COMPUTER COMMUNICATION HOMEWORK 1 – REPORT

İpek Erdoğan

150130102

In this project, our aim is to create an instant messaging platform. So I used threadining logic to provide this. Both of my client and server creates threads.

Starting from client code:

I defined a function inside of the ClientServer class to provide my client's listening. It calls by initialization function (init) with threadining and print the result to the screen.

```
def listenOtherClients(self, server):
    while True:
        clientUsername=server.recv(1024)
        clientMessage=server.recv(1024)
        if clientMessage=="exit":
            print (clientUsername , " is closed")
        else:
            print (clientUsername , " says: ", clientMessage.decode("utf-8"))
```

I defined initialization function with 3 parameters: serverport, servername and client's port. First it creates a socket on a pre-defined port (clientserverport).

```
def __init__(self,serverport, servername, clientserverport):
    try:
        clientSocket=socket(AF_INET,SOCK_STREAM)
    except:
        print ("Socket cannot be created.")
        exit(1)
    print ("Socket is created.")
    try:
```

```
        clientSocket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
except:
        print ("Socket cannot be used!")
        exit(1)
print ("Socket can be used well")
try:
        clientSocket.bind(('', clientserverport))
except:
        print ("Binding is unsuccessfull!")
        exit(1)
print ("Binding is successfull.")
clientSocket.listen(45)
```

Then, it tries to connect to the server with servername and serverport parameters it got from the main function.

```
        clientSocket.connect((servername,serverport))
```

And it sends 2 information to the server, username and message.

```
        while True:
                username=input('Username:')
                message=input('Type message:')
                clientSocket.send(username.encode())
                clientSocket.send(message.encode())
                if message=="exit":
                        clientSocket.close()
                        exit(0)
                else:
                        print("Message is sent.")
                connectionSocket,addr=clientSocket.accept()
```

Then it also opens a thread to listen other clients' information from the server.

threading.Thread(target = self.listenOtherClients,args = (connectionSocket)).start()

I defined a main function to call these functios and give the parameters.

```
if __name__=="__main__":
    serverName="192.168.0.33"
    serverPort=12000
    clientServerPort=13000
    ClientServer(serverPort, serverName, clientServerPort)
```

Continue with the server code:

class ThreadedServer(): // I defined a class named “ThreadedServer”

clientset= set() // I defined a set called “clientset” to keep the addresses of the online clients, I will use this information while I’m going to send clients’ messages each other.

And I defined a sendToClient function which sends a message that it took from a client to other clients.

```
def sendToClient(self, client, username, message):
    while True:
        for c in clientset:
            c.sendall(username)
            c.sendall(message)
```

I defined a function for server to listen the clients and receive the messages, then turn them into variables called “clientUsername” and “clientMessage”. This function creates a thread for sendToClient function.

```
def listenToClient(self, client, addr):
    while True:
        clientUsername= client.recv(1024)
        clientMessage= client.recv(1024)
        if message == "exit":
            clientset.remove(client)
            client.close()
    threading.Thread(target = self.listenToClient,args = (connectionSocket, clientUsername, clientMessage)).start()
```

Then, initialization starts. It defines a reusable socket for the application on port serverPort which defined in the main function.

```
def __init__(self,serverPort):
    try:
        serverSocket=socket(AF_INET,SOCK_STREAM)
    except:
        print ("Socket cannot be created.")
        exit(1)
    print ("Socket is created.")
    try:
        serverSocket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
    except:
        print ("Socket cannot be used.")
        exit(1)
    print ("Socket is being used.")
    try:
        serverSocket.bind(('',serverPort))
    except:
        print ("Binding cannot be done.")
        exit(1)
    print ("Binding is done.")
    try:
        serverSocket.listen(45)
    except:
        print ("Server cannot listen!")
        exit(1)
    print ("The server is ready.")
```

Then it accepts the connection which are coming and adding their address to the clientset set.

```
    while True:
        connectionSocket,addr=serverSocket.accept()
```

clientset.add(client)

And creating a thread for listenToClients function.

threading.Thread(target = self.listenToClient,args = (connectionSocket,addr)).start()

I gave parameters to the class in the main function.

```
if __name__=="__main__":
   serverPort=12000
   ThreadedServer(serverPort)
```