



## 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

tion

hism

ce

### 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

Strat  
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, or as we also know them, Observables, update Observers using a common interface.
- Observers are loosely coupled in that the Observable knows nothing about them, other than that they implement the Observer Interface.
- You can push or pull data from the Observable when using the pattern (pull is considered more “correct”).
- Don't depend on a specific order of notification for your Observers.
- Java has several implementations of the Observer Pattern, including the general purpose `java.util.Observable`.
- Watch out for issues with the `java.util.Observable` implementation.
- Don't be afraid to create your own Observable implementation if needed.
- Swing makes heavy use of the Observer Pattern, as do many GUI frameworks.
- You'll also find the pattern in many other places, including JavaBeans and RMI.

## Designing the Weather Station

How does this diagram compare with yours?

