returns COM1, COM2, COM3, and COM4, regardless of whether cards for all of these ports are installed.

The only way you can make sure that a port is available is only by attempting to open it. If the port doesn't exist at all, the *open* method throws the NoSuchPortException; if the port is in use by another program, it throws the PortInUseException. The code for handling this looks like the following:

```
//try to get port ownership
portId = CommPortIdentifier.getPortIdentifier(portName);

//if successful, open the port
    Commport cp = portId.open("SimpleComm",100);
//report success
    status.add("Port opened: "+portName);
}
catch(NoSuchPortException e){
    status.add("No such port:"+portName);
}
catch(PortInUseException e) {
    status.add (portName+" in use by: " +
                portId.getCurrent Owner());
}
```

Note that we are sending the error messages to some sort of status display. The important thing is that the CommPort *cp* is non-null only if no exceptions are thrown. We illustrate the use of these ports in the SimpleComm program shown in Figure 6.1.

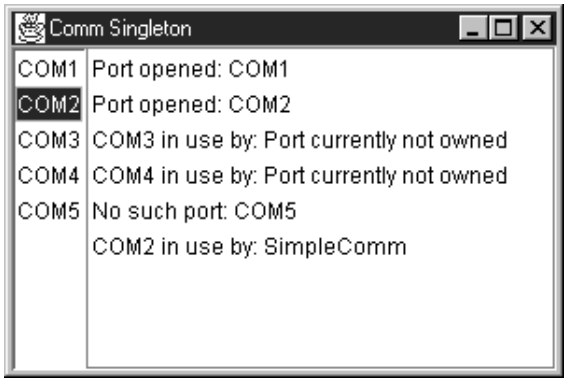

Figure 6.1 The SimpleComm program executing, showing how it represents ports that do not exist and those that are not owned.

An enumeration of the four legal ports that the PC BIOS recognizes is shown on the left. In addition, we added the bogus COM5 port to show the error message it generates. Clicking on each port in turn opens that port and assigns it to the SimpleComm program. When we click on COM3 and COM4,