```
        pr2 = new Spooler();
        }
        catch (SingletonException e)
        {System.out.println(e.getMessage());}
    }
}
```

If we execute this program, we get the following results:

```
Opening one spooler
printer opened
Opening two spoolers
Only one spooler allowed
```

The last line indicates that an exception was thrown as expected. The complete source of this program is on the example CD-ROM as singleSpooler.java.

Providing a Global Point of Access to a Singleton Pattern

Since a Singleton pattern is used to provide a single point of global access to a class, your program design must provide for a way to reference the Singleton throughout the program, even though Java has no global variables.

One solution is to create such Singletons at the beginning of the program and pass them as arguments to the major classes that might need to use them.

```
pr1 = new Spooler();
Customers cust = new Customers (pri);
```

The disadvantage is that you might not need all of the Singletons that you create for a given program execution. This could have performance implications. A more elaborate solution is to create a registry of all of the program's Singleton classes and make the registry generally available. Each time a Singleton is instantiated, it notes that in the registry. Then any part of the program can ask for the instance of any Singleton using an identifying string and get back that instance variable.

The disadvantage of the registry approach is that type checking might be reduced, since the table of Singletons in the registry probably keeps all of the Singletons as objects, for example in a Hashtable object. And, of course, the registry itself is probably a Singleton and must be passed to all parts of the program using the constructor of various set functions.

Probably the most common way to provide a global point of access is by using static methods of a class. The class name is always available, and the static methods can be called only from the class and not from its instances, so there is never more than one such instance no matter how many places in your program