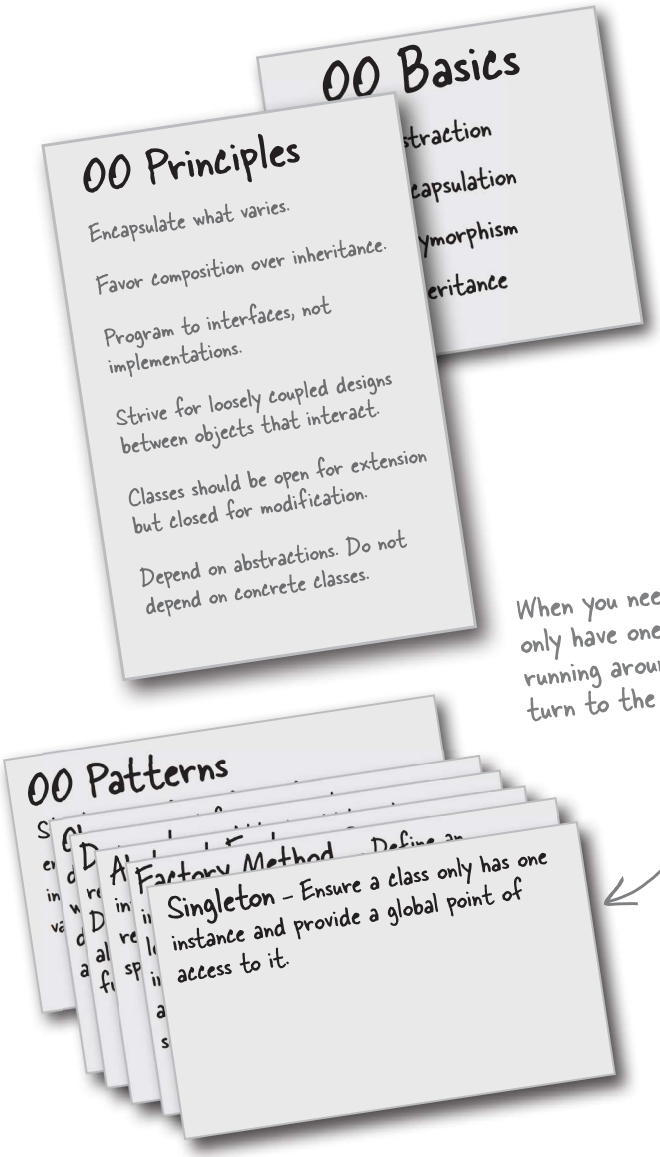
Tools for your Design Toolbox

You've now added another pattern to your toolbox. Singleton gives you another method of creating objects—in this case, unique objects.



As you've seen, despite its apparent simplicity, there are a lot of details involved in Singleton's implementation. After reading this chapter, though, you're ready to go out and use Singleton in the wild.

BULLET POINTS

- The Singleton Pattern ensures you have at most one instance of a class in your application.
- The Singleton Pattern also provides a global access point to that instance.
- Java's implementation of the Singleton Pattern makes use of a private constructor, a static method combined with a static variable.
- Examine your performance and resource constraints and carefully choose an appropriate Singleton implementation for multithreaded applications (and we should consider all applications multithreaded!).
- Beware of the double-checked locking implementation; it isn't thread safe in versions before Java 5.
- Be careful if you are using multiple class loaders; this could defeat the Singleton implementation and result in multiple instances.
- You can use Java's enums to simplify your Singleton implementation.