```
import java.util.*;

//Step 1: Create Observer interface and DisplayElement Interface for
display common method
//public
interface Observer {
 public void update(float temp, float humidity, float pressure);
}

//public
interface DisplayElement {
 public void display();
}



//Step 2: Create Observable interface Subject.java
//public
interface Subject {
 public void registerObserver(Observer o);
 public void removeObserver(Observer o);
 public void notifyObservers();
}



//Step 3: Create WeatherData class to implement Subject interface.

//public
class WeatherData implements Subject {
 private ArrayList<Observer> observers;
 private float temperature;
 private float humidity;
 private float pressure;

 public WeatherData() {
  observers = new ArrayList<>();
 }

 public void registerObserver(Observer o) {
  observers.add(o);
 }

 public void removeObserver(Observer o) {
  int i = observers.indexOf(o);
  if (i >= 0) {
   observers.remove(i);
  }
 }

 public void notifyObservers() {
  for (int i = 0; i < observers.size(); i++) {
   Observer observer = (Observer)observers.get(i);
   observer.update(temperature, humidity, pressure);
  }
```

```
 }

 public void measurementsChanged() {
  notifyObservers();
 }

 public void setMeasurements(float temperature, float humidity, float
pressure) {
  this.temperature = temperature;
  this.humidity = humidity;
  this.pressure = pressure;
  measurementsChanged();
 }

 public float getTemperature() {
  return temperature;
 }

 public float getHumidity() {
  return humidity;
 }

 public float getPressure() {
  return pressure;
 }
}



//Step 4: Create Observer ForecastDisplay class.
//public
class ForecastDisplay implements Observer, DisplayElement {
 private float currentPressure = 29.92f;
 private float lastPressure;
 private WeatherData weatherData;

 public ForecastDisplay(WeatherData weatherData) {
  this.weatherData = weatherData;
  weatherData.registerObserver(this);
 }

 public void update(float temp, float humidity, float pressure) {
                lastPressure = currentPressure;
  currentPressure = pressure;

  display();
 }

 public void display() {
  System.out.print("Forecast: ");
  if (currentPressure > lastPressure) {
   System.out.println("Improving weather on the way!");
  } else if (currentPressure == lastPressure) {
   System.out.println("More of the same");
  } else if (currentPressure < lastPressure) {
   System.out.println("Watch out for cooler, rainy weather");
  }
```

```
 }
}



//Step 5: Create second Observer HeatIndexDisplay class.
//public
class HeatIndexDisplay implements Observer, DisplayElement {
 float heatIndex = 0.0f;
 private WeatherData weatherData;

 public HeatIndexDisplay(WeatherData weatherData) {
  this.weatherData = weatherData;
  weatherData.registerObserver(this);
 }

 public void update(float t, float rh, float pressure) {
  heatIndex = computeHeatIndex(t, rh);
  display();
 }

 private float computeHeatIndex(float t, float rh) {
  float index = (float)((16.923 + (0.185212 * t) + (5.37941 * rh) -
(0.100254 * t * rh)
   + (0.00941695 * (t * t)) + (0.00728898 * (rh * rh))
   + (0.000345372 * (t * t * rh)) - (0.000814971 * (t * rh * rh)) +
   (0.0000102102 * (t * t * rh * rh)) - (0.000038646 * (t * t * t)) +
(0.0000291583 *
   (rh * rh * rh)) + (0.00000142721 * (t * t * t * rh)) +
   (0.000000197483 * (t * rh * rh * rh)) - (0.0000000218429 * (t * t *
t * rh * rh)) +
   0.000000000843296 * (t * t * rh * rh * rh)) -
   (0.0000000000481975 * (t * t * t * rh * rh * rh)));
  return index;
 }

 public void display() {
  System.out.println("Heat index is " + heatIndex);
 }
}



//Step 6: Create third Observer StatisticsDisplay class.
//public
class StatisticsDisplay implements Observer, DisplayElement {
 private float maxTemp = 0.0f;
 private float minTemp = 200;
 private float tempSum= 0.0f;
 private int numReadings;
 private WeatherData weatherData;

 public StatisticsDisplay(WeatherData weatherData) {
  this.weatherData = weatherData;
  weatherData.registerObserver(this);
 }
```

```
 public void update(float temp, float humidity, float pressure) {
  tempSum += temp;
  numReadings++;

  if (temp > maxTemp) {
   maxTemp = temp;
  }

  if (temp < minTemp) {
   minTemp = temp;
  }

  display();
 }

 public void display() {
  System.out.println("Avg/Max/Min temperature = " + (tempSum /
numReadings)
   + "/" + maxTemp + "/" + minTemp);
 }
}



//Step 7: Create fourth Observer CurrentConditionsDisplay class.
//public
 class CurrentConditionsDisplay implements Observer, DisplayElement {
 private float temperature;
 private float humidity;
 private Subject weatherData;

 public CurrentConditionsDisplay(Subject weatherData) {
  this.weatherData = weatherData;
  weatherData.registerObserver(this);
 }

 public void update(float temperature, float humidity, float pressure)
{
  this.temperature = temperature;
  this.humidity = humidity;
  display();
 }

 public void display() {
  System.out.println("Current conditions: " + temperature
   + "F degrees and " + humidity + "% humidity");
 }
}



//Step 8: Create WeatherStation class to test observer design pattern.

public class WeatherStation {

 public static void main(String[] args) {
  WeatherData weatherData = new WeatherData();
```

```
  CurrentConditionsDisplay currentDisplay =
   new CurrentConditionsDisplay(weatherData);
  StatisticsDisplay statisticsDisplay = new
StatisticsDisplay(weatherData);
  ForecastDisplay forecastDisplay = new ForecastDisplay(weatherData);

  weatherData.setMeasurements(80, 65, 30.4f);
  weatherData.setMeasurements(82, 70, 29.2f);
  weatherData.setMeasurements(78, 90, 29.2f);
 }
}

/*
Output:
Current conditions: 80.0F degrees and 65.0% humidity
Avg/Max/Min temperature = 80.0/80.0/80.0
Forecast: Improving weather on the way!
Current conditions: 82.0F degrees and 70.0% humidity
Avg/Max/Min temperature = 81.0/82.0/80.0
Forecast: Watch out for cooler, rainy weather
Current conditions: 78.0F degrees and 90.0% humidity
Avg/Max/Min temperature = 80.0/82.0/78.0
Forecast: More of the same*/
```