Day 11 Homework

1. Shape Class Abstract Class with Polymorphism and Method Overriding

```
// Abstract class Shape
abstract class Shape {
protected String shapeName;

public Shape(String shapeName) {
this.shapeName = shapeName;
}

public String getShapeName() {
return shapeName;
}

public abstract double calculateArea();
}

// Circle class extending Shape
class Circle extends Shape {
private double radius;

public Circle(double radius) {
super("Circle");
this.radius = radius;
}

@Override
public double calculateArea() {
return Math.PI * radius * radius;
}
}

// Rectangle class extending Shape
class Rectangle extends Shape {
private double length, width;

public Rectangle(double length, double width) {
super("Rectangle");
this.length = length;
this.width = width;
}

@Override
public double calculateArea() {
return length * width;
}
}
```

```
}

// Main class to test functionality
public class ShapeTest {
public static void main(String[] args) {
Shape circle = new Circle(5.0);
System.out.println(circle.getShapeName() + " Area: " +
circle.calculateArea());

Shape rectangle = new Rectangle(4.0, 6.0);
System.out.println(rectangle.getShapeName() + " Area: " +
rectangle.calculateArea());
}
}
```

2) 2. Photosynthesis and Respiration Interfaces with Multiple Inheritance

```
// Photosynthesis interface
interface Photosynthesis {
   void absorbSunlight();
}

// Respiration interface
interface Respiration {
   void releaseOxygen();
}
```

```
    }

// Plant class implementing both interfaces
class Plant implements Photosynthesis, Respiration {
    private String plantName;

    public Plant(String plantName) {
        this.plantName = plantName;
    }

    public String getPlantName() {
        return plantName;
    }

    @Override
    public void absorbSunlight() {
        System.out.println(plantName + " is absorbing sunlight for photosynthesis.");
    }

    @Override
    public void releaseOxygen() {
        System.out.println(plantName + " is releasing oxygen through respiration.");
    }
}
```

```
// Main class to test implementation
public class PlantTest {
    public static void main(String[] args) {
        Plant mangoTree = new Plant("Mango Tree");
        Plant fern = new Plant("Fern");

        System.out.println("Plant: " + mangoTree.getPlantName());
        mangoTree.absorbSunlight();
        mangoTree.releaseOxygen();

        System.out.println("\nPlant: " + fern.getPlantName());
        fern.absorbSunlight();
        fern.releaseOxygen();
    }
}
```

3. BMW Vehicle Hierarchy with Runtime Polymorphism

```
// Parent class BMW
class BMW {
    public void showDetails() {
        System.out.println("This is a BMW vehicle.");
    }
```

```java
    public void maxSpeed() {
        System.out.println("Speed varies by model.");
    }
}

// Subclasses with method overriding
class BMWSeries3 extends BMW {
    @Override
    public void showDetails() {
        System.out.println("This is BMW Series 3.");
    }

    @Override
    public void maxSpeed() {
        System.out.println("Max speed: 240 km/h.");
    }
}

class BMWSeries5 extends BMW {
    @Override
    public void showDetails() {
        System.out.println("This is BMW Series 5.");
    }
    @Override
    public void maxSpeed() {
        System.out.println("Max speed: 260 km/h.");
    }
}

class BMWSeries7 extends BMW {
    @Override
    public void showDetails() {
        System.out.println("This is BMW Series 7.");
    }

    @Override
    public void maxSpeed() {
        System.out.println("Max speed: 300 km/h.");
    }
}

// Main class to demonstrate runtime polymorphism
public class BMWTest {
    public static void main(String[] args) {
        BMW[] cars = {new BMWSeries3(), new BMWSeries5(), new BMWSeries7()};
        for (BMW car : cars) {
            car.showDetails();
            car.maxSpeed();
            System.out.println();
        }
    }
}
```