**Program #2**

**15 points; Due Oct. 6 (Monday) at 5pm**

**Program Description**

You are going to write a simple **multi-threaded Server** program to handle requests from the Client program of Program 1. The objective is to learn how to use **ServerSocket** and **Thread** in Java. Your Server program will be “listening” on port **5976** for the connection requests from Clients. Once a connection request is accepted, the server creates a new thread to handle the connection, and keeps listening on port 5976 for additional connection requests. That is, your Server must be able to handle multiple connections simultaneously. The Server is also maintaining a log file that records every connection.

**The Protocol**

All cryptographic algorithms involve substituting one thing for another. Message in original form is called **plaintext**, and encrypted message is known as **ciphertext**. Your server implements an old symmetric key algorithm known as the **Caesar cipher** (a cipher is a method for encrypting data.) to encrypt the plaintext sent from the Clients and sends the ciphertext back to the Clients. Caesar cipher would work by taking each letter in the plaintext message and substituting the letter that is ***k*** letters later (allowing wraparound; that is, having the letter ***z*** followed by the letter ***a***) in the alphabet. For example, if ***k* = 3**, then the letter ***a*** in plaintext becomes ***d*** in ciphertext (and the letter ***A*** becomes ***D***.) The protocol is summarized as follows.

* Let ***k* = 4;**
* Substitute each alphabet letter (either lowercase or uppercase) in the message by the letter that is ***k*** letters later, allowing wraparound; i.e., substitute ***a*** with ***e***, ***b*** with ***f***, ***z*** with ***d***, ***A*** with ***E***, ***B*** with ***F***, etc.
* No substitutions on all non-alphabet characters;
* If the message received is “**quit**”, send “**Good Bye!!**” back to the Client, and close the connection (socket.)

**Program Requirement**

1. Your Server must not crash under any situation, and must work with your Client program from Program 1.
2. You must try-catch everything and print out appropriate error messages identifying the specific exception.
3. You MUST follow the software development ground rules from CS2430 (posted on D2L.)
4. Name the project **<your\_login\_name>\_prog2**. Copy and paste your project folder to S:\Courses\CSSE\changl\cs3830\1dropBox by the due date. **2 points off for every hour late.**
5. You MUST DEMO your program by the due date. Once you demoed, the grade is final. Therefore, make sure you thoroughly test your program before you demo. Demo schedule will be announced on D2L.
6. You should have TWO classes in your Server program: **Server** and **ServerThread**.

* **Server** class should have exactly two public methods: main() and run() (you may have other private methods). The main() method of Server should simply create a new **Server** object and call the run() method on it. In the **run() method**, do the following:
  1. Create a new **ServerSocket** listening on port **5976** (refer to the tutorial on D2L.)
  2. Create a new PrintWriter object with a new FileOutputStream that writes to a log file called **prog2.log**, which should be stored in your project folder. Make sure the second argument to the PrintWriter constructor is **true**. Every ServerThread will write connection information to the log file. Each time your Server comes up, you can either overwrite or append to **prog2.log**.
  3. Have an infinite while loop that
     1. Listens to connection requests using .**accept()** method, and once a connection is accepted, create a new instance of Socket to handle the connection.
     2. Creates a new instance of ServerThread to handle the connection. When creating a new ServerThread object, use the socket object and the log file as arguments. So each ServerThread can communicate with the Client socket and write to the log file.
     3. Calls the start() method of the ServerThread object to start the thread.
* **ServerThread** class should extend the **Thread** class and should NOT be designated as public.
  1. Declare one Socket object, two PrintWriters (one for the socket I/O, one for the log file) object, and one BufferedReader object, outside of all methods. So they are visible in ServerThread class.
  2. Define a constructor with the interface:

**public ServerThread (Socket** clientSock**, PrintWriter** logfile**)**

* 1. SinceServerThread extends Thread, it MUST have a run() method, which is the method that eventually gets called after you call start() on a Thread. The **run() method** should do the following:
     1. Write a line to the log file indicating that a connection was received. You should write the *date/time* the connection was received, the remote *IP address* and the *remote port* of the connection. You may find the following methods useful for your log file: Date().toString(), getInetAddress() and getPort() of the Socket class.
     2. Have a while loop that implements **Caesar Cipher protocol** described above. That is, your server encrypts the message and sends the ciphertext back to the Client. You should loop until a “**quit**” message is received.
     3. When the Client disconnects (quits), you should print an appropriate message to the log file saying the connection on some port number is closed. DO NOT close the PrintWriter for the log file. Just close the Client socket and return from the run() method. Doing so effectively kills the ServerThread.
     4. All output from ServerThread must go to the log file. No output may be sent to System.out.

1. Your Server.java file should look something like this:

import java.io.\*;

import java.net.\*;

import java.util.\*;

class **Server**

{

public static void **main**(String argv[])

{

...

}

public void **run()**

{

...

}

}

class **ServerThread** extends **Thread**

{

...

public **ServerThread** (**Socket** clientSock, **PrintWriter** logfile) //constructor

{

...

}

public void **run()**

{

...

}

}

1. Log file entries may look something like these:

Got a connection: Fri Sep 05 13:40:16 CDT 2014 /137.104.121.250 Port: 9312

Connection closed. Port: 9312

Got a connection: Mon Sep 15 13:32:41 CDT 2014 /137.104.121.234 Port: 11132

ServerThread Exception: java.net.SocketException: Connection reset

1. You can run and test your Server and Client on one machine. To do so, bring up your Server first, and run your Client with “**localhost**” (**127.0.0.1**) on port **5976**. When your server program is working, run and test your Server and Client with different machines. You could use “**ipconfig**” command at the command prompt under Windows to find out the IP address of the machine running the Server program, and run your Client program with that IP address to connect to your Server.

**Program Grading**

|  |  |  |
| --- | --- | --- |
| **Exceptions/Violations** | **Each Offense** | **Max Off** |
| Program not running | 15 | 15 |
| Cannot handle multiple connections | 15 | 15 |
| Missing the log file | 3 | 3 |
| Improper implementation on the Protocol | 1 | 5 |
| Improper handling try/catch exceptions | 0.5 | 2 |
| Not recording connections properly in the log file | 0.5 | 2 |
| Not complying CS2430 programming ground rules | 0.5 | 2 |