Observer

Estimated time for completion: 30 minutes

Overview:

The observer pattern lets objects be informed when something happens elsewhere. Traditionally observers are implemented using interfaces with the observer registering an interest with an observer. In this lab you will start with that setup and refactor the code to use .NET events.

Goals:

- Understand how the observer pattern works in .NET
- Refactor existing interface based code to use events

Lab Notes:

Part 1: Using the observer pattern

In this lab you will take interface based observer code and refactor to a delegate based observer.

The lab is a 'Console Clock' application. Using the .NET 2.0 Console APIs various clocks can be display on a console window. These clocks can be updated at various intervals ranging from once every hundredth of a second to once every hour. Currently the clocks are implemented as observers. They implement an ITimer interface and are registered with the timer class. The time runs on a thread and calls the observer whenever necessary. You will take this code and use events and delegate registration

1. Open the solution (Observer.sln) and familiarize yourself with the project.
2. Run the ConsoleClock application to see the clocks in action. You will see four clocks in the top-left corner of the console, each in a different colour. The first two are updated each second, the third every tenth, the fourth every hundredth.
3. The DelegateConsoleClock application is currently a copy of the ConsoleClock application. This is where you will be working. Close all the files that are currently open in Visual Studio and make sure you only edit the files from the DelegateConsoleClock application
4. The first steps will be to update the Ticker class in Ticker.cs, open that file now.
    a. In the Ticker.cs file, before the class definition add a new delegate type. Name the type Tick
    b. Within the Ticker class definition add 5 instances of this type, one for each type of tick. Call these instances Tick OnTenths, Tick OnHundredths, Tick OnSecondss, Tick OnMinutes and Tick OnHours
    c. These are the delegates that will fire when a clock needs to be updated.

```
public delegate void Tick();

public class Ticker
```

```
{
    public Tick OnTenths;
    public Tick OnHundredths;
    public Tick OnSeconds;
    public Tick OnMinutes;
    public Tick OnHours;
    ...
}
```

d. Delete the timers list and the code that uses it inside Run
e. Inside the Run method you need to replace any code that users the timer with calls to the delegates instead. Remember to check that the delegate is not null before calling it

```
public void Run()
{
    int count = 0;
    while (!done)
    {
        Thread.Sleep(0);

        Interlocked.Increment(ref count);
        if (OnHundredths != null)
            OnHundredths();


        if (count % 10 == 0)
        {
            if (OnTenths != null)
                OnTenths();
        }
        if (count % 100 == 0)
        {
            if (OnSeconds != null)
                OnSeconds();
        }
        if (count % 6000 == 0)
        {
            if (OnMinutes != null)
                OnMinutes();
        }
        if (count % 36000 == 0)
        {
            if (OnHours != null)
                OnHours();
        }
    }
}
```

f. Remove the RegisterTimer and DeregisterTimer methods

5. At this point the code will not build as the Main method is trying to register with the ticker using the methods that you have just deleted. So the next step is to fix the Main

a. In Main remove all the calls to ticker.RegisterTimer and replace them with calls adding the appropriate method to the appropriate delegate

```
Ticker ticker = new Ticker();
ticker.OnSeconds += clock.Second;
ticker.OnSeconds += clock2.Second;
ticker.OnTenths += clock3.TenthSecond;
ticker.OnHundredths += clock4.HundredthSecond;
```

b. Build and run the code. It should now work like before.

6. You now have some tidying up to do. The delegates define in Ticker are currently public which means you can overwrite them and also call them from outside the Ticker class. This is not good. Change the delegates to be defined as events.

```
public delegate void Tick();

public class Ticker
{
    public event Tick OnTenths;
    public event Tick OnHundredths;
    public event Tick OnSeconds;
    public event Tick OnMinutes;
    public event Tick OnHours;
    ...
}
```

7. You can also delete all references to the ITimer interface and remove all the extra methods that the various clocks no longer need.
    a. Delete the ITimer.cs file from the project.
    b. Open clocks.cs and remove all references to ITimer
    c. In clocks.cs tidy up SecondClock, HundredthSecondClock and TenthSecondClock by removing the methods that are not used (those with empty bodies). Do not remove the constructors!
8. Build and run the application. It should still work as before.