An **AlarmSystem** implements **AlarmSystemI**: 

```java
public interface AlarmSystemI {
    public void reset();
    public boolean intrusionOccurred();
    public int numberOfIntrusionEvents();
}
```

The **AlarmSystem** can be in two states: **Enabled** and **Alarmed**. The system has one state-dependent method, **intrusionOccurred**, which returns true if (and only if) in the Alarmed state. The method **reset()** returns the system to the Enabled state. The system has a *collection* of instances of **Sensor**. It *observes* the individual Sensors. A Sensor implements **SensorI**: 

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

Whenever a Sensor’s **intrusionDetected** method is called, the Sensor must notify the **AlarmSystem**. If the **AlarmSystem** is in the Enabled state when notified, the system will change its state to Alarmed. A counter keeps track of the total number of intrusions detected by any of its Sensors since the last time the system was reset (the count is incremented even if already in the Alarmed state).

The **AlarmSystem** accepts **Visitors**. **Intruder** and **Tester** are two kinds of Visitor. When the **AlarmSystem** accepts a Visitor it visits with the Visitor and then passes the Visitor to its Sensors. The Visitor would be responsible for triggering an Sensor (by calling a Sensor’s intrusionDetected method). A Tester, for instance, will trigger every Sensor.

1) Draw the UML class diagrams for **AlarmSystem**, plus its State hierarchy, **Sensor**, and the Visitor hierarchy (Visitor, Intruder, and Test-Visitor).

2) Write all the code for all classes (**except** Intruder). Assume you have the classes Observable and Collection available (see reverse page).

3) Discuss your design to help appreciate special features or understand any problems you’ve encountered.
Here is a simple Observer interface

   public interface Observer {
      public void update(Observable o, Object arg);
   }

And presume you may use a (simplified) Observable class with methods:

   public void addObserver(Observer o);
   public void notifyObservers();
   public void notifyObservers(Object o);
   public void setChanged();
   public void clearChanged();

Presume you have the class Collection which implements the CollectionI interface:

   public interface CollectionI {
      boolean add(Item i);
      Item    get();
      void    start();
      boolean more();
      ... others removed so you don't get tempted to meander off into trouble
   }