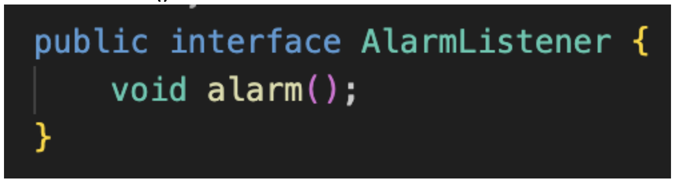
Name: E9_David_Mugwaneza

The Observer Design Pattern

Problem:
The objective of this exercise is to implement the Observer design pattern.

Observer pattern steps:

1. Create an interface called **AlarmListener**. This is the observer interface. In **AlarmListener**, there is a void **alarm**() method defined.

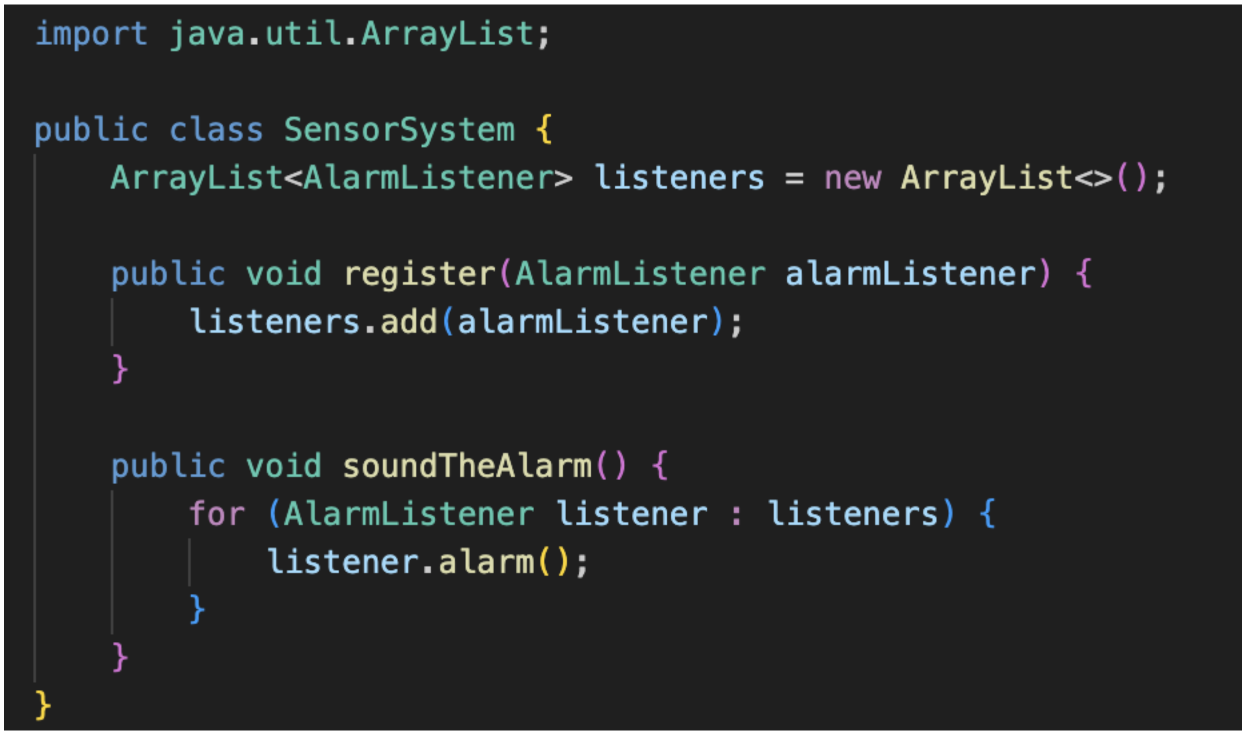


```java
public interface AlarmListener {
    void alarm();
}
```

2. Create a class called **SensorSystem**. This is the publisher class. Define one instance variable
   ArrayList< AlarmListener > listeners = new ArrayList();
   This ArrayList saves all the observers of this publisher.

3. In **SensorSystem**, define a method void **register**(AlarmListener alarmListener). What it does is to add alarmListener to the ArrayList listeners.

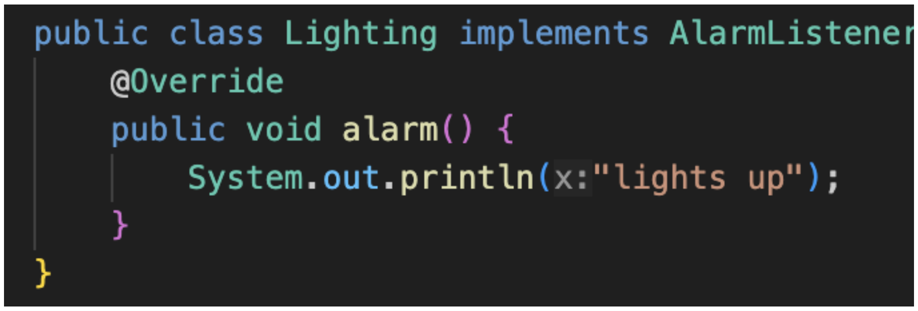
4. In **SensorSystem**, define another method void **soundTheAlarm**(). What it does is to use a for loop to loop through all the listeners/observers in the ArrayList listeners, and call their alarm() method.



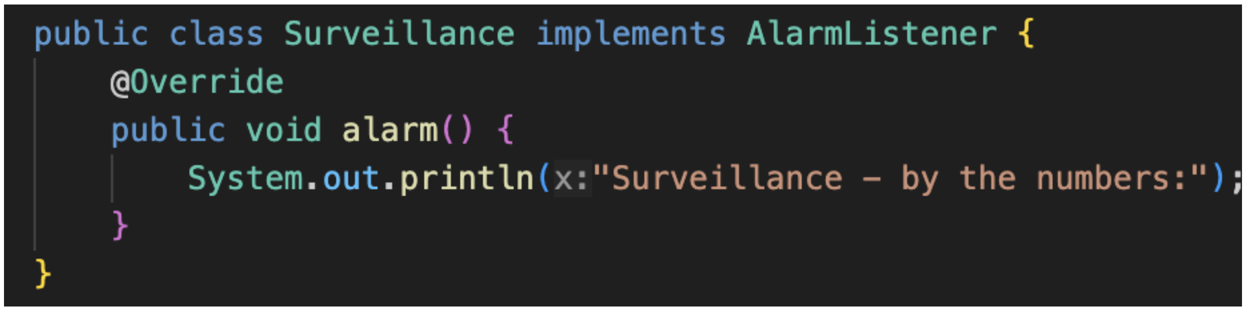
```java
import java.util.ArrayList;

public class SensorSystem {
    ArrayList<AlarmListener> listeners = new ArrayList<>();

    public void register(AlarmListener alarmListener) {
        listeners.add(alarmListener);
    }

    public void soundTheAlarm() {
        for (AlarmListener listener : listeners) {
            listener.alarm();
        }
    }
}
```

5. Different three concrete observer classes: **Lighting**, **Gates**, and **Surveillance**. Make them implement the interface **AlarmListener.** Implement the **alarm**() method in all three classes**.** In **Lighting**, **alarm**() prints out “lights up”. In **Gates**, **alarm**() prints out “gates close”. In **Surveillance**, **alarm**() prints out “Surveillance – by the numbers:”.

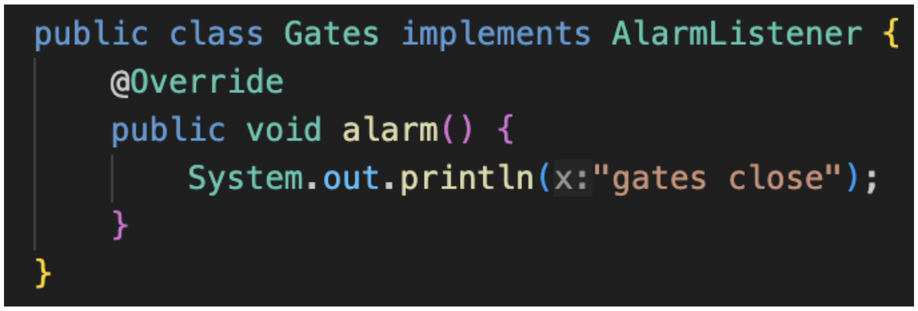
Name: E9_David_Mugwaneza



```java
public class Lighting implements AlarmListener
    @Override
    public void alarm() {
        System.out.println(x:"lights up");
    }
}
```



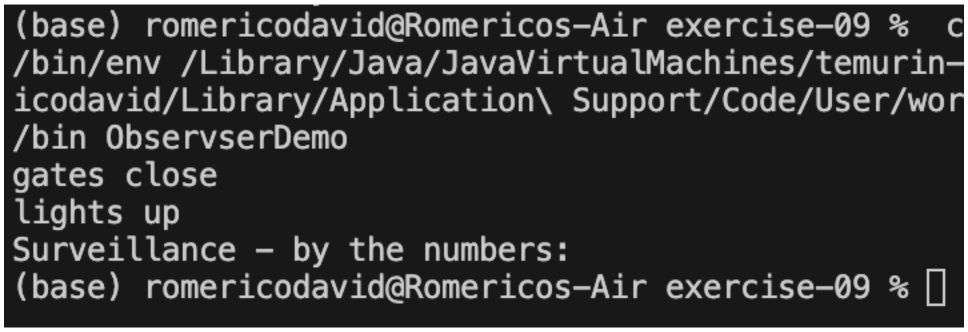
```java
public class Surveillance implements AlarmListener {
    @Override
    public void alarm() {
        System.out.println(x:"Surveillance - by the numbers:");
    }
}
```



```java
public class Gates implements AlarmListener {
    @Override
    public void alarm() {
        System.out.println(x:"gates close");
    }
}
```

6. Use the following client code to try it.

```java
public class ObservserDemo {
  public static void main( String[] args ) {
    SensorSystem sensorSystem = new SensorSystem();
    sensorSystem.register(new Gates());
    sensorSystem.register(new Lighting());
    sensorSystem.register(new Surveillance());
    sensorSystem.soundTheAlarm();
  }
}
```



```
(base) romericodavid@Romericos-Air exercise-09 %  c
/bin/env /Library/Java/JavaVirtualMachines/temurin-
icodavid/Library/Application\ Support/Code/User/wor
/bin ObservserDemo
gates close
lights up
Surveillance - by the numbers:
(base) romericodavid@Romericos-Air exercise-09 % 
```

Name: E9_David_Mugwaneza

Upload your code to the Blackboard when you are done.