

Object Oriented Programming with Java

Practical List

Last Updated on 26 / 08 / 2017

Lab #	Aim	#Hrs
1	<p>Programs to demonstrate printing to console and taking input as command line arguments</p> <ol style="list-style-type: none">1. Write a Java program to display “Hello World” on the console.2. Write a Java program to take username (e.g. Ravi) as command line argument and display “Welcome Ravi!” on the console.3. Write a Java program to find sum of digits of a number taken as a command line argument.4. Write a Java program to add two numbers entered as command line argument. [Extend this program to develop simple calculator.]	2
2	<p>Programs to understand concept of class and methods, use of array and Scanner class.</p> <ol style="list-style-type: none">1. Write a Java program to find minimum and maximum value from the given array.2. Write a Java program to demonstrate use of Scanner class.<ol style="list-style-type: none">a. Take two whole numbers (suppose x and y) from user and compute xy and display the result.b. Scan three statements (suppose s1, s2, s3). Display the output as s1@@s2##ss3. [s1 is “Good Morning”, s2 is “Good Afternoon” and s3 is “Good Night”. Output should be Good Morning@@Good Afternoon## Good Night.3. Do as directed for a given class Student as below<pre>class Student { int roll_no; String name; int marks[] = new int[5]; }</pre><ol style="list-style-type: none">a. Store details of one student by creating one object of Student class and display them.b. Add double findAverage() method in the Student class.c. Store details of 3 students by creating array of object of Student class and display the details of the student who has highest average amongst the three students.	2
3	<p>Programs to demonstrate recursion, constructor overloading, this keyword and passing objects as parameters.</p>	2

	<p>1. Write a java program to find factorial using recursion.</p> <p>2. Write a java program as follows:</p> <p><u>Box class contains following :</u></p> <p>Properties: width, height, depth</p> <p>Constructors:</p> <ul style="list-style-type: none"> • Box() : which initializes all properties to -1 • Box(double width, double height, double depth) : which sets parameter values to corresponding properties. (Hint : use this keyword) • Box(double length) : which sets length value to all properties in case Box is a cube. • Box(Box box) : which sets values of box properties to corresponding properties of current object. (i.e. copy constructor) <p>Methods:</p> <ul style="list-style-type: none"> • double findVolume() : which returns volume of the box calculated using properties of the box. <p><u>BoxDemo class :</u></p> <p>Create four box objects using each of the constructors and display volume of each box.</p>	
4	<p>Programs to demonstrate inheritance, method overriding, abstract class and final keyword.</p> <p>1. Write a java program containing following classes with mentioned features.</p> <ul style="list-style-type: none"> • Person class : <ul style="list-style-type: none"> • An instance variable fullName of type String. • A constructor with parameter to initialize fullname. • An abstract method getDescription() with return type String. • A concrete method getName() which returns fullName of that person. • Employee class: <ul style="list-style-type: none"> • An instance variable salary. • A constructor with two parameters full name and salary. • A method getSalary() which returns salary of that employee. • An implementation of getDescription() which returns description of type of that person (i.e. employee) with salary value. • Student class: <ul style="list-style-type: none"> • An instance variable branch. • A constructor with two parameters full name and branch. • A method getBranch() which returns branch in which that student is studying. • An implementation of getDescription() which returns description 	2

	<p>of type of that person (i.e. student) with his/her branch name.</p> <ul style="list-style-type: none"> • DemoAbstract class: <ul style="list-style-type: none"> • In main method, create two variables person1 and person2 of Person type. • Initialize person1 with an object of Employee type with necessary values. • Initialize person2 with an object of Student type with necessary values. • Print description of each person. • Sample output : <ul style="list-style-type: none"> • Manoj Pandey, an employee with a salary of Rs. 50,000. • Rohini Dave, a student studying in Computer Science. <p>2. Write a java program with classes Person and Employee as in above problem. Also write classes Manager (subclass of Employee) and DemoFinal as per the following description.</p> <ul style="list-style-type: none"> • Manager class : <ul style="list-style-type: none"> • A String instance variable type which stores value of manager type (i.e. HR, General, Finance..). • A constructor with three parameters full name, salary and type. • An instance method increaseSalary(), which takes an amount by which manger's salary will be incremented as a parameter and new increased amount is set as a new value to variable salary. • An overridden method getDescription() which also prints the manager type of that manager. • DemoFinal class : <ul style="list-style-type: none"> • In main method, create an instance manager1 of type Manager with suitable initial values. • Print description of manager1. • Call increaseSalary() on manager1 by passing some appropriate value. • Print description of manager1 again. • Execute a program with different cases : <ul style="list-style-type: none"> • Case A: Make an Employee class final. • Case B: Make a method getDescription() in Employee class as a final method. • Case C: Make an instance variable salary in Employee class as a final variable. 	
5	<p>Programs to demonstrate use of interface and multidimensional array</p> <p>1. Write a java program as per the given description to demonstrate use of interface.</p> <ul style="list-style-type: none"> • Define an interface RelationInterface. • Write three abstract methods: isGreater, isLess and isEqual. All methods have 	2

	<p>return type of boolean and take an argument of type</p> <ul style="list-style-type: none"> Line with which the caller object will be compared. Write Line class implements RelationInterface interface. It has 4 double variables for the x and y co-ordinates of the line. Write a constructor in Line class that initializes these 4 variables. Write a method getLength() that computes length of the line. [double length = Math.sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))]. Implement the methods in interface. In class CompareLines.java, create two objects of Line class, call the three methods to compare the lengths of the lines. <p>2. Write a java program as per the given description to demonstrate use of interface for multiple inheritances.</p> <ul style="list-style-type: none"> Define two interfaces named Terrestrial and Aquatic. In Terrestrial interface, write two methods eats() and walks(). Both methods have return type void and take no arguments. In Aquatic interface, write two methods eats() and swims(). Both methods have return type void and take no arguments. Write a class Amphibian that implements both the interfaces. Write all methods. In DemoMultipleInterface class, create Frog object of Amphibian and call all the methods. <p>3. Write a program that inputs two 3 X 3 matrices and performs matrix multiplication on them. Display properly formatted output.</p>	
6	<p>Programs to demonstrate String class methods.</p> <ol style="list-style-type: none"> Write a program that returns the string made of the first two characters of the given string. E.g. for given string "Hello" return "He". [If entered string is shorter than length 2 then return that string itself and return empty string for an empty string.] Write a program that takes two strings and returns a string in the form of short+long+short, with the shorter string on the outside and longer on inside. Write a program that returns the number of times that the string "Hi" appears anywhere in the given string. Write a java program which checks whether input string is palindrome or not and then display appropriate message [e.g. "Refer" is a palindrome string]. Write a java program that prints the words of sentence given as input in reverse order. E.g. Input : This is a java programming lab Output : lab programming java a is This 	2

7	<p>Programs to demonstrate use of Exception handling</p> <ol style="list-style-type: none"> 1. Write a program that catches divide-by-zero exception using try-catch mechanism. Take a numeric value and perform division by zero. Catch the ArithmeticException. 2. Write a program that demonstrates use of multiple catch blocks. Observe the importance of order of exception being caught in various catch blocks. Take an integer array a[] of size 5. Divide the element a[5] by zero. Catch IndexOutOfBoundsException, ArrayIndexOutOfBoundsException and Exception in the respective catch blocks. 3. Write a program that demonstrates use of finally block. Observe the output of your program for different cases as mentioned below. <p>Case A: exception does not occur. Perform 25/5 mathematical operation. Catch NullPointerException.</p> <p>Case B: exception occurs but not handled. Perform 25/0 mathematical operation. Catch NullPointerException.</p> <p>Case C: exception occurs and handled. Perform 25/0 mathematical operation. Catch ArithmeticException.</p> 	2
8	<p>Programs to demonstrate custom exceptions and multithreading</p> <ol style="list-style-type: none"> 1. Write a program that creates and throws custom exception using throw keyword. Handle the custom exception in your program and print it. <ul style="list-style-type: none"> • Create a class InvalidAgeException that extends Exception. The constructor of InvalidAgeException with a String argument calls superclass constructor. • Create a CustomExceptionDemo class that has a static method void validate that take integer argument age and returns void. This method throws InvalidAgeException. Throw a new InvalidAgeException if the age is below 18 years. From the main method, call the validate method. 5. 2. Write a program that demonstrates the use of throws keyword for propagation of checked exception <ul style="list-style-type: none"> • Create a class Simple with three methods m(), n(), p() with no arguments and void return type. m() and n() throws IOException. p() uses try-catch block. • In m(), throw new IOException("device error") as a checked exception. Call m() from n(). Call n() from p(). • In the main(), create a Simple class object and call the p() method. 3. Write a java program which creates two threads from main thread. Both newly created thread prints 1 to 5 with its name which was given to it while creating. Thread must sleep for 200 ms between two print statements. Main thread prints its 	2

	<p>name then creates two new threads and exits. Observe the output.</p> <p>4. Write a java program which creates two threads from main thread. Both newly created thread prints 1 to 5 with its name which was given to it while creating. Thread must sleep for 200 ms between two print statements. Main thread prints its name then creates two new threads and waits for those two threads to finish. Observe the output.</p>	
--	---	--