**Instantiated**

As soon as the `ServerSocket` constructor completes, the `ServerSocket` is in the Instantiated state. In this state you should really only call `ServerSocket.accept` to begin listening for a client to connect. You can call the `getLocalPort` and `getInetAddress` methods in this state, but they return useless values.

**Accepting**

Once you call the `ServerSocket.accept` method, the `ServerSocket` starts waiting for a client to connect. This method is synchronous: it either waits forever for a connection or waits for the amount of time you specified in the `ServerSocket` constructor.

**Connected**

As soon as `ServerSocket.accept` completes, you are provided with a `Socket` instance that is connected to the remote client. In this state you can use the `getInetAddress` method to obtain the address of the remote client, and the `getLocalPort` method to obtain the local port that the new `Socket` occupies. From this state you can call `ServerSocket.accept` again to obtain additional clients, or you can call close to shutdown the `ServerSocket`. Note that the `Sockets` returned by `ServerSocket.accept` need to be shut down independent of the `ServerSocket`.

To summarize, to create a server using `ServerSocket`, you do something similar to this:

```java
ServerSocket servSock = new ServerSocket(portNumber);
Socket newConnection = servSock.accept();
```

That’s really all there is to it. Of course, to create a well-behaved server, you need to wrap these calls in the right error handling code and thread environment. In the sections that follow we’ll show in detail how to create a sophisticated Web server.

**Creating a Simple Synchronous Web Server**

In this section we’ll look at a simple application of the `ServerSocket`: a simple blocking (synchronous) HTTP server. You should be able to connect to this server using a Web browser.

This example uses a utility class: the `HttpTransactionHandler`. The implementation of this class is covered in detail in “An HTTP Transaction Handler” beginning on page 291.