Program 1a – Knock Knock Client (15 points)

Due Date: Sept. 20 (Friday) 11:59pm; **extra credit**: **5 points** if you submit it by the due date.

Grace Date: Sept. 25 (Wednesday) 11:59pm

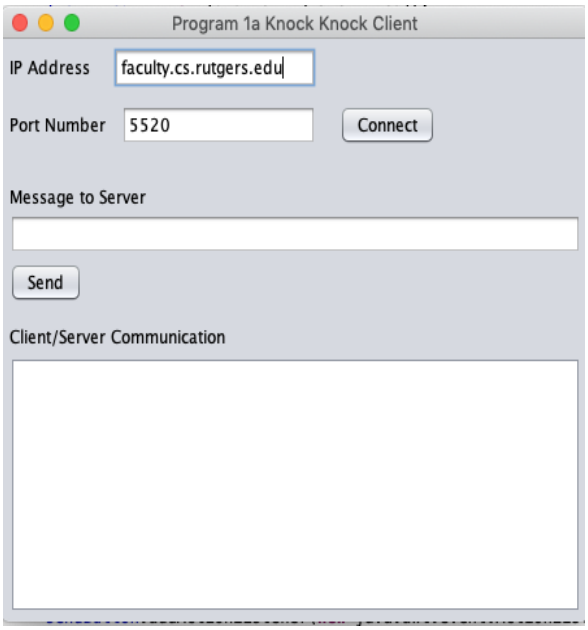
Program Submission: Submit your program as a single zip file to Canvas; click Assignment and Program 1a, you should see a "Submit Assignment" button on the upper right.

Program Description

Write a Java GUI client program that communicates with the server running at **constance.cs.rutgers.edu** with the TCP port **5520**. The server is running a proprietary protocol called **Knock Knock Protocol**. It will only respond when the client is following the protocol. You MUST meet the requirements listed below to get the full credit for this program.

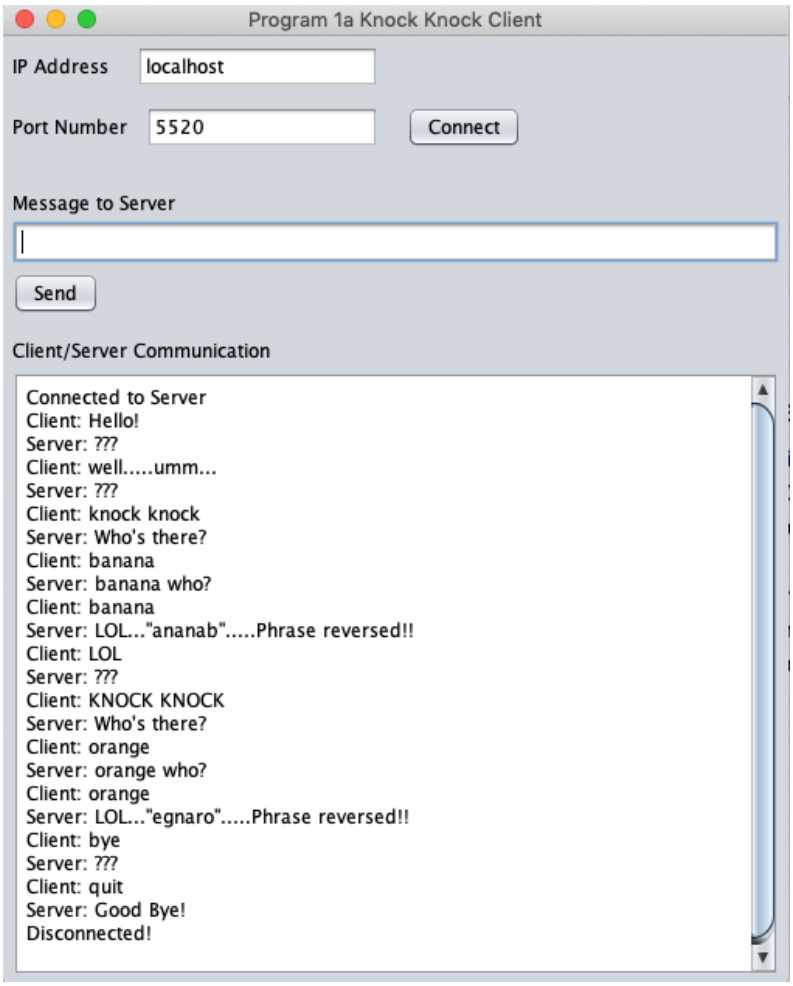
Program Requirement

1. You can use any Java packages to help you complete the GUI design, such as, Java swing JFrame, JavaFX, Eclipse WindowBuilder or Eclipse SWT. You should include the following components on the GUI. A sample GUI is shown on the right. (**Note:** the IP address could be different; check with the instructor/TAs)



   - A JLabel and a JTextField for entering the server address; set default text to **constance.cs.rutgers.edu**
   - A JLabel and a JTextField for entering the port number, set default port number to **5520**.
   - Have a Connect/Disconnect JButton and set the text to "**Connect**". When the connect button is clicked, connect to the server with the port number specified and change the text to "**Disconnect**". When the disconnect button is clicked, close the connection and change the text back to "**Connect**". If the connection dies for some reason, the button must display "**Connect**". In addition, whenever the client is connected to the server, the button must display "**Disconnect**". Make sure your Connect/Disconnect button is in the correct state!
   - A JLabel, JTextField and a **Send button** for sending a message (String) to the Server.
   - A JLabel and a read-only JTextArea for displaying messages. Use the **append()** method of JTextArea to display all messages, including the communications between the client and server, error messages and the connection status (either connected to …., or disconnected…). DO NOT CLEAR the messages in the JTextArea.

2. Knock Knock Protocol
   - First, the client says hello to the server by sending a "**Knock Knock**" message to start the conversation. The message is not case-sensitive on the server side. The server will send "**who's there?**" back to the client, but if you send anything else, the server simply doesn't understand and will send "**???**".
   - Next, the client identifies itself to the server, in response to the message "**who's there?**". For example, "**Banana**". After that, the server will send "**Banana who?**".
   - When the server send "Banana who?" the client can send a Knock Knock joke to the server. The server is very kind, so whatever joke the client sent, she will always say LOL!
   - Because the server is very accommodating, the client can tell her another Knock Knock joke even if the previous one is not funny. In this case, the client must send "Knock Knock" again!
   - The client could send a "**quit**" message to disconnect from the server. The server will then send a "**Good Bye!**" message back to the client. The "quit" message is not case-sensitive.
3. Your program **must not crash** under any situation and must always be in a sane state. You must **try-catch** everything and print out appropriate error messages identifying the specific exception.
4. You must thoroughly test your program. A sample test sequence is provided on Canvas for you to test your program.
5. You MUST submit your program before the grace date or **0 points.**
6. A sample client/server interaction is shown below.



You will get **0 points if your program is **not running** OR **not able to communicate** with the server.

Exceptions/Violations	Each Offense	Max Off
Missing GUI components	1	2
Default Host name / port number	0.5	1
Connect button malfunction	1	3
Incorrect text display on the connect button	0.5	1
Improper handling try/catch exceptions	0.5	2
Improper messages display in the text area	0.5	2

Hint

1. Import Java packages **java.io.*** and **java.net.***.
2. Instantiate 3 objects with Socket, PrintWriter and BufferedReader, so these objects are visible throughout the program.
3. If you are creating a GUI program, code the "Connect" button (connectButtonActionPerformed() method):
   - If the client is currently disconnected (the text of the button is "connect"), instantiate the objects declared in step 3. Note that you need to use **getText()** method to get the host IP address, and the port number in order to instantiate a socket object.
   - If the client is currently connected (the text of the button is "disconnect"), close the socket and I/O objects and set the text of the button to "connect". Note that you MUST display a disconnected message in the text area.
   - Remember to set the correct text for your connect button (either "connect" or "disconnect".)
   - Code the "Send" button (sendButtonActionPerformed() method):
   - If for some reason you got disconnected, you won't be able to send messages to the server. Therefore, you must handle the exception.
   - Display the message you sent in the text area in the following format: "Client: ……", write the message to the socket (printwriterObj.println() method), and wait for the server's response (bufferedreaderObj.readLine() method.)