Sharpen your pencil Solution

Based on our first implementation, which of the following apply? (Choose all that apply.)

- ☑ A. We are coding to concrete implementations, not interfaces.
- ☑ B. For every new display element, we need to alter code.
- ☑ C. We have no way to add display elements at runtime.
- ❑ D. The display elements don't implement a common interface.
- ☑ E. We haven't encapsulated what changes.
- ❑ F. We are violating encapsulation of the WeatherData class.

Design Principle Challenge Solution

Design Principle

Identify the aspects of your application that vary and separate them from what stays the same.

The thing that varies in the Observer Pattern is the state of the Subject and the number and types of Observers. With this pattern, you can vary the objects that are dependent on the state of the Subject, without having to change that Subject. That's called planning ahead!

Design Principle

Program to an interface, not an implementation.

Both the Subject and Observers use interfaces. The Subject keeps track of objects implementing the Observer interface, while the Observers register with, and get notified by, the Subject interface. As we've seen, this keeps things nice and loosely coupled.

Design Principle

Favor composition over inheritance.

The Observer Pattern uses composition to compose any number of Observers with their Subject. These relationships aren't set up by some kind of inheritance hierarchy. No, they are set up at runtime by composition!