



Tools for your Design Toolbox

You've nearly made it through the first chapter! You've already put a few tools in your OO toolbox; let's make a list of them before we move on to Chapter 2.

OO Basics

Abstraction
Encapsulation
Polymorphism
Inheritance

We assume you know the OO basics like abstraction, encapsulation, polymorphism, and inheritance. If you are a little rusty on these, pull out your favorite object-oriented book and review, then skim this chapter again.

OO Principles

Encapsulate what varies.
Favor composition over inheritance.
Program to interfaces, not implementations.

We'll be taking a closer look at these down the road and also adding a few more to the list.

OO Patterns

Strategy – defines a family of algorithms, encapsulates each one, and makes them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

One down, many to go!

Throughout the book, think about how patterns rely on OO basics and principles.



BULLET POINTS

- Knowing the OO basics does not make you a good OO designer.
- Good OO designs are reusable, extensible, and maintainable.
- Patterns show you how to build systems with good OO design qualities.
- Patterns are proven object-oriented experience.
- Patterns don't give you code, they give you general solutions to design problems. You apply them to your specific application.
- Patterns aren't *invented*, they are *discovered*.
- Most patterns and principles address issues of *change* in software.
- Most patterns allow some part of a system to vary independently of all other parts.
- We often try to take what varies in a system and encapsulate it.
- Patterns provide a shared language that can maximize the value of your communication with other developers.