# 

[Regex:](#_tepsvkvuyzlt)

[Problem 1:](#_sa1f4kki3gl4)

[String and String Buffer:](#_eiuja6lkgrze)

[Problem 2](#_ymdvqvp5x0j2)

[Lookahead](#_9iurdw8edlk8)

[Program 3](#_ok4pza4r09jk)

[Program 4](#_gx6iyzkgheuy)

# 

# Regex:

Regular expressions in java is shortly called as regex.Regex is often used in strings to perform matching,searching operations.It is also widely used to validate password.

Reference : <https://www.rexegg.com/>

Pattern Class and Matcher class: Pattern class is used to define the regex and Matcher contains matcher( ) method which is used match the regex pattern against the string.

## Problem 1:

Write a program that takes an integer ‘n’.For ‘n’,get a line of text from the user.Concat

them using a new line character and display.

Input : n - 3

Enter text : Applets for interactivity

Enter text : Multithreading for performance

Enter text : Servlets for client server communication

Output : Applets for interactivity

Multithreading for performance

Servlets for client server communication

Get input from the console and concat using append method available in StringBuffer class and display each in the new line.

### String and String Buffer:

The basic difference between them is that String is immutable(cannot be changed) where as StringBuffer is mutable.length of the String cannot be increased where as the length of the StringBuffer can be increased.

Performance of the concatenation is faster in StringBuffer compare to String.

Reference: <https://techdifferences.com/difference-between-string-and-stringbuffer.html>

## Problem 2

Password checker :

A valid password

1) Must contain at least 8 characters and maximum 15 characters

2) Can contain letters,digits and only special characters like - ! @ \_ $

3) Should contain at least one digit and two letters

Prompt the user ‘Invalid password’ if none is satisfied.

Rule 1:

Contains 3/more digits

Rule 2:

Contains digits , special characters and letters

Rule 3:

Contains combination of uppercase and lowercase

Display ‘Weak’ if none of rules gets satisfied

‘Good’ if atleast one is satisfied

‘Moderate’ if any two are satisfied

‘Strong’ if all gets satisfied

Here is where the regex comes into picture.

* To fix the min and max :{min,max} can be used. For eg:[a-z]{8,15}
* To include both uppercase and lowercase [a-zA-Z] can be used.
* \* indicates 0 or more.
* + indicates 1 or more.
* ? indicates 0 or 1.
* ^ it indicates negation[^abc] should not contain a or b or c.
* ^ it is to indicate the start and $ is to indicate the end.

Regex for valid password is: ^(?=.\*[0-9])(?=[a-zA-Z]{2,})([@$!A-Za-z0-9\_-]+)

### Lookahead

\d+(?= dollars)

Sample Match: 100 in 100 dollars

Explanation: \d+ matches the digits 100, then the lookahead (?= dollars) asserts that at that position in the string, what immediately follows is the characters " dollars"

## Program 3

Validate XML/HTML:

Given a ​ X

ML/HTML tags,print valid or invalid

A tag is valid if it has a starting and closing tag.Closing tag must end with a slash.

A tag that is opened first must end last.Input : <h1> Java basics </h1>

Output : valid

Input :<div> <p> Java basics </p> </div>

Output : Valid

Input : <div> Java basics <div>

Output : Invalid

Regex used for validating html:<.\*?>([^<]+)</.\*?>.

If the string does not match then print invalid else print valid.

.\*---Because of the greedy quantifier, the dot-star matches all the characters to the very end of the string.

.\*?--- guarantees that the quantified dot only matches as many characters as needed for the rest of the pattern to succeed.

Reference:

<https://www.rexegg.com/regex-quantifiers.html>

## Program 4

Design a class Contact with the following fields

1) Name

2) Email

3) Number

Create two instances of Contact using the input entered in the console.

Check for their equality by comparing the instances.

Compare each fields in an object with the other object fields. If equal print equal.

Alter method:

<https://stackoverflow.com/questions/13387742/compare-two-objects-with-equals-and-operator>