**ASSIGNMENT 3**

**SET A**

**Q.1 Define a “Point” class having members – x,y (coordinates). Define default constructor and parameterized constructors. Define subclass “ColorPoint” with member as color. Write display method to display the details of Point.**

class Point

{

int x, y;

public Point(int x, int y)

{

this.x = x;

this.y = y;

}

public void display()

{

System.out.println("Point coordinates: (" + x + ", " + y + ")");

}

}

class ColorPoint extends Point

{

String color;

public ColorPoint(int x, int y, String color)

{

super(x, y);

this.color = color;

}

public void display()

{

System.out.println("Color: " + color);

super.display();

}

}

public class Main

{

public static void main(String[] args)

{

Point p1 = new Point(1, 2);

ColorPoint c1 = new ColorPoint(3, 4, "Green");

ColorPoint c2 = new ColorPoint(5, 6, "Blue");

p1.display();

c1.display();

c2.display();

}

}

Output:

Point coordinates: (1, 2)

Color: Green

Point coordinates: (3, 4)

Color: Blue

Point coordinates: (5, 6)

**Q.2 Create an abstract class Shape with methods calc\_area() & calc\_volume(). Derive two classes Sphere(radius) & Cone(radius, height) from it. Calculate area and volume of both. (Use Method Overriding).**

abstract class Shape

{

abstract double calc\_area();

abstract double calc\_volume();

}

class Sphere extends Shape

{

private double radius;

Sphere(double radius)

{

this.radius = radius;

}

double calc\_area()

{

return 4 \* Math.PI \* Math.pow(radius, 2);

}

double calc\_volume()

{

return (4.0 / 3) \* Math.PI \* Math.pow(radius, 3);

}

}

class Cone extends Shape

{

private double radius;

private double height;

Cone(double radius, double height)

{

this.radius = radius;

this.height = height;

}

double calc\_area()

{

double slantHeight = Math.sqrt(Math.pow(radius, 2) + Math.pow(height, 2));

return Math.PI \* radius \* (radius + slantHeight);

}

double calc\_volume()

{

return (1.0 / 3) \* Math.PI \* Math.pow(radius, 2) \* height;

}

}

public class Main

{

public static void main(String[] args)

{

Sphere sphere = new Sphere(5);

System.out.println("Sphere Area: " + sphere.calc\_area());

System.out.println("Sphere Volume: " + sphere.calc\_volume());

Cone cone = new Cone(3, 7);

System.out.println("Cone Area: " + cone.calc\_area());

System.out.println("Cone Volume: " + cone.calc\_volume());

}

}

Output:

Sphere Area: 314.1592653589793

Sphere Volume: 523.5987755982989

Cone Area: 100.05130440467447

Cone Volume: 65.97344572538566

Q.3 Define a class Employee having private members-id, name, department, salary. Define default & parameterized constructors. Create a subclass called Manager with private member bonus. Define methods accept & display in both the classes. Create n objects of the manager class & display the details of the manager having the maximum total salary (salary+bonus).

import java.util.\*;

class Emp

{

int id;

String name;

int salary;

Emp()

{

id = 0;

name = "";

salary = 0;

}

void acceptE()

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter ID: ");

id = sc.nextInt();

System.out.print("Enter Name: ");

name = sc.next();

System.out.print("Enter Salary: ");

salary = sc.nextInt();

}

void displayE()

{

System.out.println("ID: " + id);

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

System.out.println("Salary: " + salary);

}

double sal()

{

return salary;

}

}

class Manager extends Emp {

private int bonus;

void acceptM()

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter Bonus: ");

bonus = sc.nextInt();

}

void displayM()

{

System.out.println("Bonus: " + bonus);

}

double totalSalary()

{

return sal() + bonus;

}

static int max(Manager[] m, int n)

{

double tmax = 0;

int ts = 0;

for (int i = 0; i < n; i++)

{

if (tmax < (m[i].totalSalary()))

{

tmax = m[i].totalSalary();

ts = i;

}

}

System.out.println("\nMax Total Salary: " + tmax);

return ts;

}

}

public class Main

{

public static void main(String... args)

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter how many Records you want to enter:");

int n = sc.nextInt();

Manager[] m = new Manager[n];

for (int i = 0; i < n; i++)

{

m[i] = new Manager();

m[i].acceptE();

m[i].acceptM();

}

System.out.println("\n---------- Details Of Worker Having Maximum Total Salary --------");

int ts = Manager.max(m, n);

m[ts].displayE();

m[ts].displayM();

}

}

Output:

Enter how many Records you want to enter:2

Enter ID: 1

Enter Name: Ashish

Enter Salary: 40000

Enter Bonus: 5000

Enter ID: 2

Enter Name: Mohan

Enter Salary: 35000

Enter Bonus: 5000

---------- Details Of Worker Having Maximum Total Salary --------

Max Total Salary: 45000.0

ID: 1

Name: Ashish

Salary: 40000

Bonus: 5000

=== Code Execution Successful ===

**Q. 4 Write a Java program to create a vehicle class hierarchy. The base class should be Vehicle, with subclasses Truck, Car and Motorcycle. Each subclass should have properties such as make, model, year, and fuel type. Implement methods for calculating fuel efficiency, distance traveled, and maximum speed.**

class Vehicle

{

protected String make;

protected String model;

protected int year;

protected String fuelType;

public Vehicle(String make, String model, int year, String fuelType)

{

this.make = make;

this.model = model;

this.year = year;

this.fuelType = fuelType;

}

public double calculateFuelEfficiency()

{

return 0.0;

}

public double calculateDistanceTraveled(double fuelAmount)

{

return calculateFuelEfficiency() \* fuelAmount;

}

public int getMaxSpeed()

{

return 0;

}

public void displayDetails()

{

System.out.println("Make: " + make);

System.out.println("Model: " + model);

System.out.println("Year: " + year);

System.out.println("Fuel Type: " + fuelType);

}

}

class Truck extends Vehicle

{

private double loadCapacity;

public Truck(String make, String model, int year, String fuelType, double loadCapacity)

{

super(make, model, year, fuelType);

this.loadCapacity = loadCapacity;

}

public double calculateFuelEfficiency()

{

return 8.0;

}

public int getMaxSpeed()

{

return 90;

}

public void displayDetails()

{

super.displayDetails();

System.out.println("Load Capacity: " + loadCapacity + " tons");

}

}

class Car extends Vehicle

{

private int numDoors;

public Car(String make, String model, int year, String fuelType, int numDoors)

{

super(make, model, year, fuelType);

this.numDoors = numDoors;

}

public double calculateFuelEfficiency() {

return 30.0;

}

public int getMaxSpeed()

{

return 150;

}

public void displayDetails()

{

super.displayDetails();

System.out.println("Number of Doors: " + numDoors);

}

}

class Motorcycle extends Vehicle

{

private boolean hasSidecar;

public Motorcycle(String make, String model, int year, String fuelType, boolean hasSidecar) {

super(make, model, year, fuelType);

this.hasSidecar = hasSidecar;

}

public double calculateFuelEfficiency()

{

return 60.0;

}

public int getMaxSpeed()

{

return 180;

}

public void displayDetails()

{

super.displayDetails();

System.out.println("Has Sidecar: " + (hasSidecar ? "Yes" : "No"));

}

}

public class Main

{

public static void main(String[] args)

{

Truck truck = new Truck("Ford", "F-150", 2020, "Gasoline", 3.5);

Car car = new Car("Toyota", "Corolla", 2022, "Gasoline", 4);

Motorcycle motorcycle = new Motorcycle("Harley-Davidson", "Sportster", 2023, "Gasoline", false);

System.out.println("Truck Details:");

truck.displayDetails();

System.out.println("Fuel Efficiency: " + truck.calculateFuelEfficiency() + " MPG");

System.out.println("Max Speed: " + truck.getMaxSpeed() + " mph");

System.out.println("Distance Traveled with 10 gallons: " + truck.calculateDistanceTraveled(10) + " miles");

System.out.println();

System.out.println("Car Details:");

car.displayDetails();

System.out.println("Fuel Efficiency: " + car.calculateFuelEfficiency() + " MPG");

System.out.println("Max Speed: " + car.getMaxSpeed() + " mph");

System.out.println("Distance Traveled with 10 gallons: " + car.calculateDistanceTraveled(10) + " miles");

System.out.println();

System.out.println("Motorcycle Details:");

motorcycle.displayDetails();

System.out.println("Fuel Efficiency: " + motorcycle.calculateFuelEfficiency() + " MPG");

System.out.println("Max Speed: " + motorcycle.getMaxSpeed() + " mph");

System.out.println("Distance Traveled with 10 gallons: " + motorcycle.calculateDistanceTraveled(10) + " miles");

}

}

Output:

Truck Details:

Make: Ford

Model: F-150

Year: 2020

Fuel Type: Gasoline

Load Capacity: 3.5 tons

Fuel Efficiency: 8.0 MPG

Max Speed: 90 mph

Distance Traveled with 10 gallons: 80.0 miles

Car Details:

Make: Toyota

Model: Corolla

Year: 2022

Fuel Type: Gasoline

Number of Doors: 4

Fuel Efficiency: 30.0 MPG

Max Speed: 150 mph

Distance Traveled with 10 gallons: 300.0 miles

Motorcycle Details:

Make: Harley-Davidson

Model: Sportster

Year: 2023

Fuel Type: Gasoline

Has Sidecar: No

Fuel Efficiency: 60.0 MPG

Max Speed: 180 mph

Distance Traveled with 10 gallons: 600.0 miles

**SET C**

**Create an abstract class “order” having members id, description. Create a subclass “Purchase Order” having a member as customer name. Define methods to accept and display. Create 3 objects each of Purchase Order. Accept and display the details.**

abstract class order

{

public abstract void accept(int id, String name, String description);

public abstract void display();

}

class purchaseorder extends order

{

int id;

String name;

String description;

public void accept(int id, String name, String description)

{

this.id=id;

this.name=name;

this.description=description;

}

public void display()

{

System.out.println("purchaseorder details:");

System.out.println("ID:" +id);

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

System.out.println("Description:" +description);

}

}

public class Order\_details

{

public static void main(String[] args)

{

purchaseorder p1=new purchaseorder();

purchaseorder p2=new purchaseorder();

purchaseorder p3=new purchaseorder();

purchaseorder p4=new purchaseorder();

purchaseorder p5=new purchaseorder();

p1.accept(201,"Pavan","pen");

p2.accept(202,"Prachi","pencil");

p3.accept(203,"Ram","book");

p4.accept(204,"Rohan","bottle");

p5.accept(205,"Mohon","sketchbook");

p1.display();

p2.display();

p3.display();

p4.display();

p5.display();

}

}

Output:

purchaseorder details:

ID:201

Name:Pavan

Description:pen

purchaseorder details:

ID:202

Name:Prachi

Description:pencil

purchaseorder details:

ID:203

Name:Ram

Description:book

purchaseorder details:

ID:204

Name:Rohan

Description:bottle

purchaseorder details:

ID:205

Name:Mohan

Description:sketchbook

**Q.2 Write a Java program to create a class called Shape with a method called getArea(). Create a subclass called Rectangle that overrides the getArea() method to calculate the area of a rectangle.**

class Shape

{

public double getArea()

{

return 0.0;

}

}

class Rectangle extends Shape

{

private double length;

private double width;

public Rectangle(double length, double width)

{

this.length = length;

this.width = width;

}

public double getArea()

{

return length \* width;

}

public void displayDimensions()

{

System.out.println("Length: " + length);

System.out.println("Width: " + width);

}

}

public class ShapeTest

{

public static void main(String[] args)

{

Rectangle rectangle = new Rectangle(5.0, 3.0);

rectangle.displayDimensions();

System.out.println("Area of the Rectangle: " + rectangle.getArea() + " square units");

}

}

OUTPUT:

Length: 5.0

Width: 3.0

Area of the Rectangle: 15.0 square units

**Q.3 Write a Java program to create a class called Employee with methods called work() and getSalary(). Create a subclass called HRManager that overrides the work() method and adds a new method called addEmployee().**

class Employee

{

private String name;

private double salary;

public Employee(String name, double salary)

{

this.name = name;

this.salary = salary;

}

public void work()

{

System.out.println(name + " is doing general work.");

}

public double getSalary()

{

return salary;

}

public void displayDetails()

{

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

System.out.println("Salary: " + salary);

}

public String getName()

{

return name;

}

}

class HRManager extends Employee

{

public HRManager(String name, double salary)

{

super(name, salary);

}

public void work()

{

System.out.println(getName() + " is managing HR tasks.");

}

public void addEmployee(String newEmployeeName)

{

System.out.println(getName() + " is adding new employee: " + newEmployeeName);

}

}

public class Employee\_Test

{

public static void main(String[] args)

{

Employee employee = new Employee("Raj", 50000);

employee.displayDetails();

employee.work();

System.out.println();

HRManager hrManager = new HRManager(" Shrusti", 75000);

hrManager.displayDetails();

hrManager.work();

hrManager.addEmployee("Bob Brown");

}

}

Output:

Employee Name: Raj

Salary: 50000.0

Raj is doing general work.

Employee Name: Shrusti

Salary: 75000.0

Shrusti is managing HR tasks.

Shrusti is adding new employee: Bob Brown

**Q.4 Define an abstract class Staff with members name & address. Define two sub classes FullTimeStaff(Department,Salary) and PartTimeStaff (numberOfHours, ratePerHour). Define appropriate constructors. Create n objects which could be of either FullTimeStaff or PartTimeStaff class by asking the user’s choice. Display details of FulltimeStaff and PartTimeStaff.**

import java.io.\*;

abstract class Staff {

String name, address;

public abstract void accept() throws IOException;

public abstract void display();

}

class FullTimeStaff extends Staff

{

String department;

double salary;

public void accept() throws IOException

{

System.out.println("Enter the Name, Address, Department, and Salary:");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

name = br.readLine();

address = br.readLine();

department = br.readLine();

salary = Double.parseDouble(br.readLine());

}

public void display()

{

System.out.println();

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

System.out.println("Address: " + address);

System.out.println("Department: " + department);

System.out.println("Salary: " + salary);

System.out.println("-----------------------------------");

}

}

class PartTimeStaff extends Staff

{

int hours, rate;

public void accept() throws IOException

{

System.out.println("Enter the Name, Address, No of Working Hours, and Rate per Hour:");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

name = br.readLine();

address = br.readLine();

hours = Integer.parseInt(br.readLine());

rate = Integer.parseInt(br.readLine());

}

public void display()

{

System.out.println();

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

System.out.println("Address: " + address);

System.out.println("No of Working Hours: " + hours);

System.out.println("Rate per Hour: " + rate);

System.out.println("-----------------------------------");

}

}

public class Main

{

public static void main(String[] args) throws IOException

{

int i;

System.out.println("Select Any One: ");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("1. Full Time Staff");

System.out.println("2. Part Time Staff");

int ch = Integer.parseInt(br.readLine());

switch (ch)

{

case 1:

System.out.println("Enter the number of Full Time Staff: ");

int n = Integer.parseInt(br.readLine());

FullTimeStaff[] fullTimeStaff = new FullTimeStaff[n];

for (i = 0; i < n; i++)

{

fullTimeStaff[i] = new FullTimeStaff();

fullTimeStaff[i].accept();

}

for (i = 0; i < n; i++)

{

fullTimeStaff[i].display();

}

break;

case 2:

System.out.println("Enter the number of Part Time Staff: ");

int m = Integer.parseInt(br.readLine());

PartTimeStaff[] partTimeStaff = new PartTimeStaff[m];

for (i = 0; i < m; i++)

{

partTimeStaff[i] = new PartTimeStaff();

partTimeStaff[i].accept();

}

for (i = 0; i < m; i++)

{

partTimeStaff[i].display();

}

break;

default:

System.out.println("Invalid choice. Please select 1 or 2.");

}

}

}

Output:

Select Any One:

1. Full Time Staff

2. Part Time Staff

1

Enter the number of Full Time Staff:

2

Enter the Name, Address, Department, and Salary:

Avinash

Pune

201

50000

Enter the Name, Address, Department, and Salary:

Rohan

Alandi

202

55000

Name: Avinash

Address: Pune

Department: 201

Salary: 50000.0

-----------------------------------

Name: Rohan

Address: Alandi

Department: 202

Salary: 55000.0

-----------------------------------

=== Code Execution Successful ===