

BLG433E - Computer Communications Homework 1 - Report

Kadir Emre Oto
150140032

In this homework, we are expected to implement an instant messaging program with python programming language.

Implementation Details

There are three files implemented in this project: “base.py”, “server.py”, and “client.py”.

For the sake of simplicity, I implement a base-class to generalize the sending-receiving procedure in “**base.py**” file. This class has two methods: send method which conveys a message to given connection, and recv method which gets a message from given connections and raises an error if connections is closed. Both “**server**” and “**client**” classes inherit that abstract class.

“**server.py**” implements the server functions; run, accepts, listens. First an instance of Server Class should be created with host and port values. Then “**run**” method can be called, and server would be started. This method can be called with a parameter, daemon, which denotes the number of daemon in server to accept connections in parallel using threads. The default value of daemon number is 10. “**accept**” method accepts a new connections and waits for username information, and after saving the connection and username data, calls “**listen**” method using another thread to receive messages in parallel. “**listen**” method takes one argument, a socket connection object, and waits for messages and sends them to all other connections.

“**client.py**” file is for users. This file has only one class called “Interface” which creates a GUI using built-in “Tkinter” library. After creating an interface instance, main method should be called without any parameter. Constructor of the class (`__init__` method) creates the window and some interface widgets. “**connect**” method is called when the “Connect Button” is pressed and it provides us to connect a server with given host and port information in GUI. After a successful connection, “**listen**” method is called immediately using thread. Any message is fetched in this method, and inserted to the chat area by calling the “**insert**” method. Lastly, if user wants to send a message, “**send_message**” function should be called.

Running Procedures

No additional command-line arguments are needed to run program.

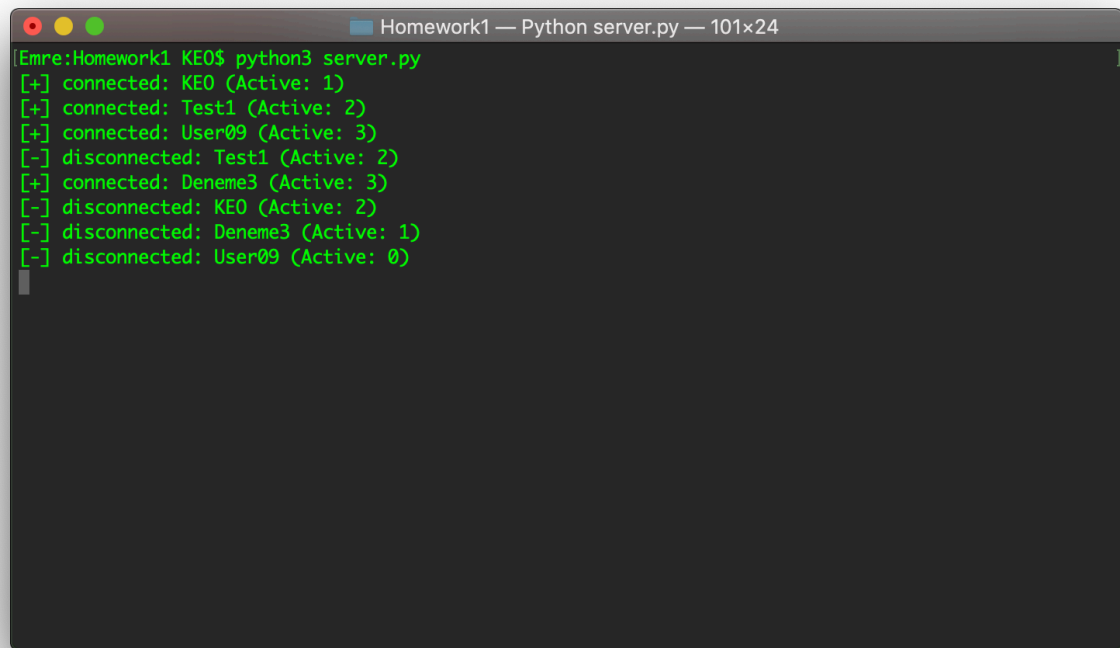
For server: `python3 server.py`
For users: `python3 client.py`

If you do not have Tkinter library in python 3.7, you may get an error while running client program (No module named: tkinter). In this case you need to install Tkinter library first:

```
sudo apt-get install python3.7-tkinter
```

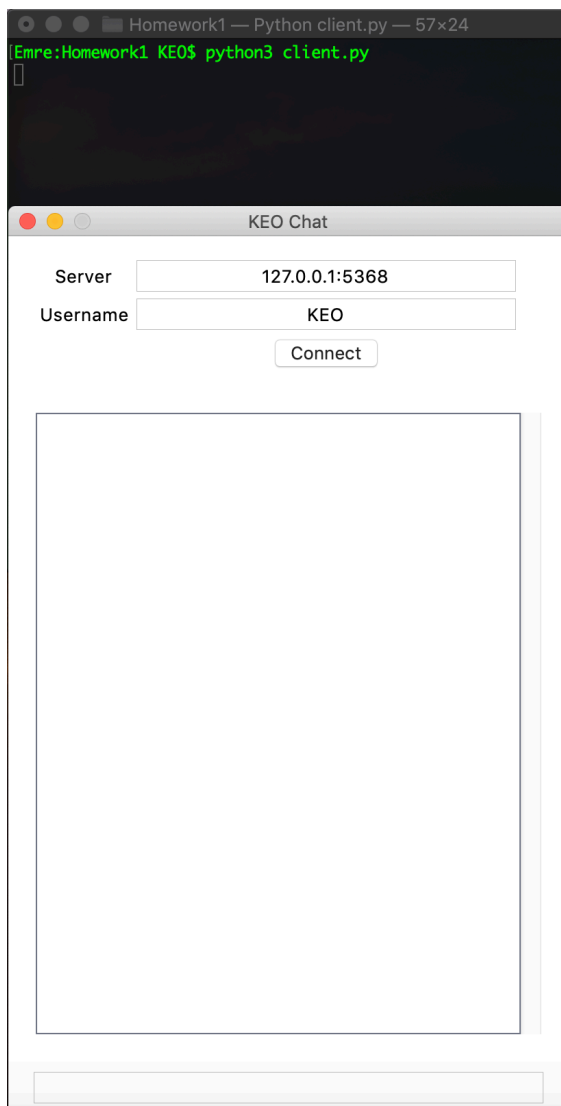
Use Case Scenarios

- 1) Server program shows the connection events and number of active connections.



```
Homework1 — Python server.py — 101x24
[Emre:Homework1 KE0$ python3 server.py
[+] connected: KE0 (Active: 1)
[+] connected: Test1 (Active: 2)
[+] connected: User09 (Active: 3)
[-] disconnected: Test1 (Active: 2)
[+] connected: Deneme3 (Active: 3)
[-] disconnected: KE0 (Active: 2)
[-] disconnected: Deneme3 (Active: 1)
[-] disconnected: User09 (Active: 0)
```

- 2) After connecting a server, all active users will be listed in chat area.



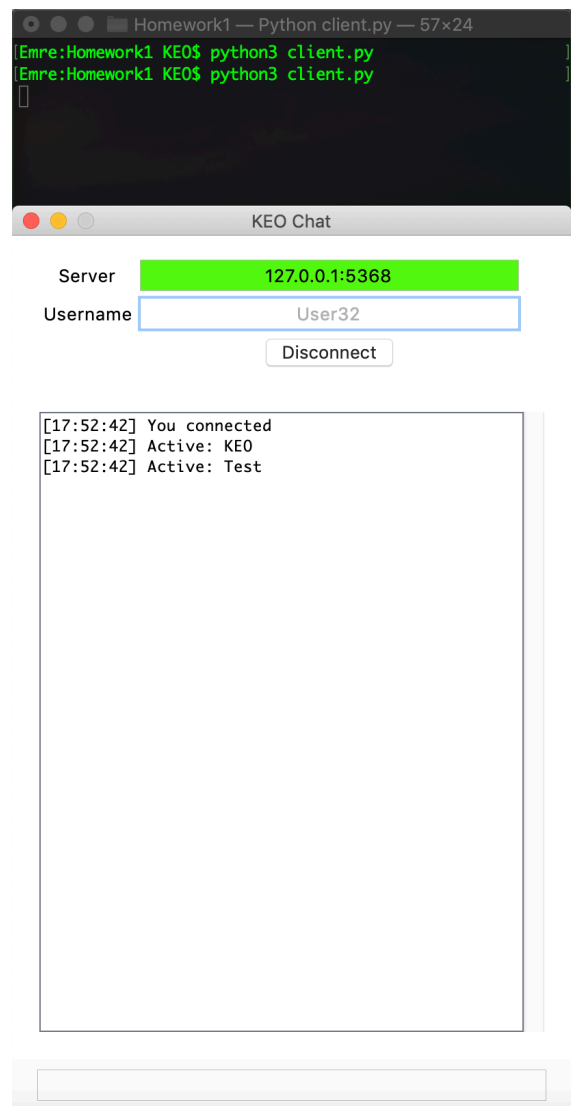
Homework1 — Python client.py — 57x24

```
[Emre:Homework1 KE0$ python3 client.py
```

KEO Chat

Server

Username



Homework1 — Python client.py — 57x24

```
[Emre:Homework1 KE0$ python3 client.py
[Emre:Homework1 KE0$ python3 client.py
```

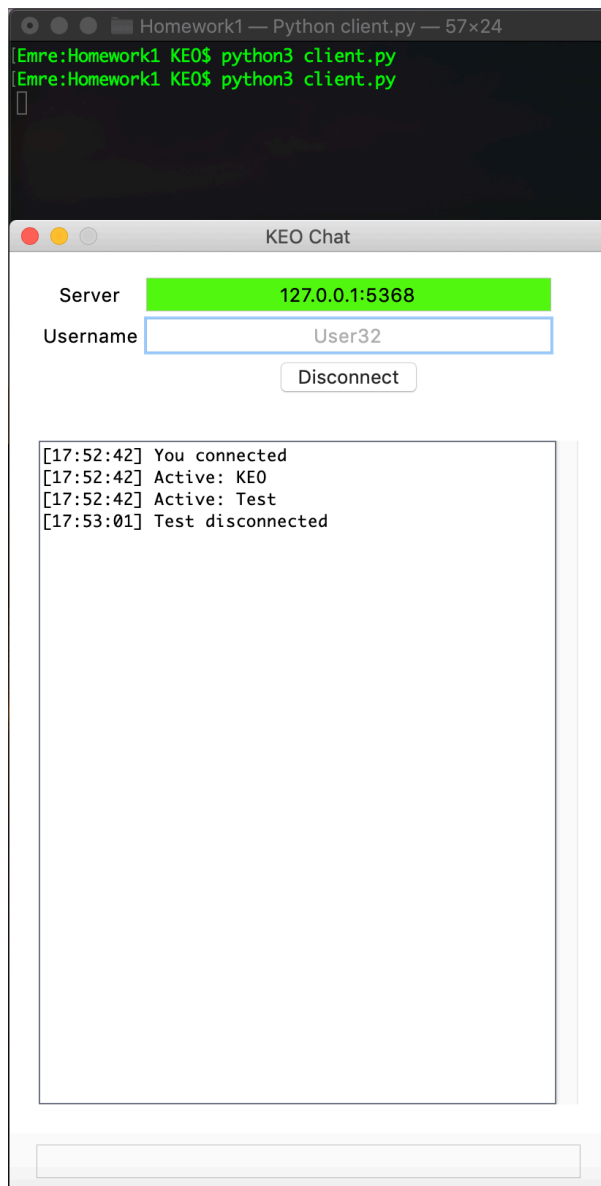
KEO Chat

Server

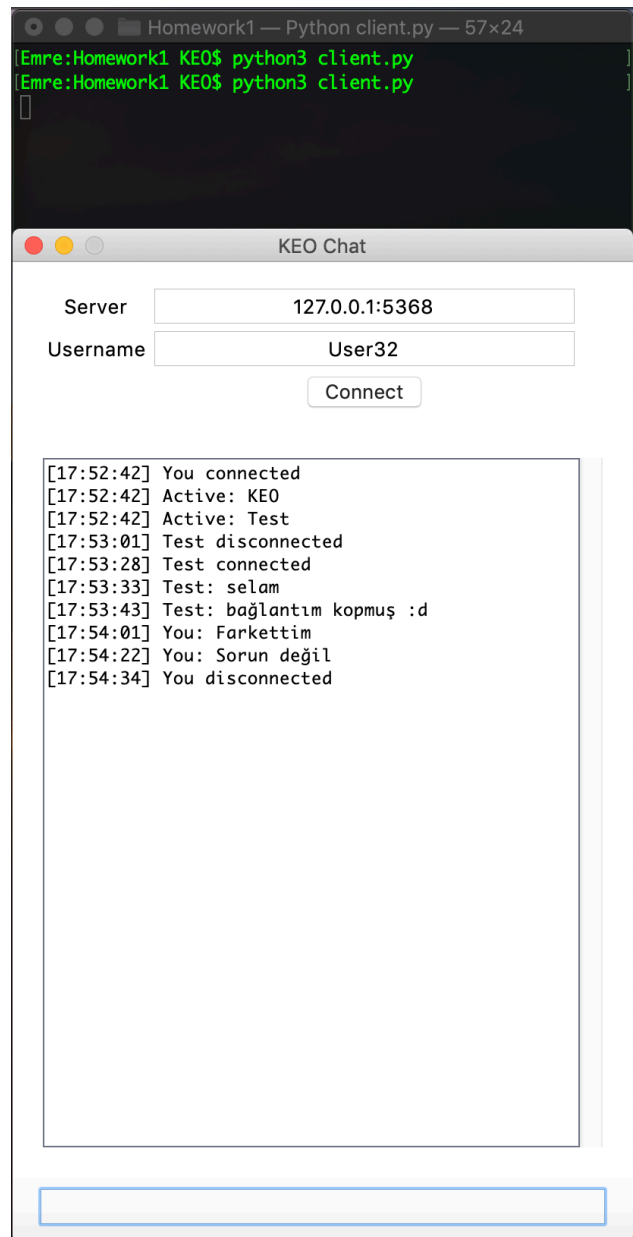
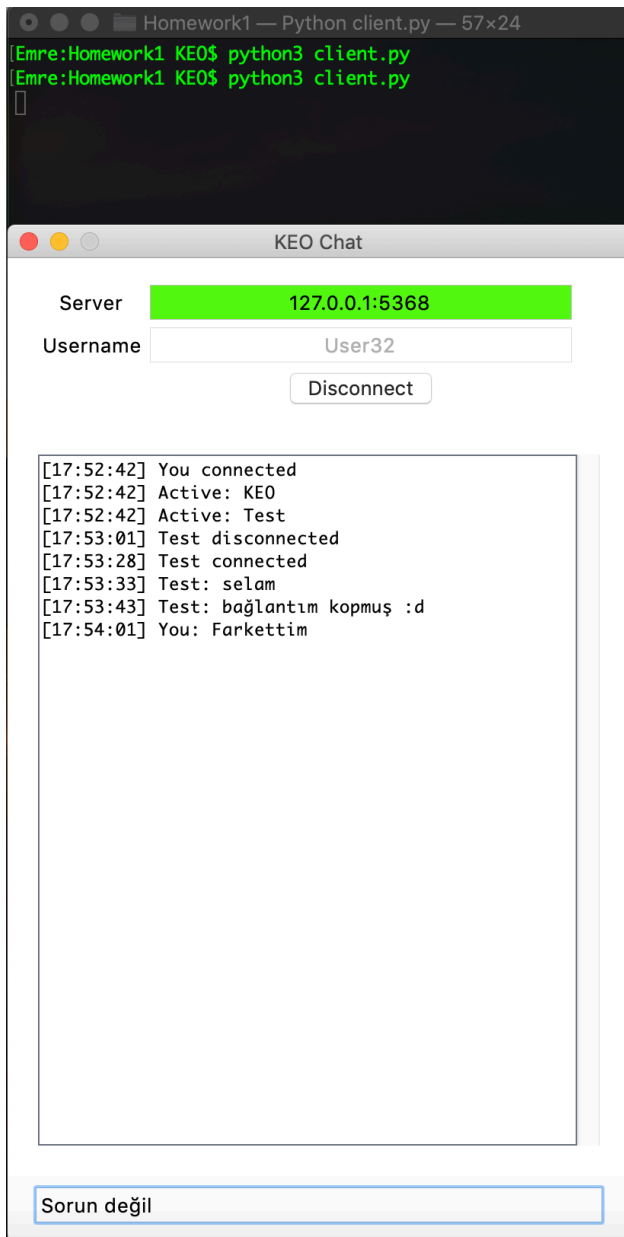
Username

[17:52:42] You connected
[17:52:42] Active: KE0
[17:52:42] Active: Test

3) If any user closes the client program or disconnects from the server or any new user connects to the server, all other users will be notified.



4) Users can send new messages by using the entry box on the bottom of window, and can disconnect from the server by pressing the disconnect button. Chat area would be updated after all this events.



4) If the user wants to send a message before connected a server, or tries to connect a non-existed server, an error message would be inserted the chat area to notify the user.

