Objective:

In this lab you learn how to create server sockets.

Background:

Up to this point we have used sockets to allow a client program to connect to a remote server. To write a server using sockets in Java one needs to create a ServerSocket object. One specifies the port to be used when creating the ServerSocket, then one listens for connections on the ServerSocket. When a connection is made a Socket is returned. The Socket can then be used to send data to and receive data from the client.

The life cycle of a server is:
1. A new ServerSocket is created on a specified port using a ServerSocket() constructor.
2. The ServerSocket listens for incoming connection attempts on that port using its accept() method. accept() blocks until a client attempts to make a connection, at which point accept() returns the Socket object that connects the client to the server.
3. The Socket’s getInputStream() method and getOutputStream() methods are used to get input and output streams to communicate with the client. (As before these streams should be converted to BufferedReader’s and PrintWriter’s to allow for more efficient line-oriented reading and writing.)
4. The server and the client interact according to an agreed upon protocol until it is time to close the connection.
5. The server, the client, or both, close the connection.
6. The server (usually) returns to step 2 and waits for the next connection.

This is the life cycle of an iterative (non-concurrent) server. That is, the server handles connections sequentially, one at a time. In a future lab we will look at concurrent servers.

Instructions:

1. Obtain a copy of the program SimpleServer.java. The SimpleServer opens a ServerSocket on port 4444. It listens for a connection using the accept() method and once the connection is made opens the connection’s output stream. The connection is closed after sending a one-line message to the client.

   Change the message sent by the server to include your name(s). Run the server on your machine. Go to another machine in the lab (or login to the terminal server) and telnet to your first machine on port 4444 to get the message.

   Print a copy of the telnet session and hand it in along with a copy of the modified Java program.
2. Make a series of modifications to SimpleServer as follows. (I recommend making the modifications one at a time and testing after each.)
   a) Open the ServerSocket on port 0. Have the server display the port number on which it is listening right after the ServerSocket has been created. What port number gets displayed in this case? Why is it necessary to display the port number used?
   b) Instead of sending a message to the client, send a time/date string to the client. In this way the server will mimic the daytime service (which is a service on the well-known port 13). To do this you can create a Date object and send it to the client. Note: Make sure to grab the date after the connection is made.
      (Extra credit if you can mimic the format used by the daytime service.)
   c) Add code to the server to display connection information after the connection is made. (You can do this by just placing the identifier for the Socket object in a System.out.println().)

   Test the server by running it on one machine and then accessing it from another machine using telnet. Record and print the results displayed by your server as well as a copy of the telnet session. Turn these in along with a copy of your server program.

3. Modify the DaytimeClient program that you wrote in Lab 2 to connect to your server from step 2. Record whose server you connected to and the results you get.
   Hand in your results.

4. Modify the server code from step 2 to accept any number of connections, however only one at a time. To do this, place the lines of code starting with where the connection to the client is opened to where the connection with the client is closed inside an infinite loop:
   ```java
   while (true) { . . . }
   ```

   Make sure the code to create the ServerSocket is not inside the loop and remove/comment out the statement to close the ServerSocket.

   Further modify the server to display (in the console of the server) the date/time of the connection and the source IP address and port of each connection.

   Now connect to your server multiple times from different lab machines.

   Hand in a copy of your server code and the results that are displayed in the console of the server.

Extra Credit: For extra credit add code to write the server’s log to a file, i.e. the record of date/time of each connection as well as the IP address and source port of each connection.

   Hand in the modified server code and your results.

Help Policy:
Help Policy in Effect for This Assignment: Group Project with Limited Collaboration

In particular, you may discuss the assignment and concepts related to the assignment with the following persons, in addition to an instructor in this course: any member of your group; any St. Bonaventure Computer Science instructor; and any student enrolled in CS 254.

You may use the following materials produced by other students: materials produced by members of your group.