МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

**ОТЧЕТ**

по лабораторной работе № 6

на тему: «Сетевое программирование»

по дисциплине: «Программирование на языке Python»

Вариант № 18

Выполнил: Шорин В.Д. Шифр: 171406

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71-ПГ

Проверили: Захарова О.В., Раков В.И.

Отметка о зачете:

Дата: «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Орел, 2019 г.

**Задание:**

Разработать клиент-серверное приложение (чат).

Требования к клиенту:

- отправка на сервер введенного пользователем сообщения;

- получение сообщений, отправленных другими клиентами;

- удобный графический интерфейс.

Требования к серверу:

- организация чата между клиентами (до 3 клиентов), а именно отправка всем клиентам полученных сообщений;

- полученные сообщения перед отправкой клиентам необходимо проверить на грамотность и исправить: после запятой должен стоять один пробел, далее должен идти непробельный символ.

Клиенты должны последовательно отправлять сообщения.

**Код:**

**«main.py» Client**

import tkinter

from socket import AF\_INET, socket, SOCK\_STREAM

from threading import Thread

def receive():

""" Handles receiving of messages. """

while True:

try:

msg = sock.recv(BUFSIZ).decode("utf8")

msg\_list.insert(tkinter.END, msg)

except OSError: # Possibly client has left the chat.

break

def send(event=None):

""" Handles sending of messages. """

msg = my\_msg.get()

my\_msg.set("") # Clears input field.

sock.send(bytes(msg, "utf8"))

if msg == "#quit":

sock.close()

top.quit()

def on\_closing(event=None):

""" This function is to be called when the window is closed. """

my\_msg.set("#quit")

send()

top = tkinter.Tk()

top.title("Simple Chat Client v1.0")

messages\_frame = tkinter.Frame(top)

my\_msg = tkinter.StringVar() # For the messages to be sent.

my\_msg.set("")

scrollbar = tkinter.Scrollbar(messages\_frame) # To navigate through past messages.

msg\_list = tkinter.Listbox(messages\_frame, height=15, width=70, yscrollcommand=scrollbar.set)

scrollbar.pack(side=tkinter.RIGHT, fill=tkinter.Y)

msg\_list.pack(side=tkinter.LEFT, fill=tkinter.BOTH)

msg\_list.pack()

messages\_frame.pack()

button\_label = tkinter.Label(top, text="Enter Message:")

button\_label.pack()

entry\_field = tkinter.Entry(top, textvariable=my\_msg, foreground="Red")

entry\_field.bind("<Return>", send)

entry\_field.pack()

send\_button = tkinter.Button(top, text="Send", command=send)

send\_button.pack()

quit\_button = tkinter.Button(top, text="Quit", command=on\_closing)

quit\_button.pack()

top.protocol("WM\_DELETE\_WINDOW", on\_closing)

HOST = "127.0.0.1"

PORT = 5000

BUFSIZ = 1024

ADDR = (HOST, PORT)

sock = socket(AF\_INET, SOCK\_STREAM)

sock.connect(ADDR)

receive\_thread = Thread(target=receive)

receive\_thread.start()

tkinter.mainloop() # Starts GUI execution.

**«processor.py» Server**

from socket import AF\_INET, socket, SOCK\_STREAM

from threading import Thread

import re

clients = {}

addresses = {}

HOST = "127.0.0.1"

PORT = 5000

BUFSIZ = 1024

ADDR = (HOST, PORT)

SOCK = socket(AF\_INET, SOCK\_STREAM)

SOCK.bind(ADDR)

def parse\_message(msg):

message = msg.decode()

# pattern\_whirespace = '[\s]+'

# re.sub(pattern\_whirespace, ' ', message)

i = 0

while i != len(message) - 3:

if message[i] == ',' \

and message[i + 1] != ' ':

message = message[:i] + ', ' + message[i + 1:]

i = 0

elif message[i] == ',' \

and message[i + 1] == ' ' \

and message[i + 2] == ' ':

message = message[:i] + ', ' + message[i + 3:]

i = 0

else:

message = message

i += 1

return message.encode()

def accept\_incoming\_connections():

"""Sets up handling for incoming clients."""

while True:

client, client\_address = SOCK.accept()

print(f"{client\_address} has connected.")

client.send("Greetings from the ChatRoom! ".encode("utf8"))

client.send("Now type your name and press enter!".encode("utf8"))

addresses[client] = client\_address

Thread(target=handle\_client, args=(client, client\_address)).start()

def handle\_client(conn, addr): # Takes client socket as argument.

"""Handles a single client connection."""

name = conn.recv(BUFSIZ).decode("utf8")

welcome = f'Welcome {name}! If you ever want to quit, type #quit to exit.'

conn.send(bytes(welcome, "utf8"))

msg = f"{name} from [{addr[0]}:{addr[1]}] has joined the chat!"

broadcast(bytes(msg, "utf8"))

clients[conn] = name

while True:

msg = conn.recv(BUFSIZ)

msg = parse\_message(msg)

if msg != bytes("#quit", "utf8"):

broadcast(msg, name + ": ")

else:

conn.send(bytes("#quit", "utf8"))

conn.close()

del clients[conn]

broadcast(bytes(f"{name} has left the chat.", "utf8"))

break

def broadcast(msg, prefix=""): # prefix is for name identification.

"""Broadcasts a message to all the clients."""

for sock in clients:

sock.send(bytes(prefix, "utf8") + msg)

if \_\_name\_\_ == "\_\_main\_\_":

SOCK.listen(3) # Listens for 3 connections at max.

print("Chat Server has Started !!")

print("Waiting for connections...")

ACCEPT\_THREAD = Thread(target=accept\_incoming\_connections)

ACCEPT\_THREAD.start() # Starts the infinite loop.

ACCEPT\_THREAD.join()

SOCK.close()