**ASSIGNMENT—26.07.2024**

**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().**

**// Employee class**

**class Employee {**

**public void work() {**

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

**}**

**public double getSalary() {**

**return 50000.0; // Sample salary for demonstration**

**}**

**}**

**// HRManager subclass**

**class HRManager extends Employee {**

**@Override**

**public void work() {**

**System.out.println("HR Manager is managing human resources.");**

**}**

**public void addEmployee() {**

**System.out.println("HR Manager is adding a new employee.");**

**}**

**}**

**// Main class for testing**

**public class Main {**

**public static void main(String[] args) {**

**Employee emp = new Employee();**

**emp.work();**

**System.out.println("Employee Salary: $" + emp.getSalary());**

**HRManager hr = new HRManager();**

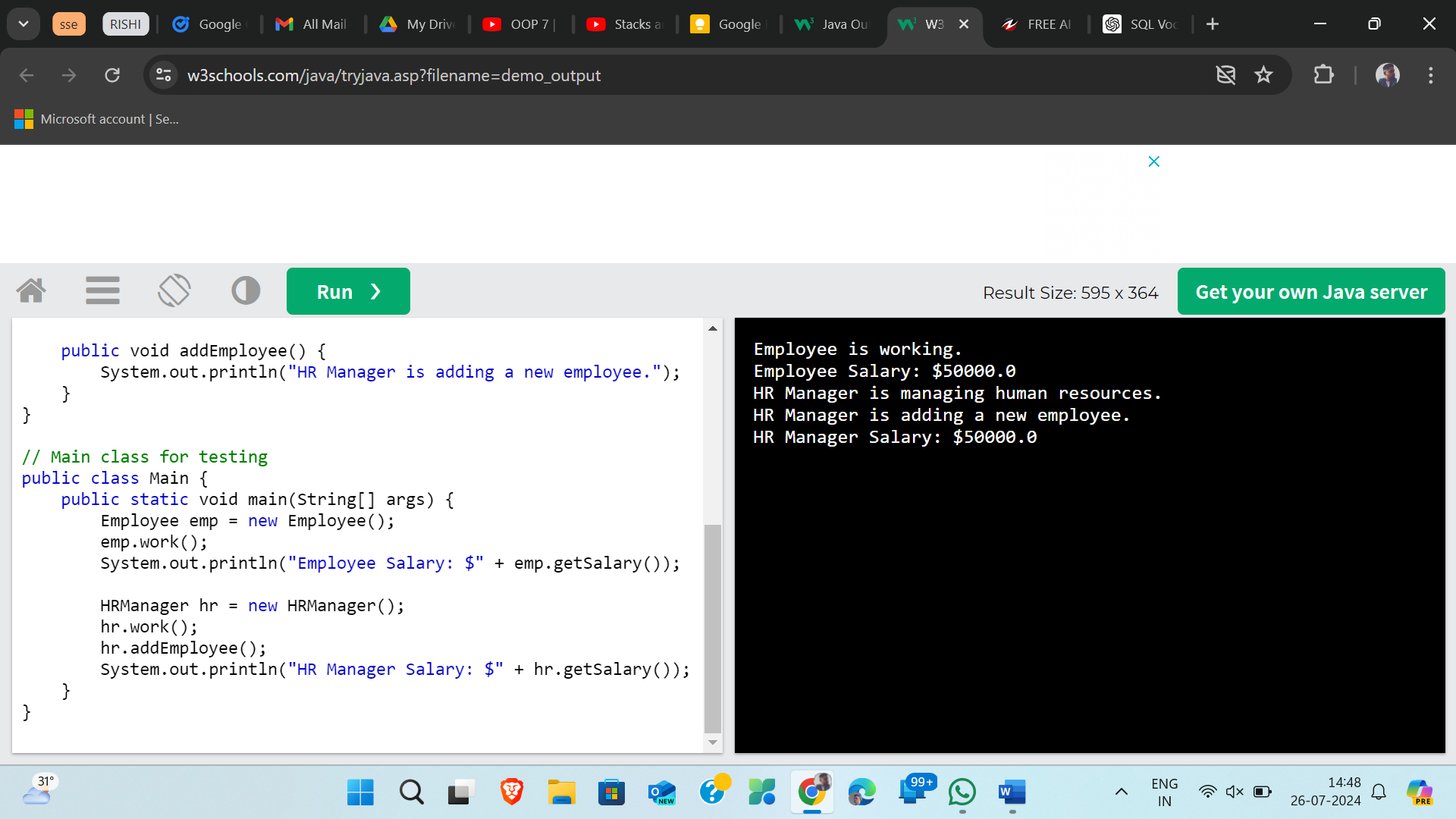
**hr.work();**

**hr.addEmployee();**

**System.out.println("HR Manager Salary: $" + hr.getSalary());**

**}**

**}**



**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.**

**class Vehicle {**

**String make;**

**String model;**

**int year;**

**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 void calculateFuelEfficiency() {**

**// Add logic to calculate fuel efficiency**

**}**

**public void calculateDistanceTraveled() {**

**// Add logic to calculate distance traveled**

**}**

**public void calculateMaxSpeed() {**

**// Add logic to calculate maximum speed**

**}**

**}**

**class Truck extends Vehicle {**

**public Truck(String make, String model, int year, String fuelType) {**

**super(make, model, year, fuelType);**

**}**

**// Additional Truck-specific properties and methods**

**}**

**class Car extends Vehicle {**

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

**super(make, model, year, fuelType);**

**}**

**// Additional Car-specific properties and methods**

**}**

**class Motorcycle extends Vehicle {**

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

**super(make, model, year, fuelType);**

**}**

**// Additional Motorcycle-specific properties and methods**

**}**

**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.**

**class Employee {**

**String name;**

**String address;**

**double salary;**

**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 double calculateBonus() {**

**// Bonus calculation logic**

**return 0.0;**

**}**

**public void generatePerformanceReport() {**

**// Performance report generation logic**

**}**

**public void manageProject() {**

**// Project management logic**

**}**

**}**

**class Manager extends Employee {**

**// Additional properties and methods specific to Manager**

**}**

**class Developer extends Employee {**

**// Additional properties and methods specific to Developer**

**}**

**class Programmer extends Employee {**

**// Additional properties and methods specific to Programmer**

**}**

**4.Write 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**

**abstract class Shape {**

**public abstract double calculateArea();**

**public abstract double calculatePerimeter();**

**}**

**// Subclass Circle**

**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;**

**}**

**}**

**// Subclass Triangle**

**class Triangle extends Shape {**

**private double side1, side2, side3;**

**public Triangle(double side1, double side2, double side3) {**

**this.side1 = side1;**

**this.side2 = side2;**

**this.side3 = side3;**

**}**

**@Override**

**public double calculateArea() {**

**double s = (side1 + side2 + side3) / 2;**

**return Math.sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));**

**}**

**@Override**

**public double calculatePerimeter() {**

**return side1 + side2 + side3;**

**}**

**}**