WEEK-1: Fundamentals of Classes and Objects.

1. Design a Java class "Demo" to display "Hello Everyone...!! Welcome to RVR&JC...!!" on the console.

2. Design a Java class "Circle" with the following criteria:

   Instance var: pi (should be declared as static with constant value)

   Local var's: radius

   Methods: area( ), perimeter( )

   Note: a. The above methods should be non-static.

   b. The above methods should accept radius as an argument and should return a value.

   c. Input should be read dynamically.

3. Design a Java class "Student" with the following criteria:

   Instance var's: Rollno, Name, Gender, Sec, Dept, College

   Methods: disp( )

   Note: a. No constructors should be used.

   b. Create an object "s1" and use the created object to initialize the instance variables.

   c. The above method should be non-static and use that method to display the values of all the variables.

   d. Input should be read dynamically.

4. Design a Java class "Employee" with the following criteria:

   Instance var's: ID, Name, Gender, Salary, Location

   Methods: disp( )

   Note: a. Use parameterized constructor to initialize the instance variables.

   b. The names of formal parameters in the parameterized constructor should not match with the names of instance variables.

   c. The above method should be non-static and use that method to display the values of all the variables.

   d. Input should be read dynamically.

WEEK-2: Usage of static and this keyword.

1. Design a Java class "Student" with the following criteria:

   Instance var's: Rollno, Name, Gender, Sec, Dept, College

   Methods: disp( ), static block

   Note: a. Use parameterized constructor to initialize the instance variables.

   b. All the variables should be declared as "public".

   c. "College" should be declared as static with a constant "RVR&JC".

   d. The disp( ) method should be static and use that method to display the values of all variables.

   e. The static block should display the statement as "The statements of static block will be executed first".

   f. Input should be read dynamically, except "College".

2. Design a Java class "Employee" with the following criteria:

   Instance var's: ID, Name, Gender, Salary, Location

   Methods: disp( )

   Note: a. All the variables should be declared as "public".

   b. Use parameterized constructor to initialize the instance variables.

   c. The names of formal parameters in the parameterized constructor should match with the names of instance variables.

   d. Use this keyword to resolve the name conflicts.

   e. Use this keyword to invoke the disp( ) method.

   f. The above method should be non-static and use that method to display the values of all variables.

   g. Input should be read dynamically.

WEEK-3: Nested Classes (static), Inner Classes (Non-static), Constructor Overloading.

1. Design a Java class "StaticNestedDemo" with the following criteria:

   Instance var's: College

   Methods: disp( )

   Nested class: create a static nested class "Nested"

   Note: a. "College" should be declared as public & static with a constant "RVR&JC".

   b. The disp( ) method should be non-static and use that method to display the value of "College" variable.

   c. The disp( ) method should be enclosed in the static nested class "Nested".

   d. Create an object for the static nested class "Nested" to call the disp( ) method.

2. Design a Java class "NonStaticNestedDemo" with the following criteria:

   Instance var's: College

   Methods: disp( )

   Nested class: create a non-static nested class "Nested"

   Note: a. "College" should be declared as public with a constant "RVR&JC".

   b. The disp( ) method should be non-static and use that method to display the value of "College" variable.

   c. The disp( ) method should be enclosed in the non-static nested class "Nested".

   d. Create an object for the non-static nested class "Nested" to call the disp( ) method.

3. Design a Java class "ConstructorOverloading" with the following criteria:

   Instance var's: pi, radius

   Constructors: default, 2 parameterized

   Methods: disp( )

   Note: a. All the variables should be declared as public.

   b. "pi" should be declared as static with a constant value.

   c. Create 3 objects to the class "ConstructorOverloading" and pass radius as an argument in the constructor calling statement.

   d. The disp( ) method should be non-static and use that method to display area and perimeter of a circle.

WEEK-4: Types of Inheritance.

1. Design a Java program with the following criteria:

   Classes: A (Super class), B (Sub class), SingleInheritance (Main class)

   Super class members: a (Instance var)

   Sub class members: b (Instance var), add( ) method

   Note: a. Initialize both the variables with some constant values and declare variable "a" as protected.

   b. The add( ) method should be non-static and use that method to display the addition result of both the variables.

   c. Create an object for the sub class "B" in the main class and call add( ) method.

2. Design a Java program with the following criteria:

   Classes: A (Grandparent class), B (Parent class), C (Child class),

   MultilevelInheritance (Main class)

   Grandparent class members: a (Instance var)

   Parent class members: b (Instance var), add( ) method

   Child class members: c (Instance var), mul( ) method

   Note: a. Initialize all the variables with some constant values and declare variable "a" & "b" as protected.

   b. The add( ) method should be non-static, protected and use that method to display the addition result of "a" & "b" variables.

   c. The mul( ) method should be non-static and use that method to display the multiplication result of "b" & "c" variables.

   d. Create an object for the child class "C" in the main class and call add( ), mul( ) methods.

3. Design a Java program with the following criteria:

   Classes: Bank (Super class), SBI (Sub class1), ICICI (Sub class2), HDFC (Sub class3),

   HierarchicalInheritance (Main class)

   Super class members: ROI (Instance var)

   Sub class1 members: disp( ) method

   Sub class2 members: disp( ) method

   Sub class3 members: disp( ) method

Note: a. Declare the super class variable as double, protected and Initialize with "0".

b. In Sub class1, use the super class variable and reinitialize it with "9.6" and display the initialized value with disp( ) method.

c. In Sub class2, use the super class variable and reinitialize it with "10.2" and display the initialized value with disp( ) method.

d. In Sub class3, use the super class variable and reinitialize it with "10.5" and display the initialized value with disp( ) method.

e. In all the sub classes, the disp( ) method should be non-static.

f. Create an object for each sub class in the main class and call their disp( ) method.

4. Design a Java program with the following criteria:

Classes: Employee (Super class), Salary (Sub class), SingleInheritance (Main class)

Super class members: Instance var's: Name, ID, Designation, Company

Sub class members: Instance var's: Basic, HRA, DA, PF, readInfo( ), readSal( ), calcSal( ) & disp( ) methods.

Note: a. Use the readInfo( ) method to read the input and initialize the super class variables. Input should be read dynamically.

b. Use the readSal( ) method to read the input and initialize the sub class variables. Input should be read dynamically.

c. Use the calcSal( ) method to calculate the Net salary of an employee.

Formula: Netsal = Basic + HRA + DA + PF

d. Use the disp( ) method to display the complete information of an employee.

e. Create an object for the sub class "Salary" in the main class and call all the methods.

WEEK-5: Method Overloading, Method Overriding, Usage of final and super keywords.

1. Design a Java class "MethodOverloading" with the following criteria:

   Methods: add( )

   Note: a. add( ) method needs to be overloaded 4 times within the same class.

   b. In the 1st version, add( ) method needs to perform addition between 2 integers and it should return integer.

   c. In the 2nd version, add( ) method needs to perform addition between 3 integers and it should return integer.

   d. In the 3rd version, add( ) method needs to perform addition between 2 floating numbers and it should return float.

   e. In the 4th version, add( ) method needs to perform addition between an integer and a floating number and it should return float.

2. Design a Java program with the following criteria:

   Classes: Bank (Super class), SBI (Sub class1), ICICI (Sub class2), HDFC (Sub class3), AXIS (Sub class4), MethodOverriding (Main class)

   Super class members: dispROI( ) method

   Sub class1 members: dispROI( ) method

   Sub class2 members: dispROI( ) method

   Sub class3 members: dispROI( ) method

   Sub class4 members: dispROI( ) method

   Note: a. In Super class, the dispROI( ) method should return "0".

   b. In Sub class1, the dispROI( ) method should return "8.8".

   b. In Sub class2, the dispROI( ) method should return "9.5".

   b. In Sub class3, the dispROI( ) method should return "9.4".

   b. In Sub class4, the dispROI( ) method should return "9.2".

   c. Create an object for each sub class in the main class and call their dispROI( ) method and display the returned value.

3. Design a Java program to demonstrate the following:
   a. Declaring and initializing final variables.
   b. Preventing a class from being derived from the other class.
   c. Preventing a method being overridden in the child class.

4. Design a Java program to demonstrate the following:
    a. Accessing super class variables in the child class.
    b. Calling super class constructors from the child class.
    c. Calling super class methods from the child class.

WEEK-6: Abstract Classes, Interfaces.

1. Design a Java program with the following criteria:

   Classes: ArithmeticOperations (Abstract & Parent class), AO (Child class), AbstractDemo (Main class)

   Abstract class instance var's: x, y

   Abstract methods: add( ), sub( )

   Concrete methods: mul( ), div( )

   Note: a. Define a parameterized constructor to initialize instance var's of abstract class.

   b. Object should be created for child class.

   c. Use super keyword to call parent class constructor.

   d. Input should be read dynamically.

2. Design a simple Java program to demonstrate defining and implement an interface.

3. Design a Java program with the following criteria:

   Interfaces: Interface1, Interface2, Interface3

   Classes: Demo (Implementation class), InterfaceDemo (Main class)

   Methods: Interface1: displ1( ), disp2( )

   Interface2: disp3( ), disp4( )

   Interface3: disp5( ), disp6( )

   Note: a. In every interface method try to display a statement.

   b. Interface2 is child of Interface1.

   c. Interface3 is child of Interface2.

   d. Object should be created for the implementation class.

4. Design a Java program to demonstrate Multiple Inheritance through interfaces.

WEEK-7: Packages

1. Design a simple Java program with the following criteria:

   Package name: ArithmeticOperations

   Class: PackageDemo (Main class)

   Methods in package: add( ), sub ( ), mul( ), div( )

   Note: a. Created package and Main class should be maintained in different directories & set the CLASSPATH before execution.

   b. Each method of package should accept 2 arguments and it must return a value.

   c. Call each package method in the main class to perform arithmetic operations.

   d. Input should be read dynamically.

WEEK-8: String class, StringBuffer class and its methods

1. Design a simple Java program to demonstrate creating a string and various String class methods.

2. Design a simple Java program to demonstrate creating a string using StringBuffer and various StringBuffer class methods.

WEEK-9: Exception Handling

1. Design a simple Java program to demonstrate how to handle an Arithmetic Exception using try and catch.

2. Design a simple Java program to demonstrate how to handle multiple exceptions using nested try block.

3. Design a simple Java program to demonstrate how to use multiple catch blocks.

4. Design a simple Java program to demonstrate how to use universal catch block.

5. Design a simple Java program to demonstrate how to throw an exception manually using throw keyword.

6. Design a simple Java program to demonstrate when and how to use throws keyword.

7. Design a simple Java program to demonstrate how to rethrow an exception.

8. Design a simple Java program to demonstrate how to create & handle a user defined exception. (NotEligileForVoteException)

9. Design a simple Java program to demonstrate when and how to use a finally block.