial, at index exponent1(exponent from input polynomial)+exponent2(exponent from result polynomial) the vector will take the value coefficient1\*coefficient2;
* div: this method is used for dividing the two polynomials. Three auxiliary arrays are used for computing the polynomial. The operation in this method are translated in the pseudo code presented at the “data structures and algorithms used” section. At the end of the method in the result polynomial we will store the result of the operation approximated to integer. Because in this operation we can deal with negative coefficient in the results and in intermediate calculus also we use local arrays in order to store the results that can take non-integer values. The remainder will take the last value stored in the array which stores the intermediate results of the result polynomial. Also, the results (quotient and remainder) will be displayed in this method in their corresponding text areas having only 2 decimals.
* deriv: Computes the derivative for the input polynomial. The result will be stored in the result polynomial. So, in the result polynomial at the index i-1 we will have the value of the coefficient of the input polynomial having the index i multiplied by i (which represents the exponent of the term).
* integral: Computes the integral for the input polynomial. Because the input and the result polynomial has integer coefficients and the result of this operation can have rational coefficients the result of this operation is memorized into an auxiliary array of type double and displayed on the result text area as polynomial with real coefficients having 2 decimals. In the result polynomial we will have the result of this polynomial with its coefficients approximated to their integer value.
* setResult: Displays the result stored in the result polynomial into the result text area. For each coefficient that is different from 0 we check if the coefficient is 1. In this case we don’t display it. Also, we check if we are in the case of the last coefficient in order not to display the exponent.
* actionPerformed: Responds to user actions. Checks what button was pressed and responds by calling methods or setting error message if the inputs does not corresponds to the specifications.

**Conclusion**

In conclusion the application realizes all the requirements of the specification, realizing the main operations specific to polynomials with one variable and having integer coefficients.

The application is easy to use by anyone, even by the users which are not familiar with the Java programming language or any other technical domain, the results being generated by simply introducing some data and clicking some buttons. The only errors can appear at introducing data, the user has to stick with some rules for inputting.

The program can have future possibilities of development to become more complex and to cover a more diverse range of processing polynomials. That been told we can extend the application for:

* Computing the polynomial’s value in a given point;
* Computing the roots of the polynomial;
* To extend the range of the polynomials having real or complex coefficients;
* To display the graphic of the polynomial;
* To compute the derivative of a greater order;
* etc.

After realizing this project I became more familiar with working with swing interface. I developed my skills in coding and became more familiar with operations between polynomials.

**Reference**

* <http://rosettacode.org/wiki/Polynomial_long_division>
* <http://users.utcluj.ro/~jim/OOPE/index.html>