**JAVA PROGRAMMING**

**ASSIGNMENT-4**

**1.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 {

public void work() {

System.out.println("Employee is working.");

}

public double getSalary() {

return 50000.0;

}

}

class HRManager extends Employee {

@Override

public void work() {

System.out.println("HR Manager is managing employee relations and recruiting.");

}

public void addEmployee(String employeeName) {

System.out.println("Adding new employee: " + employeeName);

}

}

public class Main {

public static void main(String[] args) {

HRManager hrManager = new HRManager();

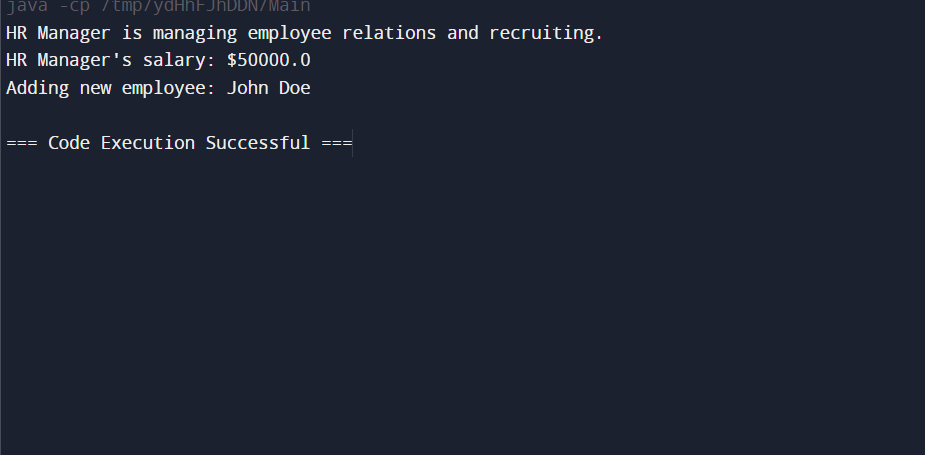
hrManager.work();

System.out.println("HR Manager's salary: $" + hrManager.getSalary());

hrManager.addEmployee("John Doe");

}

}



**2.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.**

// Base class Vehicle

abstract 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 abstract double calculateFuelEfficiency();

public abstract double calculateDistanceTraveled(double fuel);

public abstract double getMaximumSpeed();

}

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;

}

@Override

public double calculateFuelEfficiency() {

return 15.0 - (loadCapacity \* 0.5);

}

@Override

public double calculateDistanceTraveled(double fuel) {

return calculateFuelEfficiency() \* fuel;

}

@Override

public double getMaximumSpeed() {

return 80.0;

}

}

class Car extends Vehicle {

private boolean isElectric;

public Car(String make, String model, int year, String fuelType, boolean isElectric) {

super(make, model, year, fuelType);

this.isElectric = isElectric;

}

@Override

public double calculateFuelEfficiency() {

return isElectric ? 250.0 : 30.0;

}

@Override

public double calculateDistanceTraveled(double fuel) {

return calculateFuelEfficiency() \* fuel;

}

@Override

public double getMaximumSpeed() {

return 120.0;

}

}

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;

}

@Override

public double calculateFuelEfficiency() {

return 50.0 - (hasSidecar ? 5.0 : 0.0);

}

@Override

public double calculateDistanceTraveled(double fuel) {

return calculateFuelEfficiency() \* fuel;

}

@Override

public double getMaximumSpeed() {

return 160.0;

}

}

public class Main {

public static void main(String[] args) {

// Create instances of Truck, Car, and Motorcycle

Truck truck = new Truck("Ford", "F-150", 2023, "Diesel", 3.0);

Car car = new Car("Tesla", "Model S", 2023, "Electric", true);

Motorcycle motorcycle = new Motorcycle("Harley-Davidson", "Street 750", 2023, "Gasoline", false);

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

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

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

System.out.println("Maximum Speed: " + truck.getMaximumSpeed() + " mph");

System.out.println("\nCar:");

System.out.println("Fuel Efficiency: " + car.calculateFuelEfficiency() + " miles per charge");

System.out.println("Distance Traveled with 100 kWh: " + car.calculateDistanceTraveled(100) + " miles");

System.out.println("Maximum Speed: " + car.getMaximumSpeed() + " mph");

// Display information for Motorcycle

System.out.println("\nMotorcycle:");

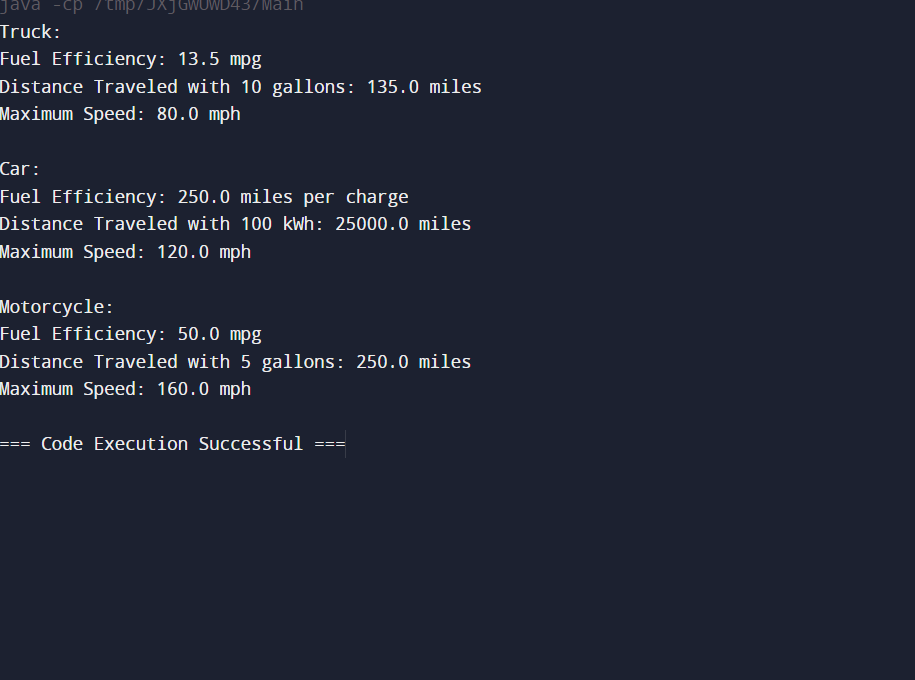
System.out.println("Fuel Efficiency: " + motorcycle.calculateFuelEfficiency() + " mpg");

System.out.println("Distance Traveled with 5 gallons: " + motorcycle.calculateDistanceTraveled(5) + " miles");

System.out.println("Maximum Speed: " + motorcycle.getMaximumSpeed() + " mph");

}

}



**3.WRITE A JAVA PROGRAM THAT CREATES A CLASS HIERARCHY FOR EMPLOYEES OF A COMPANY. THE BASE CLASS SHOULD BE EMPLOYEE, WITH SUBCLASSES MANAGER, DEVELOPER, AND PROGRAMMER. EACH SUBCLASS SHOULD HAVE PROPERTIES SUCH AS NAME, ADDRESS, SALARY, AND JOB TITLE. IMPLEMENT METHODS FOR CALCULATING BONUSES, GENERATING PERFORMANCE REPORTS, AND MANAGING PROJECTS.**

// Base class Employee

abstract class Employee {

protected String name;

protected String address;

protected double salary;

protected String jobTitle;

public Employee(String name, String address, double salary, String jobTitle) {

this.name = name;

this.address = address;

this.salary = salary;

this.jobTitle = jobTitle;

}

public abstract double calculateBonus();

public abstract String generatePerformanceReport();

public abstract void manageProjects();

}

class Manager extends Employee {

public Manager(String name, String address, double salary) {

super(name, address, salary, "Manager");

}

@Override

public double calculateBonus() {

return salary \* 0.15;

}

@Override

public String generatePerformanceReport() {

return "Manager " + name + " has achieved the company goals for this quarter.";

}

@Override

public void manageProjects() {

System.out.println(name + " is managing the company's projects.");

}

}

class Developer extends Employee {

public Developer(String name, String address, double salary) {

super(name, address, salary, "Developer");

}

@Override

public double calculateBonus() {

return salary \* 0.10;

}

@Override

public String generatePerformanceReport() {

return "Developer " + name + " has successfully completed the assigned tasks.";

}

@Override

public void manageProjects() {

System.out.println(name + " is working on software development projects.");

}

}

class Programmer extends Employee {

public Programmer(String name, String address, double salary) {

super(name, address, salary, "Programmer");

}

@Override

public double calculateBonus() {

return salary \* 0.05;

}

@Override

public String generatePerformanceReport() {

return "Programmer " + name + " has delivered high-quality code.";

}

@Override

public void manageProjects() {

System.out.println(name + " is contributing to programming tasks.");

}

}

public class Main {

public static void main(String[] args) {

Manager manager = new Manager("Alice Johnson", "123 Elm St", 90000);

Developer developer = new Developer("Bob Smith", "456 Oak St", 80000);

Programmer programmer = new Programmer("Charlie Brown", "789 Pine St", 70000);

System.out.println("Manager:");

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

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

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

System.out.println("Bonus: $" + manager.calculateBonus());

System.out.println("Performance Report: " + manager.generatePerformanceReport());

manager.manageProjects();

System.out.println("\nDeveloper:");

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

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

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

System.out.println("Bonus: $" + developer.calculateBonus());

System.out.println("Performance Report: " + developer.generatePerformanceReport());

developer.manageProjects();

System.out.println("\nProgrammer:");

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

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

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

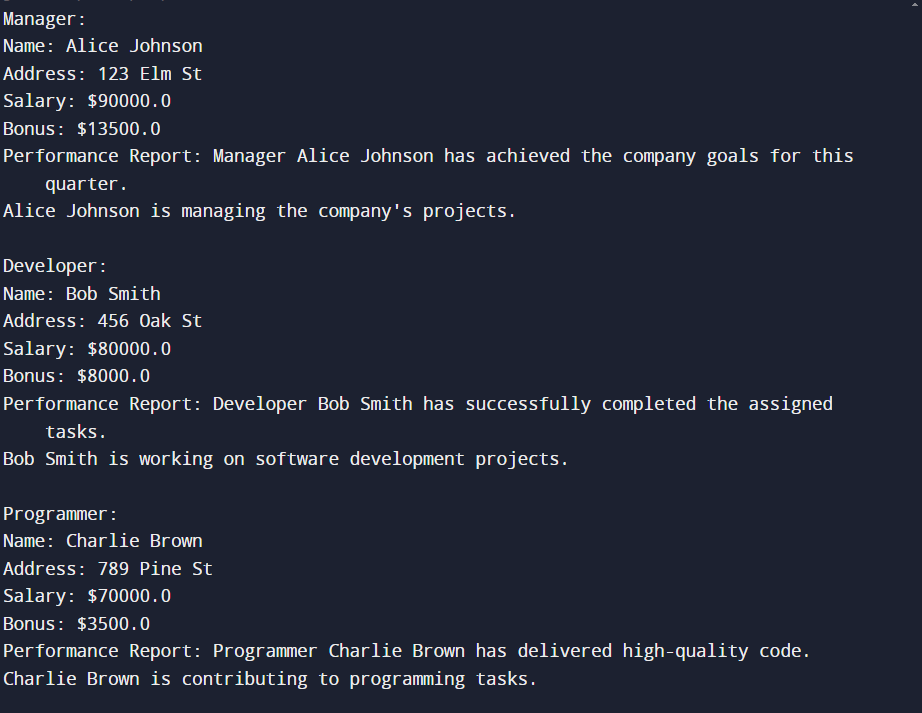
System.out.println("Bonus: $" + programmer.calculateBonus());

System.out.println("Performance Report: " + programmer.generatePerformanceReport());

programmer.manageProjects();

}

}



**4.A JAVA PROGRAM TO CREATE AN ABSTRACT CLASS SHAPE WITH ABSTRACT METHODS CALCULATEAREA() AND CALCULATEPERIMETER(). CREATE SUBCLASSES CIRCLE AND TRIANGLE THAT EXTEND THE SHAPE CLASS AND IMPLEMENT THE RESPECTIVE METHODS TO CALCULATE THE AREA AND PERIMETER OF EACH SHAPE**

abstract class Shape {

public abstract double calculateArea();

public abstract double calculatePerimeter();

}

class Circle extends Shape {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

@Override

public double calculateArea() {

return Math.PI \* radius \* radius;

}

@Override

public double calculatePerimeter() {

return 2 \* Math.PI \* radius;

}

}

class Triangle extends Shape {

private double sideA;

private double sideB;

private double sideC;

public Triangle(double sideA, double sideB, double sideC) {

this.sideA = sideA;

this.sideB = sideB;

this.sideC = sideC;

}

@Override

public double calculateArea() {

double s = calculatePerimeter() / 2;

return Math.sqrt(s \* (s - sideA) \* (s - sideB) \* (s - sideC));

}

@Override

public double calculatePerimeter() {

return sideA + sideB + sideC;

}

}

public class Main {

public static void main(String[] args) {

Circle circle = new Circle(5.0);

System.out.println("Circle:");

System.out.println("Area: " + circle.calculateArea());

System.out.println("Perimeter: " + circle.calculatePerimeter());

Triangle triangle = new Triangle(3.0, 4.0, 5.0);

System.out.println("\nTriangle:");

System.out.println("Area: " + triangle.calculateArea());

System.out.println("Perimeter: " + triangle.calculatePerimeter());

}

}

