Computer Networks Lab 9a

A Message Server and Client

Purpose:

To introduce the Socket class and the ServerSocket class in some simple Java programs.

Overview:

One of the Java classes for connecting a client application to a port on a remote computer is the Socket class. Read about it in JavaDocs at:
http://java.sun.com/j2se/1.5.0/docs/api/java/net/Socket.html.

On the server side of things, the Java class to listen to a port and respond to requests is the ServerSocket class. Read about it in JavaDocs at:
http://java.sun.com/j2se/1.5.0/docs/api/java/net/ServerSocket.html.

Procedures:

1.Read about Socket and ServerSocket in JavaDocs.

2.Open Eclipse. Start a new Project called Lab9 and a package called lab9.

3.Create a class with a *main* method, and call it SocketTest1. Type in the following program, or adapt it to create your own program that opens a Socket to a time-of-day server. Run the program to verify that it connects properly.

```
package lab9;
import java.io.*;
import java.net.*;
import java.util.*;

public class SocketTest1  {
        /**
         * Open a socket connection to the NIST Daytime server
         * in Boulder Colorado and print out the text that the server sends.
         */
        public static void main(String[] args) throws IOException {
                try {
                        Socket sock = new Socket("time-A.timefreq.bldrdoc.gov", 13);
                        try {
                                InputStream inStream = sock.getInputStream();
                                inStream = new BufferedInputStream(inStream);
                                StringBuffer time = new StringBuffer();

                                 int c;

                                while ((c = inStream.read()) != -1) time.append((char) c);

                                String timeString = time.toString();
```

```
                        System.out.println(timeString);
                }
                finally {
                        sock.close();
                }
        }
        catch (IOException ioe)  {
                ioe.printStackTrace();
        }
    }
}
```

1.Next, you will create your own time of day server, using the ServerSocket class. Create a separate class with a main method called MyTimeServer.

2.Pick out a port number in the range of 10001-19999 as the port for your server and use that port when constructing the ServerSocket.

3.Add a while block to run for a count of 3 times. [Note: This seems easier than running forever because it is difficult to stop the server process from within Eclipse.] You can modify the count during debugging.

4.Here is some help you get your server running. Inside the while loop put code such as this:

```
Socket client = serverSocket.accept();
Writer out = new OutputStreamWriter(client.getOutputStream(), "UTF-8");
out.write(new java.util.Date().toString());  // Date string gets written to client
pout.close();
client.close();
```

5.Add exception handling where needed.

6.Run your time server and try using telnet to connect to the port, for example:

```
telnet 127.0.0.1  10001   // use your port number
```

You may have to repeat three times to cause the server to exit.

7.Now modify your client, SocketTest, to connect to your server.

8.Optional:

1.Improve the quality of information returned by your server by sending back multiple lines of text including a message-of-the-day or inspiring quote.

2.Improve the client by putting it in a user interface that asks the user for a host name or number and a port number. Display the results in the user interface.

3.Move MyTimerServer to a neighboring computer and run it from there. Then test it from your computer using the client.

9. Submit your code to the class directory.