Technical University of Cluj-Napoca

Faculty of Automation and Computer Science

2nd Semester 2016-2017

**Programming Techniques**

**Assignment no. 1**

Student: Crișan Dan

Group: 30424

Supervising teacher: Viorica Chifu

**Table of contents**

1. **Objective** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3
2. **Problem analysis, modeling, scenarios and use cases** .3

2.1. Problem analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

2.2. Modeling . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

2.3. Scenarios and use cases . . . . . . . . . . . . . . . . . . . . . . . 3

1. **Design** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .4

3.1. Design decisions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .4

3.2. UML diagrams (Unified Modeling Language) . . . . . . . . 4

3.3. Data Structures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

3.4. Class design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

3.5. Algorithms . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

3.6. GUI (Graphical User Interface) . . . . . . . . . . . . . . . . . . . 7

1. **Implementation and testing** . . . . . . . . . . . . . . . . . . . . . . . .8
2. **Results** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .9
3. **Conclusions** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
   1. Things I have learned . . . . . . . . . . . . . . . . . . . . . . . . . .9
   2. Future improvements . . . . . . . . . . . . . . . . . . . . . . . . . 10
4. **Bibliography** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
5. **Objective**

The main objective of this assignments is to propose, design and implement a system for polynomial processing. We consider the polynomials of one variable and take the coefficients as integers.­ The program will present a GUI through which the user may perform operations on one or two polynomials. The available operations will be the following: Addition, Subtraction, Multiplication, Division, Derivation and Integration.

1. **Problem analysis, modeling, scenarios and use cases**
   1. Problem analysis

The problem requires us to perform different operations on 2 polynomials. This requires the creation of a Polynomial class in order to store the information given by the user. Furthermore, in order to perform said operations we need to create a Monomial class. In this way all operations will be done termwise.

* 1. Modeling

The polynomials are conceived as a list of monomials. Each monomial will have its own coefficient and exponent. The exponent is of type integer but the coefficient is made of type double so that we will be able to perform divisions on it when required (e.g.: when performing the integration operation).

* 1. Scenarios and use cases

Addition scenario:

- User inserts polynomial number 1

- User inserts polynomial number 2

- User clicks on the button “+” for the addition operation to be made

- System prints the result in the “Result:” text box

e.g.: Polynomial 1: -3x^2 + 5

Polynomial 2: -2x - 5

Result: -2x - 3x^2

Derivation scenario:

- User needs to insert only the first polynomial

- User clicks on the button “Derivation” for the derivation operation to be made

- System prints the derived polynomial in the “Result:” text box

e.g.: Polynomial 1: -3x^2 + 5

Result: -6x

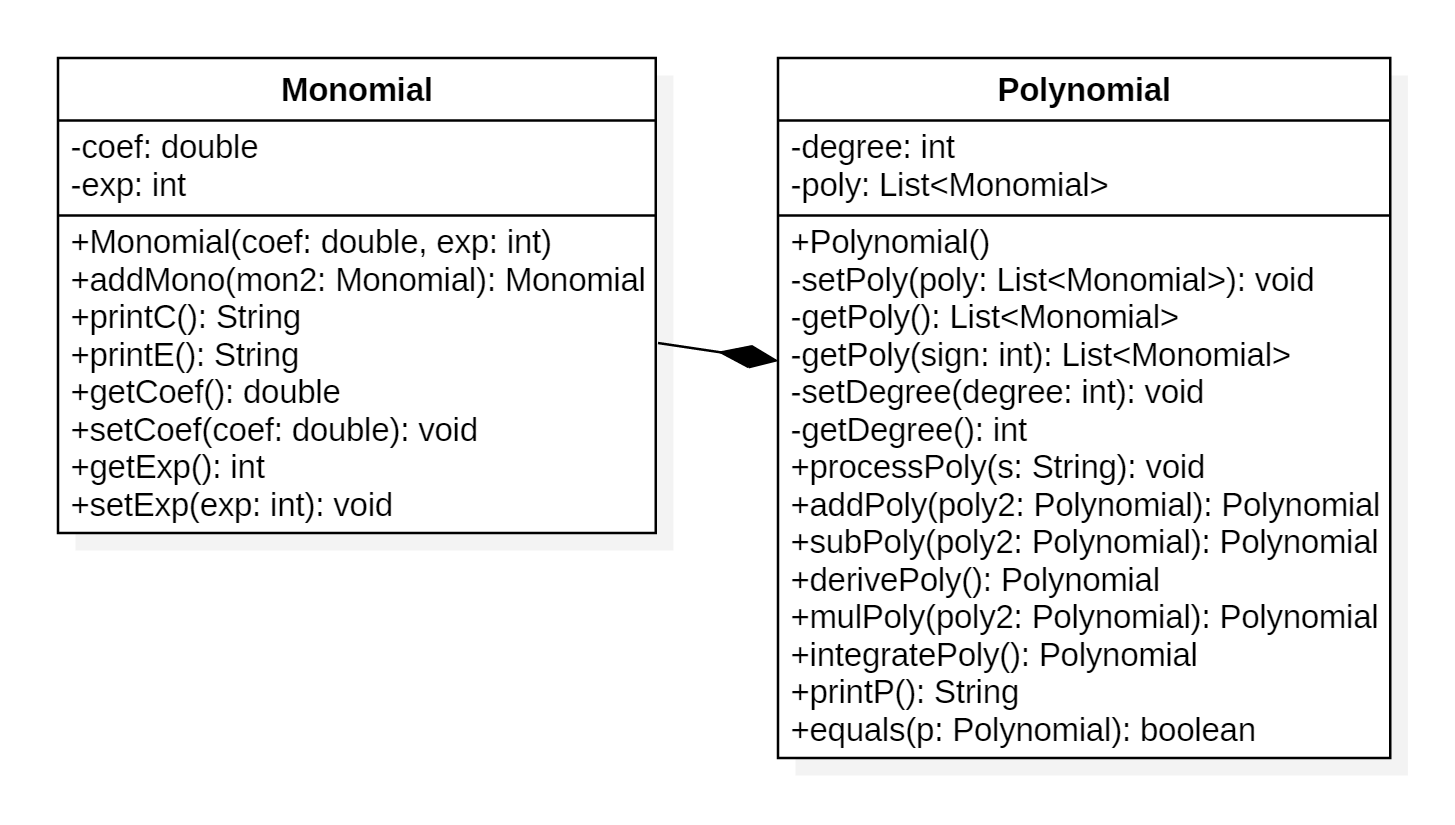
The subtraction, multiplication and division operations are made the same way the addition one is done.

For integration, the user should follow the derivation instructions.

1. **Design**
   1. Design decisions

As said in a previous section, I decided it was best to divide the polynomial into monomials, operations being easier to be executed term by term. Aside from the list of monomials, the polynomial has a degree. Most of the time this degree is used in order to determine which of the two polynomials is of higher grade so that we do not make redundant operations. For example if a polynomial is of degree 5 and the other one is of degree 2, we will use the degree to tell the addPoly function to add monomials equal in grade from both polynomials but only up to the point where there can still be a monomial of the same grade. So in this example for monomials of degree 3 and above, the algorithm will know to add them directly to the result since they will not be modified.

* 1. UML diagram (Unified Modeling Language)



* 1. Data structures
* Class Monomial:

Has 2 parameters: coef, which stores the coefficient of the monomial and which is of type double, so that when operations such as integration are made they can be performed correctly. The other parameter is exp, of type integer, and it holds the exponent of the monomial.

* Class Polynomial:

It also has 2 parameters. One named degree, where we keep

the degree of the polynomial. The other one is a list of monomials (List<Monomial>).

* 1. Class design
* Class Monomial

The constructor here is plain and simple having a coefficient coef of type double and the exponent exp of type int as parameters.

Besides the straight forward getters and setters, this class has several other methods such as: addMono, printC and printE.

Method addMono is called from the polynomial class. It is used to add two monomials of the same exponent. This method is used both for adding and substracting the polynomials. Methods printC and printE are used for printing the coefficient and exponent of the monomials. I used two different methods because coef and exp are of different type.

* Class Polynomial

Given the fact that I determine the degree of the polynomial in the processPoly method, I cannot initialize the list of monomials in a constructor, thus the initialization is made in the processPoly method and occasionally in a few other places such as the mulPoly method given that I have to create additional space for the newly created terms. The constructor for this class will contain only the initialization with 0 of the polynomial’s degree.

In class Polynomial we have setters and getters for both parameters, but for the list of monomials we have two getters, a normal one and one with a parameter named sign. The 2nd getter is used so that when we perform the subtraction operation we multiply the polynomial by -1 and then perform the addition operation between the two. Then we have the processPoly method wich transforms the string we just read into a list of monomials. Next we have the addition, subtraction, derivation, multiplication and integration methods named accordingly addPoly, subPoly, derivePoly, mulPoly and integratePoly. Their functionality will be further described in the algorithms section of this documentation. Finally we have the printP method which outputs the result of the operation performed.

* 1. Algorithms
* Class Monomial

This class does not present any complicated algorithms.

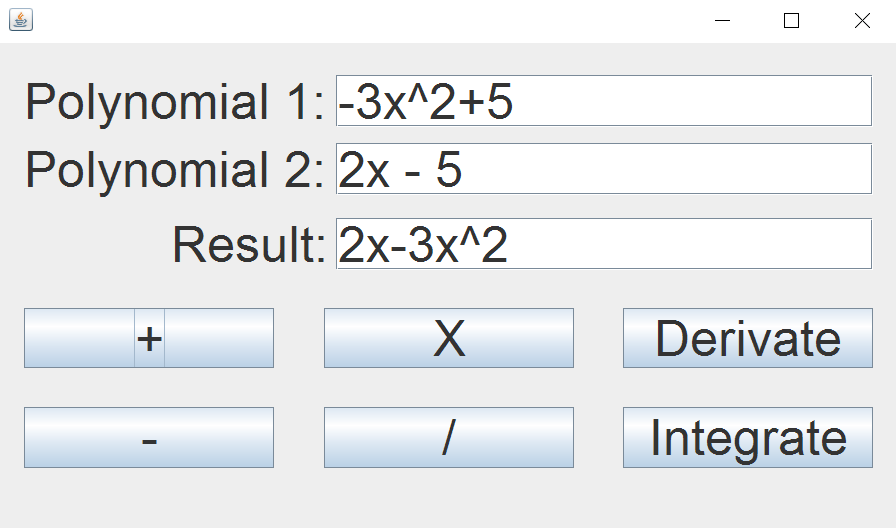
* The **addMono** method returns a monomial with the sum of coefficients and with the same exponent as the monomials received when called.
* Methods **printE** and **printC** return the exponent and coefficient as a string.
* Class Polynomial
* The getter for the list of monomials with the parameter sign, multiplies each monomial with the value of sign (-1 in this case).
* Method **processPoly** uses function .replaceAll to remove all white space inserted by the user and it is used once again to replace every - sign with a +-. In this way when we use the function .split in order to divide the string every time we find the sign +, so that every piece obtained will become a monomial and will preserve the - sign for its coefficient.
* Method **addPoly** and **subPoly** are almost the same with small differences between them. In both methods we use the degree of the polynomials to determine which on is the bigger one. Then we have a list called rezPoly which receives the values from the bigger polynomial, and another list called smallPoly which will hold the values for the smaller polynomial. Now the addMono function is called for every monomial with the exponent smaller or equal than the degree of the polynomial smallPoly. Then the value is updated in rezPoly. The difference made in the subPoly algorithm is that the polynomial which is subtracted uses the getter with the parameter sign equal to -1.
* The **derivePoly** method has an exception in the algorithm, so that when the exponent of the monomial is equal to 0, the resulted monomial will also be 0. This exception manages the case when the monomial is a constant. If the monomial does not enter the exceptional case, the program follows the derivation rules; the exponent is decreased by 1 and the coefficient is multiplied by the initial exponent of the monomial.
* First, in method **mulPoly** we update the degree of the resulted polynomial to be equal to the sum of the first two. Then we initialize the resulting polynomial with monomials with coefficient 0 and exponent i, where i goes from 0 to degPoly, where degPoly is the degree we just computed. After this we proceed with updating the monomials to their final result. Their final exponent will be the sum of the exponents of the monomials multiplied, this also being their index in the list holding the result polynomial.

The coefficient of the resulting monomial will simply be the multiplication between the coefficients of the initial monomials.

* Method integratePoly is maybe the most simple method of the ones mentioned above. The algorithm just follows the rules of integration. The exponent is increased by 1 and the coefficient will be divided by the new exponent.
* The printP method is the one created to transform the list of monomials back into a string so it can be printed to the screen. The algorithm goes monomial by monomial and checks in each if the conditions to see what to do with it. For example if the coefficient is 0 the monomial will not be concatenated to the final strig, if the coefficient is -1 or 1 the monomial will be printed without the coefficient, except if the monomial is of degree 0 (e.g.: x^2 instead of 1x^2 and -1 instead of just -). Finally if the monomial is not of negative coefficient the string concatenates with the sign + before each monomial, exception making the first one.
* Last, the method equals of type Boolean checks if two polynomials are equal. First it checks that both polynomials have the same degree. If the condition is sufficed the algorithm proceeds to check if each monomials of equal exponent have the coefficients also equal. If a pair is found not to be equal the function returns false. If the algorithm finishes going through all the monomials the function will return true.
  1. GUI (Graphical User Interface)

In my opinion, the Graphical User Interface I have designed is user friendly and it is easy to be used by non-expert users also. The buttons are straight forward with clear meaning. From top to bottom, the GUI has 3 text boxes, followed by 2 rows each with 3 buttons.

In the 1st and 2nd box the user may introduce his/her polynomials of choice. The 3rd box holds the result and it will print an output after one of the operation buttons has been pressed.



1. **Implementation and testing**

Although quite permissive, the input should follow some format rules. The program is implemented in such a way that all whitespace is ignored, so inputs such as “x^2 + 3” and “x^2+3” are both accepted. So no matter the spacing the following rules should always be fulfilled: between each 2 terms there should be a “+” or a “-” sign but not both; after each sign the user may choose to type in a coefficient or if the coefficient is 1, he/she can type in “x” and skip the coefficient input. Except for the case where the exponent is 1 or 0, it is mandatory that the user types in the “^” symbol followed by the number which will represent the power of the current element. Another rule would be that each polynomial may have only the symbol “x” representing the unknown variable. The polynomials must be entered from the biggest grade to the smallest in order. Aside from the characters “x”, “+”, “-”, “^” and digits ranging “0-9”, any other input will result in an error.

The application was tested with several inputs so that any particular cases observed can be dealt with in the implementation of the program so that no computation errors should be faced.

Furthermore, a Junit Test Case was created for verifying that each of the operations outputs the expected result. The test inputs can be modified. For the time being they are currently set for all operations as follows: Polynomial 1 is set to - 3x^2 + 5 and Polynomial 2 is set to - 2x - 5. For each operation test the expect output is modified accordingly and for the derivation and integration tests only the first polynomial is provided, the 2nd one being left empty.

1. **Results**

As it can also be seen in the Junit test, here are a few examples on what the outputs should look like for different inputs and operations:

* Addition

Polynomial 1: - 3x^2 + 5

Polynomial 2: - 2x - 5

Expected result: - 2x - 3x^2

* Subtraction

Polynomial 1: - 3x^2 + 5

Polynomial 2: - 2x - 5

Expected result: 10 + 2x - 3x^2

* Derivation

Polynomial 1: - 3x^2 + 5

Expected result: - 6x

* Multiplication

Polynomial 1: - 3x^2 + 5

Polynomial 2: - 2x - 5

Expected result: - 25 - 10x + 15x^2 + 6x^3

* Integration

Polynomial 1: - 3x^2 + 5

Expected result: 5x - x^3

Once again, for the derivation and integration operations, the 2nd polynomial can be optionally specified, although not needed.

1. **Conclusions**
   1. Things I have learned

After this project, I came to realize that creating a GUI is not that hard after a little bit of practice. Also I got the chance to learn how to create Junit tests. Among other pre-defined existing methods and functions I stumbled upon. split and .replaceAll, 2 functions which I have found to be extremely useful for my problem. And as any project not only did it teach me new stuff but it also helped me maintain my programming skills, having small problems at first recalling what programming in Java was like. With this assignment, although a bit embarrassing, I got to remember what operations with polynomials are like (e.g.: division) and it was further challenging to think how to implement those mathematical operations into Java code. Another thing would be learning to code according to a convention so that my code is easier to read and process.

* 1. Future improvements

Future improvements include derivation and integration of a degree higher than one. Also the ability to calculate the integral between to points a and b of the polynomial could be a possible new feature.

Other useful features may include finding the value of a polynomial for a give value and checking if two polynomials are equal. Another good thing would be for the program to accept polynomials of more than one variable. For a more tough challenge, one can implement algorithms for calculating the Fourier transform, raising at power or maybe finding the roots of the polynomial. As for the GUI, a fun feature would be for the user to be able to display if needed the drawing of the resulting function.

1. **Bibliography**

For different functions for operating on strings, I have found information on the following websites:

* <http://stackoverflow.com/questions/13415573/how-to-extract-polynomial-coefficients-in-java>
* <http://stackoverflow.com/questions/26883037/polynomial-string-not-split>
* <http://stackoverflow.com/questions/1701839/string-replaceall-single-backslashes-with-double-backslashes>

For Junit test I gathered information from a couple of youtube videos, some of them listed below:

* <https://www.youtube.com/watch?v=tkzJsP7NP54>
* <https://www.youtube.com/watch?v=o5k9NOR9lrI>
* <https://www.youtube.com/watch?v=tkzJsP7NP54>

For a further understanding on how to create good and friendly user interface I have also found a couple of tutorials on youtube:

* <https://www.youtube.com/watch?v=r8Qiz9Bn1Ag>