NAME : SOHAM SAHA

SEC : CS2A

ROLL : 61

ENROLLMENT : 12019009001389

SUB : OOP WITH JAVA LAB

Q1. Design an abstract class having two methods. Create Rectangle and Triangle classes by inheriting the shape class and override the above methods to suitably implement for Rectangle and Triangle class.

SOLUTION

abstract class Area\_Peri{

public abstract int area(int w,int l);

public abstract double perimeter(int a,int b);

}

class Rectangle extends Area\_Peri{

public int area(int w, int l) {

return w\*l;

}

public double perimeter(int w, int l) {

return 2\*(w+l);

}

}

class Triangle extends Area\_Peri{

public int area(int h,int b) {

return (h\*b)/2;

}

public double perimeter(int h, int b) {

return (h+h+b);

}

}

public class q1 {

public static void main(String args[]) {

Area\_Peri obj=new Rectangle();

System.out.println("Area of the Rectangle "+obj.area(10,20));

System.out.println("perimeter of the Rectangle "+obj.perimeter(10,20));

Area\_Peri obj1=new Triangle();

System.out.println("Area of the Triangle "+obj1.area(10,20));

System.out.println("perimeter of the Triangle "+obj1.perimeter(20,10));

}

}

Output :

Area of the Rectangle 200

perimeter of the Rectangle 60.0

Area of the Triangle 100

perimeter of the Triangle 50.0

Q2. Write a program in Java to illustrate the use of interface in Java.

SOLUTION

interface Example{

public void show();

}

class Test\_Example implements Example{

public void show() {

System.out.print("This is Interface");

}

}

public class q2 {

public static void main(String args[]) {

Example obj=new Test\_Example();

obj.show();

}

}

Output :

This is Interface

Q3. Create a general class ThreeDObject and derive the classes Box, Cube, Cylinder and Cone from it. The class ThreeDObject has methods wholeSurfaceArea ( ) and volume( ). Override these two methods in each of the derived classes to calculate the volume and whole surface area of each type of three- dimensional objects. The dimensions of the objects are to be taken from the users and passed through the respective constructors of each derived class. Write a main method to test these classes.

SOLUTION

import java.util.Scanner;

abstract class ThreeDObject{

abstract int SurfaceArea(int h, int w, int l);

abstract int volume(int h, int w, int l);

}

class Box extends ThreeDObject{

int SurfaceArea(int h, int w, int l) {

return (2\*(h\*w)+2\*(h\*l)+2\*(w\*l));

}

int volume(int h, int w, int l) {

return (h\*w\*l);

}

}

class Cube extends ThreeDObject{

int SurfaceArea(int a,int b,int c) {

return (6\*a\*b);

}

int volume(int a,int b,int c) {

return (a\*b\*c);

}

}

class Cone extends ThreeDObject{

int SurfaceArea(int h,int r,int a) {

return (int) (Math.PI\*a\*(r+Math.sqrt(Math.pow(h, 2)+Math.pow(r, 2))));

}

int volume(int h,int r,int a) {

return (int) (Math.PI\*a\*r\*(h/3));

}

}

class Cylinder extends ThreeDObject{

int SurfaceArea(int h,int r,int a) {

return (int) (2\*Math.PI\*r\*h+2\*Math.PI\*r\*a);

}

int volume(int h,int r,int a) {

return (int) (Math.PI\*r\*a\*h);

}

}

public class q3 {

public static void main(String args[])

{

ThreeDObject obj=new Box();

ThreeDObject obj1=new Cube();

ThreeDObject obj2=new Cone();

ThreeDObject obj3=new Cylinder();

System.out.println("Box surface area "+obj.SurfaceArea(10,20,30));

System.out.println("Cube surface area "+obj1.SurfaceArea(10,10,10));

System.out.println("Cone surface area "+obj2.SurfaceArea(10,20,20));

System.out.println("Cylinder surface area "+obj3.SurfaceArea(10,20,20));

System.out.println("Box volume " +obj.volume(10,20,30));

System.out.println("Cube volume "+obj1.volume(10,10,10));

System.out.println("Cone volume "+obj2.volume(10,20,20));

System.out.println("Cylinder volume "+obj3.volume(10,20,20));

}

}

Output :

Box surface area 2200

Cube surface area 600

Cone surface area 2661

Cylinder surface area 3769

Box volume 6000

Cube volume 1000

Cone volume 3769

Cylinder volume 12566

Q4. Write a program to create a class named Vehicle having protected instance variables regnNumber, speed, color, ownerName and a method showData ( ) to show “This is a vehicle class”. Inherit the Vehicle class into subclasses named Bus and Car having individual private instance variables routeNumber in Bus and manufacturerName in Car and both of them having showData ( ) method showing all details of Bus and Car respectively with content of the super class’s showData ( ) method.

SOLUTION

class Vehicle\_Main {

protected int regnNumber;

protected int speed;

protected String color;

protected String ownerName;

public Vehicle\_Main(int regnNumber, int speed, String color, String ownerName) {

this.regnNumber = regnNumber;

this.speed = speed;

this.color = color;

this.ownerName = ownerName;

}

}

class Bus extends Vehicle\_Main{

int routeNumber;

public Bus(int regnNumber, int speed, String color, String ownerName, int routeNumber) {

super(regnNumber, speed, color, ownerName);

this.routeNumber = routeNumber;

}

public String ShowData() {

return "Bus [routeNumber=" + routeNumber + ", regnNumber=" + regnNumber + ", speed=" + speed + ", color="+ color + ", ownerName=" + ownerName + "]";

}

}

class Car extends Vehicle\_Main{

String manufacturerName ;

public Car(int regnNumber, int speed, String color, String ownerName, String manufacturerName ) {

super(regnNumber, speed, color, ownerName);

this.manufacturerName = manufacturerName ;

}

public String ShowData() {

return "Car [manufacturerName=" + manufacturerName + ", regnNumber=" + regnNumber + ", speed=" + speed + ", color="+ color + ", ownerName=" + ownerName + "]";

}

}

public class q4 {

public static void main(String args[]) {

Bus obj=new Bus(1010,70,"Black","Krishnendu",199);

System.out.println(obj.ShowData());

Car obj1=new Car(1010,70,"Black","Krishnendu","TATA");

System.out.println(obj1.ShowData());

}

}

Output :

Bus [routeNumber=199, regnNumber=1010, speed=70, color=Black, ownerName=Krishnendu]

Car [manufacturerName=TATA, regnNumber=1010, speed=70, color=Black, ownerName=Krishnendu]

Q5. Create three interfaces, each with two methods. Inherit a new interface from the three, adding a new method. Create a class by implementing the new interface and also inheriting from a concrete class. Now write four methods, each of which takes one of the four interfaces as an argument. In main ( ), create an object of your class and pass it to each of the methods.

SOLUTION

interface Test1{

public void show1();

public void show2();

}

interface Test2{

public void show3();

public void show4();

}

interface Test3{

public void show5();

public void show6();

}

interface Test4{

public void show7();

public void show8();

}

interface MultiTest extends Test1,Test2,Test3,Test4{

public void newMethod();

}

class Multi implements MultiTest{

public void show1() {

System.out.println("Mehtod 1");

}

public void show2() {

System.out.println("Mehtod 2");

}

public void show3() {

System.out.println("Mehtod 3");

}

public void show4() {

System.out.println("Mehtod 4");

}

public void show5() {

System.out.println("Mehtod 5");

}

public void show6() {

System.out.println("Mehtod 6");

}

public void show7() {

System.out.println("Mehtod 7");

}

public void show8() {

System.out.println("Mehtod 8");

}

public void newMethod() {

System.out.println("Mehtod newMethod()");

}

}

public class q5 {

public static void main(String args[]) {

MultiTest obj=new Multi(); obj.show1();

obj.show2();

obj.show3();

obj.show4();

obj.show5();

obj.show6();

obj.show7();

obj.show8(); obj.newMethod();

}

}

Output :

Mehtod 1

Mehtod 2

Mehtod 3

Mehtod 4

Mehtod 5

Mehtod 6

Mehtod 7

Mehtod 8

Mehtod newMethod()

Q6. Create an interface Department containing attributes deptName and deptHead. It also has abstract methods for printing the attributes. Create a class hostel containing hostelName, hostelLocation and numberofRooms. The class contains methods for getting and printing the attributes. Then write Student class extending the Hostel class and implementing the Department interface. This class contains attributes studentName, regdNo, electiveSubject and avgMarks. Write suitable getData and printData methods for this class. Also implement the abstract methods of the Department interface. Write a driver class to test the Student class. The program should be menu driven containing the options:

i) Admit new student

ii) Migrate a student

iii) Display details of a student

For the third option a search is to be made on the basis of the entered registration number.

SOLUTION

import java.util.Scanner;

interface Department{

public final String deptName="CSE";

public final String deptHead="XYZ\_Sir";

public abstract String printData();

}

class Hostel{

protected String hostelName,hostelLocation;

protected int numberofRooms;

public Hostel(String hostelName, String hostelLocation, int numberofRooms) {

this.hostelName = hostelName;

this.hostelLocation = hostelLocation;

this.numberofRooms = numberofRooms;

}

}

class Student extends Hostel implements Department{

protected String studentName, regdNo, electiveSubject;

protected int avgMarks;

public Student(String hostelName, String hostelLocation, int numberofRooms, String studentName, String regdNo, String electiveSubject, int avgMarks) { super(hostelName, hostelLocation, numberofRooms); this.studentName = studentName;

this.regdNo = regdNo;

this.electiveSubject = electiveSubject;

this.avgMarks = avgMarks;

}

public String printData() {

return "Student [studentName=" + studentName + ", regdNo=" + regdNo + ", electiveSubject=" + electiveSubject + ", avgMarks=" + avgMarks + ", hostelName=" + hostelName + ", hostelLocation=" + hostelLocation + ", numberofRooms=" + numberofRooms + ", deptName=" + deptName + ", deptHead=" + deptHead + "]";

}

}

public class q6 {

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

String studentName, regdNo, electiveSubject;

int avgMarks,numberofRooms;

String hostelName,hostelLocation;

Student[] obj=new Student[10];

int x=1,i=1;

while(x!=0)

{

System.out.println("1: Admit");

System.out.println("2: Migrate");

System.out.println("3: Show");

int z=sc.nextInt();

if(z==1)

{

System.out.println("Enter Student Name");

studentName=sc.next();

regdNo="UEMK2NDCSE"+i;

System.out.println("Enter elected Subject");

electiveSubject=sc.next();

System.out.println("Enter avgMarks");

avgMarks=sc.nextInt();

System.out.println("Enter hostelName");

hostelName=sc.next();

System.out.println("Enter hostelLocation");

hostelLocation=sc.next();

System.out.println("Enter numberofRooms");

numberofRooms=sc.nextInt();

obj[i++]=new Student( hostelName, hostelLocation, numberofRooms, studentName, regdNo,electiveSubject, avgMarks);

System.out.println("Student Added::Reg Id"+regdNo);

}

else if(z==2)

{

System.out.print("Enter the last digit of the id ");

int o=sc.nextInt();

obj[o]=null;

}

else if(z==3){

System.out.print("Enter the last digit of the id ");

int o=sc.nextInt();

System.out.println(obj[o].printData());

}

else

{

x=0;

}

}

}

}

Output :

1: Admit

2: Migrate

3: Show

1

Enter Student Name

Krishnendu

Enter elected Subject

JAVA

Enter avgMarks

90

Enter hostelName

UEM\_hostel

Enter hostelLocation

NewTown

Enter numberofRooms

1

Student Added::Reg IdUEMK2NDCSE1

1: Admit

2: Migrate

3: Show

3

Enter the last digit of the id 1

Student [studentName=Krishnendu, regdNo=UEMK2NDCSE1, electiveSubject=JAVA, avgMarks=90, hostelName=UEM\_hostel, hostelLocation=NewTown, numberofRooms=1, deptName=CSE, deptHead=XYZ\_Sir]

Q7. Create an interface called Player. The interface has an abstract method called play() that displays a message describing the meaning of “play” to the class. Create classes called Child, Musician, and Actor that all implement Player. Create an application that demonstrates the use of the classes(UsePlayer.java )

SOLUTION

interface Player{

abstract public void play();

}

class Child implements Player{

public void play() {

System.out.println("This is Child");

}

}

class Musician implements Player{

public void play() {

System.out.println("This is Musician");

}

}

class Actor implements Player{

public void play() {

System.out.println("This is Actor ");

}

}

public class q7 {

public static void main(String args[]) {

Player obj=new Child();

Player obj1=new Musician();

Player obj2=new Actor();

obj.play();

obj1.play();

obj2.play();

}

}

Output :

This is Child

This is Musician

This is Actor

Q9. Create an abstract class MotorVehicle with the following details:

Data Members:

(a) modelName (b)modelNumber (c) modelPrice

Methods:

(a) display() to show all the details

Create a subclass of this class Carthat inherits the class MotorVehicle and add the following details:

Data Members:

(b) discountRate

Methods:

(a) display() method to display the Car name, model number, price and the discount rate.

(b) discount() method to compute the discount.

SOLUTION

abstract class MotorVehicle {

protected String modelName;

protected int mmdelPrice;

protected int modelNumber;

public MotorVehicle(String modelName, int mmdelPrice, int modelNumber) {

super();

this.modelName = modelName;

this.mmdelPrice = mmdelPrice;

this.modelNumber = modelNumber;

}

abstract public String display();

}

class Carthat extends MotorVehicle{

double discountRate;

public Carthat(String modelName, int mmdelPrice, int modelNumber, double d) {

super(modelName, mmdelPrice, modelNumber);

this.discountRate = d;

}

public String display() {

return "Carthat [discountRate=" + discountRate + ", modelName=" + modelName + ", mmdelPrice=" + mmdelPrice + ", modelNumber=" + modelNumber + "]";

}

public void discount()

{

System.out.println("Discount Price:"+(mmdelPrice-(discountRate\*mmdelPrice)));

}

}

public class q9 {

public static void main(String args[]) {

Carthat obj=new Carthat("i20", 1200000, 7768, 0.2); obj.discount();

System.out.println(obj.display());

}

}

Output :

Discount Price:960000.0

Carthat [discountRate=0.2, modelName=i20, mmdelPrice=1200000, modelNumber=7768]

Q10. Implement the below Diagram. Here, Asset class is an abstract class containing an abstract method displayDetails() method. Stock, bond and Savings class inherit the Asset class and displayDetails() method is defined in every class.

SOLUTION

abstract class Asset{

protected String descriptor;

protected String date;

protected int currentvalue;

public Asset(String descriptor, String date, int currentvalue) {

super();

this.descriptor = descriptor;

this.date = date;

this.currentvalue = currentvalue;

}

abstract public String display();

}

class Stock extends Asset{

int num\_share;

int share\_price;

int asset;

public Stock(String descriptor, String date, int currentvalue, int num\_share, int share\_price, int asset){

super(descriptor, date, currentvalue);

this.num\_share = num\_share;

this.share\_price = share\_price;

this.asset = asset;

}

public String display() {

return "Stock [num\_share=" + num\_share + ", share\_price=" + share\_price + ", asset=" + asset + ", descriptor=" + descriptor + ", date=" + date + ", currentvalue=" + currentvalue + "]";

}

}

class Bond extends Asset{

int intrest\_rate;

int asset;

public Bond(String descriptor, String date, int currentvalue, int intrest\_rate, int asset) {

super(descriptor, date, currentvalue);

this.intrest\_rate = intrest\_rate;

this.asset = asset;

}

public String display() {

return "Bond [intrest\_rate=" + intrest\_rate + ", asset=" + asset + ", descriptor=" + descriptor + ", date=" + date + ", currentvalue=" + currentvalue + "]";

}

}

class Savings extends Asset{

int intrest\_rate;

int asset;

public Savings(String descriptor, String date, int currentvalue, int intrest\_rate, int asset) {

super(descriptor, date, currentvalue);

this.intrest\_rate = intrest\_rate;

this.asset = asset;

}

public String display() {

return "Savings [intrest\_rate=" + intrest\_rate + ", asset=" + asset + ", descriptor=" + descriptor + ", date=" + date + ", currentvalue=" + currentvalue + "]";

}

}

public class q10 {

public static void main(String args[]) {

Asset obj=new Stock("Bruce", "07-04-2021", 50000, 10, 2500, 25000);

Asset obj1=new Bond("Barry", "06-04-2021", 40000, 5, 75000);

Asset obj2=new Savings("Krishnendu", "07-04-2021", 70000, 3,50000);

System.out.println(obj.display());

System.out.println(obj1.display());

System.out.println(obj2.display());

}

}

Output :

Stock [num\_share=10, share\_price=2500, asset=25000, descriptor=Bruce, date=07-04-2021, currentvalue=50000]

Bond [intrest\_rate=5, asset=75000, descriptor=Barry, date=06-04-2021, currentvalue=40000]

Savings [intrest\_rate=3, asset=50000, descriptor=Krishnendu, date=07-04-2021, currentvalue=70000]

Q11. Implement the below Diagram. Here AbstractProduct is only abstract class.

SOLUTION

abstract class AbstractProduct{

protected int productId;

protected String name;

protected String description;

public AbstractProduct(int productId, String name, String description) {

super();

this.productId = productId;

this.name = name;

this.description = description;

}

}

class Product extends AbstractProduct{

protected int price;

public Product(int productId, String name, String description, int price) {

super(productId, name, description);

this.price = price;

}

}

class Book extends Product{

protected int ISBN;

protected String Author,Title;

public Book(int productId, String name, String description, int price, int iSBN, String author, String title) {

super(productId, name, description, price);

ISBN = iSBN;

Author = author;

Title = title;

}

public String display() {

return "Book [ISBN=" + ISBN + ", Author=" + Author + ", Title=" + Title + ", price=" + price + ", productId=" + productId + ", name=" + name + ", description=" + description + "]";

}

}

class Travel\_Guide extends Book{

protected String Country;

public Travel\_Guide(int productId, String name, String description, int price, int iSBN, String author, String title, String country) {

super(productId, name, description, price, iSBN, author, title);

Country = country;

}

public String display() {

return "Travel guide [Country = "+ Country + ", ISBN = " + ISBN + ", Author = " + Author + ", Title = " + Title + ", price = "+price + ", product id = "+productId+", name = "+name+", description = "+description+"]";

}

}

class CompactDisc extends Product{

protected String Artist;

protected String Title;

public CompactDisc(int productId, String name, String description, int price, String artist, String title){

super(productId, name, description, price);

Artist = artist;

Title = title;

}

public String display() {

return "CompactDisc [Artist=" + Artist + ", Title=" + Title + ", price=" + price + ", productId=" + productId + ", name=" + name + ", description=" + description + "]";

}

}

public class q11 {

public static void main(String args[]) {

Book obj=new Book(101,"Chronicles","Chronicles, Volume One is a memoir written by American musician Bob Dylan.",700,1786,"Bob Dylan","The Chronicles:Vol 1");

Travel\_Guide obj1=new Travel\_Guide(102,"My Travel Journal","My Travel Journal:1st Edition",300,1786,"by The Unscripted Life (Author)","My Travel Journal","India");

CompactDisc obj2=new CompactDisc(103,"Abbey Road","Abbey Road:1969/The Beatles",5000,"The Beatles","Abbey Road:1969");

System.out.println(obj.display());

System.out.println(obj1.display());

System.out.println(obj2.display());

}

}

Output :

Book [ISBN=1786, Author=Bob Dylan, Title=The Chronicles:Vol 1, price=700, productId=101, name=Chronicles, description=Chronicles, Volume One is a memoir written by American musician Bob Dylan.]

Travel guide [Country = India, ISBN = 1786, Author = by The Unscripted Life (Author), Title = My Travel Journal, price = 300, product id = 102, name = My Travel Journal, description = My Travel Journal:1st Edition]

CompactDisc [Artist=The Beatles, Title=Abbey Road:1969, price=5000, productId=103, name=Abbey Road, description=Abbey Road:1969/The Beatles]

Q12. Implement the below diagram.

SOLUTION

abstract class Vechicle{

protected int ID;

protected String name;

protected String LicenseNumber;

public Vechicle(int iD, String name, String licenseNumber) {

ID = iD;

this.name = name;

LicenseNumber = licenseNumber;

}

abstract public String display();

}

class Two\_Wheeler extends Vechicle{

int SteeringHandle;

public Two\_Wheeler(int iD, String name, String licenseNumber, int steeringHandle) {

super(iD, name, licenseNumber);

SteeringHandle = steeringHandle;

}

@Override

public String display() {

return "Two\_Wheeler [SteeringHandle=" + SteeringHandle + ", ID=" + ID + ", name=" + name + ", LicenseNumber=" + LicenseNumber + "]";

}

}

class Four\_Wheeler extends Vechicle{

int SteeringWheel;

public Four\_Wheeler(int iD, String name, String licenseNumber, int steeringWheel) {

super(iD, name, licenseNumber); SteeringWheel = steeringWheel;

}

public String display() {

return "Four\_Wheeler [SteeringWheel=" + SteeringWheel + ", ID=" + ID + ", name=" + name + ", LicenseNumber=" + LicenseNumber + "]";

}

}

public class q12 {

public static void main(String args[]) {

Vechicle obj=new Two\_Wheeler(1011,"Hero Splender","WBIO0897",1);

Vechicle obj1=new Four\_Wheeler(1011,"Hyundai i20","WBIO4547",1);

System.out.println(obj.display());

System.out.println(obj1.display());

}

}

Output :

Two\_Wheeler [SteeringHandle=1, ID=1011, name=Hero Splender, LicenseNumber=WBIO0897]

Four\_Wheeler [SteeringWheel=1, ID=1011, name=Hyundai i20, LicenseNumber=WBIO4547]

Q13. Write a program to implement the Multiple Inheritance (Bank Interface, Customer & Account classes).

SOLUTION

interface Bank{

final String name="HDFC";

public String details();

}

class Customer{

protected String CustomerName;

public Customer(String name) {

super();

this.CustomerName = name;

}

}

class Account\_Main extends Customer implements Bank{

int acNo;

public Account\_Main(String name, int acNo) {

super(name); this.acNo = acNo;

}

public String details() {

return "Account [acNo=" + acNo + ", CustomerName=" + CustomerName + "]";

}

}

public class q13 {

public static void main(String args[]) {

Bank obj=new Account\_Main("Krishnendu",12585550);

System.out.println(obj.details());

}

}

Output :

Account [acNo=12585550, CustomerName=Krishnendu]

Q14. Write a program to implement the Multiple Inheritance (Gross Interface, Employee & Salary classes).

SOLUTION

interface Gross{

public String details();

}

class Employee1{

protected String EmployeeName;

public Employee1(String name) {

super();

this.EmployeeName = name;

}

}

class Salary extends Employee1 implements Gross{

int sal;

public Salary(String name, int sal) {

super(name); this.sal = sal;

}

@Override

public String details() {

return "Salary [Salary=" + sal + ", EmployeeName=" + EmployeeName + "]";

}

}

public class q14 {

public static void main(String args[]) {

Gross obj=new Salary("Krishnendu",80000);

System.out.println(obj.details());

}

}

Output :

Salary [Salary=80000, EmployeeName=Krishnendu]

Q15. Program to create a interface 'Mango' and implement it in classes 'Winter' and 'Summer'.

SOLUTION

interface Mango{

abstract public String Availability();

}

class Summer implements Mango{

boolean Availability ;

public Summer(boolean availability) {

Availability = availability;

}

public String Availability() {

return "Summer [Availability=" + Availability + "]";

}

}

class Winter implements Mango{

boolean Availability ;

public Winter(boolean availability) {

Availability = availability;

}

public String Availability() {

return "Winter [Availability=" + Availability + "]";

}

}

public class q15 {

public static void main(String args[]) {

Mango obj=new Summer(true);

Mango obj1=new Winter(false);

System.out.println(obj.Availability());

System.out.println(obj1.Availability());

}

}

Output :

Summer [Availability=true]

Winter [Availability=false]

Q16. Program to implement the Multiple Inheritance (Exam Interface, Student & Result classes).

SOLUTION

interface Exam{

abstract public String markSheet();

}

class Student\_Exam{

protected String name;

public Student\_Exam(String name) {

super();

this.name = name;

}

public void get()

{

}

}

class Results extends Student\_Exam implements Exam{

int percent;

public Results(String name, int percent) {

super(name);

this.percent = percent;

}

public String markSheet() {

return "Result [percentage=" + percent + ", name=" + name + "]";

}

}

public class q16 {

public static void main(String[] args) {

Exam obj=new Results("Krishnendu",97);

System.out.println(obj.markSheet());

}

}

Output :

Result [percentage=97, name=Krishnendu]

Q17. Program to demonstrate use of hierarchical inheritance using interface.

SOLUTION

interface A

{

public void displayA();

}

interface B extends A

{

public void displayB();

}

interface C extends A

{

public void displayC();

}

class Hierarchy implements B,C{

public void displayC() {

System.out.println("Hi this is C");

}

public void displayB() {

System.out.println("Hi this is B");

}

public void displayA() {

System.out.println("Hi this is A");

}

}

public class q17 {

public static void main(String args[])

{

Hierarchy obj=new Hierarchy();

obj.displayA();

obj.displayB();

obj.displayC();

}

}

Output :

Hi this is A

Hi this is B

Hi this is C

Q18. Java program to Perform Payroll Using Interface (Multiple Inheritance).

SOLUTION

import java.lang.\*;

interface Gross

{

double ta=800.0;

double da=1500.0;

void gross\_sal();

}

class Employee2

{

String name;

float basic\_sal;

Employee2(String n, float b)

{

name=n;

basic\_sal=b;

}

void display()

{

System.out.println("Name of Employee : "+name);

System.out.println("Basic Salary of Employee : "+basic\_sal);

}

}

class Salary extends Employee2 implements Gross {

float hra;

Salary(String n, float b, float h){

super(n,b);

hra=h;

}

void disp()

{

display();

System.out.println("HRA of Employee : "+hra);

}

public void gross\_sal()

{

double gross\_sal=basic\_sal + ta + da + hra;

System.out.println("TA of Employee : "+ta);

System.out.println("DA of Employee : "+da);

System.out.println("Gross Salary of Employee : "+gross\_sal);

}

}

public class q18 {

public static void main(String args[])

{

Salary s = new Salary("Sachin",8000,3000);

s.disp();

s.gross\_sal();

}

}

Output :

Name of Employee : Sachin

Basic Salary of Employee : 8000.0

HRA of Employee : 3000.0

TA of Employee : 800.0

DA of Employee : 1500.0

Gross Salary of Employee : 13300.0

Q19. Implement the following diagram.

SOLUTION

interface Exam\_{

public void percentCall();

}

class Student\_{

protected String name;

protected int roll;

protected int marks1;

protected int marks2;

public Student\_(String name, int roll, int marks1, int marks2) {

super();

this.name = name;

this.roll = roll;

this.marks1 = marks1;

this.marks2 = marks2;

}

public String show() {

return "Student [name=" + name + ", roll=" + roll + ", marks1=" + marks1 + ", marks2=" + marks2 + "]";

}

}

class Result extends Student\_ implements Exam\_{

int per;

public Result(String name, int roll, int marks1, int marks2) {

super(name, roll, marks1, marks2);

}

public void percentCall() {

per=(marks1+marks2)/2;

System.out.println("Percentage :"+per);

}

public String display() {

return "Result [percentage=" + per + ", name=" + name + ", roll=" + roll + ", marks1=" + marks1 + ", marks2=" + marks2 + "]";

}

}

public class q19 {

public static void main(String args[]) {

Result obj=new Result("Krishnendu",35,85,90);

obj.percentCall();

System.out.println(obj.show());

System.out.println(obj.display());

}

}

Output :

Percentage :87

Student [name=Krishnendu, roll=35, marks1=85, marks2=90]

Result [percentage=87, name=Krishnendu, roll=35, marks1=85, marks2=90]