there are no Dumb Questions

Q: Open for extension and closed for modification? That sounds very contradictory. How can a design be both?

A: That's a very good question. It certainly sounds contradictory at first. After all, the less modifiable something is, the harder it is to extend, right?

As it turns out, though, there are some clever OO techniques for allowing systems to be extended, even if we can't change the underlying code. Think about the Observer Pattern (in Chapter 2)...by adding new Observers, we can extend the Subject at any time, without adding code to the Subject. You'll see quite a few more ways of extending behavior with other OO design techniques.

Q: Okay, I understand Observer, but how do I generally design something to be extensible yet closed for modification?

A: Many of the patterns give us time-tested designs that protect your code from being modified by supplying a means of extension. In this chapter you'll see a good example of using the Decorator Pattern to follow the Open-Closed Principle.

Q: How can I make every part of my design follow the Open-Closed Principle?

A: Usually, you can't. Making OO design flexible and open to extension without modifying existing code takes time and effort. In general, we don't have the luxury of tying down every part of our designs (and it would probably be wasteful). Following the Open-Closed Principle usually introduces new levels of abstraction, which adds complexity to our code. You want to concentrate on those areas that are most likely to change in your designs and apply the principles there.

Q: How do I know which areas of change are more important?

A: That is partly a matter of experience in designing OO systems and also a matter of knowing the domain you are working in. Looking at other examples will help you learn to identify areas of change in your own designs.

While it may seem like a contradiction, there are techniques for allowing code to be extended without direct modification.

Be careful when choosing the areas of code that need to be extended; applying the Open-Closed Principle EVERYWHERE is wasteful and unnecessary, and can lead to complex, hard-to-understand code.