# Assignment 1 documentation

# Polynomials Operations

Dreghici Popa Vlad

Group 30422

1. *Problem description*

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

Operations to be implemented:

* Addition
* Subtraction
* Multiplication
* Division
* Derivation
* Integration

1. *Problem analysis*

From the beginning we assume the user inserts the polynomials correct. Since we have to implement operations, the project will require 2 polynomials for those operations. We will have to implement a MVC model for the project. The user will be able to insert the coefficients separated with ‘ , ‘. For example “1,4,5” will denote the polynomial 1\*x^0 + 4\*x^1 + 5\*x^2. After the polynomial insertion, the user will be prompted to select the operation he wants to execute. Those operations will be implemented on buttons. Each operation will have it’s button. After the user will press the operation he desires, the result will be displayed on the same window.

To introduce another pair of polynomial, the user will have to refresh the window. This will be done with another button which basically clears everything. After this, the user will be able to introduce a new set of values.

Use cases:

Open eclipse

Subtraction

Addition

Division

Derivation

Integration

User

Run program

Multiplication

Display Result

Do operation

Insert polynomials

Reset polynomials

Choose operation

1. *Design*
   1. *Classes*

A package is a collection of classes, interfaces and so on that provides access, protection and namespace management

A class is a template/blueprint that describes the behavior/state of the object it supports

An object has states and behaviors. An object is an instance of a class.

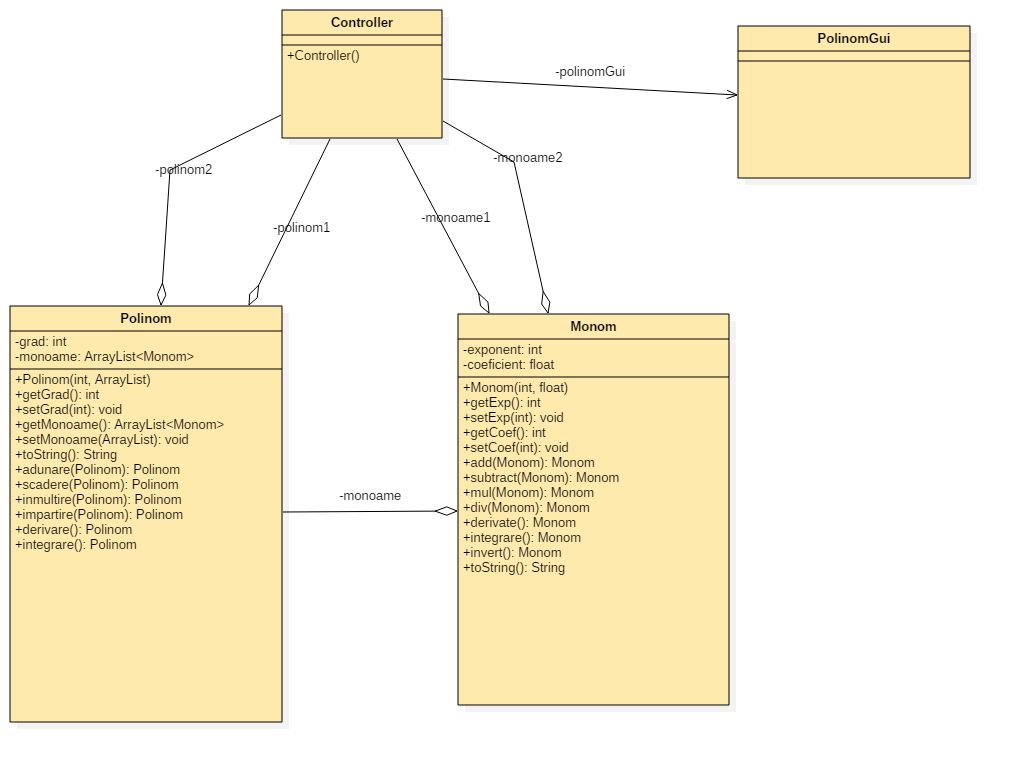
In mathematics, a **polynomial** is an expression consisting of variables (also called in determinates ) and coefficients, that involves only the operations of addition, subtraction, multiplication, and non-negative integer exponents of variables. An example of a polynomial of a single indeterminate *x* is *x*2 − 4*x* + 7. An example in three variables is *x*3 + 2*xyz*2 − *yz* + 1.

The project will be split in a certain number of classes, organized in packages, in a MVC manner. There is a Model package, which contains the Main class, Polinom and Monom classes. Those classes must execute the back-end algorithms, the operations the user requests. In the main class we have only the call for the controller. In Monom class, we describe a basic monom, for example “4x^3” where we declare the coefficient of float type (real) “4” and exponent of integer type “3”. This class will form the array for the polynomial with the help of Framework collections. To store monomials we used a List interface, which contains ordered elements. List interface has more benefits:

* Access elements based on Index
* Search for element
* Use iterators to traverse list

So the result will be an Array List of Monom type, where Monom represents an object of Element type. In this way we build the polynomial which contains more monomials. Another role for this class is to execute low level operations on the monoms. The polinom class will have an array of monoms and the rank. Inside the polynomial class there are the algorithms used for operations which will be described in detail on the Algorithms section of the documentation. Another package is the Controller package which contains only the controller Class. Controller links all the user interactions with the project execution. The class contains all the events for pressing buttons.

The last package is the View package. This package contains the GUI of the project. In the GUI class you can find all of the GUI elements like Textfields, labels, buttons and so on. Also there are a number of methods which are used to get or modify the text from certain Textfields or Label.

* 1. *UML Diagram*
  2. *Algorithms*

The methods that execute the operations on the polynomial traverse one or two structure of Array List type. The traversing is done with for each loops. Because of this we can easily compute the complexity of the algorithms. For the operations which require 2 for loop( Addition, Subtraction, Division, Multiplication) the complexity is O(n^2). For the Integrate and Derivate operations the complexity is only O(n) because we have only one for loop.

Addition operation:

The operation is made on 2 branches:

1. First grade is bigger than second grade
   1. Create a new Array List which takes monomials from the first polynomial
   2. Add coefficients when exponents are equal
2. First grade is smaller than second grade
   1. Create new Array List which takes monomials from second polynomial
   2. Add coefficients when exponents are equal

Subtraction operation:

Similar to addition, subtraction is executed on 2 branches:

1. First grade is bigger than second grade
   1. Create a new Array List with monomial from first polynomial
   2. Subtract coefficients at equal exponents
2. First grade is smaller than second grade
   1. Create a new Array List of monomials from second polynomial
   2. Subtract coefficients at equal exponents
   3. At the end, invert the remaining coefficients from the second polynomial

Multiplication operation:

1. Build a new Array List and fill it with 0’s.
2. Traverse the two given polynomials
3. Set each element to be equal to him plus the sum of the same exponent elements

Derivation operation:

1. Create a new Array List
2. Use the derivate operation from monom class to derivate each monom
3. Add the derivated monom to the newly created Monom Array List

Integration operation:

1. Create a new array list
2. Use the integrate operation from the monom class to integrate each monom
3. Add the integrated monomial to the newly created Monom Array List

The addition, subtraction, multiplication and division are executed with the help of the low level operations in the monomial class: Add, Subtract, Multiply and Divide. Those methods execute operations only on the monomial given so it is much more easier to write the final polynomial.

1. *Implementation*
   1. *Monom Class*

This class contains two variables: Integer “exp” which is for the power of the monomial and the float “coef” which is the coefficient of the monomial.

We also have a constructor which takes exponent and coefficient and assigns them to the monomial.

The class also contains the usual get and set methods for each attribute: getExp(), setExp(), getCoef(), setCoef(). Also for ease of use, in the monomial class there are declared the low level operations as method. We have add() method which sums the coefficients of monomials, subtract() which subtract their coefficients, mul() which multiplies the coefficients and adds the exponents, integrate() which adds 1 to exponent and divides the coefficient by the exponent and finally derivate() which subtracts one from the exponent and multiplies the coefficient with the exponent.

In the end we have method which is used to print the monomial. It overrides the toString() method and returns the polynomial under the following form : “2x^3” and a plus or minus in front depending on the sign of the coefficient.

* 1. *Polinom Class*

The polinom class contains a integer variable “grad” which sets the grade of the polynomial and an ArrayList of type Monom, which will contain all the monomials of the polynomial.

There also is a constructor which will set the grade and Array List of monomials

The class contains the regular getters and setters: getGrad(), setGrad() used to get and set the grade of the polynomial, getMonoame(), setMonoame() used to get or modify the array of monomials from the polynomial.

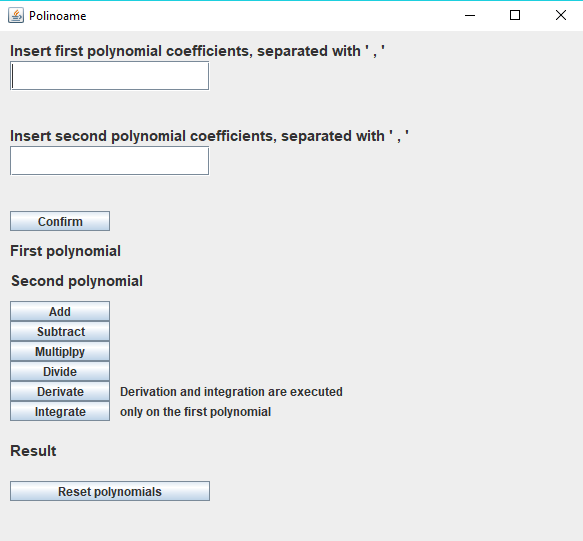
As explained earlier, the polynomial contains all the operations needed to be executed on the two polynomials. Those operations are all implemented as methods as follows:

* + - The addition method as “adunare(Polinom p)”, which will take the current polynomial and p polynomial and add them as explained in the algorithms section.
    - The subtraction method as “scadere(Polinom p)”, which acts almost the same as the addition method, algorithm explained earlier.
    - The multiplication method as “inmultire(Polinom p)”
    - The division method as “impartire(Polinom p)”
    - The integration method as “integrare()”
    - The derivation method as “derivare()”

The last method implemented in the polinom class is the toString() method which overrides the regular toString() method, in order to print the polynomial to the GUI.

* 1. *Polinom GUI Class*

This class will contain all the GUI elements for the interaction USER – PROGRAM. We have a bunch of GUI attributes and GUI methods.



Starting with the beginning we will have 2 labels which are basically instructions for the user. After that we have 2 Text fields where the user will be able to insert the coefficients of the polynomials. After the two coefficients are inserted, the user must press submit in order to save the two Polinom type objects. Those two will be displayed on the labels under the “Confirm” button. After the two polynomials are submitted, the user will be able to execute operations on the two polynomials. The operations will be executed by pressing the respective buttons. The result will be displayed under the operation buttons. After the user is done with the two polynomials, he can easily reset them by pressing the “Reset” button. This will clear the application.

All the buttons are activated by using ActionListeners, which is notified whenever the user clicks on the button. It is notified agains ActionEvent. ActionListener is basically an interface that has one method: actionPerformed().

* 1. *Controller Class*

The controller is the class that basically links the two parts of the project: Back-End and Front-End. In the controller class we import all the classes declared before.

We have as parameters the following:

* + - PolinomGui which is the GUI Class, interface.
    - ArrayList <Monom> monoame1 and monoame2 which are used to initialize the two polynomials.
    - Polinom polinom1 and polinom2 which are the 2 polynomials declared as objects.

We also have a constructor which initializes our hole graphical interface. We start by declaring a new object of type PolinomGui. After that we initialize all the action events for the buttons. There are the same number of action listeners as buttons and for each button a certain set of instructions is executed. Perhaps the most complicated one is the confirmPress instructions. For this button press, the program will create the two polynomials by reading the values from the 2 text fields and separating the values in between the ‘ , ‘. This way we will end with a string of digit characters. Those digits will be than parsed to Integers and inserted as coefficients in the Monom Array List. The exponents will be added automatically by the index of the coefficient.

1. *Testing*

The testing part is implemented by using JUnit tests. For this, a new class is created called PolinomTest in which we implement as methods the testing for each operation in the project. This will be called in the main class. The method will take the polynomials, compute the necessary operation and compare the result with a hardcoded one. If the results are equal then the test is a success. Otherwise, the implementation of the algorithm or the test is wrong. All the testing is done with the junit package and assert operation which will compare the result with the hardcoded one.

1. *Results*

The two polynomials are successfully computed after pressing the buttons. Success is given by the display of a new polynomial with correct values. The project is functional for executing basic operations on one or more polynomials, being able to work with input from user and show output back to user.

1. *Conclusions*

By working on this assignments I acquired more knowledge about how the Swing framework works. I have also acquired a set of skills needed to build the MVC model that will help me for future assignments. A future feature would be to implement a way to add a certain exponent in the monom class, so the user will be able to insert for example x^999 without having to enter 998 zeroes before. That would be great. The reading could be done using the register execution method in such a way that we read a string of symbols and the program knows how to interpret it.

1. *Bibliography*

http://mathworld.wolfram.com/Polynomial

http://stackoverflow.com/

https://en.wikipedia.org/

https://www.google.ro/

http://www.dreamincode.net/