**JAVA BASICS**

**CONVENTIONS TO BE FOLLOWED**

1. Camel case style of variables is to be followed.
2. Proper variable naming is to followed according to the context.

**CONCEPTS LEARNED**

1. Java inputs and outputs
2. Regex.
3. Capturing groups.
4. Back referencing.

**PROGRAMS SOLVED**

1. String appending.
2. Password checker.
3. Validate XML/HTML tags.
4. Object comparison.

**JAVA INPUTS AND OUTPUTS**

1. Initialize and declare the variables.
2. Develop the business logic.

#### **REGEX**

1. Writing regular expression for any specified condition.
2. [0-9] - your input can have any digits between 0 to 9.
3. [a-zA-Z] - can have uppercase or lowercase characters.
4. . - can be anything (characters or digits).
5. ^ - the input should start with the expression that follows ^.
6. \* - 0 or more instances.
7. + - 1 or more instances.
8. $ - the input should end with the expression that is before $.
9. [0-9]{2} - no.of occurences of digits should be 2 .
10. [0-9]{2,5} - no.of occurences of digits should be between 2 and 5.
11. [0-9] {1,} - no.of occurences should be atleast 1 and maximum of any.
12. (?= .\*\d) - this means the pattern should be 0 or more instances of any character 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.

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

#### **BACK REFERENCING**

Example: (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

**SOLUTION**

## **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

### **SOLUTION**

## **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

### **SOLUTION**

## **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.

### **SOLUTION**