

ASSIGNMENT-5**NAME: ARITRA SARKAR****ROLL NO: 002311001048****SEC: A2****Problem 1:**

```
public interface Shape {
    void draw();
}

class Circle implements Shape {
    public void draw() {
        System.out.println("Inside Circle::draw() method.");
    }
}

class Rectangle implements Shape {
    public void draw() {
        System.out.println("Inside Rectangle::draw() method.");
    }
}

class Square implements Shape {
    public void draw() {
        System.out.println("Inside Square::draw() method.");
    }
}

class ShapeFactory {
    public Shape getShape(String shapeType) {
        if (shapeType == null) {
            return null;
        }
        if (shapeType.equalsIgnoreCase("CIRCLE")) {
            return new Circle();
        } else if (shapeType.equalsIgnoreCase("RECTANGLE")) {
            return new Rectangle();
        } else if (shapeType.equalsIgnoreCase("SQUARE")) {
            return new Square();
        }
        return null;
    }
}

class FactoryPatternDemo {
    public static void main(String[] args) {
        ShapeFactory shapeFactory = new ShapeFactory();

        Shape shape1 = shapeFactory.getShape("CIRCLE");
        shape1.draw();
    }
}
```

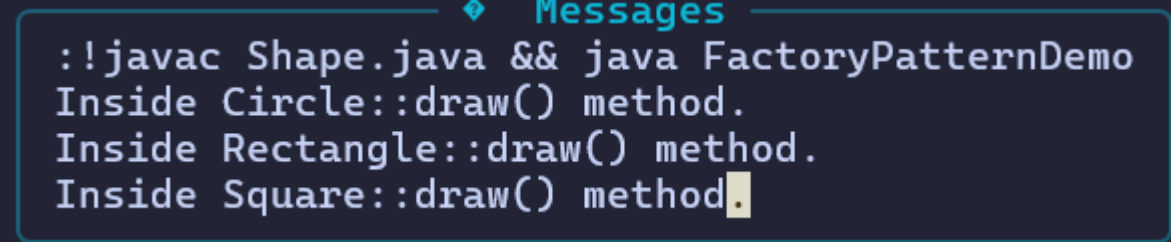
```

        Shape shape2 = shapeFactory.getShape("RECTANGLE");
        shape2.draw();

        Shape shape3 = shapeFactory.getShape("SQUARE");
        shape3.draw();
    }
}

```

Output:



```

Messages
:!javac Shape.java && java FactoryPatternDemo
Inside Circle::draw() method.
Inside Rectangle::draw() method.
Inside Square::draw() method.

```

Problem 2:

```

interface Shape {
    void draw();
}

class Circle implements Shape {
    public void draw() {
        System.out.println("Shape: Circle");
    }
}

class Rectangle implements Shape {
    public void draw() {
        System.out.println("Shape: Rectangle");
    }
}

abstract class ShapeDecorator implements Shape {
    protected Shape decoratedShape;

    public ShapeDecorator(Shape decoratedShape) {
        this.decoratedShape = decoratedShape;
    }

    public void draw() {
        decoratedShape.draw();
    }
}

class RedShapeDecorator extends ShapeDecorator {
    public RedShapeDecorator(Shape decoratedShape) {

```

```

        super(decoratedShape);
    }

    public void draw() {
        decoratedShape.draw();
        setRedBorder(decoratedShape);
    }

    private void setRedBorder(Shape decoratedShape){
        System.out.println("Border Color: Red");
    }
}

class DecoratorPatternDemo {
    public static void main(String[] args) {
        Shape circle = new Circle();
        Shape redCircle = new RedShapeDecorator(new Circle());
        Shape redRectangle = new RedShapeDecorator(new Rectangle());

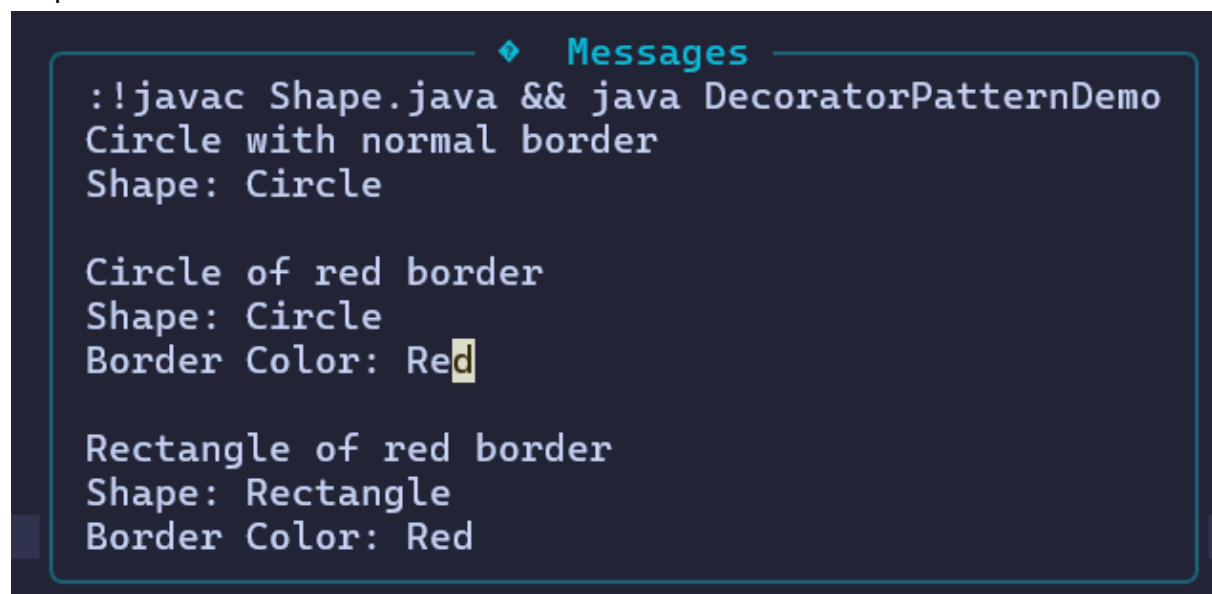
        System.out.println("Circle with normal border");
        circle.draw();

        System.out.println("\nCircle of red border");
        redCircle.draw();

        System.out.println("\nRectangle of red border");
        redRectangle.draw();
    }
}

```

Output:



```

❖ Messages
: !javac Shape.java && java DecoratorPatternDemo
Circle with normal border
Shape: Circle

Circle of red border
Shape: Circle
Border Color: Red

Rectangle of red border
Shape: Rectangle
Border Color: Red

```

Problem 3:

```
import java.util.*;
```

```
interface ChatMediator {  
    void sendMessage(String message, User user);  
    void addUser(User user);  
}
```

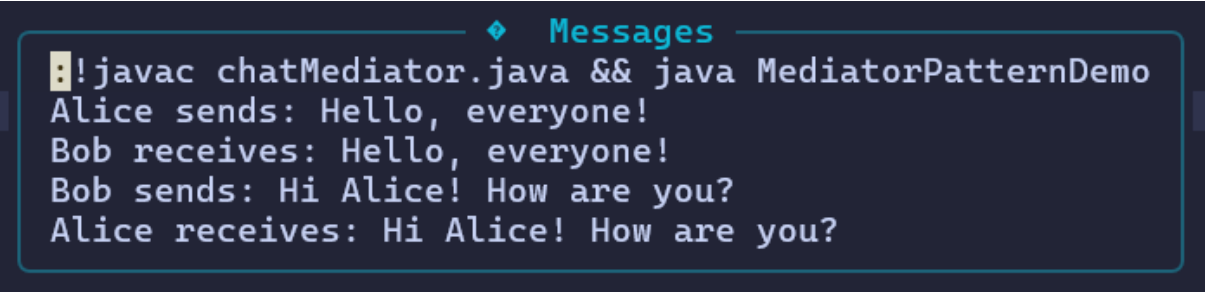
```
class ChatRoom implements ChatMediator {  
    private List<User> users;  
  
    public ChatRoom() {  
        this.users = new ArrayList<>();  
    }  
  
    public void addUser(User user) {  
        this.users.add(user);  
    }  
  
    public void sendMessage(String message, User user) {  
        for (User chatUser : this.users) {  
            if (chatUser != user) {  
                chatUser.receive(message);  
            }  
        }  
    }  
}
```

```
interface User {  
    void send(String message);  
    void receive(String message);  
}
```

```
class ChatUser implements User {  
    private String name;  
    private ChatMediator mediator;  
  
    public ChatUser(String name, ChatMediator mediator) {  
        this.name = name;  
        this.mediator = mediator;  
        this.mediator.addUser(this);  
    }  
  
    public void send(String message) {  
        System.out.println(this.name + " sends: " + message);  
        this.mediator.sendMessage(message, this);  
    }  
}
```

```
    public void receive(String message) {  
        System.out.println(this.name + " receives: " + message);  
    }  
}  
  
class MediatorPatternDemo {  
    public static void main(String[] args) {  
        ChatMediator mediator = new ChatRoom();  
  
        User user1 = new ChatUser("Alice", mediator);  
        User user2 = new ChatUser("Bob", mediator);  
  
        user1.send("Hello, everyone!");  
        user2.send("Hi Alice! How are you?");  
    }  
}
```

Output:



```
!javac chatMediator.java && java MediatorPatternDemo  
Alice sends: Hello, everyone!  
Bob receives: Hello, everyone!  
Bob sends: Hi Alice! How are you?  
Alice receives: Hi Alice! How are you?
```