Exercises: Network Servers

Note: again, there is no need to use my NetworkServer class for these exercises; you could easily write everything from scratch. Using NetworkServer will just save you a few lines of code. If you do use it, you would need to copy NetworkServer.java and SocketUtil.java into your Eclipse project, then do something like this:

```
public class MyServer extends NetworkServer {
  public MyServer(int port) {
    super(port);
  }

  @Override

  public void handleConnection(Socket s) throws IOException {
    // This is the main code you need to write
  }
}
```

Then, your driver routine (the one that has main) would do:

```
MyServer server = new MyServer(some-port);
server.listen(); // Start listening for incoming connections
```

1. Make a server that sends a random number to anyone that connects to it. Connect to it from a Web browser. Also connect to it using telnet. (Open a DOS window and type "telnet localhost *portnumber*". Kill telnet by hitting Control-] and then entering "Quit". Remember that recent versions of Windows have telnet disabled by default: Google "enable telnet windows *x*" to see how to turn it on.)

2. Make a Java client program that connects to it and prints out the number that is sent. (Hint: you may have already written this in your solutions to the networking clients exercises.)

3. Make a server that takes two numbers on separate lines, adds them together, and returns the sum. (Recall that Double.parseDouble will turn a String into a double). Test with telnet, but note that Windows users will have to type "blind" because the Windows telnet client does not echo the characters you type by default. Linux and MacOS telnet clients do echo properly, but even on Windows, telnet is a useful way to test.

4. Make a client that takes a host, port, and two numbers from the command line, uses the addition server to sum the two numbers, and prints the result.

5. Make a multithreaded version of your SumServer. Your networking code will be the same in both cases, but where you put it will be very slightly different. Put it in the handleConnection method of a subclass of MultithreadServer instead of in a subclass of NetworkServer. There are examples of doing this in the lecture notes. If you have #3 running already, this is *very* easy. Verify that it handles multiple connections by telneting to it from one window, leaving the connection open, then connecting to it from a second window and sending the right data there.