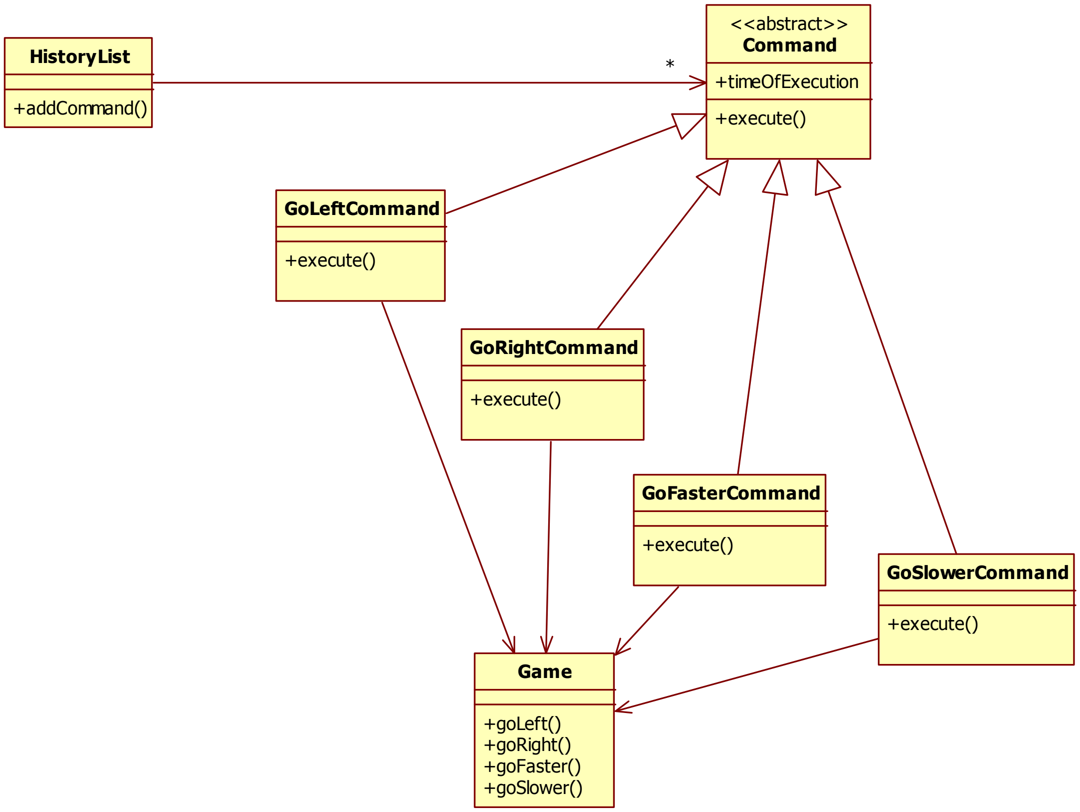
Question 1:

a. Command pattern

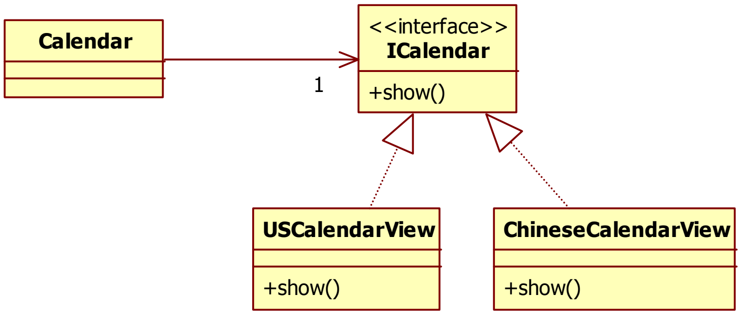
b.



Question 2:

a. Strategy pattern

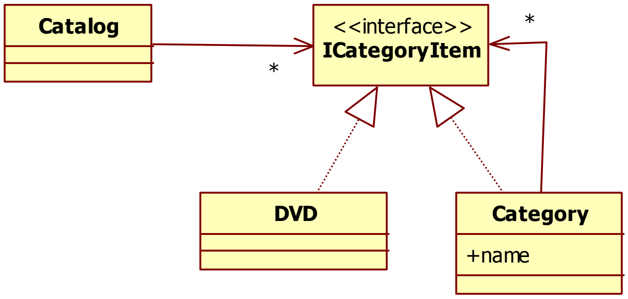
b.



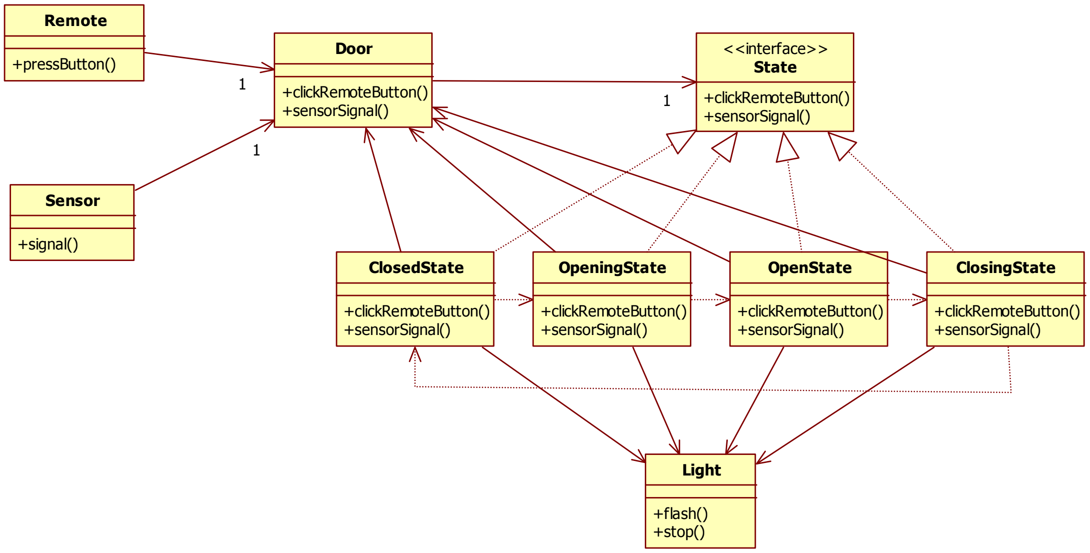
Question 3:

a. Composite pattern

b.



Question 4:



```java
public class Door {
        private State state;
        private Light light;

        public Door(Light light) {
                this.light = light;
                state = new ClosedState(this,light);
        }
        public void clickRemoteButton() {
                state.clickRemoteButton();
        }
        public void sensorSignal(String signal) {
                state.sensorSignal(signal);

        }
        public void setState(State state) {
                this.state = state;
        }
}
```

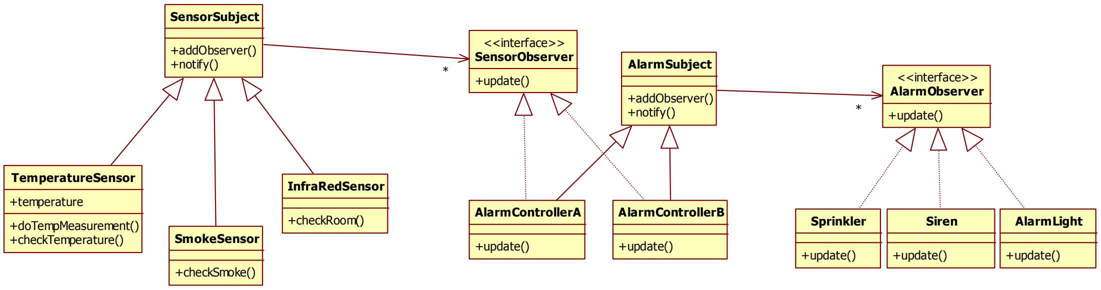
```java
public class ClosingState implements State{
        private Door door;
        private Light light;

        public ClosingState(Door door, Light light) {
                   this.door = door;
                   this.light=light;
                   light.flash("yellow");
                System.out.println(", new state = closing");
        }

             public void clickRemoteButton() {
                   System.out.print("old state = closing ");
                   door.setState(new OpeningState(door,light));
             }

             public void sensorSignal(String signal) {
                   System.out.print("old state = closing ");
                   if (signal.equals("close"))
                         door.setState(new ClosedState(door,light));
             }
}
```

Question 5


SensorSubject
+addObserver()
+notify()
*
<<interface>>
SensorObserver
+update()
AlarmSubject
+addObserver()
+notify()
*
<<interface>>
AlarmObserver
+update()
TemperatureSensor
+temperature
+doTempMeasurement()
+checkTemperature()
SmokeSensor
+checkSmoke()
InfraRedSensor
+checkRoom()
AlarmControllerA
+update()
AlarmControllerB
+update()
Sprinkler
+update()
Siren
+update()
AlarmLight
+update()