



Tools for your Design Toolbox

Welcome to the end of Chapter 2. You've added a few new things to your OO toolbox...

OO Basics

Abstraction

OO Principles

Encapsulate what varies.

Favor composition over inheritance.

Program to interfaces, not implementations.

Strive for loosely coupled designs between objects that interact.

Here's your newest principle. Remember, loosely coupled designs are much more flexible and resilient to change.

OO Patterns

Strap
encap
inter
vary

Observer – defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically

A new pattern for communicating state to a set of objects in a loosely coupled manner. We haven't seen the last of the Observer Pattern—just wait until we talk about MVC!



BULLET POINTS

- The Observer Pattern defines a one-to-many relationship between objects.
- Subjects update Observers using a common interface.
- Observers of any concrete type can participate in the pattern as long as they implement the Observer interface.
- Observers are loosely coupled in that the Subject knows nothing about them, other than that they implement the Observer interface.
- You can push or pull data from the Subject when using the pattern (pull is considered more “correct”).
- Swing makes heavy use of the Observer Pattern, as do many GUI frameworks.
- You'll also find the pattern in many other places, including RxJava, JavaBeans, and RMI, as well as in other language frameworks, like Cocoa, Swift, and JavaScript events.
- The Observer Pattern is related to the Publish/Subscribe Pattern, which is for more complex situations with multiple Subjects and/or multiple message types.
- The Observer Pattern is a commonly used pattern, and we'll see it again when we learn about Model-View-Controller.