CECS 277 LAB OBSERVER PATTERN

OBJECTIVE: Build on the code that we went over in lecture so that you get a better picture of how it all works together.

INTRODUCTION: Please remember the coding standards here.

The benefit of the observer pattern is that it gives us flexibility. Certain types of changes require little if any rewrite of the code. In this lab, we're going to start with the code that we went over in lecture. You can find that here.

Then we want to add a new pair of weather measurements, and a new display class to the mix and make this into a **pull** rather than a push circumstance. The new measurements are wind direction and speed. The wind direction is captured as an integer value of the angle from true north. So, if the wind comes from the north, then the direction is 180. The speed we will assume to be in miles per hour on average.

PROCEDURE:

1. Update the `Weatherdata` class:
    a. To include the two new instance variables: one for the direction of the wind, and the other for the wind speed in miles/hour.
    b. Getter methods for all the weather measurements. The `Observer` concrete implementations will use those getter methods to extract only those weather measurements of interest from the `WeatherData` instance that they are subscribing to.
2. Update the `Observer` interface so that the update method passes a reference to a `Subject`. This essentially makes this into a pull approach. The observers now will have to use the getter methods in the Subject instance to get the new values.
3. Update the Observers: `CurrentConditionsDisplay`, `ForecastDisplay`, `StatisticsDIsplay`, and `WindConditionsDisplay` so that their update methods accept a reference to **`Subject.`** Extract the necessary data from there using the new getter methods that you added to `WeatherData`. In order to do that, you will need to cast the Subject instance to `WeatherData` before you attempt to use the `WeatherData` getter methods.
4. Clone one of the Observers to create a new Observer concrete implementation: WindConditionsDisplay.
    a. WindConditionsDisplay only cares about the direction and speed of the wind.
    b. Have your `display` method print out the direction of the wind (just North, South, East, West will do. Do not get any finer than that).
    c. Have your display method warn the user to get into a shelter if the wind velocity > 100.
    d. If the wind velocity is 50-100 mph and getting worse, warn the user.

   e. If the wind velocity is 50-100 mph and getting better (i.e. the velocity is getting lower) let the user know things are moderating.

5. Update the ObserverRunner application
   a. Create an instance of WindConditionsDisplay so that your new Observer gets to see some action.
   b. Add wind direction and speed measurements to your `setMeasurements` method calls.

WHAT TO TURN IN:

- CurrentConditionsDisplay.java
- ForecastDisplay.java
- Observer.java
- ObserverRunner.java
- StatisticsDisplay.java
- WeatherData.java
- WindConditionsDisplay.java
- Your sample console output, named `console.txt`

SAMPLE CONSOLE 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!
Wind Forecast: Good kite flying weather.
Wind blowing out of the: North
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
Wind Forecast: Conditions moderate.
Wind blowing out of the: North
Current conditions: 78.0F degrees and 90.0% humidity
Avg/Max/Min temperature = 80.0/82.0/78.0
Forecast: More of the same
Wind Forecast: Conditions moderate.
Wind blowing out of the: West
Current conditions: 40.0F degrees and 40.0% humidity
Avg/Max/Min temperature = 70.0/82.0/40.0
Forecast: Watch out for cooler, rainy weather
Wind Forecast: Conditions moderate and worsening.
Wind blowing out of the: North
Completed satisfactorily.
```