Week - 3

Laya Gollamandala

1.A base class Vehicle and derived classes Car and Bike. Implement method overriding and demonstrate polymorphism.

Program:

```
class Vehicle {
        public void start() {
        System.out.println("Vehicle is starting");
        }
        public void stop() {
        System.out.println("Vehicle is stopping");
        }
}
class Car extends Vehicle {
        @Override
        public void start() {
        System.out.println("Car is starting");
        }
        @Override
        public void stop() {
        System.out.println("Car is stopping");
        }
}
class Bike extends Vehicle {
        @Override
        public void start() {
        System.out.println("Bike is starting");
        }
        @Override
        public void stop() {
        System.out.println("Bike is stopping");
        }
}
public class Main {
        public static void main(String[] args) {
        Vehicle vehicle1 = new Car();
        Vehicle vehicle2 = new Bike();
        vehicle1.start();
        vehicle1.stop();
        vehicle2.start();
        vehicle2.stop();
```

```
        }
}
```

Output:

```
Car is starting
Car is stopping
Bike is starting
Bike is stopping
```

2.Implement a Shape class with derived classes Circle, Rectangle, and Triangle, each having a method to calculate the area.

Program:

```
abstract class Shape {
        public abstract double calculateArea();
}
class Circle extends Shape {
        private double radius;
        public Circle(double radius) {
        this.radius = radius;
        }
        @Override
        public double calculateArea() {
        return Math.PI * radius * radius;
        }
}
class Rectangle extends Shape {
        private double width;
        private double height;
        public Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
        }
        @Override
        public double calculateArea() {
        return width * height;
        }
}
class Triangle extends Shape {
        private double base;
        private double height;
        public Triangle(double base, double height) {
        this.base = base;
        this.height = height;
```

```
        }
        @Override
        public double calculateArea() {
        return 0.5 * base * height;
        }
}
public class Main {
        public static void main(String[] args) {
        Shape circle = new Circle(5);
        Shape rectangle = new Rectangle(4, 6);
        Shape triangle = new Triangle(3, 7);
        System.out.println("Circle Area: " + circle.calculateArea());
        System.out.println("Rectangle Area: " + rectangle.calculateArea());
        System.out.println("Triangle Area: " + triangle.calculateArea());
        }
}
```

Output:

```
Circle area: 78.53981633974483
Rectangle area: 24.0
Triangle area: 6.0
```

3.Develop a simple program to demonstrate the concept of abstract classes and interfaces.

Program:

```
interface Animal {
        void sound();
        void eat();
}
abstract class Mammal implements Animal {
        public void eat() {
        System.out.println("This mammal is eating.");
        }
        abstract void sleep();
}
class Dog extends Mammal {
        public void sound() {
        System.out.println("The dog barks.");
        }
        public void sleep() {
        System.out.println("The dog is sleeping.");
        }
}
```

```
class Cat extends Mammal {
        public void sound() {
        System.out.println("The cat meows.");
        }
        public void sleep() {
        System.out.println("The cat is sleeping.");
        }
}
public class Main {
        public static void main(String[] args) {
        Mammal myDog = new Dog();
        myDog.sound();
        myDog.eat();
        myDog.sleep();
        Mammal myCat = new Cat();
        myCat.sound();
        myCat.eat();
        myCat.sleep();
        }
}
```

Output:

```
The dog barks.
This mammal is eating.
The dog is sleeping.
The cat meows.
This mammal is eating.
The cat is sleeping.
```