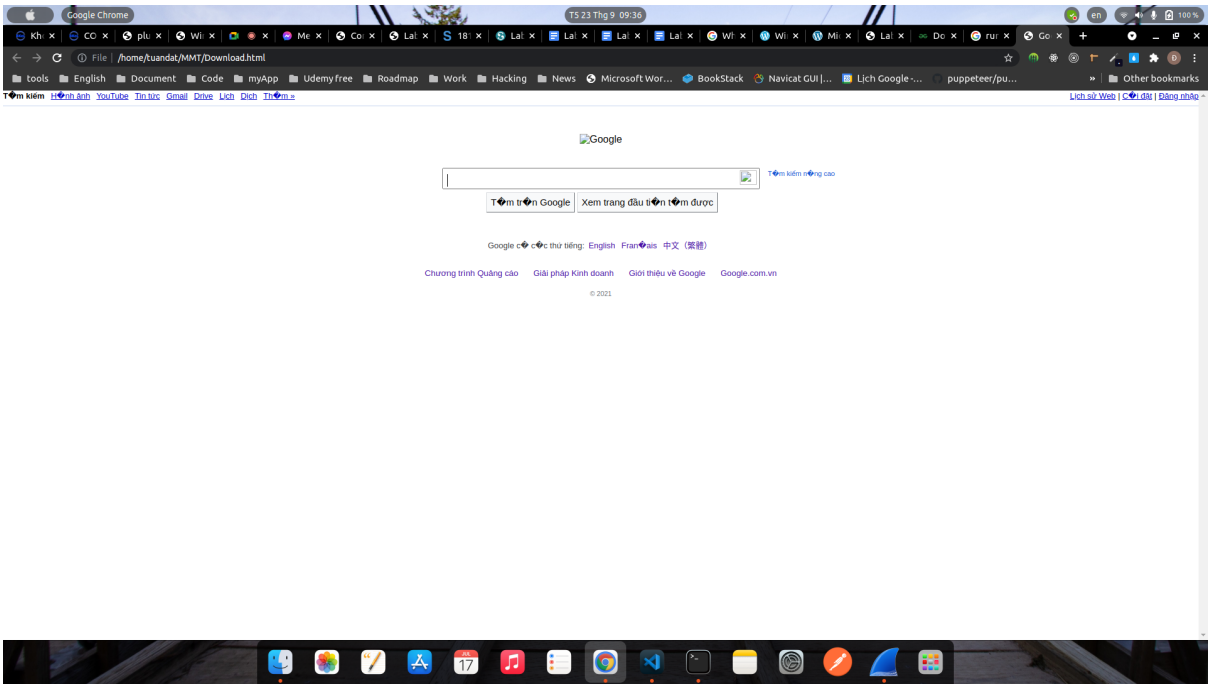
Socket Programming in Java: Chat Application

Exercise 1: Create a program that connects to a web server and downloads the homepage of this website to local computer.

- Code:

```java
package normal;
// Java program to read and download
// webpage in html file
import java.io.*;
import java.net.URL;
import java.net.MalformedURLException;

public class download {

    public static void DownloadWebPage(String webpage)
    {
        try {

            // Create URL object
            URL url = new URL(webpage);
            BufferedReader readr =
              new BufferedReader(new InputStreamReader(url.openStream()));

            // Enter filename in which you want to download
            BufferedWriter writer =
              new BufferedWriter(new FileWriter("Download.html"));

            // read each line from stream till end
            String line;
            while ((line = readr.readLine()) != null) {
                writer.write(line);
            }

            readr.close();
            writer.close();
            System.out.println("Successfully Downloaded.");
        }

        // Exceptions
        catch (MalformedURLException mue) {
            System.out.println("Malformed URL Exception raised");
        }
        catch (IOException ie) {
            System.out.println("IOException raised");
        }
    }
    public static void main(String args[])
        throws IOException
    {
        String url = "https://www.google.com/";
        DownloadWebPage(url);
    }
}
```

- Result:



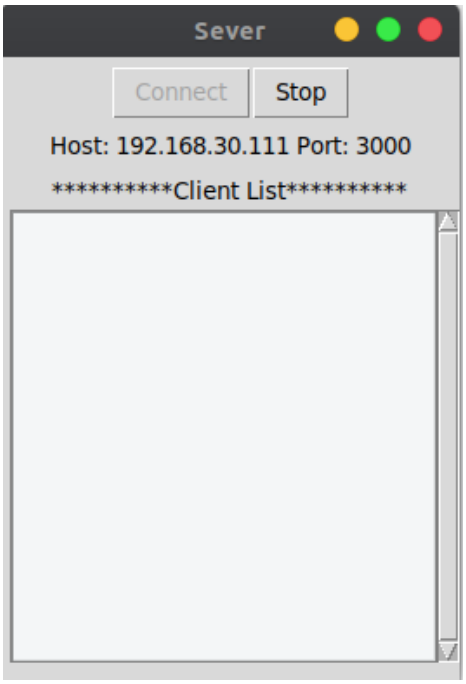
Exercise 2: Design the user interface for the chat application

I will be using the popular Python GUI library: Tkinter for the User Interfaces

Code manager server UI:

```
import tkinter as tk
import socket
import threading


window = tk.Tk()
window.title("Sever")

# Top frame consisting of two buttons widgets (i.e. btnStart, btnStop)
topFrame = tk.Frame(window)
btnStart = tk.Button(topFrame, text="Connect", command=lambda: start_server())
btnStart.pack(side=tk.LEFT)
btnStop = tk.Button(topFrame, text="Stop",
                    command=lambda: stop_server(), state=tk.DISABLED)
btnStop.pack(side=tk.LEFT)
topFrame.pack(side=tk.TOP, pady=(5, 0))

# Middle frame consisting of two labels for displaying the host and port info
middleFrame = tk.Frame(window)
lblHost = tk.Label(middleFrame, text="Host: X.X.X.X")
lblHost.pack(side=tk.LEFT)
lblPort = tk.Label(middleFrame, text="Port:XXXX")
lblPort.pack(side=tk.LEFT)
middleFrame.pack(side=tk.TOP, pady=(5, 0))

# The client frame shows the client area
clientFrame = tk.Frame(window)
lblLine = tk.Label(clientFrame, text="**********Client List**********").pack()
scrollBar = tk.Scrollbar(clientFrame)
scrollBar.pack(side=tk.RIGHT, fill=tk.Y)
tkDisplay = tk.Text(clientFrame, height=15, width=30)
tkDisplay.pack(side=tk.LEFT, fill=tk.Y, padx=(5, 0))
scrollBar.config(command=tkDisplay.yview)
tkDisplay.config(yscrollcommand=scrollBar.set, background="#F4F6F7",
                 highlightbackground="grey", state="disabled")
clientFrame.pack(side=tk.BOTTOM, pady=(5, 10))
```

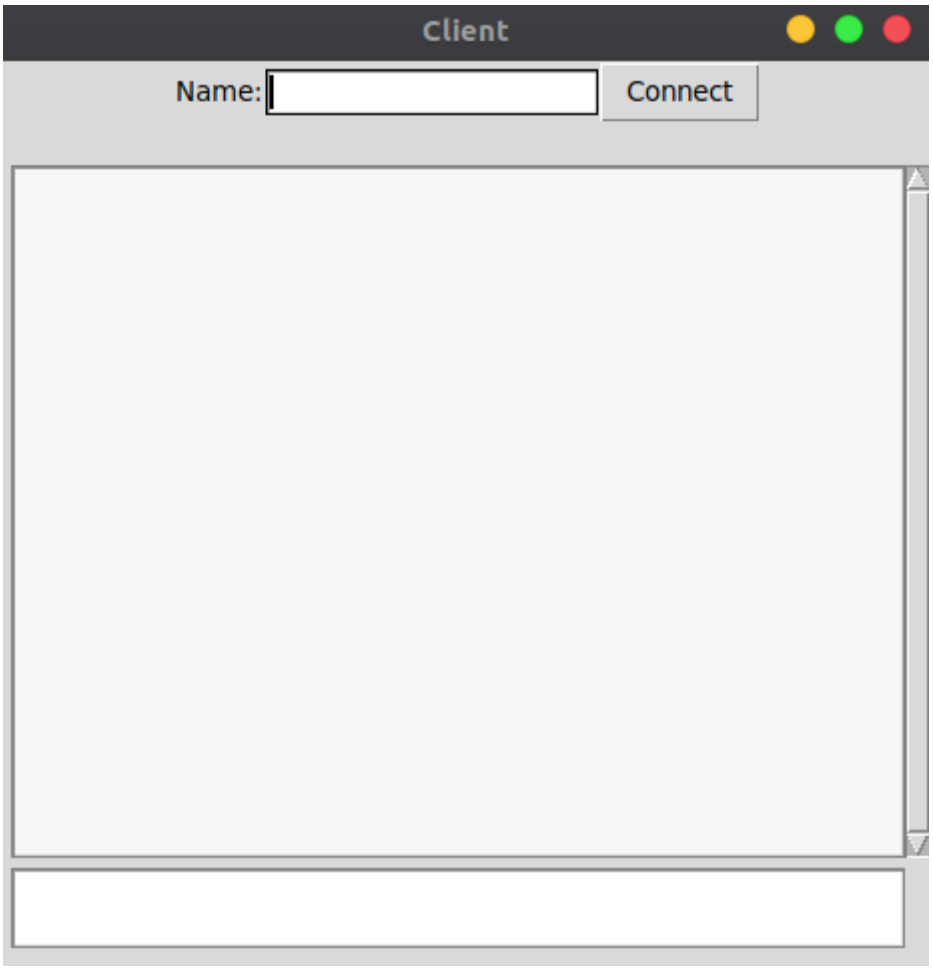
UI of manage server:



Code client UI:

```
import tkinter as tk
import socket
import threading

window = tk.Tk()
window.title("Client")
username = " "


topFrame = tk.Frame(window)
lblName = tk.Label(topFrame, text="Name:").pack(side=tk.LEFT)
entName = tk.Entry(topFrame)
entName.pack(side=tk.LEFT)
btnConnect = tk.Button(topFrame, text="Connect", command=lambda: connect())
btnConnect.pack(side=tk.LEFT)
#btnConnect.bind('<Button-1>', connect)
topFrame.pack(side=tk.TOP)

displayFrame = tk.Frame(window)
lblLine = tk.Label(
    displayFrame, text="").pack()
scrollBar = tk.Scrollbar(displayFrame)
scrollBar.pack(side=tk.RIGHT, fill=tk.Y)
tkDisplay = tk.Text(displayFrame, height=20, width=55)
tkDisplay.pack(side=tk.LEFT, fill=tk.Y, padx=(5, 0))
tkDisplay.tag_config("tag_your_message", foreground="blue")
scrollBar.config(command=tkDisplay.yview)
tkDisplay.config(yscrollcommand=scrollBar.set, background="#F4F6F7",
                 highlightbackground="grey", state="disabled")
displayFrame.pack(side=tk.TOP)


bottomFrame = tk.Frame(window)
tkMessage = tk.Text(bottomFrame, height=2, width=55)
tkMessage.pack(side=tk.LEFT, padx=(5, 13), pady=(5, 10))
tkMessage.config(highlightbackground="grey", state="disabled")
tkMessage.bind(
    "<Return>", (lambda event: getChatMessage(tkMessage.get("1.0", tk.END))))
bottomFrame.pack(side=tk.BOTTOM)
```

UI of client:



Exercise 3: Using multi thread programming model to make the chat application can talk to many different users concurrently.

Code server:

- Setup host and port for socket and logic socket in server

```
39  server = None
40  HOST_ADDR = "192.168.30.111"
41  HOST_PORT = 3000
42  client_name = " "
43  clients = []
44  clients_names = []
45
46
47  # Start server function
48  def start_server():
49      btnStart.config(state=tk.DISABLED)
50      btnStop.config(state=tk.NORMAL)
51
52      server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
53      print(socket.AF_INET)
54      print(socket.SOCK_STREAM)
55
56      server.bind((HOST_ADDR, HOST_PORT))
57      server.listen()  # server is listening for client connection
58
59      threading._start_new_thread(accept_clients, (server, " "))
60
61      lblHost["text"] = "Host: " + HOST_ADDR
62      lblPort["text"] = "Port: " + str(HOST_PORT)
```

Ở dòng 56, kết nối server với host cũng như port đã được sắp xếp trước và bắt đầu lắng nghe client ở dòng 57.
Các yêu cầu phía cầu kết nối và gửi nhận tin nhận phái client được thiết lập ở dòng 59 (tạo 1 thread mới)

- Accept client to connect server

```
71  def accept_clients(the_server, y):
72      while True:
73          client, addr = the_server.accept()
74          clients.append(client)
75
76          # use a thread so as not to clog the gui thread
77          threading._start_new_thread(
78              send_receive_client_message, (client, addr))
```

Server chấp nhận một yêu cầu kết nối từ client mới và lưu trữ thông tin client (đối tượng kết nối) trong một danh sách . Điều này cho phép server theo dõi tất cả client được kết nối.

- Handle sending/receiving clients messages

```python
def send_receive_client_message(client_connection, client_ip_addr):
    client_msg = " "

    # send welcome message to client
    client_name = client_connection.recv(4096).decode()
    welcome_msg = "Welcome " + client_name + ". Use 'exit' to quit"
    client_connection.send(welcome_msg.encode())

    clients_names.append(client_name)

    update_client_names_display(clients_names)  # update client names display

    while True:
        data = client_connection.recv(4096).decode()
        if not data:
            break
        if data == "exit":
            break

        client_msg = data

        idx = get_client_index(clients, client_connection)
        sending_client_name = clients_names[idx]

        for c in clients:
            if c != client_connection:
                server_msg = str(sending_client_name + "->" + client_msg)
                c.send(server_msg.encode())

    # find the client index then remove from both lists(client name list and connection list)
    idx = get_client_index(clients, client_connection)
    del clients_names[idx]
    del clients[idx]
    server_msg = "BYE!"
    client_connection.send(server_msg.encode())
    client_connection.close()

    update_client_names_display(clients_names)  # update client names display
```

Ultis:

```python
# Return the index of the current client in the list of clients
def get_client_index(client_list, curr_client):
    idx = 0
    for conn in client_list:
        if conn == curr_client:
            break
        idx = idx + 1

    return idx


# Update client name display when a new client connects OR
# When a connected client disconnects
def update_client_names_display(name_list):
    tkDisplay.config(state=tk.NORMAL)
    tkDisplay.delete('1.0', tk.END)

    for c in name_list:
        tkDisplay.insert(tk.END, c+"\n")
    tkDisplay.config(state=tk.DISABLED)
```

get_client_index (): trả về chỉ mục khách hàng hiện tại trong danh sách khách hàng.
update_client_names_display (): về cơ bản cập nhật hiển thị tên client khi client kết nối (thêm) hoặc ngắt kết nối
Server nhận tên client bằng phương thức socket.recv và gửi thông báo chào mừng bằng cách sử dụng socket.send cho client. Tên client mới được kết nối được lưu trữ trong danh sách được cập nhật trong và được cập nhật để hiển thị ở dòng 93 . Tiếp theo, server đi vào một vòng lặp cho phép nó tiếp tục nhận và gửi tin nhắn đến client. Vòng lặp này kết thúc nếu kết nối client bị mất hoặc thông báo client nhập "exit". Sau đó client bị ngắt kết nối được khỏi danh sách tương ứng, đóng kết nối client và xóa tên client khỏi vùng hiển thị của server. Nếu không, thông điệp client này được chuyển tiếp đến tất cả các client khác đang được kết nối khác.

Code client:

- Connect to server

```
43  def connect():
44      global username, client
45      if len(entName.get()) < 1:
46          tk.messagebox.showerror(
47              title="ERROR!!!", message="You MUST enter your first name <e.g. John>")
48      else:
49          username = entName.get()
50          connect_to_server(username)
```

Chức năng kiểm tra xem tên người dùng đã được nhập chưa trước khi cố gắng kết nối với server. Sau đó, nó gọi một hàm connect_to_server() nơi yêu cầu kết nối server thực sự được bắt đầu.

```
# network client
client = None
HOST_ADDR = "192.168.30.111"
HOST_PORT = 3000
```

```
57  def connect_to_server(name):
58      try:
59          client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
60          client.connect((HOST_ADDR, HOST_PORT))
61          client.send(name.encode())  # Send name to server after connecting
62
63          entName.config(state=tk.DISABLED)
64          btnConnect.config(state=tk.DISABLED)
65          tkMessage.config(state=tk.NORMAL)
66
67          # start a thread to keep receiving message from server
68          # do not block the main thread :)
69          threading._start_new_thread(receive_message_from_server, (client, "m"))
70      except Exception as e:
71          tk.messagebox.showerror(title="ERROR!!!", message="Cannot connect to host: " + HOST_ADDR +
72                                  " on port: " + str(HOST_PORT) + " Server may be Unavailable. Try again later")
73
```

Tạo 1 soket bên phía client và kết nối đến server thông qua host và port đã setup trước

- Receive message from server

```
75  def receive_message_from_server(sck, m):
76      while True:
77          from_server = sck.recv(4096).decode()
78
79          if not from_server:
80              break
81
82          # display message from server on the chat window
83
84          # enable the display area and insert the text and then disable.
85          # why? Apparently, tkinter does not allow us insert into a disabled Text widget :(
86          texts = tkDisplay.get("1.0", tk.END).strip()
87          tkDisplay.config(state=tk.NORMAL)
88          if len(texts) < 1:
89              tkDisplay.insert(tk.END, from_server)
90          else:
91              tkDisplay.insert(tk.END, "\n\n" + from_server)
92
93          tkDisplay.config(state=tk.DISABLED)
94          tkDisplay.see(tk.END)
95
96          # print("Server says: " +from_server)
97
98      sck.close()
99      window.destroy()
```

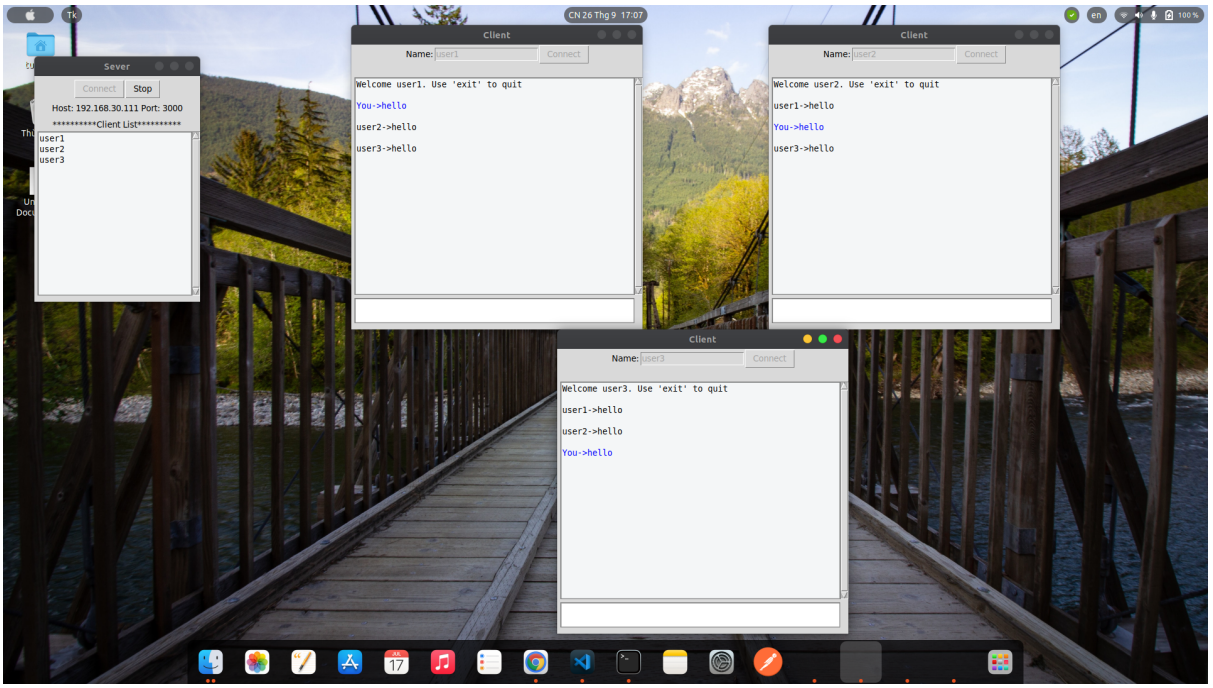
Chúng tôi chạy một vòng lặp liên tục để tiếp tục nhận thông báo từ server (thông qua socket client). Các tin nhắn đã nhận được thêm vào khu vực hiển thị trò chuyện (dòng 86 -94)

- Send message to server

```
102 def getChatMessage(msg):
103
104     msg = msg.replace('\n', '')
105     texts = tkDisplay.get("1.0", tk.END).strip()
106
107     # enable the display area and insert the text and then disable.
108     # why? Apparently, tkinter does not allow use insert into a disabled Text widget :(
109     tkDisplay.config(state=tk.NORMAL)
110     if len(texts) < 1:
111         tkDisplay.insert(tk.END, "You->" + msg, "tag_your_message")  # no line
112     else:
113         tkDisplay.insert(tk.END, "\n\n" + "You->" + msg, "tag_your_message")
114
115     tkDisplay.config(state=tk.DISABLED)
116
117     send_mssage_to_server(msg)
118
119     tkDisplay.see(tk.END)
120     tkMessage.delete('1.0', tk.END)
121
122
123 def send_mssage_to_server(msg):
124     client_msg = str(msg)
125     client.send(client_msg.encode())
126     if msg == "exit":
127         client.close()
128         window.destroy()
129     print("Sending message")
```

Nhận tin nhắn từ ô nhập và gửi tin nhắn đó đến từ server thông qua phương thức send() của socket.

Demo:


Sever
Connect
Stop
Host: 192.168.30.111 Port: 3000
**********Client List**********
user1
user2
user3
Client
Name: user1
Connect
Welcome user1. Use 'exit' to quit
You->hello
user2->hello
user3->hello
Client
Name: user2
Connect
Welcome user2. Use 'exit' to quit
user1->hello
You->hello
user3->hello
Client
Name: user3
Connect
Welcome user3. Use 'exit' to quit
user1->hello
user2->hello
You->hello