Day-8

Quiz-2

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.

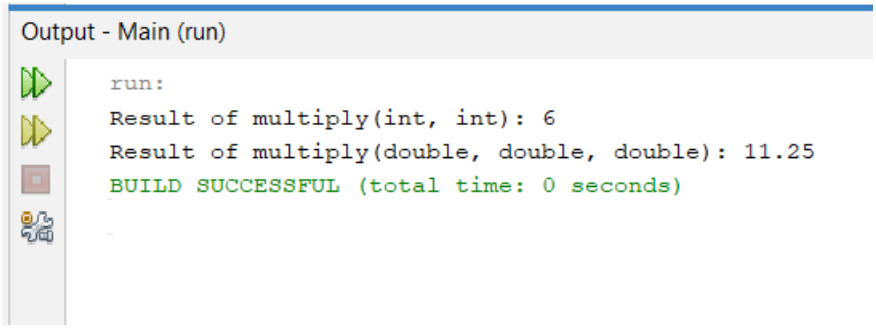
Write a simple program to demonstrate the use of method overloading by calling both versions of the multiply method and printing the results

CODE:

```
public class Calculator {
    // Method to multiply two integers
    public int multiply(int a, int b) {
        return a * b;
    }
    // Method overloading to multiply three doubles
    public double multiply(double x, double y, double z) {
        return x * y * z;
    }

    public static void main(String[] args) {
        // Create an instance of the Calculator class
        Calculator calculator = new Calculator();

        // Call the first version of the multiply method with integers
        int result1 = calculator.multiply(2, 3);
        System.out.println("Result of multiply(int, int): " + result1);

        // Call the second version of the multiply method with doubles
        double result2 = calculator.multiply(2.5, 3.0, 1.5);
        System.out.println("Result of multiply(double, double, double): " + result2);
    }
```

}

OUTPUT:



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:

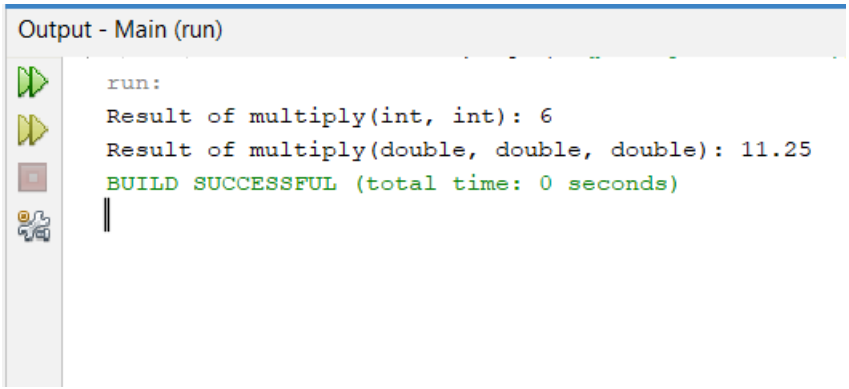
```
package main;

class Employee {

    private String name;

    private int employeeId;


    // Constructor

    public Employee(String name, int employeeId) {

        this.name = name;

        this.employeeId = employeeId;

    }
```

```
    // Method to calculate basic salary
    public double calculateSalary() {
        // Basic salary for all employees is $50,000
        return 50000.0;
    }

    // Getters
    public String getName() {
        return name;
    }

    public int getEmployeeId() {
        return employeeId;
    }
}

class Manager extends Employee {
    private double bonusPercentage;

    // Constructor
    public Manager(String name, int employeeId, double bonusPercentage) {
        super(name, employeeId);
        this.bonusPercentage = bonusPercentage;
    }

    // Override calculateSalary() method to include bonus
    @Override
    public double calculateSalary() {
        // Basic salary for all employees is $50,000
        // Adding bonus for managers
```

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

    // Getter for bonusPercentage
    public double getBonusPercentage() {
        return bonusPercentage;
    }
}

class Developer extends Employee {
    private String programmingLanguage;

    // Constructor
    public Developer(String name, int employeeId, String programmingLanguage) {
        super(name, employeeId);
        this.programmingLanguage = programmingLanguage;
    }

    // Override calculateSalary() method to include extra allowance for developers
    @Override
    public double calculateSalary() {
        // Basic salary for all employees is $50,000
        // Adding extra allowance for developers
        return super.calculateSalary() + 10000.0;
    }

    // Getter for programmingLanguage
    public String getProgrammingLanguage() {
        return programmingLanguage;
    }
```

```
}

public class Company {
    public static void main(String[] args) {
        // Create instances of Manager and Developer
        Manager manager = new Manager("John Doe", 101, 15.0);
        Developer developer = new Developer("Jane Smith", 102, "Java");

        // Print details and calculate salary for Manager
        System.out.println("Manager Details:");
        System.out.println("Name: " + manager.getName());
        System.out.println("Employee ID: " + manager.getEmployeeId());
        System.out.println("Bonus Percentage: " + manager.getBonusPercentage() + "%");
        System.out.println("Calculated Salary: $" + manager.calculateSalary());
        System.out.println();

        // Print details and calculate salary for Developer
        System.out.println("Developer Details:");
        System.out.println("Name: " + developer.getName());
        System.out.println("Employee ID: " + developer.getEmployeeId());
        System.out.println("Programming Language: " +
developer.getProgrammingLanguage());
        System.out.println("Calculated Salary: $" + developer.calculateSalary());
    }
}
```

OUTPUT:



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.

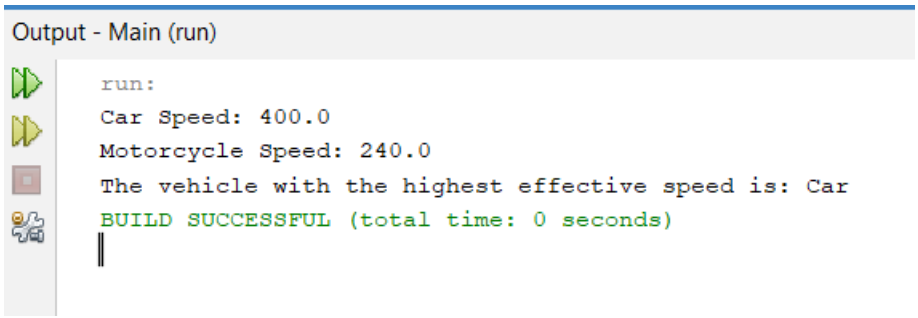
CODE:

```
package main;

class Vehicle {
    private double speed;

    public Vehicle(double speed) {
        this.speed = speed;
    }

    public double calculateSpeed() {
        return speed;
    }
```

```
}

class Car extends Vehicle {
    private int numberOfPassengers;

    public Car(double speed, int numberOfPassengers) {
        super(speed);
        this.numberOfPassengers = numberOfPassengers;
    }

    @Override
    public double calculateSpeed() {
        // Override calculateSpeed to consider the number of passengers
        return super.calculateSpeed() * numberOfPassengers;
    }
}

class Motorcycle extends Vehicle {
    private int numberOfWheels;

    public Motorcycle(double speed, int numberOfWheels) {
        super(speed);
        this.numberOfWheels = numberOfWheels;
    }

    @Override
    public double calculateSpeed() {
        // Override calculateSpeed to consider the number of wheels
        return super.calculateSpeed() * numberOfWheels;
    }
```

```
}

public class VehicleProgram {
    public static void main(String[] args) {
        // Create instances of Car and Motorcycle
        Car car = new Car(100, 4); // Speed: 100, Passengers: 4
        Motorcycle motorcycle = new Motorcycle(120, 2); // Speed: 120, Wheels: 2

        // Call calculateSpeed method on each
        double carSpeed = car.calculateSpeed();
        double motorcycleSpeed = motorcycle.calculateSpeed();

        // Determine the vehicle with the highest effective speed
        String fastestVehicle;
        if (carSpeed > motorcycleSpeed) {
            fastestVehicle = "Car";
        } else if (carSpeed < motorcycleSpeed) {
            fastestVehicle = "Motorcycle";
        } else {
            fastestVehicle = "Both vehicles have the same speed";
        }

        // Print the details
        System.out.println("Car Speed: " + carSpeed);
        System.out.println("Motorcycle Speed: " + motorcycleSpeed);
        System.out.println("The vehicle with the highest effective speed is: " + fastestVehicle);
    }
}
```

OUTPUT:


Output - Main (run)
run:
Car Speed: 400.0
Motorcycle Speed: 240.0
The vehicle with the highest effective speed is: Car
BUILD SUCCESSFUL (total time: 0 seconds)