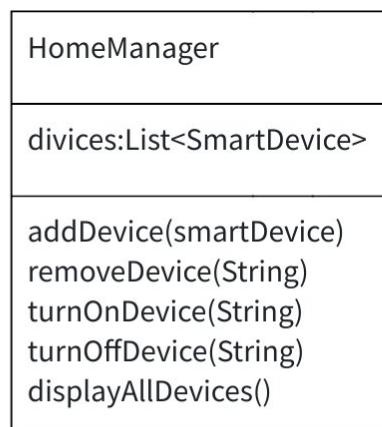
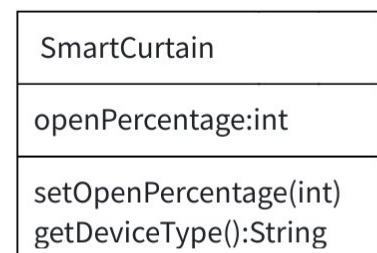
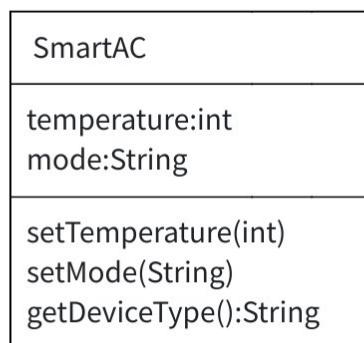
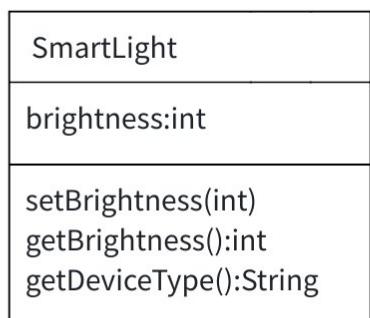
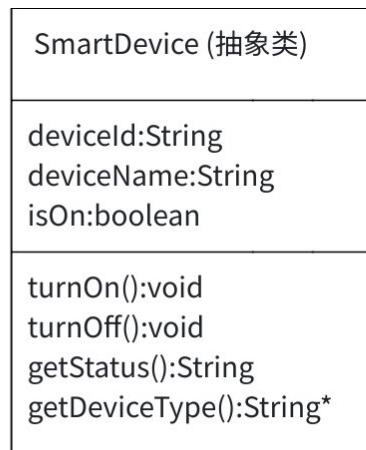


软件构造实验2

1. 类设计与规则

- 类图



2. 编码实现

代码见文件夹

3. 测试与验证

结果如图

The screenshot shows a Windows Command Prompt window titled 'C:\Windows\System32\cmd.exe'. The command entered was 'D:\Desktop\大三上\软件构造\2023214461+吴静+实验 2>javac -encoding UTF-8 TestMain.java'. The output of the application execution is displayed below:

```
D:\\Desktop\大三上\软件构造\2023214461+吴静+实验 2>java -Dfile.encoding=UTF-8 TestMain
智能灯添加成功
智能空调添加成功
智能窗帘添加成功
灯已开启
智能灯亮度已设为 80
空调已开启
空调温度设置为 22°C
空调模式设置为制热
窗帘已开启
窗帘打开 70%

当前设备状态：
设备ID: L001, 名称: 灯, 状态: 开
设备ID: A001, 名称: 空调, 状态: 开
设备ID: C001, 名称: 窗帘, 状态: 开

D:\Desktop\大三上\软件构造\2023214461+吴静+实验 2>
```

4. 问题回答

- 封装分析：哪些属性被封装了？为什么要将这些属性设为private？
 - 被封装的属性：`deviceId`, `deviceName`, `isOn`, 以及子类特有属性
 - 原因：防止外部直接修改设备状态，保证数据安全和一致性，通过公共方法控制访问。
- 继承层次：画出完整的类继承图，说明每个子类继承了哪些特性？
 - 见步骤1的类图
 - `SmartDevice` 是基类，`SmartLight`、`SmartAC`、`SmartCurtain` 都继承自它。每个子类继承了公共方法 (`turnOn`、`turnOff`、`getStatus`) 和公共属性 (`deviceId`、`deviceName`、`isOn`)
- 多态体现：在 `HomeManager` 的哪个方法中体现了多态？如果添加新的设备类型，需要修改 `HomeManager` 类吗？为什么？
 - `HomeManager` 中的 `turnOnDevice()` 与 `turnOffDevice()` 体现了多态。调用 `SmartDevice` 的 `turnOn()` 方法时，实际执行的是子类重写或继承的实现。
 - 添加新设备时，不需要修改 `HomeManager`，只需要继承 `smartDevice`，体现了开闭原则
- 设计原则：这个设计是否符合开闭原则？请举例说明。每个类的职责是否单一？分析各个类的职责。
 - 符合开闭原则（添加新设备时，不需要修改 `HomeManager`，只需要继承 `smartDevice`）
 - 每个类的职责都单一。分别负责定义通用特性、控制灯、控制空调、控制窗帘和负责设备管理

- 改进建议：你认为这个系统在面向对象设计方面还有哪些可以改进的地方？

- 添加新设备类型、场景模式、异常处理、接口设计

5. 扩展

- 接口

代码块

```
1 public interface Controllable {  
2     void turnOn();  
3     void turnOff();  
4     String getStatus();  
5 }  
6  
7 在SmartDevice.java中  
8 public abstract class SmartDevice implements Controllable{  
9     ...  
10 }
```

- smartTV

代码块

```
1 public class SmartTV extends SmartDevice {  
2     private int channel;  
3     private int volume;  
4  
5     public SmartTV(String id, String name) {  
6         super(id, name);  
7         this.channel = 1;  
8         this.volume = 20;  
9     }  
10  
11    public void setChannel(int channel) {  
12        if (channel <= 0) {  
13            System.out.println("频道号必须大于0");  
14            return;  
15        }  
16        this.channel = channel;  
17        System.out.println(getDeviceName() + "切换到频道" + channel);  
18    }  
19  
20    public void setVolume(int volume) {  
21        if (volume < 0 || volume > 100) {  
22            System.out.println("音量范围为0-100");  
23            return;
```

```
24     }
25     this.volume = volume;
26     System.out.println(getDeviceName() + "音量设置为" + volume);
27 }
28
29     public int getChannel() {
30         return channel;
31     }
32
33     public int getVolume() {
34         return volume;
35     }
36
37     @Override
38     public String getDeviceType() {
39         return "智能电视";
40     }
41 }
42
```

- 设备不存在异常

代码块

```
1  public class DeviceNotFoundException extends Exception {
2      public DeviceNotFoundException(String message) {
3          super(message);
4      }
5  }
```

- 场景模式

代码块

```
1 import java.util.*;
2
3 public class SceneMode {
4     private String sceneName;
5     private List<SmartDevice> deviceList;
6
7     public SceneMode(String sceneName) {
8         this.sceneName = sceneName;
9         this.deviceList = new ArrayList<>();
10    }
11}
```

```
12     public void addDevice(SmartDevice device) {
13         deviceList.add(device);
14     }
15
16     public void activate() {
17         System.out.println("激活场景: " + sceneName);
18         for (SmartDevice device : deviceList) {
19             device.turnOn();
20         }
21     }
22
23     public void deactivate() {
24         System.out.println("关闭场景: " + sceneName);
25         for (SmartDevice device : deviceList) {
26             device.turnOff();
27         }
28     }
29 }
30 }
```