CHAPTER 6

The Singleton Pattern

In this chapter, we'll take up the Singleton pattern. This pattern is grouped with the other Creational patterns, although it is to some extent a pattern that limits, rather than promotes, the creation of classes. Specifically, it ensures that there is one and only one instance of a class and provides a global point of access to that instance. Any number of cases in programming in which you need to ensure that there can be one and only one instance of a class are possible. For example, your system might have only one window manager or print spooler or one point of access to a database engine. Or, your computer might have several serial ports, but there can only be one instance of COM1.

Creating a Singleton Using a Static Method

The easiest way to create a class that can have only one instance is to embed a static variable inside of the class that is set on the first instance and then check for it each time that you enter the constructor. A *static variable* is a variable for which there is only one instance, no matter how many instances of the class exist. To prevent instantiating the class more than once, make the constructor private so that an instance can be created only from within the static method of the class. Then we create a method called *Instance,* which will return an instance of *Spooler,* or *null,* if the class has already been instantiated. This is illustrated in the following code:

```
public class PrintSpooler {
    private static PrintSpooler spooler;

    private PrintSpooler() {
    }
```