```
    //return only one spooler instance
    public static synchronized PrintSpooler getSpooler() {
        if (spooler == null)              //if none created
            spooler = new PrintSpooler(); //create one
        return spooler;                   //return it
    }
    public void print(String s) {
        System.out.println(s);
    }
}
```

This approach has the major advantage that you don't have to worry about exception handling if the Singleton already exists—you always get the same instance of the spooler.

And, should you try to create instances of the PrintSpooler class directly, creating instances will fail at compile time because the constructor has been declared as private.

```
    //fails at compile time because constructor is privatized
    PrintSpooler pr3 = new PrintSpooler();
```

Finally, should you need to change the program to allow two or three instances, this class is easily modified to allow this.

If instead you want to know whether a spooler has been created and that you cannot create another, you can return a null if the spooler already exists, or you can throw an exception.

## Exceptions and Instances

If you want to know whether you have created a new instance, you must check the *getSpooler* method return to make sure it is not null. Assuming that programmers will always remember to check for errors is the beginning of a slippery slope that many prefer to avoid.

Instead, you can create a class that throws an exception if you attempt to instantiate it more than once. This requires the programmer to take action and is thus a safer approach.

Let's create our own exception class for this case.

```
public class SingletonException extends RuntimeException {
    //new exception type for singleton classes
    public SingletonException() {
        super();
    }
    public SingletonException(String s) {
        super(s);
    }
}
```