Polynomials processing

UTCN CTI ENGLISH

Muresan Daniel, 30422

Contents

1. **Goal of the application**

Polynomials are a very important and commonly encountered in the mathematics domain. From polynomial functions to power series, calculus, abstract algebra and many other applications, polynomials are everywhere.

This application though is resumed to more simple things: polynomials with one variable and integer coefficients. The main goal of the application is to receive some polynomials as strings and then make operations on them like: addition, subtraction, multiplication and differentiation.

This application is useful because is it can make those operations faster than a person and on larger polynomials.

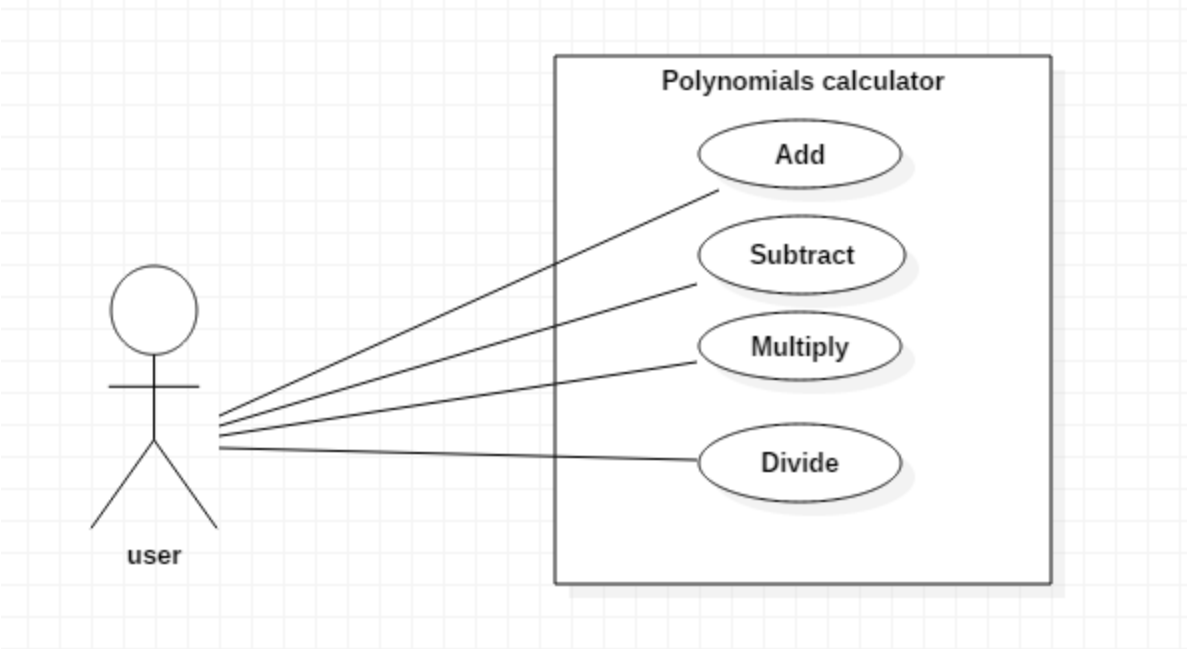
This application will also have an easy understandable graphic user interface with intuitive buttons, labels and text fields. All those will make it easy to use for any user regardless the experience and the knowledge in this domain.

2. **Problem analysis**

The main requirements for this application are the following ones: to receive the polynomials as strings, to transform them into a more appropriate type of data, to implement correct methods to calculate the sum, difference, multiplication and differentiation respectively for polynomials and then to transform the polynomials back and give the results as strings.

The assumptions I made for this application are the following ones: the user introduces only valid data-polynomials with only one variable, x for this case, the symbol “^” for representing the exponent of a variable and does not uses other symbols excepting “+” and “-“without brackets or any other types of symbols.

The use cases of the application are presented in the next use case diagram.



- **Use case: Add**
- **Primary actor: user**
- **Main success scenario:**
    - The user introduces data in the text fields corresponding for the two polynomials.
    - The user presses the button for addition.
    - The application checks if the text fields are filled and gets the strings form there
    - The application transforms the strings into polynomial objects.
    - The application performs the addition operation on the polynomials.
    - The application transforms the polynomials back into strings.
    - The application displays the result of the operation in the text field labeled as result
- **Alternate case scenario:**
    - The user introduces data in the text fields corresponding for the two polynomials.
    - The user presses the button for addition.
    - The application checks if the text fields are filled and observes that the text fields are not filled properly;

    - The application provides an error message.
    - The sequence goes back to the step one and the user needs to introduce another set of data, valid if possible.

- **Use case: Subtract**
- **Primary actor: user**
- **Main success scenario:**
    - The user introduces data in the text fields corresponding for the two polynomials.
    - The user presses the button for subtraction.
    - The application checks if the text fields are filled and gets the strings form there
    - The application transforms the strings into polynomial objects.
    - The application performs the subtraction operation on the polynomials.
    - The application transforms the polynomials back into strings.
    - The application displays the result of the operation in the text field labeled as result
- **Alternate case scenario:**
    - The user introduces data in the text fields corresponding for the two polynomials.
    - The user presses the button for subtraction.
    - The application checks if the text fields are filled and observes that the text fields are not filled properly;
    - The application provides an error message.
    - The sequence goes back to the step one and the user needs to introduce another set of data, valid if possible.

- **Use case: Multiply**
- **Primary actor: user**

- **Main success scenario:**
    - The user introduces data in the text fields corresponding for the two polynomials.
    - The user presses the button for multiplication.
    - The application checks if the text fields are filled and gets the strings form there
    - The application transforms the strings into polynomial objects.
    - The application performs the multiplication operation on the polynomials.
    - The application transforms the polynomials back into strings.
    - The application displays the result of the operation in the text field labeled as result
- **Alternate case scenario:**
    - The user introduces data in the text fields corresponding for the two polynomials.
    - The user presses the button for multiplication.
    - The application checks if the text fields are filled and observes that the text fields are not filled properly;
    - The application provides an error message.
    - The sequence goes back to the step one and the user needs to introduce another set of data, valid if possible.

- **Use case: Differentiation**
- **Primary actor: user**
- **Main success scenario:**
    - The user introduces data in the text field corresponding for the polynomial.
    - The user presses the button for differentiation.
    - The application checks if the text fields are filled and gets the string form there.
    - The application transforms the string into a polynomial object.

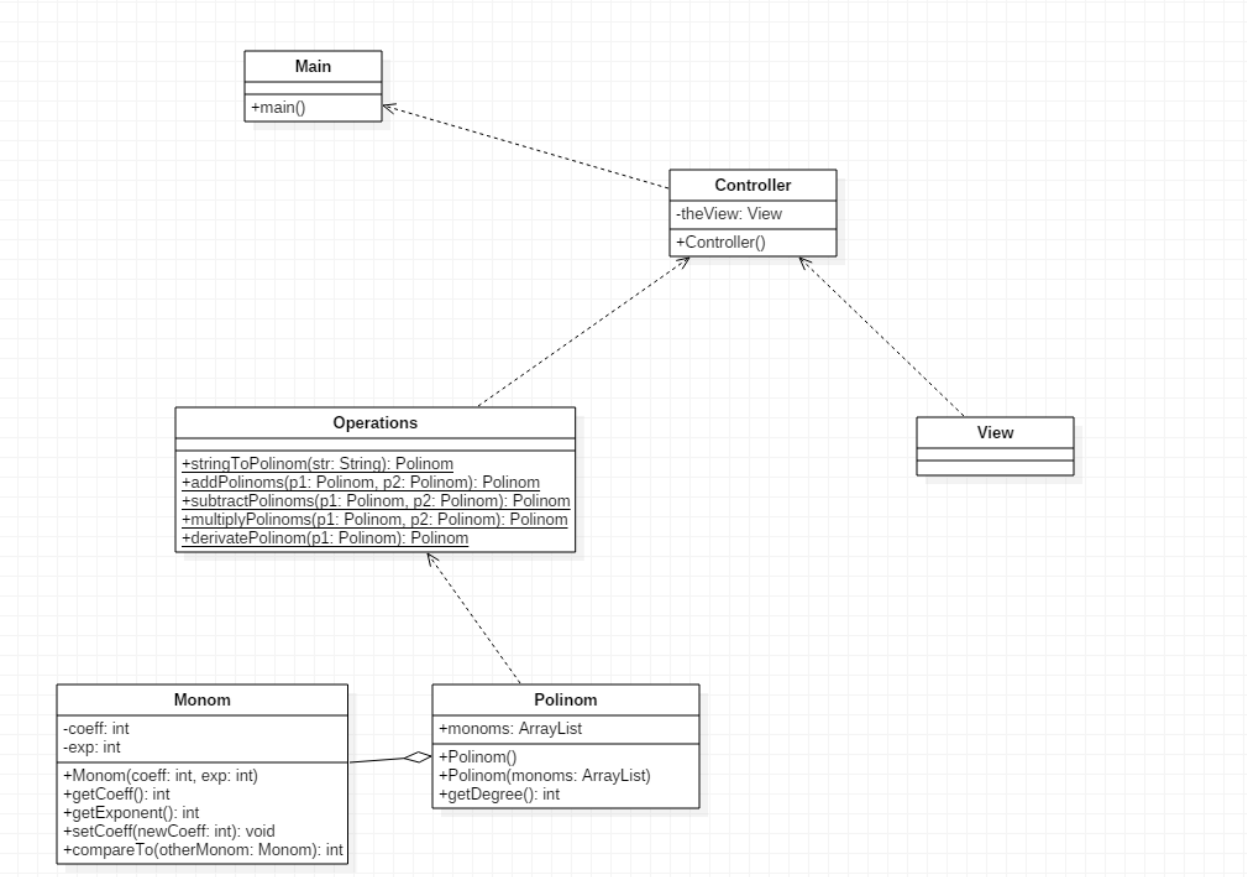
- The application performs the differentiation operation on the polynomial.
- The application transforms the polynomial back into string.
- The application displays the result of the operation in the text field labeled as result.

- **Alternate case scenario:**
  - The user introduces data in the text field corresponding for the polynomial.
  - The user presses the button for differentiation.
  - The application checks if the text fields are filled and observes that the text fields are not filled properly;
  - The application provides an error message.
  - The sequence goes back to the step one and the user needs to introduce another set of data, valid if possible.

3. Projecting

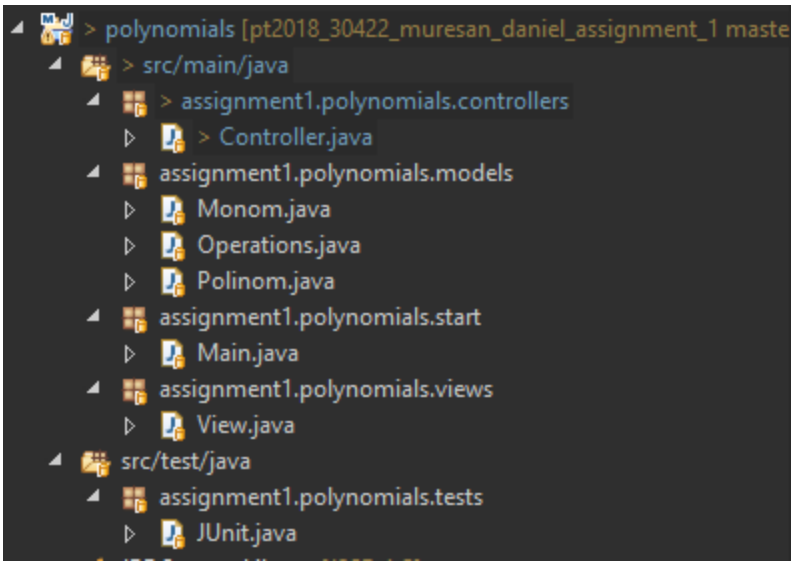
The classes I used for developing this application are the following ones: Monom, Polinom and Operations as models; View as view; Controller as controller; Main with the main method that launches the application and JUnit for the tests I ran for this application.

All the classes and the relationships between them are displayed in the next UML class diagram.



I used the model view controller pattern for organizing the packages. I used the next packages as follows: assignment1.polynomials.models, assignment1.polynomials.views, assignment1.polynomials.controllers and assignment1.polynomials.start. I also have a package called assignment1.polynomials.test for the tests class.

The structure of the packages is presented in the next image.



4. **Implementation**

Monom class

Is the class that corresponds to the monomials which together form the polynomial. The class has only two attributes: coeff and exp both integer representing the coefficient and the exponent of the monomial. It has a constructor with two integers given as parameters for the two attributes of the class. It also has getters for both attributes and setter for the coefficient and one last method compareTo() used to sort an Array List of objects of this class.

Polinom class

Is the class meant to model the polynomial in order for the operations to be applied. The class has only one attribute: an Array List of Monom objects representing the full polynomial. It has two constructors , one without parameters and one with an array list of Monom objects as parameter.One last method is the getDegree() method which returns an integer representing the largest exponent from the list of Monom objects.

Operations class

Is the core of the application, where all the operations are performed on Polinom objects. There are 6 static methods for the required operations. The first method is called stringToPolinom() and receives as parameter a string and returns an object of class Polinom. For this method I used regex expressions to convert the string into a polynomial object. The second method is called addPolinoms() and receives as parameters 2 objects of the class Polinom and returns also an object of the class Polinom with the sum of the two input polynomials. The next method is called subtractPolinoms() and also receives 2 objects of the class Polinom and returns an object of the class Polinom. The next method is called multiplyPolinoms() and receives as parameters 2 objects of the class Polinom and returns as result an object of class Polinom with the result of the multiplication of the input objects. The next method is called derivatePolinom() and receives as parameter an object of class Polinom and returns another object of class Polinom with the differentiated input polynomial. The last method from this class is called polinomToString() and receives as parameter an object of class Polinom and returns the String that represents that particular polynomial.

View class

Is the main component of the graphic user interface and represents the frame that the user will see and interact with. It has labels, text fields and buttons corresponding to the implemented operations and the required inputs. All the buttons are triggered from the Controller class.

Controller class

Is another important component of the graphic user interface and controls the functionality of the View class. It has as attribute an object of the class View which is instantiated in the constructor of this class. Also in the constructor of the class, the functionality of the buttons from the view frame is set with lambda expressions and action listeners.

Main class

Is the class that contains the main() method which runs the application. There is instantiated an object of the class Controller and so the application is launched.

JUnit class

Is the class where I used the JUnit framework to run some test for the main operations implemented by the methods from the class Operations.

5. **Testing**

For testing the application I used the JUnit framework and I run separate tests for all methods that perform required operations on polynomials.

The first test is called testAdd() and there are instantiated two objects of class Polinom and an object of class Polinom with the expected result and then the results are checked for equality.

Inputs:

String1: x^3 - 7x^2 + 2x + 1;

String2: x^4 + 2x^3 - 8;

Expected result: x^4 + 3x^3 - 7x^2 + 2x - 7;

Obtained result: x^4 + 3x^3 -7x^2 +2x - 7;

The next test is called testSub() and there are instantiated two objects of class Polinom and an object of class Polinom with the expected result and then the results are checked for equality.

Inputs:

String1: x^3 - 7x^2 + 2x + 1;

String2: x^4 + 2x^3 - 8;

Expected result: - x^4 - x^3 - 7x^2 + 2x + 9;

Obtained result: - x^4 - x^3 - 7x^2 + 2x + 9;

The next test is called testMul() and there are instantiated two objects of class Polinom and an object of class Polinom with the expected result and then the results are checked for equality.

Inputs:

String1: x^3 - 7x^2 + 2x + 1;

String2: x^4 + 2x^3 - 8;

Expected result: x^7 - 5x^6 - 12x^5 + 5x^4 - 6x^3 + 56x^2 - 16x - 8;

Obtained result: x^7 - 5x^6 - 12x^5 + 5x^4 - 6x^3 + 56x^2 - 16x - 8;

The last test is called testDer() and there are instantiated one object of class Polinom and an object of class Polinom with the expected result and then the results are checked for equality.

Inputs:

String: x^3 - 7x^2 + 2x + 1;

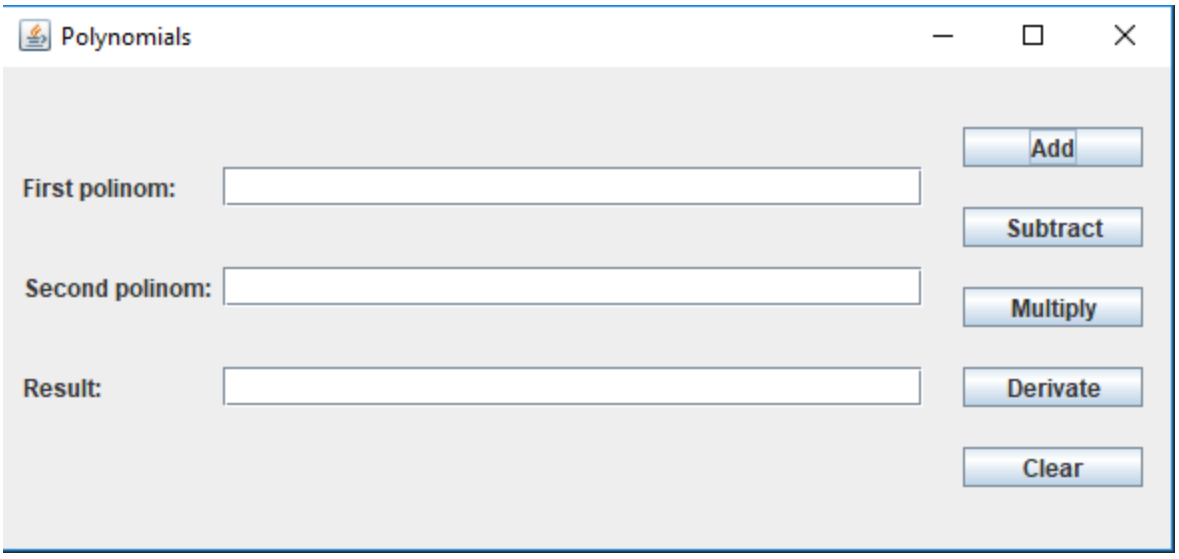
Expected result: 3x^2 - 14x + 2;

Obtained result: 3x^2 - 14x + 2;

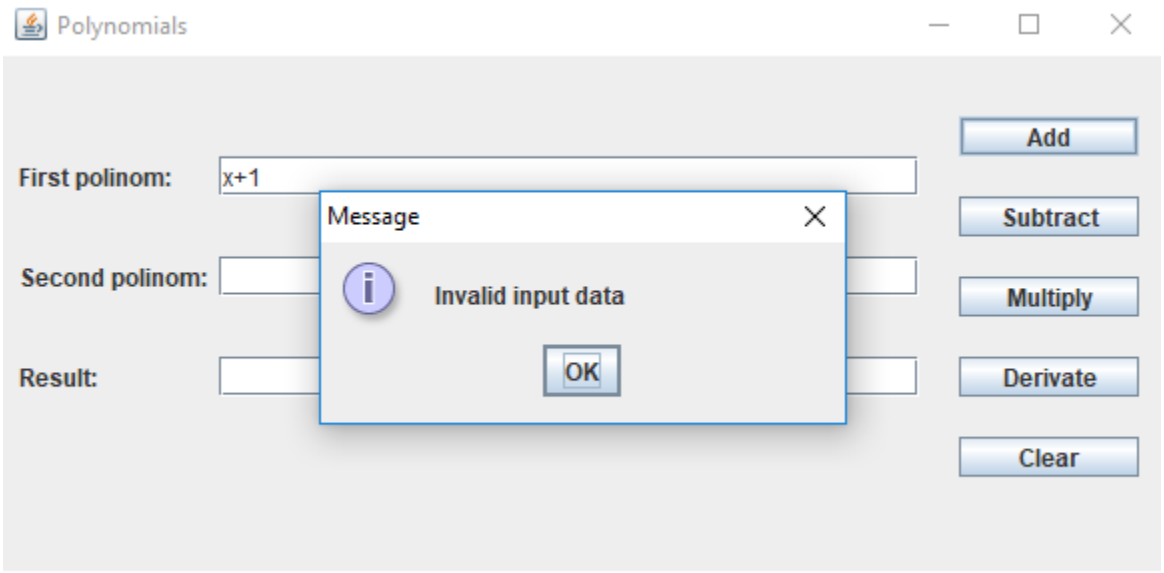
6. **Results**

In this section I will present a few screenshots with the final application.

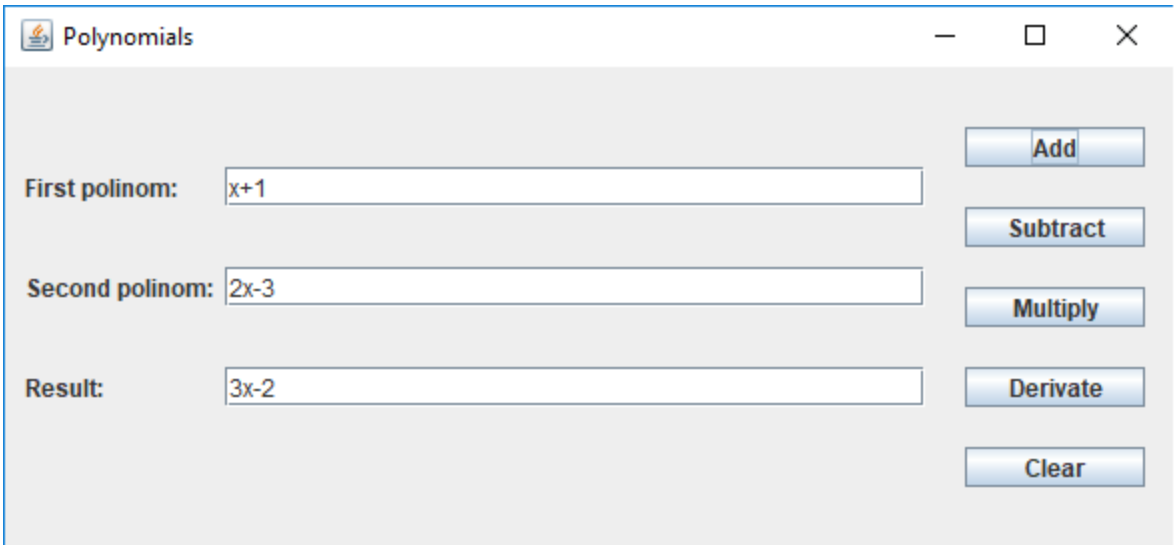
The first image is with the frame that the user sees when the application is launched.



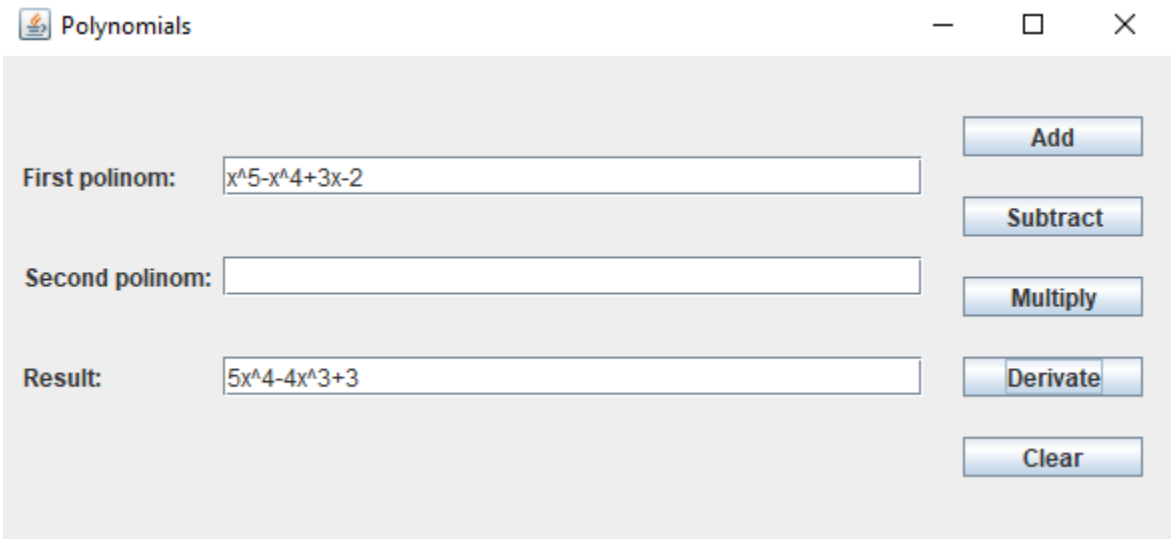
In the next image is presented the error message provided by the application when the fields are not filled properly.



The next image is the result of a successful sum operation.



The last image is the result of a differentiation operation.



7. **Conclusions**

In developing this application I learnt to use regex expressions. The main challenge in developing this application was the conversion from string to polynomial and back. For further developments the first step is to implement integration and division which require a special treatment because the operations might return float results.

8. **Bibliography**

https://regex101.com/

http://www.dummies.com/programming/java/how-to-use-lambda-expressions-to-handle-events-in-java/

http://www.mkyong.com/unittest/maven-and-junit-example/