



## Tools for your Design Toolbox

Your toolbox is starting to get heavy! In this chapter we've added a couple of patterns that allow us to alter interfaces and reduce coupling between clients and the systems they use.

### OO Principles

Encapsulate what varies.  
 Favor composition over inheritance.  
 Program to interfaces, not implementations.  
 Strive for loosely coupled designs between objects that interact.  
 Classes should be open for extension but closed for modification.  
 Depend on abstractions. Do not depend on concrete classes.  
 Talk only to your friends.

### OO Basics

abstraction  
 encapsulation  
 polymorphism  
 inheritance

We have a new technique for maintaining a low level of coupling in our designs (remember, talk only to your friends)...

### OO Patterns

Singleton  
 Factory Method  
 Simulator  
 Command  
 Adapter  
 Facade

**Adapter** - Converts the interface of a class into another interface clients expect. Lets classes work together that couldn't otherwise because of incompatible interfaces.

...and TWO new patterns. Each changes an interface, the adapter to convert, and the facade to unify and simplify.

**Facade** - Provides a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use.



## BULLET POINTS

- When you need to use an existing class and its interface is not the one you need, use an adapter.
- When you need to simplify and unify a large interface or complex set of interfaces, use a facade.
- An adapter changes an interface into one a client expects.
- A facade decouples a client from a complex subsystem.
- Implementing an adapter may require little work or a great deal of work depending on the size and complexity of the target interface.
- Implementing a facade requires that we compose the facade with its subsystem and use delegation to perform the work of the facade.
- There are two forms of the Adapter Pattern: object and class adapters. Class adapters require multiple inheritance.
- You can implement more than one facade for a subsystem.
- An adapter wraps an object to change its interface, a decorator wraps an object to add new behaviors and responsibilities, and a facade "wraps" a set of objects to simplify.