Lab 7: Concurrent Web Server

Purpose:
In this lab you will take the program you wrote for the previous lab and modify it to be a concurrent web server. You will need to use Java threads to do this. The first part of this lab introduces Java threads.

Instructions:
1. Open Eclipse, create a project and add ThreadsTest.java to the project.

   The file ThreadsTest.java defines a class named ThreadsTest with a main method that creates three SimpleThread threads and starts the threads.

   Run the program three times. Do you get the same output each time? Explain.

   Print a copy of an execution of the program. Hand in a copy of the output along with your explanations.

2. Answer the following questions about the Java code in the program you created and ran:
   a) What is the parent class of SimpleThread?
   b) What is the purpose of the statement
      
      super(str);
      
      (in the constructor)?
   c) getName() is a method of what class?
   d) Which method defines the behavior of a thread?
   e) How does one start a thread?
   f) Use the Java API to determine how a thread terminates.

3. Open your iterative web server from Lab 6. For this lab there is no need to maintain the state so you can comment out any code that records the number of visits in a HashMap or code that manages the cookie. Run your web server and make sure you can browse the web page from two other lab machines. Leave code in the server so that the date/time of the connection and the IP address of the client machine appear in the content served to the browser.

   Now we will restructure the code so that whenever the server accepts a connection it spawns a thread to handle the connection. As with the ThreadsTest program in the previous step you will need two classes. I suggest naming them ConcurrentWebServer and ServerThread.

   ConcurrentWebServer will contain the main method of your program. It will be responsible for creating the ServerSocket and publicizing the port on which it is listening. It will then have a (infinite) loop where it waits for a connection and creates and starts a thread once a connection is established. (Hint: Pass the connection to the constructor of the thread.)

   ServerThread is the Thread class with a constructor and a run() method. Include an instance variable to store the connection. The run() method needs to create a BufferedReader and a
PrintWriter using the connection. Read the request from the browser and display it in the server’s console. Then serve the web page, including the date/time and the IP address of the client in the web page. The server should also display in the console the date/time of the connection along with the IP address of the machine the browser request came from. Finally, close everything.

Execute your server. Browse to your server multiple times from at least two different lab machines. Record the output that is displayed in the server’s console and turn this in with your lab write-up.

Make sure your Javadoc commenting is accurate and turn in a copy of all your code.

4. What evidence can you find that the browser requests are handled concurrently and not iteratively? How could you design an experiment that shows that they are handled concurrently?

Hand In:

Hand in a record of the results from Steps 1 and 3. Hand in your answers to the questions in Step 2. Turn in the code from Step 3. Also, email your code from Step 3 to your instructor.