**AlarmSystem** implements **AlarmI**:

```java
public interface AlarmI {
    public void reset();
    public boolean intrusionOccurred(); // state dependent method!
    public int getIntrusionCount();
}
```

The AlarmSystem can be in one of two states: **Enabled** and **Alarmed**. It has one state-dependent method, `intrusionOccurred()`, which returns true if (and only if) it is in the Alarmed state. The method `reset()` returns the system to the Enabled state.

The AlarmSystem owns two instances of **Sensor** which implement **SensorI**:

```java
public interface SensorI {
    public void intrusionDetected();
}
```

A Sensor’s `intrusionDetected` method notifies the parent AlarmSystem. If the AlarmSystem is in the Enabled state, the system will then change state to Alarmed. The `getIntrusionCount` method returns the number of such intrusions since the last reset.

The AlarmSystem accepts **Visitors** such as **IntruderV** and **TestV**. Here is sample code for the accept method in AlarmSystem:

```java
public void accept(Visitor v) {
    sensor1.accept(v);
    sensor2.accept(v);
    v.visit(this);
}
```

Every Visitor to a Sensor causes an intrusion to be detected (the Sensor will call its own intrusionDetected method). A TestV is like an IntruderV, but it also resets the AlarmSystem.

**Write: AlarmSystem (with two Sensor instances), Visitor, TestV, and IntruderV. Use the driver you have been given.**

It is recommended that you hand-sketch the UML class diagrams if you need to think about your design.

Use the Java Observable class and Observer interface in your design!