Maria Ali, Summer Alrayyashi, Xavier Begerow, and Sara Dokter

Professor Baugh

CIS 427-001

Due April 18,2022

**GITHUB LINK:** <https://github.com/XavusZookie/project3-cis-427>

**Project 3 README**

## **Instructions to Build and Run:**

## Step One: Start the server class.

## Step Two: Start the client class.

## Step Three: Have the client login to the server. Note, multiple clients are able to login to the server by running a new client class.

## Step Four: Perform any of the following functions within the program: solve, list or message other users on the system with either client class.

## Step Five: The client will be able to message any user using the command Message [name] [message]. This will then be sent to the other user unless they are not logged in. If they are not logged in, there is a message sent to the sender that they are not online.

## Step Six: At any time, any client will be able to log out of the server using the logout function

## Step Six: After all the clients are finished with the server, the last client will shutdown the server, which will end the program.

**Any Known Bugs:**

We had a bug where the login file wasn’t included in the project, so the system would crash on login. To solve this issue, we changed the pathing of the file search, then added the file to the local folder, this solved the issue because the file could now be found.

**Program Commands:**

**Main:** Creates the login file with valid username, password combos, and the files for each username's solutions, calls the create communication loop while the server still wants to run, and then deletes the files after the server is done being used.

**CreateMultiThreadCommunicationLoop:** Establishes connection with the client and calls the client handler class that interacts with the client and handles their input

**ClientHandler:** client handler initiates the connection with the client

**Run:** run takes the input and sends input to the client and performs all the requested functions

**Login:** Takes the input from the user and the stream path to the client. Login accesses the login text file and compares the clients inputted username and password with the options from the file, it allows access to the rest of the program and returns the username that logged in if they match, otherwise it tells the user if they have invalid credentials or if their input was not valid.

**Solve:** Takes the input from the client and will be able to solve the problem based on the commands given. If the client writes “-r”, the program will find the perimeter and area of the rectangle using the values that the client has entered. If the client writes “-c”, the program will find the circumference and area of the circle using the values that the client has entered. The results of each of the following commands will then be outputted to the client and written into their own solution text file. If the client were to make any errors in regards to entering the necessary information, the server will print out an error message.

**List:** Takes the users input, the stream path to the client and the username that's logged in and opens the file associated with the username and reads the contents of what has been inputted then sends that to the client, the login function also validates if the user is root if they input the modifier -all then the solutions files of all the usernames are accessed and sent to the user

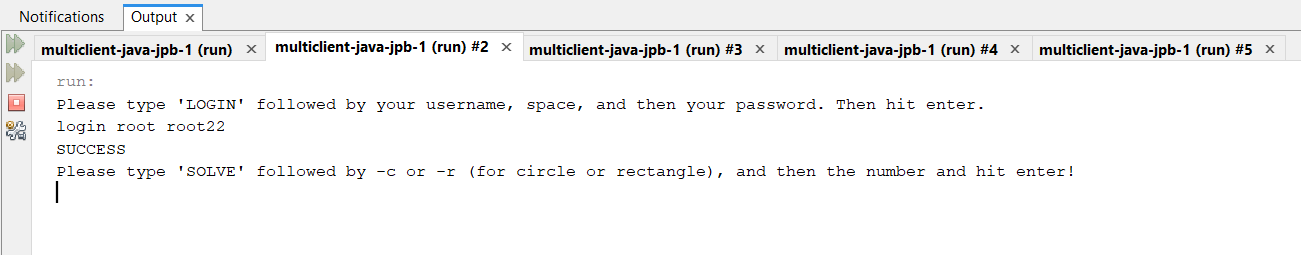
**Message:** Message uses the tag of either the name of the user who you want to send it to or all if you are root. This then checks who’s online, tells you if they aren’t there and sends it to the required person who can then receive the message after their next command. In the case of the all function, if some users aren't online, the message is still sent to the others

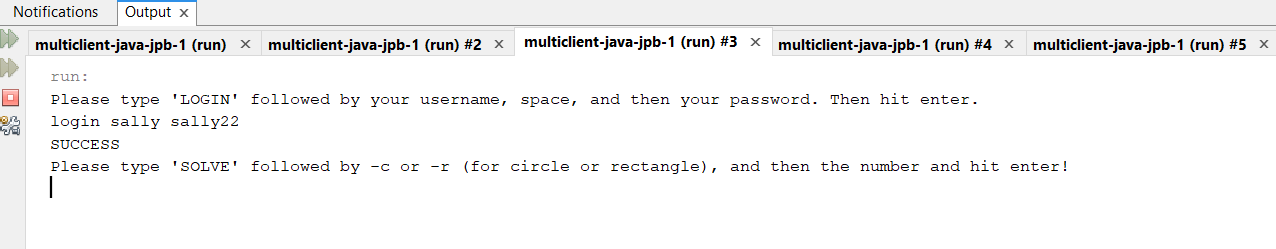
**Logout:** Closes the socket associated with the client and ends the client program, the server stays on and continues its communication loop

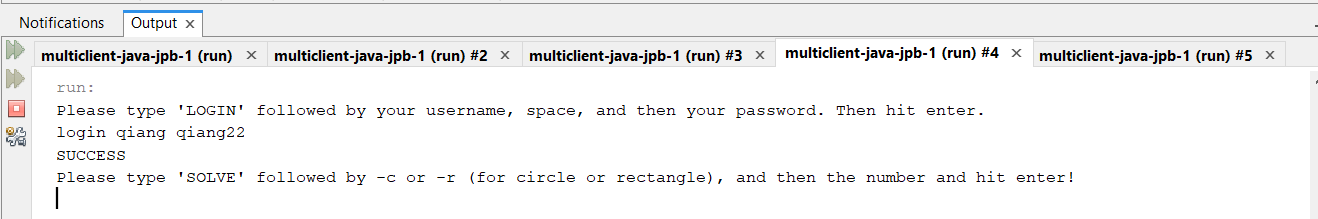
**Shutdown:** First closes the socket associated with the client and then closes the socket associated with the server, both programs then end.

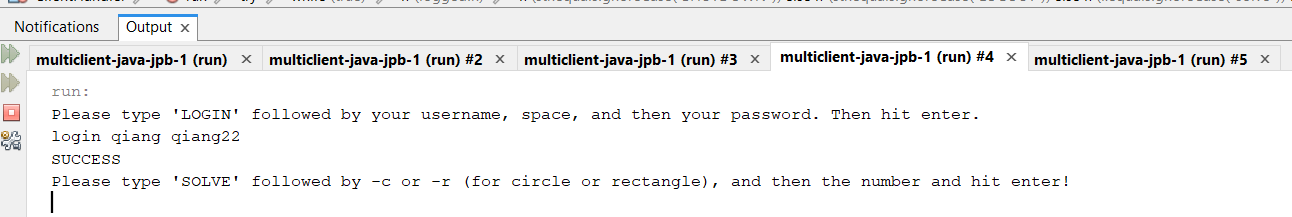
**Output with Screenshots:**

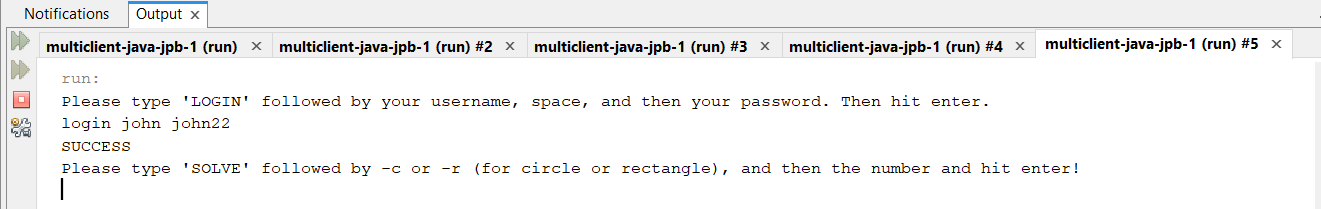
Login:



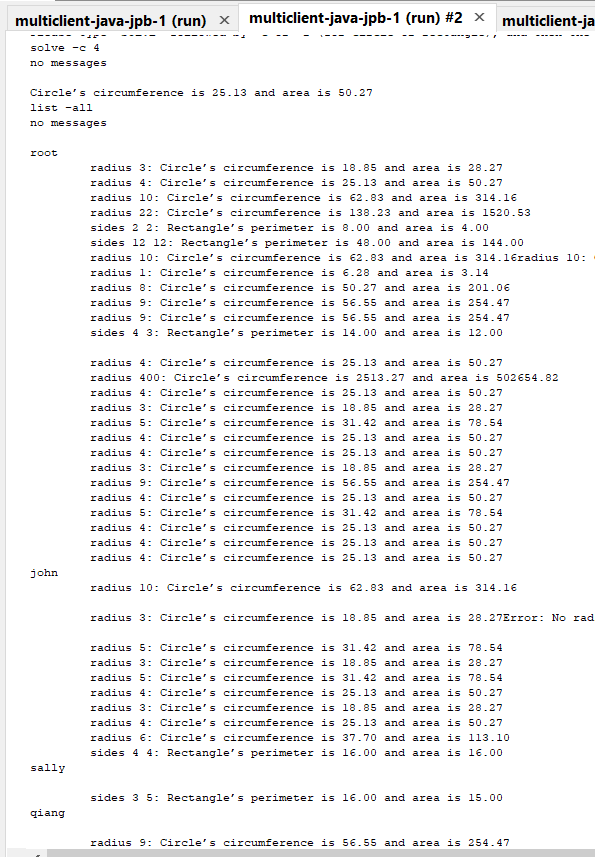


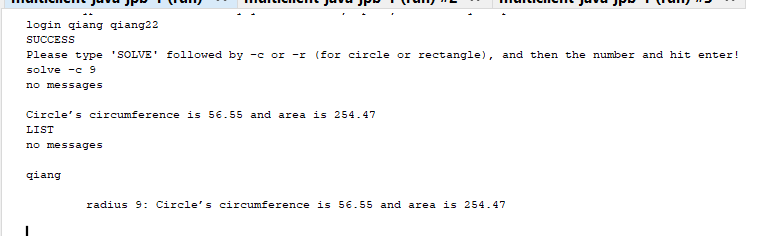




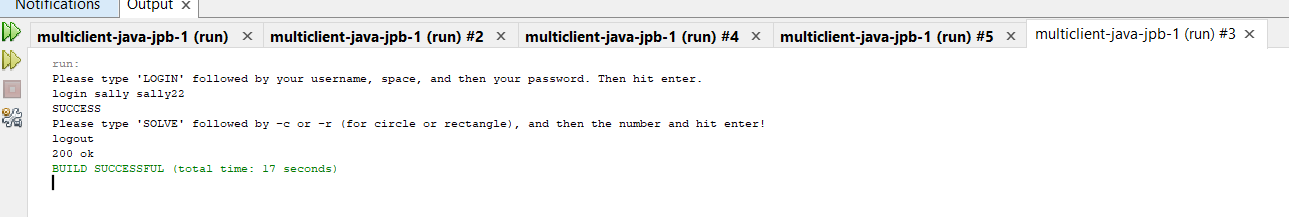
****

List:

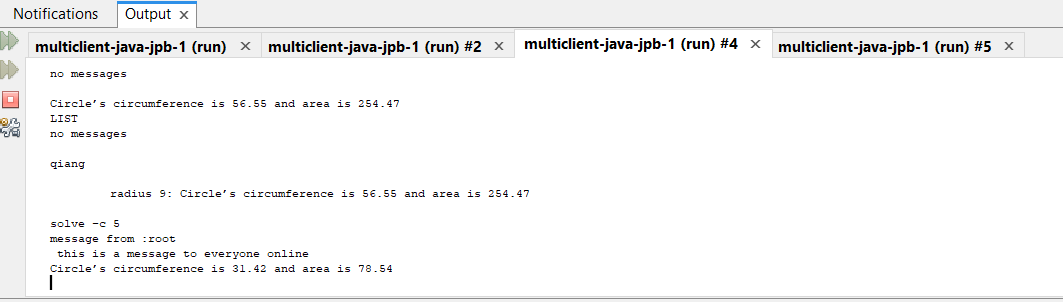
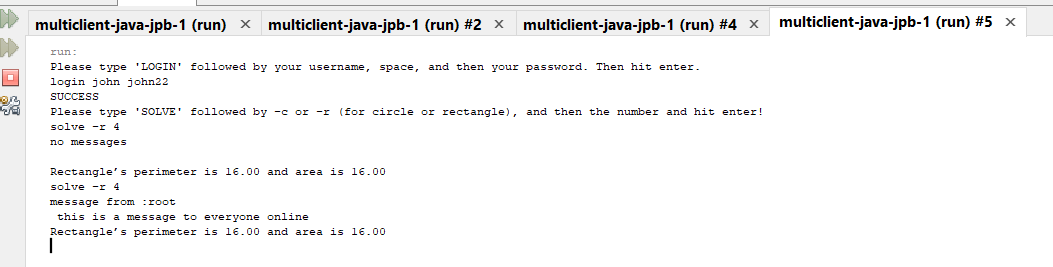
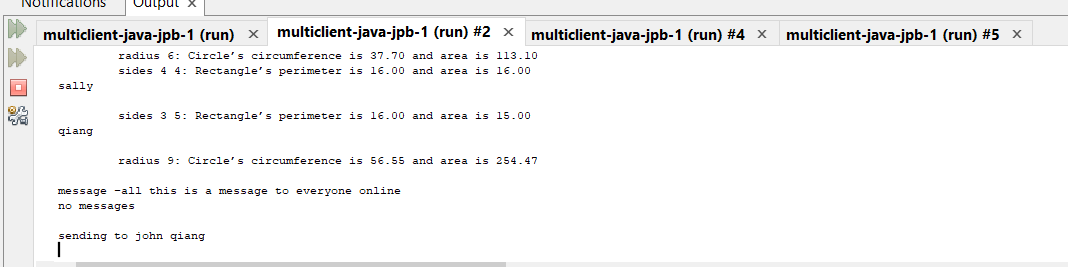


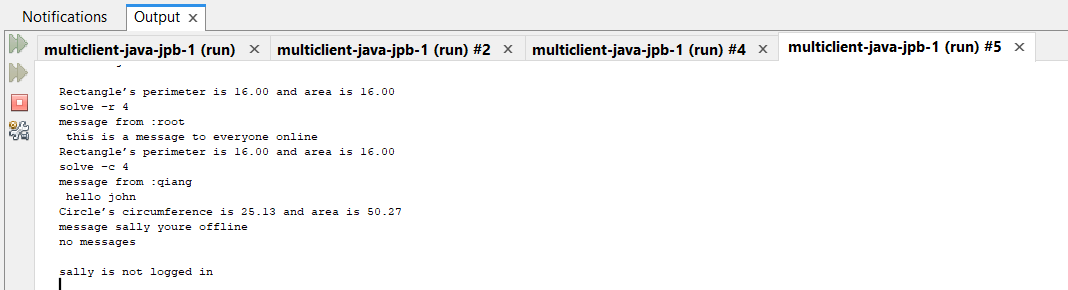
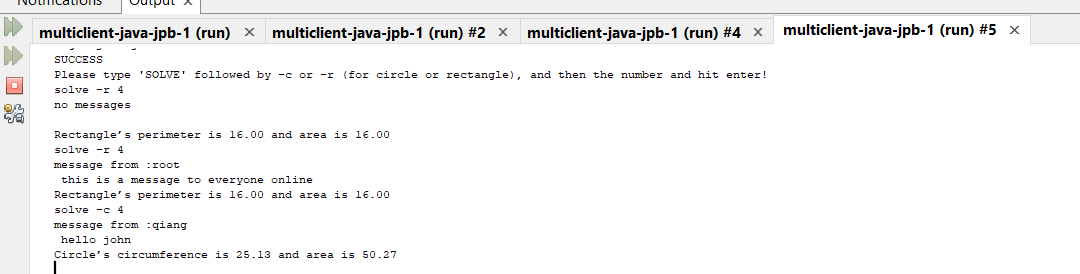
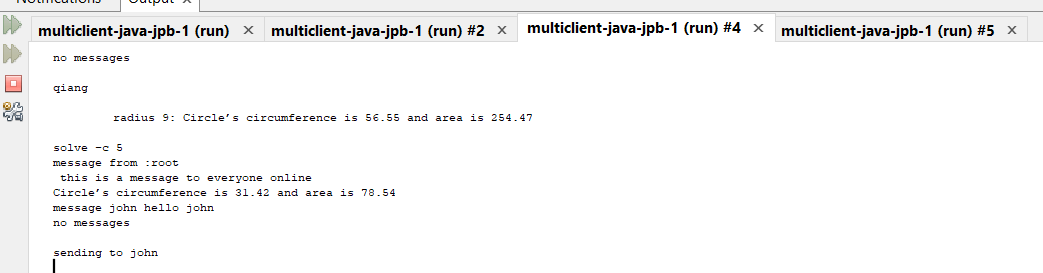
****

Logout:

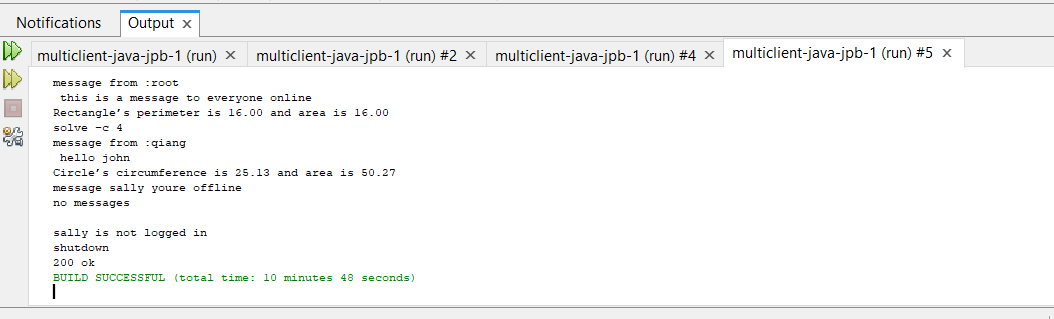
****

Message:

****

****

Shutdown:

****