```
1. 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 multiply method with two integers
      int resultInt = calculator.multiply(3, 4);
      System.out.println("Product of two integers: " + resultInt);

      // Call the multiply method with three doubles
      double resultDouble = calculator.multiply(2.5, 1.5, 3.0);
      System.out.println("Product of three doubles: " + resultDouble);
   }
}
2. // Base class Employee
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() {
      return 50000; // Basic salary for all employees is $50,000
   }

   // Getters for name and employee ID
   public String getName() {
      return name;
   }

   public int getEmployeeID() {
      return employeeID;
   }
}

// Manager class (subclass of Employee)
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() {
      // Calculate salary with bonus
      return super.calculateSalary() + (super.calculateSalary() * bonusPercentage / 100);
   }
}

// Developer class (subclass of Employee)
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 allowance
   @Override
   public double calculateSalary() {
      // Calculate salary with allowance
      return super.calculateSalary() + 10000; // Additional $10,000 allowance for developers
   }
}

// Main program to create instances of managers and developers, call calculateSalary, and print details
public class CompanyMain {
   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("Alice Smith", 102, "Java");
      // Call calculateSalary method and print details
      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(); // Blank line for separation

      System.out.println("Developer Details:");
      System.out.println("Name: " + developer.getName());
      System.out.println("Employee ID: " + developer.getEmployeeID());
      System.out.println("Salary: $" + developer.calculateSalary());
   }
}
```

3.

```
// Base class Vehicle
class Vehicle {
    private double speed;

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

    // Method to calculate speed (to be overridden by subclasses)
    public double calculateSpeed() {
        return speed;
    }
}

// Car class (subclass of Vehicle)
class Car extends Vehicle {
    private int numberOfPassengers;

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

    // Override calculateSpeed method to include the number of passengers
    @Override
    public double calculateSpeed() {
        // Calculate speed with the number of passengers
        return super.calculateSpeed() * numberOfPassengers;
    }
}

// Motorcycle class (subclass of Vehicle)
class Motorcycle extends Vehicle {
    private int numberOfWheels;

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

    // Override calculateSpeed method to include the number of wheels
    @Override
    public double calculateSpeed() {
        // Calculate speed with the number of wheels
        return super.calculateSpeed() * numberOfWheels;
    }
}


// Main program to create instances of Car and Motorcycle, call calculateSpeed, and determine the highest effective speed
public class VehicleMain {
    public static void main(String[] args) {
        // Create instances of Car and Motorcycle
        Car car = new Car(60.0, 4);
        Motorcycle motorcycle = new Motorcycle(80.0, 2);

        // Call calculateSpeed method and print details
        System.out.println("Car Details:");
        System.out.println("Effective Speed: " + car.calculateSpeed() + " mph");
        System.out.println(); // Blank line for separation

        System.out.println("Motorcycle Details:");
        System.out.println("Effective Speed: " + motorcycle.calculateSpeed() + " mph");
        System.out.println(); // Blank line for separation

        // Determine the vehicle with the highest effective speed
        if (car.calculateSpeed() > motorcycle.calculateSpeed()) {
            System.out.println("The car has the highest effective speed.");
        } else if (motorcycle.calculateSpeed() > car.calculateSpeed()) {
            System.out.println("The motorcycle has the highest effective speed.");
        } else {
            System.out.println("Both vehicles have the same effective speed.");
        }
    }
}
```