JAVA BASICS

[INPUT AND OUTPUT:](#_gz0tavbfp1sq)

[REGEX:](#_yf84e5vaxryt)

[CAPTURING GROUPS :](#_lfcw08pek4lb)

[BACK REFERENCING:](#_hah06spo117n)

[PROGRAM 1:](#_avte8l3iknjh)

[Solved program 1:](#_wd4wpjydswfc)

[PROGRAM 2:](#_rmu7ov4umg3n)

[Solved program 2:](#_7ticltyqqxkz)

[PROGRAM 3:](#_jsthuc4bn1os)

[Solved program 3:](#_1xnoz4ahcnk3)

[PROGRAM 4:](#_jlx1psdqdrax)

[Solved program 4:](#_m49af7nqxx0i)

#### INPUT AND OUTPUT:

initialization and declaration of variables.

displaying the text.

#### 

#### REGEX:

Writing regular expression for any specified condition.

[0-9] ---> your input can have any digits between 0 to 9.

[a-zA-Z] ---> can have uppercase or lowercase characters.

. --> can be anything (characters or digits).

^ ---> the input should start with the expression that follows ^.

\* ---> 0 or more instances.

+ ---> 1 or more instances.

$ ---> the input should end with the expression that is before $.

[0-9]{2} --> no.of occurences of digits should be 2 .

[0-9]{2,5}---> no.of occurences of digits should be between 2 and 5.

[0-9] {1,} --->no.of occurences should be atleast 1and maximum of any.

(?= .\*\d) --> this means the pattern should be 0 or more instances of anycharacter followed by digits and that is not captured ---?= is called lookahead

#### CAPTURING GROUPS :

Parentheses group the regex between them. They capture the text matched by the regex inside them into a numbered group that can be reused with a numbered backreference.

EG: (abc) here abc is captured and can be referred to as \\1 in future

#### BACK REFERENCING:

EG:(abc|def)=\1 matches abc=abcor def=def, but not abc=def or def=abc.

## PROGRAM 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

### Solved program 1:

<https://github.com/sruthiviswanathan/Javabasics/blob/master/javabasics/src/com/zilker/basics/Display.java>

Get the number of lines the user wishes to enter.

Use for loop to get the line one by one and concat it to a string.

Print the string finally.

## PROGRAM 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

Input : goodmorning

Output : Invalid password

Input : good@Morning1

Output : Moderate

### [Solved program 2:](https://github.com/sruthiviswanathan/Javabasics/blob/master/javabasics/src/com/zilker/basics/Password.java)

<https://github.com/sruthiviswanathan/Javabasics/blob/master/javabasics/src/com/zilker/basics/Password.java>

Get the input string from user and check if the length lies between 8 and 15.

If the above condition satisfies, match the input string to the regex expression.

String ***passwordPattern***="^(?=.\*[0-9])(?=[a-zA-Z]{2,})([@$!A-Za-z0-9\_-]+)";

If the string does not match then print invalid and stop,else move to next step.

Have separate regex expression for rule1,rule2,rule3.

String r1 = "(.\*?\\d){3,}";

String r2 = "([@$!a-zA-Z0-9\_-]+)";

String r3 = "(.\*a-z){1,}";

String r4 = "(.\*A-Z){1,}";

Match rule1,rule2,rule3 with user input, when the string matches each rule increment counter variable by 1.

If counter is 1 print weak

else if counter is 2 print moderate

else if counter is 3 print strong

## 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

### Solved program 3:

<https://github.com/sruthiviswanathan/Javabasics/blob/master/javabasics/src/com/zilker/basics/HtmlValidator.java>

Get the input string from user

Have a regex expression to validate html tags.

String ***htmlPattern*** = "<(.\*?)>([^<]+)</\\1>";

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

## 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.

### Solved program 4:

<https://github.com/sruthiviswanathan/Javabasics/blob/master/javabasics/src/com/zilker/basics/Contact.java>

Create a class with members in it.

Create 2 instances and get the input from user.

If all the members of both the instances have same values print same else not same.