the system throws the PortInUseException, as if ports 3 and 4 were assigned to the system and in use by it. However, the *getCurrentOwner* method returns the message "Port currently not owned," which means it really didn't exist in the first place. When we click on COM5, we get the NoSuchPortException. When we again click on COM2, we find that it is now in use as well, and by this program.

You might ask whether each port is a Singleton. There can be many instances of a port, but you can have only one open at a time. This certainly is providing a single global point of access and making sure that only one can be used at a time. Thus we can probably view the opened ports as Singletons as well.

Building a CommPortManager

Let's consider constructing an enclosing class for these Singleton methods and calling it a PortManager. We expect the class to allow us to enumerate the available ports and open any of them. Some of the methods in such a class might be as follows:

```
public abstract class PortManager {
public static Enumeration  getAllPorts();
public static Enumeration  getAvailablePorts();
public static CommPort     openPort(String portName);
public static CommPort     openPort();
}
```

Because of the Java syntax rules, we can't have abstract static classes, so we actually declare each of them in the parent abstract class to return null.

The *getAllPorts* method is just like the one shown previously, when we enumerated the ports. However, the *getAvailablePorts* method is interesting because we don't need to do anything at all when we catch the port exceptions. We get the port and try to open it. If neither operation fails, we add the port name to the list of available ports.

```
public static Enumeration getAvailablePorts() {
    Vector portOpenList = new Vector();
    CommPortIdentifier portId;

    Enumeration enum = getAllPorts();
    while (enum.hasMoreElements ()) {
        String portName = (String) enum.nextElement () ;
        try {
            //try to get port ownership
            portId =
                CommPortIdentifier.getPortIdentifier(portName);
            //if successful, open the port
```