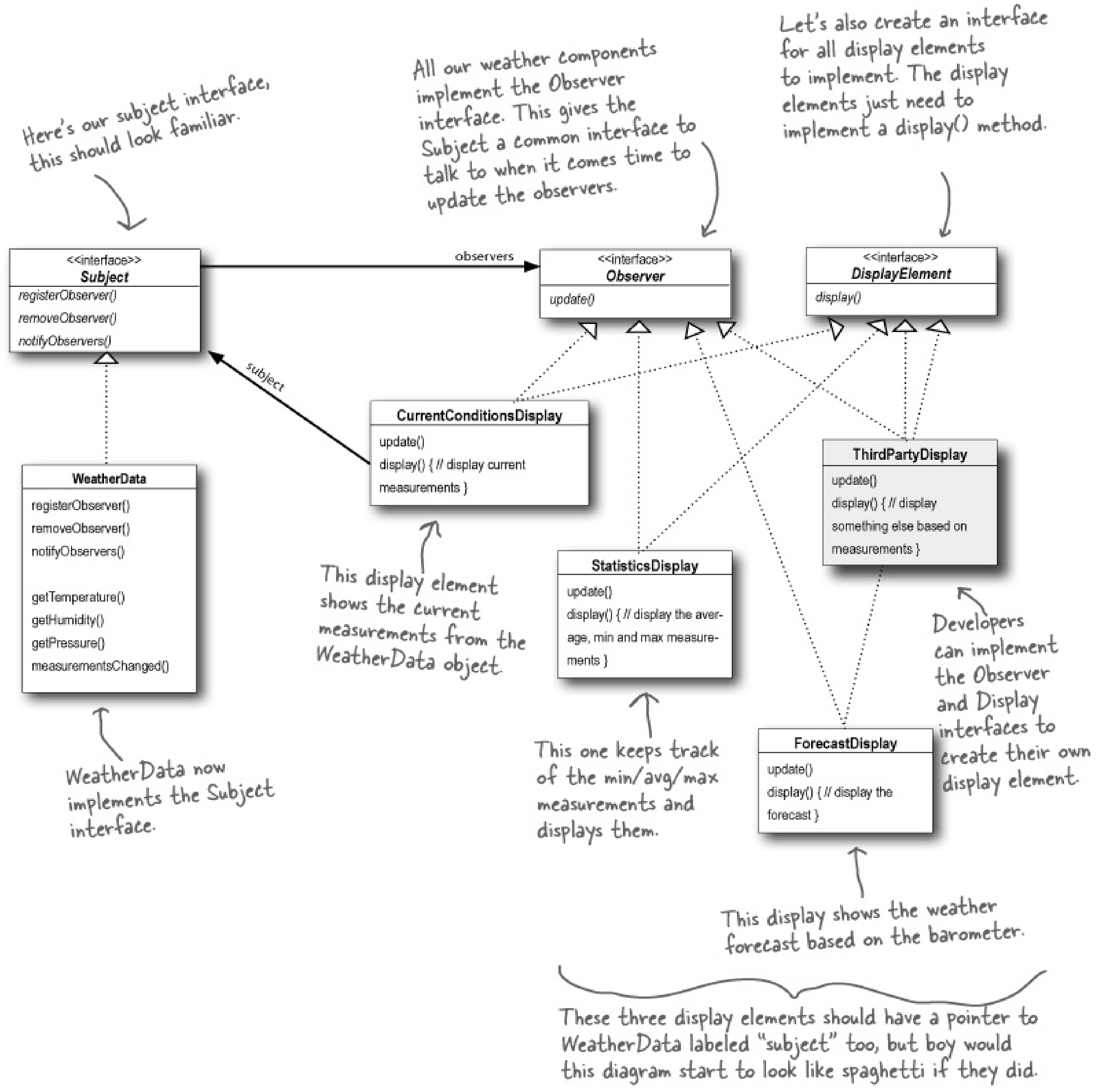
158.326 Software Construction

Tutorial 5

In this tutorial you will be implementing the Observer pattern covered in the lecture 5, using the Weather Station case study found in chapter 2 of "Head First Design Patterns". Below is the class diagram for this case study, implemented using the Observer pattern.

We will implement the Observer pattern for this case study as a Windows Forms application that comprises of 4 forms. From the diagram below, WeatherData, CurrentConditionDisplay, ForecastDisplay and StatisticsDisplay classes will be a separate Form class each. Ignore the ThirdPartyDisplay class in the diagram.

We will build this application incrementally, one display form at a time.

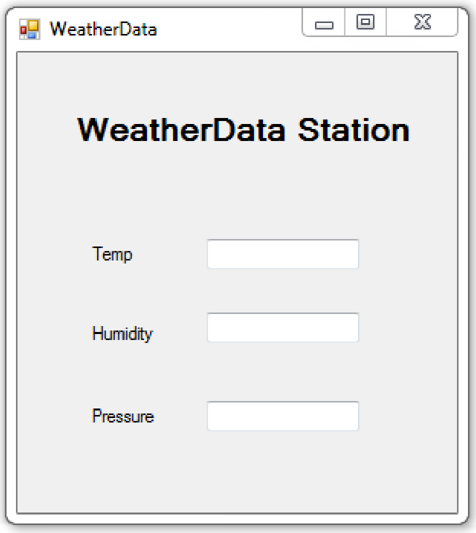


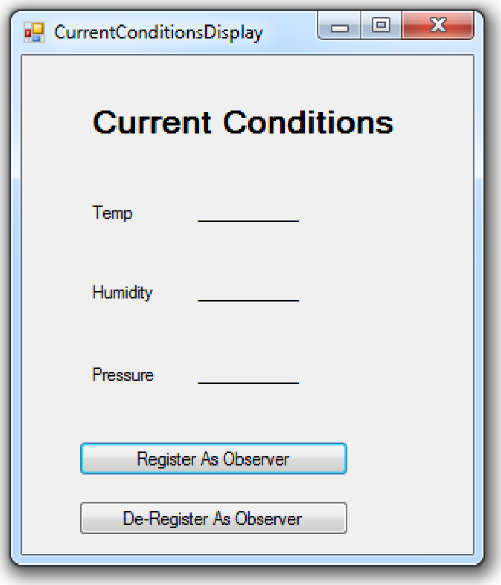
Here's our subject interface, this should look familiar.
All our weather components implement the Observer interface. This gives the Subject a common interface to talk to when it comes time to update the observers.
Let's also create an interface for all display elements to implement. The display elements just need to implement a display() method.
<<interface>> Subject
registerObserver()
removeObserver()
notifyObservers()
observers
<<interface>> Observer
update()
<<interface>> DisplayElement
display()
subject
CurrentConditionsDisplay
update()
display() { // display current measurements }
WeatherData
registerObserver()
removeObserver()
notifyObservers()
getTemperature()
getHumidity()
getPressure()
measurementsChanged()
ThirdPartyDisplay
update()
display() { // display something else based on measurements }
StatisticsDisplay
update()
display() { // display the average, min and max measurements }
ForecastDisplay
update()
display() { // display the forecast }
This display element shows the current measurements from the WeatherData object.
Developers can implement the Observer and Display interfaces to create their own display element.
WeatherData now implements the Subject interface.
This one keeps track of the min/avg/max measurements and displays them.
This display shows the weather forecast based on the barometer.
These three display elements should have a pointer to WeatherData labeled "subject" too, but boy would this diagram start to look like spaghetti if they did.

Image from "Head First Design Patterns" by Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra

Exercise 1

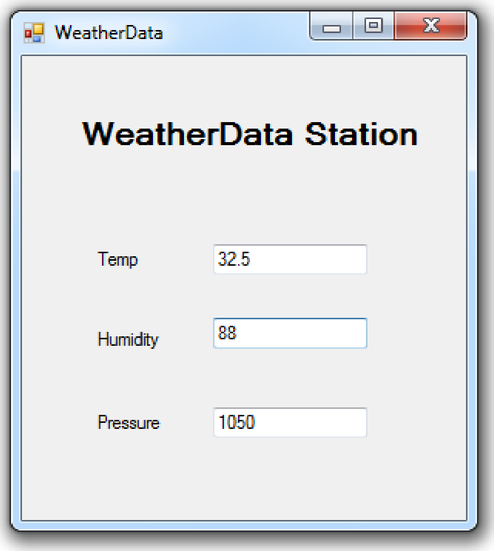
Create a C# windows forms project. Define the 3 interfaces from the diagram. First create the WeatherData form that looks something like the image below left, in which one can enter the temperature, humidity and pressure values. Choose yourself what type of a user event you would like to be triggered that will read the text box values and store them into variables in your form (floats or ints). Next, implement the interface methods in the form and any other methods you might need.



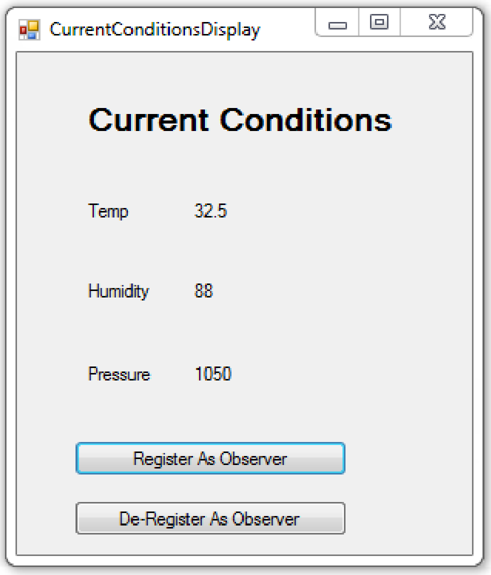


Next create the CurrentConditionsDisplay form that looks something like the form above right, whose functionality is to display values from the WeatherData form when it is registered as an observer. Implement the interfaces' methods and the button click events. For the `Update`() method in the Observer interface, use the **push technique** to update the observer objects as shown in the book. That means that the `Update`() method will be taking the temperature, humidity and pressure values as arguments and passing them to all the observers. Define the constructors in both the CurrentConditionsDisplay and WeatherData forms as shown in the class as well as in the supplementary lecture slides in order to create a multi-form application.

Once you have connected everything up and launched your application, whatever is typed into the WeatherData form should be reflected in the CurrentConditions form when the observer form registers as an observer with the subject, as shown in the images below.



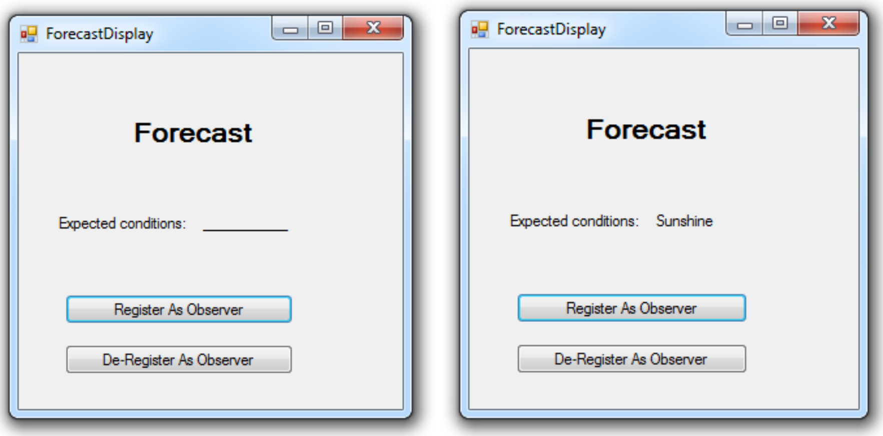
Temp 32.5
Humidity 88
Pressure 1050



Temp 32.5
Humidity 88
Pressure 1050

Exercise 2

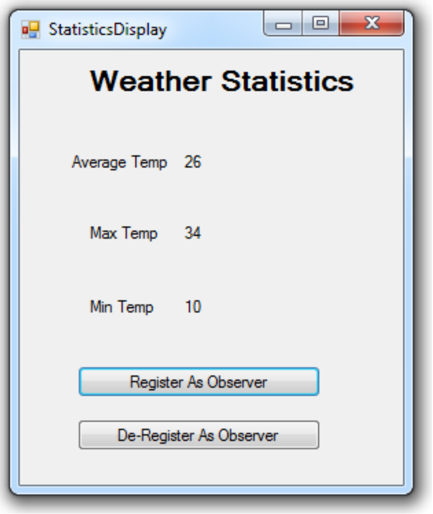
Extend the application by adding the ForecastDisplay form as pictured below.



Implement very naïve logic which sets the “Expected conditions” label as “Sunshine” if the pressure in the WeatherData form is above 1000, otherwise as “Rain”.

Exercise 3

Extend the application further by adding the StatisticsDisplay form as pictured below.



This form will keep track of all the temperature entries during the periods when it was registered as an observer, and will provide simple statistics such as the average, maximum and the minimum temperatures experienced.

Exercise 4

You have now seen how the Observer patterns works. There are, however, different ways of implementing it. You have implemented the Observer pattern using the push technique, whereby it is the Subject who decides what data to update the observers with. As an exercise, now modify your code to implement the **pull technique**. Start by adding a new method to the Observer interface called `UpdatePull`() which unlike the Update() method, takes no arguments. Modify your program in such a way that enables the observers to fetch the data they specifically need from the Subject.

Exercise 5

Add another interface `INotes` to the three Display Elements which has one method `WeatherNotes()` which gives both new and previous reading of relevant weather changes, i.e., Weather Statistics gives temperature changes only, etc.