**1. Create a Java class named Calculator with two methods:  
i) multiply method that takes two integers and returns their product.**

**ii) multiply method overload that takes three doubles and returns their product.**

**Write a simple program to demonstrate the use of method overloading by calling both versions of the multiply method and printing the results.**

**CODE :**

public class Calculator {

    public int multiply(int a, int b) {

        return a \* b;

    }

    public double multiply(double a, double b, double c) {

        return a \* b \* c;

    }

    public static void main(String[] args) {

        Calculator calc = new Calculator();

        int result1 = calc.multiply(5, 10);

        System.out.println("Multiplication of integers: " + result1);

        double result2 = calc.multiply(2.5, 3.0, 4.0);

        System.out.println("Multiplication of doubles: " + result2);

    }

}

**2.  Create a class hierarchy representing different types of employees in a company. Design a base class Employee with fields for the employee's name, employee ID, and a method named calculateSalary() that returns the basic salary. Implement two subclasses: Manager and Developer.**

**Manager class should have an additional field for the bonus percentage. Developer class should have an additional field for the programming language.**

**Override the calculateSalary() method in both the Manager and Developer classes to include the bonus for managers and an extra allowance for developers. The basic salary for all employees is $50,000.**

**Write a program to create instances of managers and developers, call the calculateSalary method on each, and print the details.**

**CODE :**

class Employee {

    private String name;

    private int employeeID;

    public Employee(String name, int employeeID) {

[this.name](http://this.name/) = name;

        this.employeeID = employeeID;

    }

    public double calculateSalary() {

        return 50000;

    }

}

class Manager extends Employee {

    private double bonusPercentage;

    public Manager(String name, int employeeID, double bonusPercentage) {

        super(name, employeeID);

        this.bonusPercentage = bonusPercentage;

    }

    @Override

    public double calculateSalary() {

        return super.calculateSalary() + (super.calculateSalary() \* (bonusPercentage / 100));

    }

}

class Developer extends Employee {

    private String programmingLanguage;

    public Developer(String name, int employeeID, String programmingLanguage) {

        super(name, employeeID);

        this.programmingLanguage = programmingLanguage;

    }

    @Override

    public double calculateSalary() {

        return super.calculateSalary() + 1000; // Extra allowance for developers

    }

}

public class Main {

    public static void main(String[] args) {

        Manager manager = new Manager("John Doe", 1001, 20);

        Developer developer = new Developer("Alice Smith", 2001, "Java");

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

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

        System.out.println("Employee ID: " + manager.getEmployeeID());

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

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

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

        System.out.println("Employee ID: " + developer.getEmployeeID());

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

    }

}

**3. Implement a class hierarchy with a base class Vehicle and two derived classes Car and Motorcycle.**

**The Vehicle class should have a method named calculateSpeed() that returns the speed of the vehicle. Override it in other two classes, where the speed is calculated as the product of the vehicle's speed and the number of passengers or wheels.**

**Note:**

**a) Car class should have an additional field for the number of passengers.**

**b) Motorcycle class should have an additional field for the number of wheels.**

**Write a program to create instances of car and motorcycle, call the calculateSpeed method on each, and determine the vehicle with the highest effective speed.**

**CODE :**

class Vehicle {

    public double calculateSpeed() {

        return 0;

    }

}

class Car extends Vehicle {

    private int numPassengers;

    public Car(int numPassengers) {

        this.numPassengers = numPassengers;

    }

    @Override

    public double calculateSpeed() {

        return super.calculateSpeed() \* numPassengers;

    }

}

class Motorcycle extends Vehicle {

    private int numWheels;

    public Motorcycle(int numWheels) {

        this.numWheels = numWheels;

    }

    @Override

    public double calculateSpeed() {

        return super.calculateSpeed() \* numWheels;

    }

}

public class Main {

    public static void main(String[] args) {

        Car car = new Car(4);

        Motorcycle motorcycle = new Motorcycle(2);

        double carSpeed = car.calculateSpeed();

        double motorcycleSpeed = motorcycle.calculateSpeed();

        System.out.println("Car speed: " + carSpeed + " mph");

        System.out.println("Motorcycle speed: " + motorcycleSpeed + " mph");

        if (carSpeed > motorcycleSpeed) {

            System.out.println("The car has the highest effective speed.");

        } else if (motorcycleSpeed > carSpeed) {

            System.out.println("The motorcycle has the highest effective speed.");

        } else {

            System.out.println("Both vehicles have the same effective speed.");

        }

    }

}