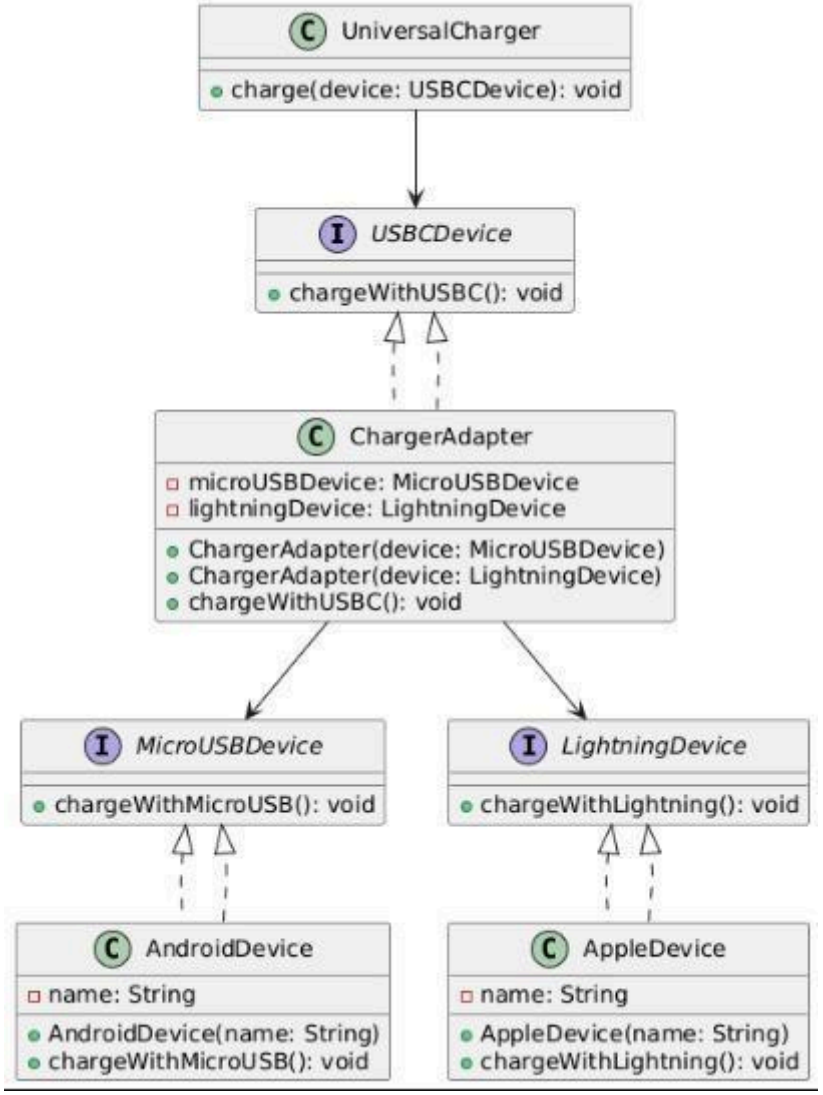
PES UNIVERSITY
ObjectOrientedAnalysisandDesign(UE22CS352B)
LAB ASSIGNMENT

Name: Rohit Yakkundi
SRN:PES2UG23CS819
Sec:G

1. Adapter Pattern :

Class Diagram :



Java Implementation :

```java
// USBCDevice.java
public interface USBCDevice {
    void chargeWithUSBC();
}

// MicroUSBDevice.java
public interface MicroUSBDevice {
    void chargeWithMicroUSB();
}

// LightningDevice.java
public interface LightningDevice {
    void chargeWithLightning();
}

// UniversalCharger.java
public class UniversalCharger {
    public void charge(USBCDevice device) {
        device.chargeWithUSBC();
    }
}

// AndroidDevice.java
public class AndroidDevice implements MicroUSBDevice {
    private String name;

    public AndroidDevice(String name) {
        this.name = name;
    }
public void chargeWithMicroUSB() {
        System.out.println("Charging " + name + " via Micro-USB using
USB-C adapter");
    }
}

// AppleDevice.java
public class AppleDevice implements LightningDevice {
    private String name;
    public AppleDevice(String name) {
        this.name = name;
    }
```

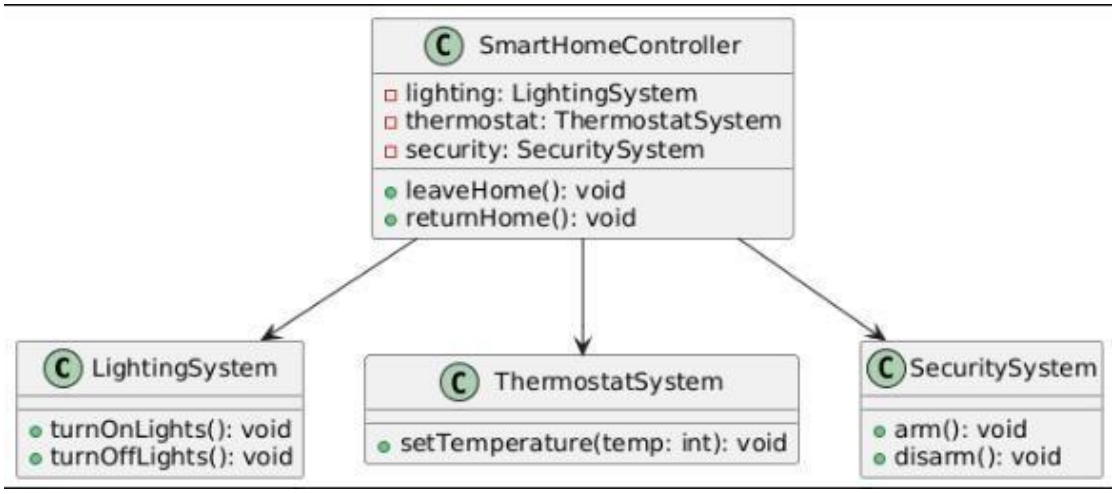
```java
// ChargerAdapter.java
public class ChargerAdapter implements USBCDevice {
    private MicroUSBDevice microUSBDevice;
private LightningDevice lightningDevice;
    public ChargerAdapter(MicroUSBDevice microUSBDevice) {
        this.microUSBDevice = microUSBDevice;
    }
    public ChargerAdapter(LightningDevice lightningDevice) {
        this.lightningDevice = lightningDevice;
    }
    public void chargeWithUSBC() {
        if (microUSBDevice != null) {
microUSBDevice.chargeWithMicroUSB();
        } else if (lightningDevice != null) {
            lightningDevice.chargeWithLightning();
        }
    }
}

// Main.java
public class Main {
    public static void main(String[] args) {
        UniversalCharger charger = new UniversalCharger();

        charger.charge(new USBCDevice() {
            public void chargeWithUSBC() {
System.out.println("Charging Laptop with USB-C");
            }
        });
        charger.charge(new ChargerAdapter(new AndroidDevice("Samsung
Galaxy")));
charger.charge(new ChargerAdapter(new AppleDevice("iPhone")));
    }
} {

}
```

Output :

```
D:\PES2UG23CS819\ooad\AdapterPattern>javac *.java

D:\PES2UG23CS819\ooad\AdapterPattern>java Main
Charging Laptop with USB-C
Charging Samsung Galaxy Via Micro-USB using USB-C adapter
Charging iPhone via Lightning using USB-C adapter

D:\PES2UG23CS819\ooad\AdapterPattern>
```

2. Facade Pattern :

Class Diagram :



Java Implementation :

```java
// LightingSystem.java
public class LightingSystem {
    public void turnOnLights() {
System.out.println("Lights turned on");
    }

    public void turnOffLights() {
        System.out.println("Lights turned off");
    }
}

// ThermostatSystem.java
public class ThermostatSystem {
    public void setTemperature(int temperature) {
        System.out.println("Temperature set to " + temperature + "°C");
    }
}

// SecuritySystem.java
public class SecuritySystem {
    public void arm() {
System.out.println("Security system armed");
    }
    public void disarm() {
        System.out.println("Security system disarmed");
    }
}
```

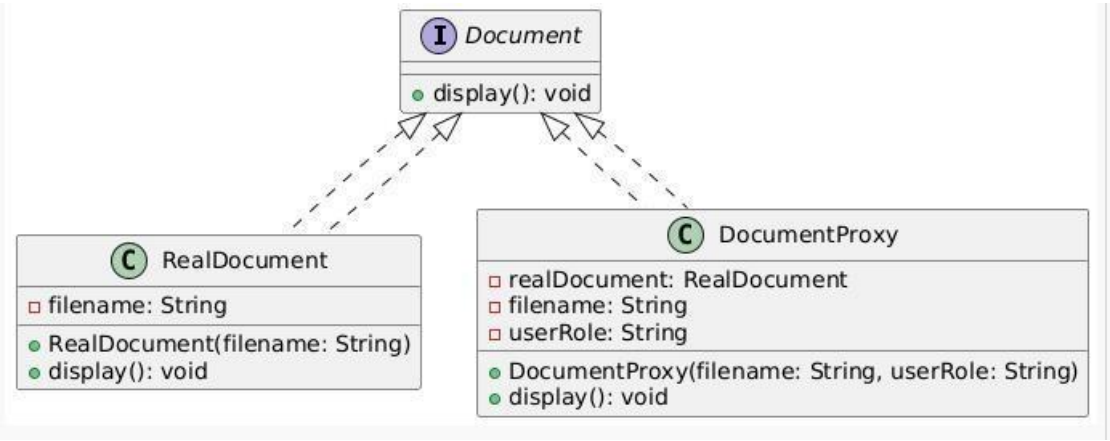
```java
// SmartHomeController.java
public class SmartHomeController {
    private LightingSystem lighting;
    private ThermostatSystem thermostat;
    private SecuritySystem security;
    public SmartHomeController() {
lighting = new LightingSystem(); thermostat = new
        ThermostatSystem(); security = new
        SecuritySystem();
    }
    public void leaveHome() {
        lighting.turnOffLights();
        thermostat.setTemperature(18);
        security.arm();
    }
    public void returnHome() {
        lighting.turnOnLights();
        thermostat.setTemperature(22);
        security.disarm();
    }
}

// Main.java
public class Main {
    public static void main(String[] args) {
        SmartHomeController home = new SmartHomeController();
        home.leaveHome();
home.returnHome();
    }
}
```

Output :

```
D:\PES2UG23CS819\ooad\FacadePattern>javac *.java

D:\PES2UG23CS819\ooad\FacadePattern>java Main
Lights turned off
Temperature set to 18°C
Security system armed
Lights turned on
Temperature set to 22°C
Security system disarmed

D:\PES2UG23CS819\ooad\FacadePattern>
```

3. Proxy Pattern :

Class Diagram :



Java Implementation :

```
// Document.java
public interface Document {
    void display();
}

// RealDocument.java
public class RealDocument implements Document {
    private String filename;
    public RealDocument(String filename) {
        this.filename = filename;
        loadFromStorage();
    }
private void loadFromStorage() {
        System.out.println("Loading " + filename + " from secure
storage...");
    }
@Override
public void display() {
System.out.println("Displaying document: " + filename);
    }
}

// DocumentProxy.java
public class DocumentProxy implements Document {
    private RealDocument realDocument;
private String filename; private
    String userRole;
    public DocumentProxy(String filename, String userRole) {
        this.filename = filename;
this.userRole = userRole;
```

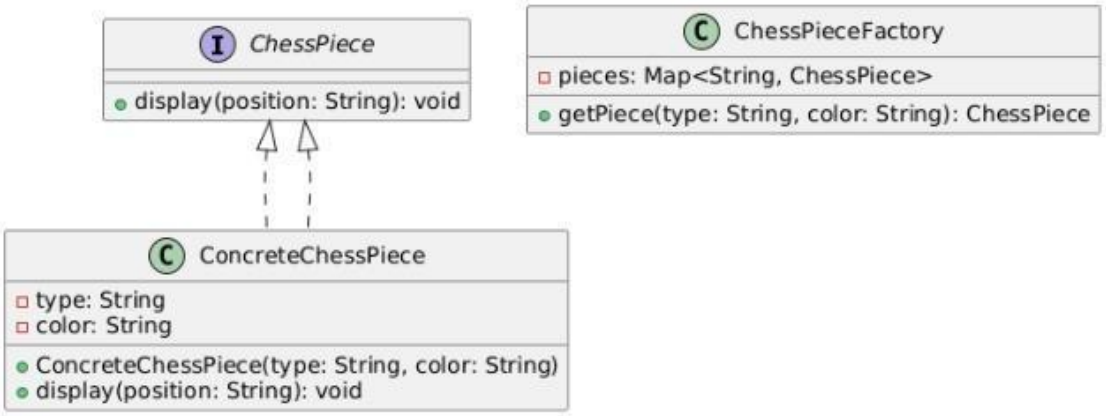
```
    }
@Override
public void display() {
        if ("PremiumUser".equalsIgnoreCase(userRole)) {
            if (realDocument == null) {
realDocument = new RealDocument(filename);
            }
realDocument.display();
} else {
System.out.println("Access denied for " + userRole + " to "
+ filename);
        }
    }
}

// Main.java
public class Main {
public static void main(String[] args) {
Document doc1 = new DocumentProxy("confidential.pdf", "Guest");
        doc1.display();
System.out.println();
        Document doc2 = new DocumentProxy("project_plan.pdf",
"PremiumUser");
doc2.display();
    }
}
```

Output :

```
D:\PES2UG23CS819\ooad\ProxyPattern>javac *.java

D:\PES2UG23CS819\ooad\ProxyPattern>java Main
Access denied for Guest to confidential.pdf

Loading project_plan.pdf from secure storage...
Displaying document: project_plan.pdf

D:\PES2UG23CS819\ooad\ProxyPattern>
```

4. Flyweight Pattern :

Class Diagram :



Java Implementation :

```
// ChessPiece.java
public interface ChessPiece {
    void display(String position);
}

// ConcreteChessPiece.java
public class ConcreteChessPiece implements ChessPiece {
    private final String type;
private final String color;
    public ConcreteChessPiece(String type, String color) {
        this.type = type;
this.color = color;
System.out.println("Creating new piece: " + type + " - " +
color);
    }
@Override
    public void display(String position) {
        System.out.println("Displaying " + type + " (" + color + ") at "
+ position);
    }
}

// ChessPieceFactory.java
import java.util.HashMap;
import java.util.Map;
public class ChessPieceFactory {
private final Map<String, ChessPiece> pieces = new HashMap<>();
    public ChessPiece getPiece(String type, String color) {
String key = type + "-" + color;
if (!pieces.containsKey(key)) {
```

```
pieces.put(key, new ConcreteChessPiece(type, color));
        }
return pieces.get(key);
    }
}

// Main.java public
class Main {
    public static void main(String[] args) {
        ChessPieceFactory factory = new ChessPieceFactory();
ChessPiece pawnWhite = factory.getPiece("Pawn", "White"); ChessPiece
        pawnBlack = factory.getPiece("Pawn", "Black"); ChessPiece
        bishopWhite = factory.getPiece("Bishop", "White");
        pawnWhite.display("A2");
pawnWhite.display("A3");
        pawnBlack.display("B7");
        bishopWhite.display("C1");
    }
}
```

Output :

```
D:\PES2UG23CS819\ooad\FlyweightPattern>javac *.java

D:\PES2UG23CS819\ooad\FlyweightPattern>java Main
Creating new piece: Pawn - White
Creating new piece: Pawn - Black
Creating new piece: Bishop - White
Displaying Pawn (White) at A2
Displaying Pawn (White) at A3
Displaying Pawn (Black) at B7
Displaying Bishop (White) at C1

D:\PES2UG23CS819\ooad\FlyweightPattern>ChessPiece.class_
```