Zane Wonsey
2-3-15

88/100

zw_Readme

1. The project consists of a client and server program.
   The client is to be able to connect to the server and send it messages.
   At the end of the session the server sends a count of the messages
   plus their content then the total duration of the connection.
2. To run the programs on a unix machine you first compile them using
   > javac zw_TCPClient.java
   followed by
   > javac zw_TCPServer.java
   After they are compiled you start the server first and then run the client.
   (the commands in [] are optional and can be in any order)
   > java zw_TCPServer [-p [port]]
   > java zw_TCPClient [-p [port]] [-h [host]] [-u [username]]
   To end the client session type DONE then press enter. To turn off the server
   press ctrl+c
3. Server has to be stopped with ctrl+c. I did not notice any other bugs.
4. I spent about 10 hours as a whole working on the project. Most of the
   time was spent refreshing myself with Java as I have not programmed
   in Java in roughly a year.

- tar file is not named "zw.tar". It is called xy.tar!! −2
- client doesn't prompt for user name when it is not given at the command line. −3
- Server doesn't accept '-p' option for the port number. −3
- can't see the contents of the chat file −4
- client echos the command line options

-Name !
-Description !

```
// Server program
// File name: "TCPServer.java"

import java.io.*;
import java.net.*;
import java.util.Scanner;
public class zw_TCPServer {
        private static ServerSocket servSock;
        public static void main(String[] args) {
                System.out.println("Opening port...\n");
                try {
                        // Create a server object
                        servSock = new ServerSocket(Integer.parseInt(args[0]));
                } catch (ArrayIndexOutOfBoundsException e) {
                        try {
                                System.out.println("Attaching to default port
22394");
                                servSock = new ServerSocket(22394);
                        } catch (IOException e1) {
                                System.out.println("Unable to attach to port!");
                                System.exit(1);
                        }
                } catch(IOException e){
                        System.out.println("Unable to attach to port!");
                        System.exit(1);
                }

                do {
                        run();
                } while (true);

        }

        private static void run() {
                Socket link = null;
                try {
                        // Put the server into a waiting state
                        link = servSock.accept();

                        long durationStart = System.currentTimeMillis();
                        System.out.println(durationStart);

                        // Set up input and output streams for socket
                        BufferedReader in = new BufferedReader(new
InputStreamReader(link.getInputStream()));
                        PrintWriter out = new
PrintWriter(link.getOutputStream(),true);

                        // print local host name
                        String host = InetAddress.getLocalHost().getHostName();
                        System.out.println("Client has estabished a connection to "
+ host);

                        File file = new File("zw_chat.txt");
                        PrintWriter writer = new PrintWriter(file);

                        // Receive and process the incoming data
                        int numMessages = 0;
                        String username = in.readLine();
                        String message = in.readLine();

                        while (!message.equals("DONE")) {
                                System.out.println(username +": " + message);
```

?? Not your assigned port !

Didn't check for command line argument!!

C

C

, true

this will automatically flushes buffer after each write.

or you can flush buffer here!

writer.flush();

```
                    writer.println(username + " " + message);
                    numMessages++;
                    message = in.readLine();
                }
                writer.close();
                // Send a report back and close the connection
                // The report includes duration of the session and number of
messages
                // sent to the server
                long durationEnd = System.currentTimeMillis() -
durationStart;
                out.println("Server received " + numMessages + " messages");
                System.out.println("SERVER: session duration in
milliseconds: " + durationEnd);
                Scanner sc = new Scanner(file);
                while (sc.hasNextLine()) {
                    out.println(sc.nextLine());
                }
                int ms = (int) (durationEnd % 1000);
                int s = (int) (durationEnd / 1000);
                int min = 0;
                int hr = 0;
                if (s > 60) {
                    if (s == 60) {
                        min = 1;
                    } else {
                        min = s / 60;
                        s = s % 60;
                        if (min == 60) {
                            hr = 1;
                        } else {
                            hr = min / 60;
                            min = min % 60;
                        }
                    }
                }
                String finalMessage = hr+"::"+min+"::"+s+"::"+ms;
                out.println(finalMessage);
                sc.close();
                file.delete();
            } catch(IOException e){
                e.printStackTrace();
            } finally {
                try {
                    System.out.println("!!!!! Closing connection...
!!!!!\n" +"!!! Waiting for the next connection... !!!");
                    link.close();
                } catch(IOException e){
                    System.out.println("Unable to disconnect!");
                    System.exit(1);
                }
            }
        }
    }
```

Name
Description

```
// Client program
// File name: TCPClient.java

import java.io.*;
import java.net.*;

public class zw_TCPClient {

        private static InetAddress host = null;

        public static void main(String[] args) {
                if (args != null){
                        String username = null;
                        int portNumber = -1;

                        try {
                                int i = 0;
                                while (true) {
                                        try {
                                                System.out.println(args[i]);
                                                if (args[i].charAt(1) == 'u') {
                                                        username = args[i+1];

System.out.println(username);
                                                } else if (args[i].charAt(1) == 'h')
{
                                                        host =
InetAddress.getByName(args[i+1]);
                                                        System.out.println(host);
                                                } else if (args[i].charAt(1) == 'p')
{
                                                        portNumber =
Integer.parseInt(args[i+1]);
System.out.println(portNumber);
                                                } else {
                                                        System.out.println("invalid
input");
                                                }
                                                i = i + 2;
                                        } catch (ArrayIndexOutOfBoundsException e) {
                                                break;
                                        }
                                }
                                //if the user did not specify a host assume
localhost as default
                                if (host == null) {
                                        host = InetAddress.getByName("localhost");
                                }
                                //if the user does not specify a port use the
default port
                                if (portNumber == -1) {
                                        portNumber = 22394;
                                }
                                //if the user does not specify a name use 'default'
for name
                                if (username == null) {
                                        username = "default"; X
                                }
```

Must ask the user to enter a user name

```
                } catch (UnknownHostException e) {
                        System.out.println("Host ID not found!");
                        System.exit(1);
                }

                System.out.println("Calling RUN");
                run(portNumber, username);

        }

    }

    private static void run(int port, String username) {

        Socket link = null;

        try {
                // Establish a connection to the server
                link = new Socket(host,port);

                // Set up input and output streams for the connection
                BufferedReader in = new BufferedReader(
                                        new
InputStreamReader(link.getInputStream()));
                PrintWriter out = new PrintWriter(
                                        link.getOutputStream(),true);

                //Set up stream for keyboard entry
                BufferedReader userEntry = new BufferedReader(new
InputStreamReader(System.in));
                String message, response;

                // Get data from the user and send it to the server
                out.println(username); ✓
                do {
                        System.out.print("Enter message: ");
                        message = userEntry.readLine();
                        out.println(message);          ✓
                } while (!message.equals("DONE"));

                // Receive the final report and close the connection
                //response = in.readLine();
                //System.out.println(response); //Time
                response = in.readLine();
                System.out.println(response);   //Number of messages
                while (true) {
                        response = in.readLine();  ok    ✓
                        if (response != null) {
                                System.out.println(response);
                        } ~~else {~~
                                ~~break~~;  don't need an else!
                        }
                }
                response = in.readLine();
                if (response != null) {
                        System.out.println(response);    ✓
                }
        } catch(IOException e) {
                e.printStackTrace();
        } finally {
                try {
                        System.out.println("\n!!!!! Closing connection...
 !!!!!");
```

```
                                          1
                                link.close();
                        } catch(IOException e) {
                                System.out.println("Unable to disconnect!");
                                System.exit(1);
                        }

                }

        }

}
```