**Report**

과 목 : TCP/IP

담당교수 : 김 한 결 교수님

학 과 : 컴퓨터정보

학 번 : 201744028

이 름 : 이 기 훈

제 출 일 : 2021.06.16.

1. 개요

1.1 프로젝트 개요

▶ 만들게 된 이유 : 멀티 스레드와 소켓 프로그래밍을 가장 잘 활용할 수 있을 것 같은 주제를 생각하다가

채팅 프로그램이 가장 적합하다고 생각하여 채팅 프로그램의 개발을 하게 되었습니다

▶ 프로젝트의 목적 : 1:1채팅이 아니라 복수의 사용자들이 접속하여 메시지를 전송하는 즉시 모든

사용자들에게 보여줄 수 있는 것입니다. 또한 스레드를 사용하여 메인 스레드는 서버의 작동 및 연결을

담당하고 다른 스레드는 송/수신을 담당하게 만드는 것입니다.

▶ 개발 언어 및 개발 환경 : Python3.9.5 / PyCharm

▶ 기능 소개 : 여러 클라이언트와 동시 채팅 기능, 계산기 기능, 대화 기록 저장 기능

1.2 요구 사항

멀티 스레드를 사용한 TCP 소켓 프로그래밍

사용자가 채팅을 보낸 시간이 출력되도록 한다

각 사용자들에게 채팅을 보여준다

종료를 누르면 통신이 끊기게 한다

각 접속자들에게 접속 정보를 나타내게 한다

채팅방에 들어가기 전에 서버 아이피와 포트 입력하고 접속한다

DB를 이용하여 사용자 정보와 채팅 기록들을 저장

채팅 초기화 버튼을 이용하여 채팅창 Clear 기능

채팅이 길어지면 스크롤바를 이용하여 지난 채팅 기록을 볼 수 있게 한다

1.3 프로젝트 산출물

- 서버 Python 파일 / exe파일

- 사용자 Python 파일 / exe파일

- 채팅기록이 저장될 파일

2. 설계

2.1 기능 및 사용 환경 설계

멀티 스레드를 기반으로 한 TCP 소켓 프로그래밍 다중 채팅 프로그램 개발

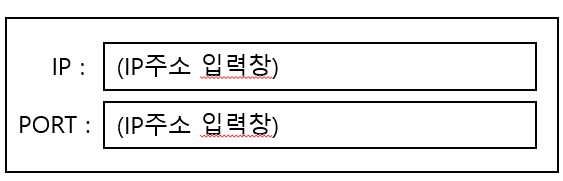
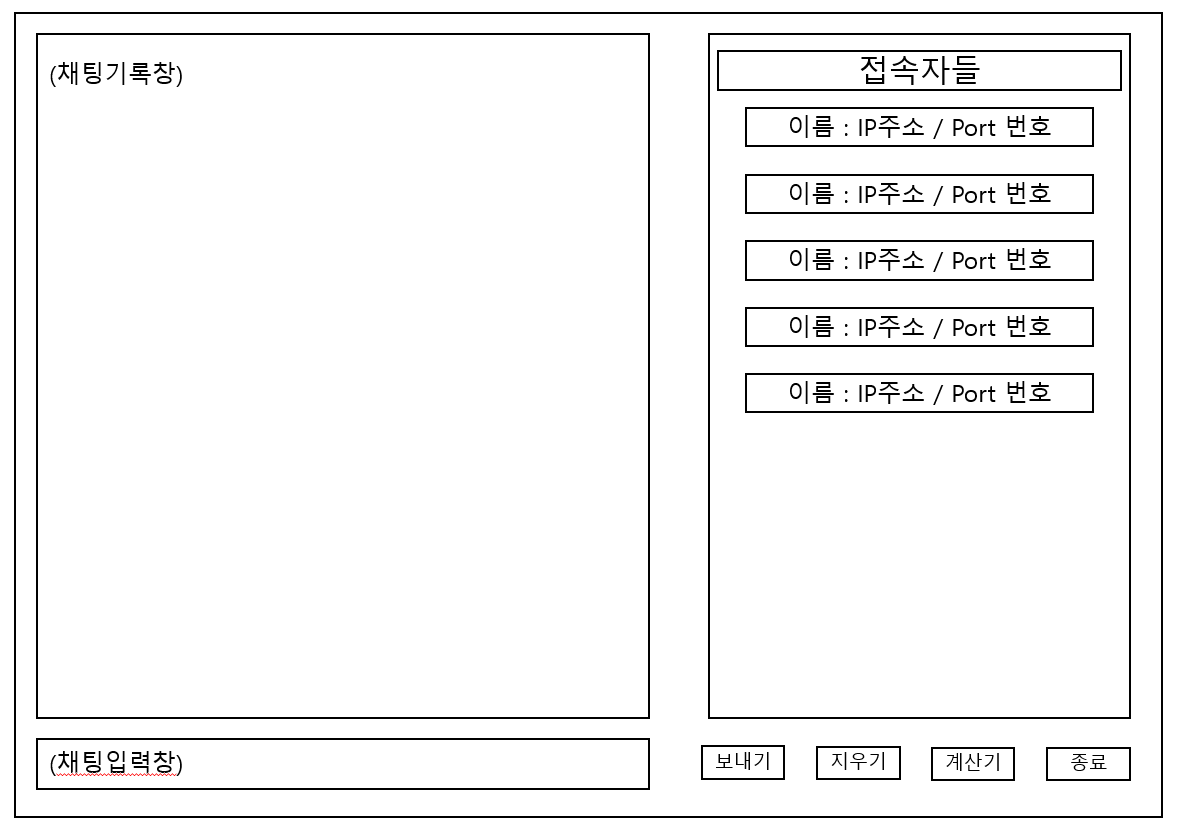
개발 언어는 Python을 이용하며, IDE는 JetBrains사의 PyCharm을 사용한다.

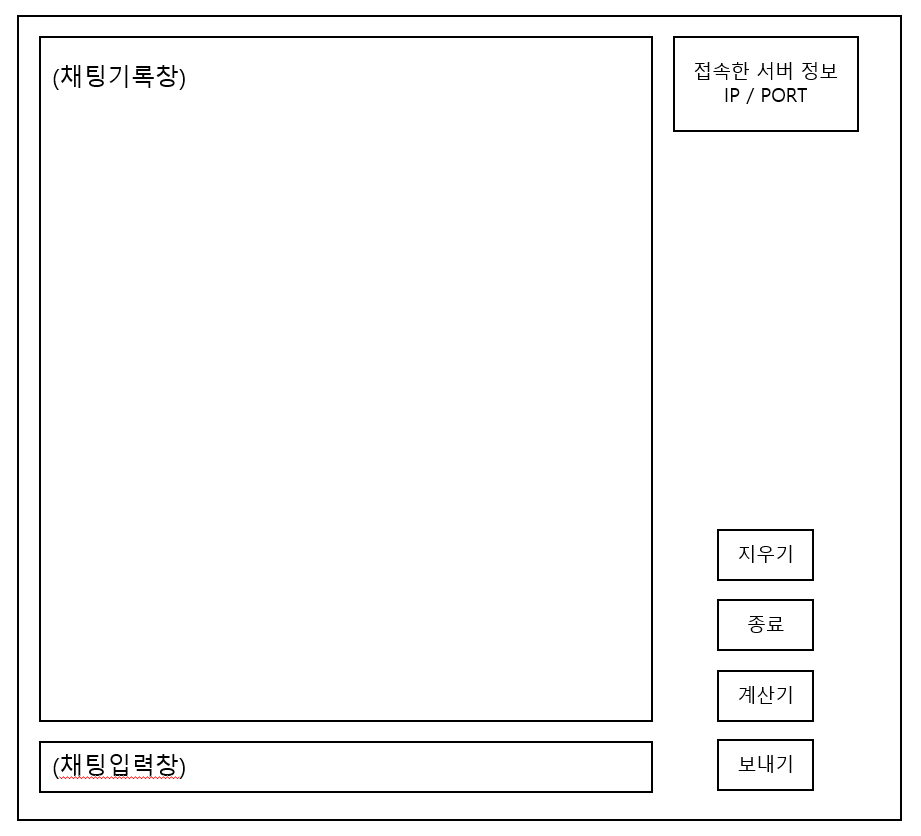
GUI(Graphical User Interface)는 Python tkinter을 사용하여 구현한다.

에코서버 개념으로 각 클라이언트들이 메시지를 전송하면 서버측에서 모든 클라이언트에게 뿌려주도록 한다

클라이언트의 접속 제한 수는 따로 제한을 두지 않고 구현한다.

2.2 GUI화면 설계

 (처음 생각했던 구현 모습)

 (확정 GUI 모습)

3. 차이점

▶ 1:1채팅 때는 클라이언트1이 채팅을 입력하고 있는 도중에 클라이언트2가 채팅을 송신하면 클라이언트1은

도중에 입력하던 채팅이 초기화되버리는 현상이 발생하는데, 이번에 구현할 채팅은 그런 걱정 없이 채팅을

이어나갈 수 있다.

▶ 여러 다른 채팅 프로그램을 사용하면서 더치페이, 단위 계산, 복잡한 숫자의 사칙연산 등 필요할 때가

많았는데 채팅창 안에 계산기 기능을 추가하여 원할 때 사용할 수 있다.

▶ 대화 기록을 다른 공간에 저장할 수 있다. 프로그램이 종료되어서 지난 기록을 보고 싶어도 못 볼 때,

대화 기록이 저장된 파일을 열어서 지난 대화 기록을 볼 수 있다.

▶ 대화가 송신, 수신될 때마다 시간을 남겨서 며칠 몇 시 몇 분에 송신하고 수신했는지 볼 수 있다.

4. 소스코드

**guiServer.py**

**from** **socket** **import** \*

**import** **threading**

**import** **datetime**

stop = **True**

**class** **chatRoom**: *# 채팅방에 관한 설정을 하는 클래스이다*

**def** \_\_init\_\_(self): *# \_\_init\_\_메서드로 변수들을 초기화 한다*

self.clients = [] *# 각 클라이언트을 담을 리스트 변수이다*

self.allChat = **None** *# 모든 대화기록들을 저장할 변수이다*

**def** sendMessageAll(self, message): *# 채팅방에 있는 모든 클라이언트에게 메시지를 전송한다*

**for** i **in** self.clients: *# 리스트안에 있는 클라이언트의 수 만큼 반복한다*

print(i)

i.sendMessage(message)

**def** addClient(self, t): *# 리스트에 새 클라이언트를 추가하는 메서드이다.*

self.clients.append(t)

**def** delClient(self, t): *# 리스트에서 해당 클라이언트를 삭제하는 메서드이다.*

self.clients.remove(t)

**class** **chatClient**: *# 채팅의 송/수신을 담당하는 부분이다.*

**def** \_\_init\_\_(self, room, sock): *# 사용할 변수들을 초기화해준다.*

self.room = room

self.id = **None**

self.sock = sock

**def** readMessage(self):

now = str(datetime.datetime.now().strftime("%A, **%d**. %I:%M %p")) *# 처음 접속하고 닉네임을 지정하는 시간 저장*

self.id = self.sock.recv(1024).decode() *# 클리아언트로 부터 받은 id를 수신받는다*

message ='[' + now + '] ' + self.id + '님이 입장하셨습니다.'

self.room.sendMessageAll(message) *# 모든 클라이언트에게 메시지를 전송*

**while** **True**:

now = str(datetime.datetime.now().strftime("%A, **%d**. %I:%M %p")) *# 채팅 입력할 때마다 갱신하는 시간*

message = self.sock.recv(1024).decode() *# 받은 데이터를 decode함수로 byte코드를 문자열로 변환한다*

*# if message == "/종료": # 해당 주석 부분은 “/종료”라는 메시지를 받으면 종료되게 구현을 하려했으나*

*# self.sock.sendall(message) # 구현에 실패한 부분입니다*

*# self.room.delClient(self)*

*# stop = False*

*# break*

message = '[' + now + '] ' + self.id + ' : ' + message

self.room.sendMessageAll(message)

*# self.room.sendMessageAll('[' + now + '] ' + self.id + '님이 퇴장하셨습니다')*

**def** sendMessage(self, message): *#*

*# print(type(message))*

self.sock.sendall(message.encode(encoding='utf-8')) *# sendall함수는 모두 전송할 때까지 send함수를 호출한다*

**class** **chatSever**: *# 서버의 생성 및 설정을 담당*

ip = '127.0.0.1'

port = 8080

**def** \_\_init\_\_(self):

self.server\_sock = **None**

self.room = chatRoom() *# chatRoom클래스의 인스턴스 생성*

**def** open(self):

*# 서버 소켓 생성, AF\_INET으로 IPv4인터넷 프로토콜로 설정/SOCK\_STREAM으로 TCP프로토콜을 사용*

self.server\_sock = socket(AF\_INET, SOCK\_STREAM)

self.server\_sock.setsockopt(SOL\_SOCKET, SO\_REUSEADDR, 1) *# setsockopt로 소켓 옵션을 설정한다*

*# 프로그램을 강제 종료하고나서, 커널단에서 해당 소켓을 바인딩해서 사용하고 있어서 프로그램을 재실행하면 오류가발생하는데,*

*# SO\_REUSEADDR 옵션을 주면 기존에 바인디된 주소를 다시 사용할 수 있게 할 수 있다.*

self.server\_sock.bind((chatSever.ip,chatSever.port)) *# 소켓을 ip와 포트 번호와 연결하는데 사용한다*

self.server\_sock.listen() *# 클라이언트 연결 대기상태*

**def** run(self): *# 서버 가동 함수*

self.open()

print('서버 가동')

**while** **True**:

client\_sock, addr = self.server\_sock.accept() *# 클라이언트의 연결을 허용한다*

print(addr, '접속') *# 연결을 허용한 클라이언트의 ip번호와 포트번호가 출력된다*

t = chatClient(self.room, client\_sock) *# 인스턴스 생성*

self.room.addClient(t) *# 클라이언트 리스트에 새로 append한다*

print('클라이언트 : ', self.room.clients)

th = threading.Thread(target=t.readMessage) *# 스레드 생성*

*# target=t.readMessage로 쓰레드가 실행할 함수를 선언한다.*

th.start() *# 스레드 시작*

*# if stop == False:*

*# self.server\_sock.close()*

*# break*

**def** main(): *# 위의 모든 코드가 실행될 메인 부분*

cs = chatSever()

cs.run()

main()

**guiClient.py**

**import** **threading**

**from** **socket** **import** \*

**from** **tkinter** **import** \*

**import** **tkinter** **as** **tk**

**from** **tkinter.scrolledtext** **import** ScrolledText

**class** **chatUi**:

ip = '127.0.0.1'

port = 8080

*# ip = tkinter.StringVar()*

*# port = tkinter.IntVar()*

**def** \_\_init\_\_(self): *# 사용 할 변수들을 초기화한다*

*# self.root = None*

self.conn\_sock = **None**

*# self.window2 = None*

self.window = **None**

self.chatContent = **None**

self.enterChat = **None**

self.sendBtn = **None**

self.allChat = ''

self.num1 = **None**

self.num2 = **None**

self.op = **None**

self.result = **None**

**def** connect(self): *# 서버와의 연결을 담당하는 함수*

self.conn\_sock = socket(AF\_INET, SOCK\_STREAM)

self.conn\_sock.connect((chatUi.ip, chatUi.port))

*# def connect(self): # 해당 주석부분은 IP와 PORT번호를 입력받아서 처리하는 기능의 구현을 실패하여 주석처리 하였습니다.*

*# self.conn\_sock = socket(AF\_INET, SOCK\_STREAM)*

*#*

*# try:*

*# self.conn\_sock.connect((self.ip, self.port))*

*# except Exception as e:*

*# print('Connect Error : ', e)*

*# return False*

*# else:*

*# th2 = threading.Thread(target=self.recvMessage)*

*# th2.start()*

*# print('Connected')*

*#*

*# return True*

*# def connetInfo(self): # 아이피와 포트번호 입력 창 GUI (기능 구현 실패로 인한 주석처리)*

*# self.window2 = Tk()*

*# self.window2.title('주소정보입력')*

*# self.window2.geometry('220x50')*

*# self.ipInfo = tk.Label(self.window2, text='IP : ', textvariable=self.ip)*

*# self.portInfo = tk.Label(self.window2, text='Port : ', textvariable=self.port)*

*# self.ipEntry = tk.Entry(self.window2, width=30)*

*# self.portEntry = tk.Entry(self.window2, width=30)*

*#*

*# self.ipInfo.grid(row=0, column=0)*

*# self.portInfo.grid(row=1, column=0)*

*# self.ipEntry.grid(row=0, column=1)*

*# self.portEntry.grid(row=1, column=1)*

*#*

*# self.ipEntry.bind('<Return>', self.sendInfo)*

*# self.portEntry.bind('<Return>', self.sendInfo)*

*#*

*# self.window2.mainloop()*

*# def sendInfo(self, e):*

*# self.ip =self.ipEntry.get()*

*# self.port = self.portEntry.get()*

*#*

*# self.window2.destroy()*

**def** setWindow(self): *# 채팅프로그램의 GUI*

self.window = Tk()

self.window.title('인하챗2021')

self.window.geometry('570x520')

self.chatContent = tk.Label(self.window, borderwidth=4, relief="sunken",bg='white', width=60, height=30, text='닉네임을 입력하세요')

self.joinUser = tk.Label(self.window, borderwidth=2, relief="sunken",bg='white', width=15, height=4,text='접속한 서버 정보' + '**\n**' + chatUi.ip + '**\t**' + str(chatUi.port))

self.enterChat = tk.Entry(self.window, borderwidth=3, relief="sunken",width=54)

self.sendBtn = tk.Button(self.window, borderwidth=4,width=10, text="보내기", command=self.sendMessage)

self.exitBtn = tk.Button(self.window, borderwidth=4,width=10, text="종료", command=self.exitChat)

self.cal = tk.Button(self.window, borderwidth=4, width=10, text='계산기', command=self.CalculatorUI)

self.refresh = tk.Button(self.window, borderwidth=4, width=10, text='클리어', command=self.chatClear)

*# scroll = tk.Scrollbar(self.window, orient="vertical")*

*# scroll.grid(row=0, column=1, sticky='ns',padx=10)*

*# self.window.configure()*

*# self.scrollbar = tk.Scrollbar(self.window)*

self.chatContent.grid(row=0, column=0) *# 대화기록이 출력되는 창*

self.joinUser.grid(row=0, column=1, padx=10, sticky='n') *# 접속정보가 출력되는 창*

self.enterChat.grid(row=1, column=0, ipadx=20) *# 대화내용을 입력하는 창*

self.sendBtn.grid(row=1, column=1) *# 대화내용을 전송하는 버튼*

self.exitBtn.grid(row=0, column=1,sticky='s',pady=60) *# 통신 종료 버튼*

self.cal.grid(row=0, column=1,sticky='s',pady=15) *# 계산기 버튼*

self.refresh.grid(row=0,column=1, sticky='s', pady=100) *# 대화내용 초기화 버튼*

*# self.scrollbar.grid(row=0, column=1,sticky='ns')*

*# bind함수를 사용하여 Enter키를 누르면 채팅이 보내지도록 연결*

self.enterChat.bind('<Return>', self.sendMessage)

**def** chatClear(self): *# 채팅초기화 함수*

self.chatContent.config(text='')

**def** sendMessage(self, e): *# 채팅 전송 함수*

message = self.enterChat.get() *# 입력받은 문자열을 메시지 변수에 저장*

self.enterChat.delete(0, tk.END) *# 채팅 전송 후 입력창을 초기화한다*

self.enterChat.config(text='')

print(type(message))

message = message.encode(encoding='utf-8')

*# encode함수로 문자열을 byte코드로 변환한다 변환을 해야 서버쪽에서 수신받을 수 있다*

print(self.conn\_sock)

self.conn\_sock.sendall(message) *# sendall함수를 통해 요청한 데이터의 모든 버퍼 내용을 모두 전송한다*

print('메시지를 전송하였습니다')

**def** recvMessage(self): *# 채팅 수신 함수*

**while** **True**:

print('메시지를 수신하였습니다')

message = self.conn\_sock.recv(1024) *# 서버로부터 1024바이트 크기만큼 수신*

print(message)

message = message.decode() + '**\n**' *# byte코드를 문자열로 변환 / 가독성을 위해 줄 바꿈을 추가한다*

self.allChat += message

print('서버 : ', self.allChat)

self.chatContent.config(text=self.allChat, justify='left')

*# 대화내용을 config함수로 출력한다 / justify는 문자열의 정렬방법을 설정한다*

*# if message == '/종료':*

*# self.conn\_sock.close()*

*# print('종료되었습니다')*

*# break*

*# self.conn\_sock.close()*

**def** exitChat(self): *# 통신 종료 함수*

*# self.conn\_sock.sendall('접속을 종료하였습니다.'.encode(encoding='utf-8'))*

print(self.allChat, file=self.fp) *# 통신을 종료하면서 지금까지 대화한 내용들을 파일에 모두 저장*

self.chatContent.config(text='접속을 종료하였습니다' ,justify='left')

self.conn\_sock.close() *# 소켓을 종료한다*

self.fp.close() *# 파일을 닫아준다*

**def** run(self): *# 가동 기능 담당 함수*

*# self.connetInfo()*

self.fp = open('chatHistory.txt', 'a')

*# 대화 내용을 기록할 파일을 연결한다, 'a'값을 줘서 기존내용에 이어쓰도록 설정*

self.connect() *# 연결*

self.setWindow() *# GUI창 띄우기*

th1 = threading.Thread(target=self.recvMessage) *# 스레드 생성*

*# target=t.recvMessage 쓰레드가 실행할 함수를 선언한다.*

th1.start()

self.window.mainloop() *# 윈도우에서 수행되는 마우스 클릭 같은 이벤트들이 발생하게끔 유지해주는 함수*

**def** CalculatorUI(self): *# 계산기의 GUI 세팅 함수*

newWindow = tk.Toplevel(self.window)

*# Toplevel 함수를 사용하여 외부 위젯을 종료하지않고 별도의 창을 작동할 수 있다*

newWindow.title('계산기')

self.resultWindow = tk.Label(newWindow, width=25) *# 입력한 값이나 결과값이 출력되는 부분*

self.resultWindow.grid(row=0, column=0, columnspan=4)

*# 각 버튼들을 생성 및 배치*

self.b1 = tk.Button(newWindow, text='1', width=25, command=**lambda**: self.numBtn('1'))

self.b1.grid(row=1, column=0)

self.b2 = tk.Button(newWindow, text='2', width=25, command=**lambda**: self.numBtn('2'))

self.b2.grid(row=1, column=1)

self.b3 = tk.Button(newWindow, text='3', width=25, command=**lambda**: self.numBtn('3'))

self.b3.grid(row=1, column=2)

self.b4 = tk.Button(newWindow, text='+', width=25, command=**lambda**: self.opBtn('+'))

self.b4.grid(row=1, column=3)

self.b5 = tk.Button(newWindow, text='4', width=25, command=**lambda**: self.numBtn('4'))

self.b5.grid(row=2, column=0)

self.b6 = tk.Button(newWindow, text='5', width=25, command=**lambda**: self.numBtn('5'))

self.b6.grid(row=2, column=1)

self.b7 = tk.Button(newWindow, text='6', width=25, command=**lambda**: self.numBtn('6'))

self.b7.grid(row=2, column=2)

self.b8 = tk.Button(newWindow, text='-', width=25, command=**lambda**: self.opBtn('-'))

self.b8.grid(row=2, column=3)

self.b9 = tk.Button(newWindow, text='7', width=25, command=**lambda**: self.numBtn('7'))

self.b9.grid(row=3, column=0)

self.b10 = tk.Button(newWindow, text='8', width=25, command=**lambda**: self.numBtn('8'))

self.b10.grid(row=3, column=1)

self.b11 = tk.Button(newWindow, text='9', width=25, command=**lambda**: self.numBtn('9'))

self.b11.grid(row=3, column=2)

self.b12 = tk.Button(newWindow, text='\*', width=25, command=**lambda**: self.opBtn('\*'))

self.b12.grid(row=3, column=3)

self.b13 = tk.Button(newWindow, text='0', width=25, command=**lambda**: self.numBtn('0'))

self.b13.grid(row=4, column=0)

self.b14 = tk.Button(newWindow, text='C', width=25, command=**lambda**: self.clear())

self.b14.grid(row=4, column=1)

self.b15 = tk.Button(newWindow, text='=', width=25, command=**lambda**: self.resultBtn())

self.b15.grid(row=4, column=2)

self.b16 = tk.Button(newWindow, text='/', width=25, command=**lambda**: self.opBtn('/'))

self.b16.grid(row=4, column=3)

*# C 클릭시 발생하는 함수 / 초기화 기능을 담당*

**def** clear(self):

self.num1 = **None**

self.num2 = **None**

self.op = **None**

self.resultWindow.config(text='')

*# 숫자 클릭시 발생하는 함수*

**def** numBtn(self, num):

txt = self.resultWindow.cget('text')

txt += num

**if** self.op **is** **None**:

self.num1 = txt

self.resultWindow.config(text=self.num1)

**else**:

self.num2 = txt

self.resultWindow.config(text=self.num2)

*# 연산자 클릭시 발생하는 함수*

**def** opBtn(self, op):

self.op = op

self.resultWindow.config(text='')

*# = 버튼 클릭시 발생하는 함수*

**def** resultBtn(self):

**if** self.num1 **is** **not** **None** **and** self.num2 **is** **not** **None**:

**if** self.op == '+':

self.num1 = int(self.num1) + int(self.num2)

self.num2= **None**

self.resultWindow.config(text=self.num1)

**elif** self.op == '-':

self.num1 = int(self.num1) - int(self.num2)

self.num2 = **None**

self.resultWindow.config(text=self.num1)

**elif** self.op == '\*':

self.num1 = int(self.num1) \* int(self.num2)

self.num2 = **None**

self.resultWindow.config(text=self.num1)

**elif** self.op == '/':

self.num1 = int(self.num1) / int(self.num2)

self.num2 = **None**

self.resultWindow.config(text=self.num1)

**def** main():

connect = chatUi()

connect.run()

main()

5. 구현 여부

▶ 지우기 버튼을 누를 시 누르는 그 직후에만 채팅창이 초기화되고, 다시 입력을 시작하면 원래 있던 대화

기록이 출력되게 되었습니다.서버 쪽에서 클라이언트들의 모든 채팅기록 들을 한 변수에 모두 저장하고

송신하는 방식이라 이 저장된 변수 쪽에서 문제가 생긴 것 같습니다. 변수에 모든 기록을 저장하는 방법

이외에 방법이 생각이 나지 않아서 해결하지 못했습니다

▶ 모든 접속자들이 오른쪽 창에 닉네임, IP, PORT 번호가 출력되게 하려고 했으나, 구현을 하고 나니

각 클라이언트 자기 자신의 정보만 출력되고 새로운 접속자가 들어와도 추가가 되지 않았습니다.

이 문제를 해결하지 못하여 접속한 서버의 정보만을 출력하게 GUI 및 기능을 바꾸었습니다.

▶ 입력 창에 “/종료” 를 입력하면 서버와의 접속을 끊어지게 구현을 하려했으나,구현한 코드가 제대로 기능을

하지 않아서 버튼 형식으로 바꾸게 되었습니다.

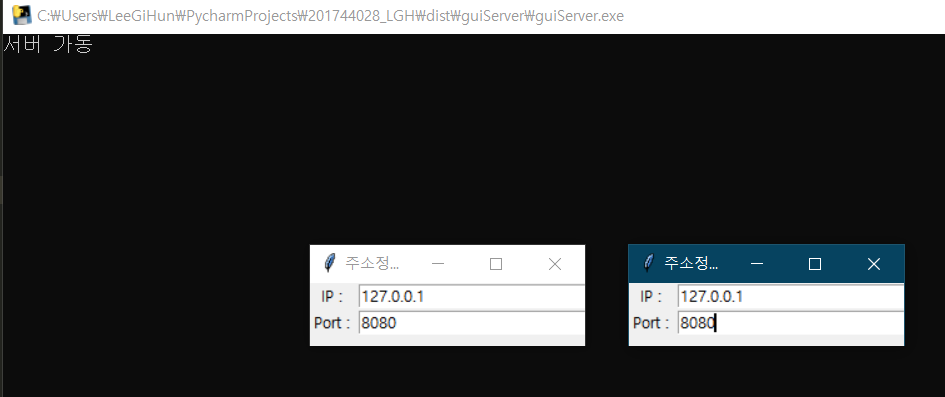
▶ 처음 실행 시 위의 GUI설계대로 IP와 PORT번호를 입력하고 서버와 연결될 수 있게 구현을 할려했으나,

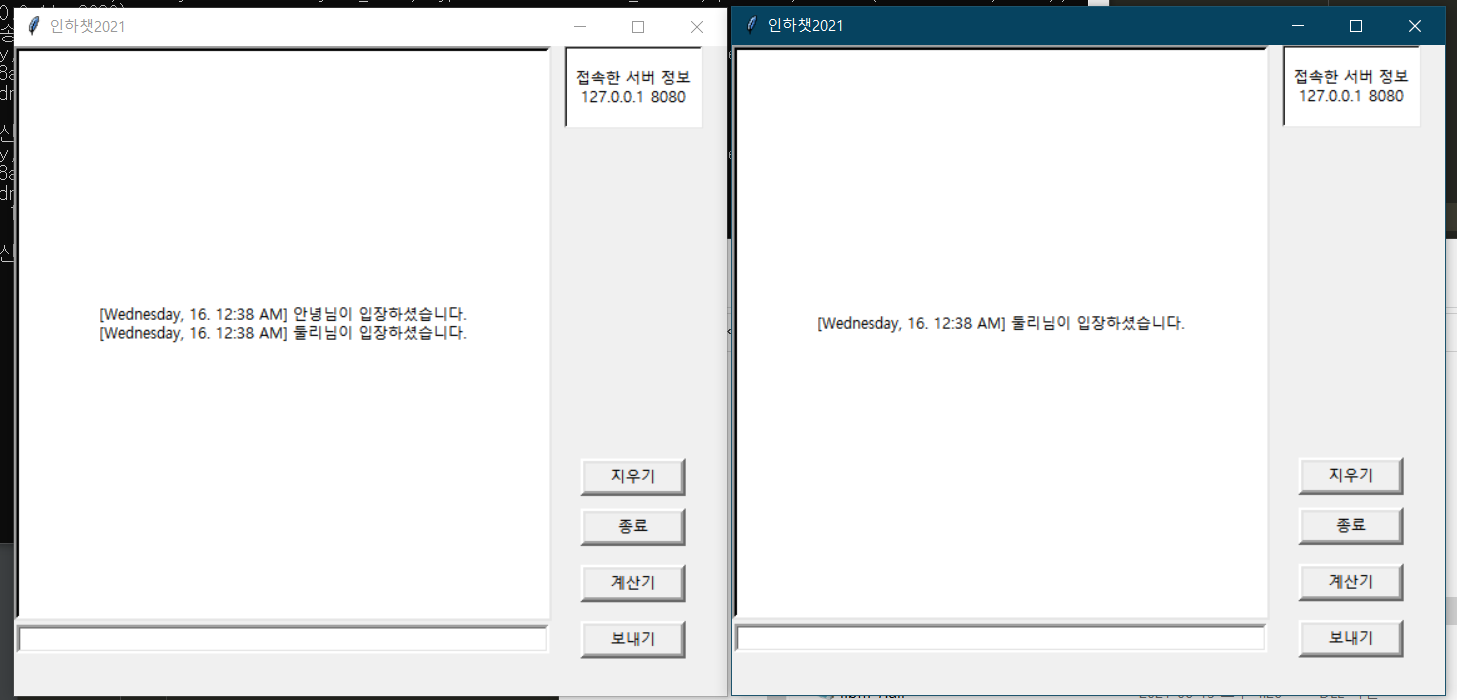
an integer is required (got type str)라는 에러를 해결하지 못하여 코드에 적어서 가동하였습니다

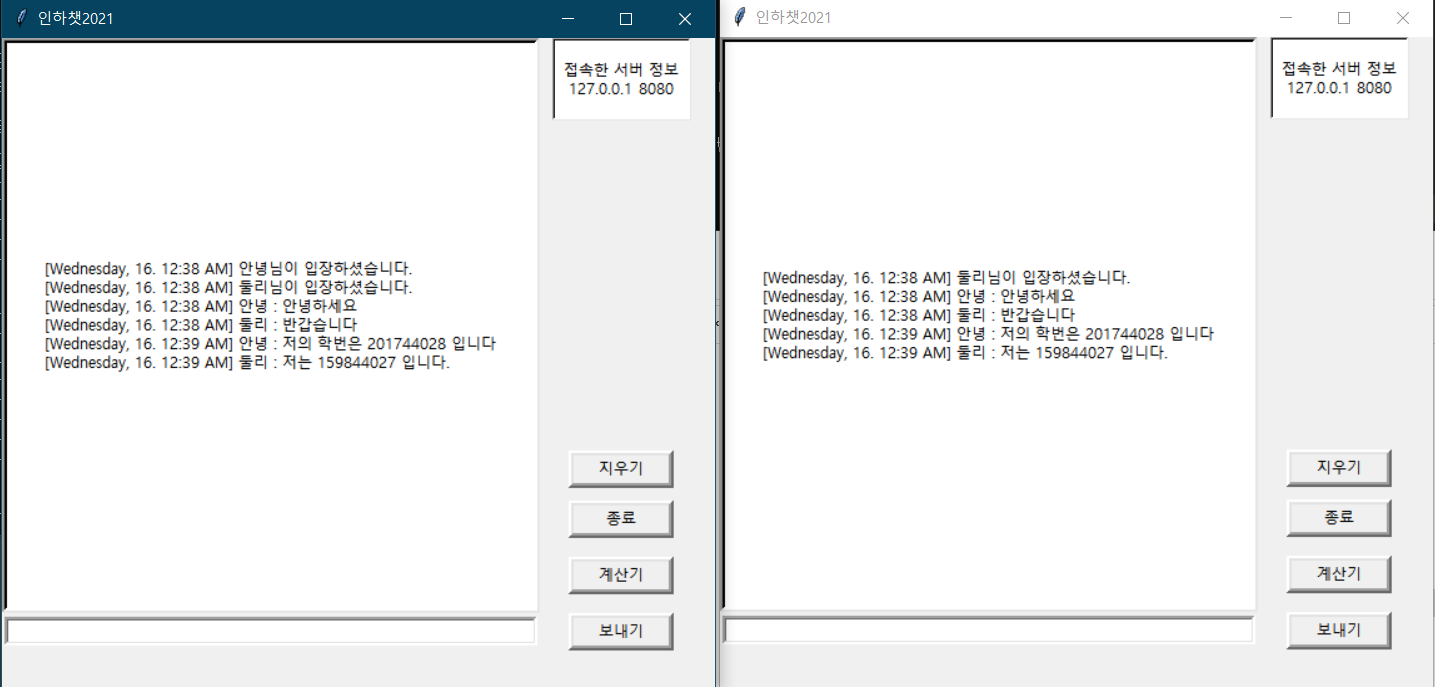
▶ 종료버튼을 누르면 누른 클라이언트 1개만 종료되지않고 다른 클라이언트도 모두 종료되게됩니다.

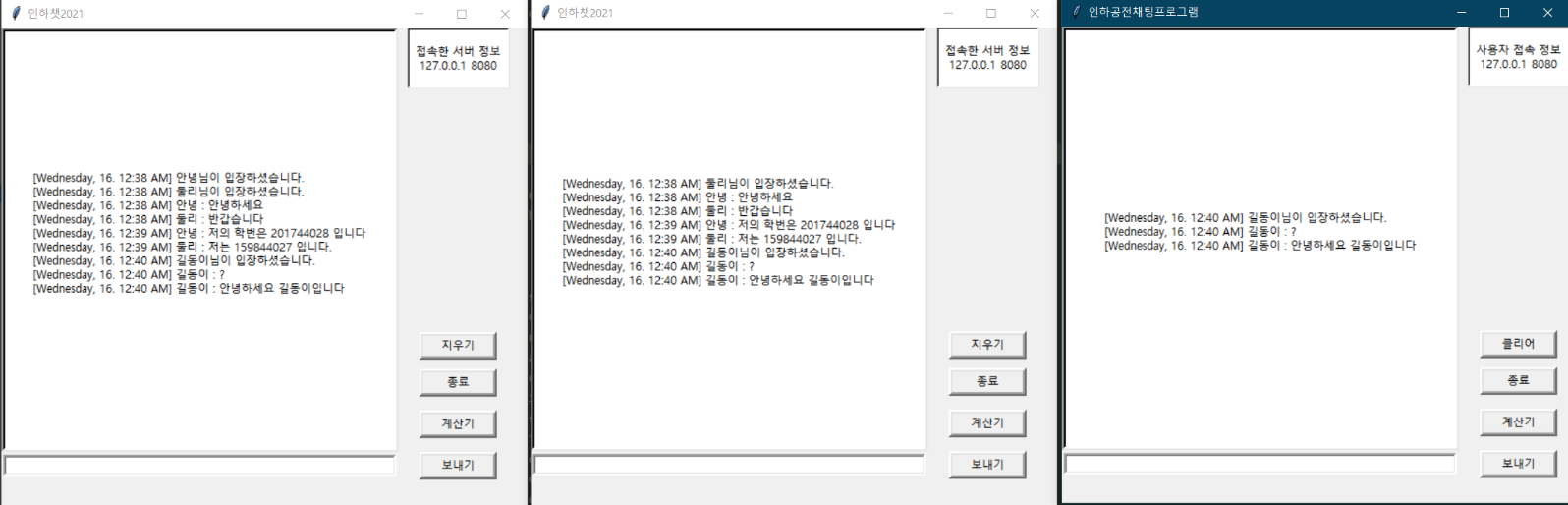
의도는 누른 클라이언트만 종료되게 하는 것이지만 거기까지 구현하지 못했습니다.

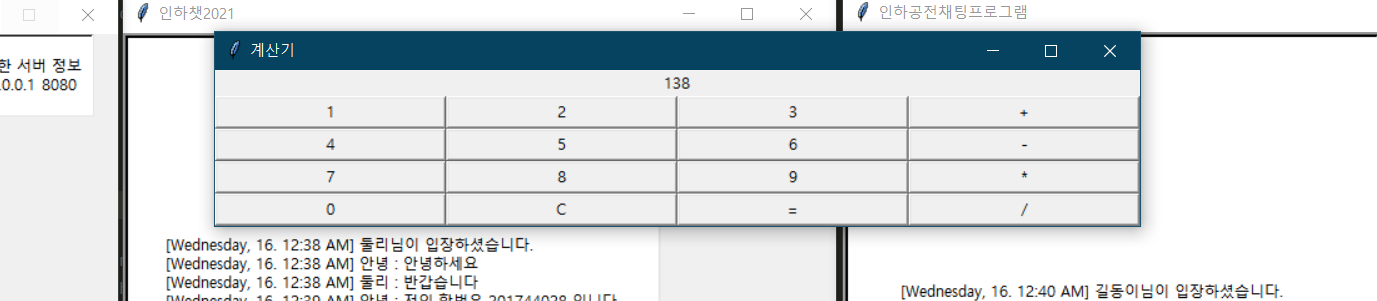
6. 실행화면

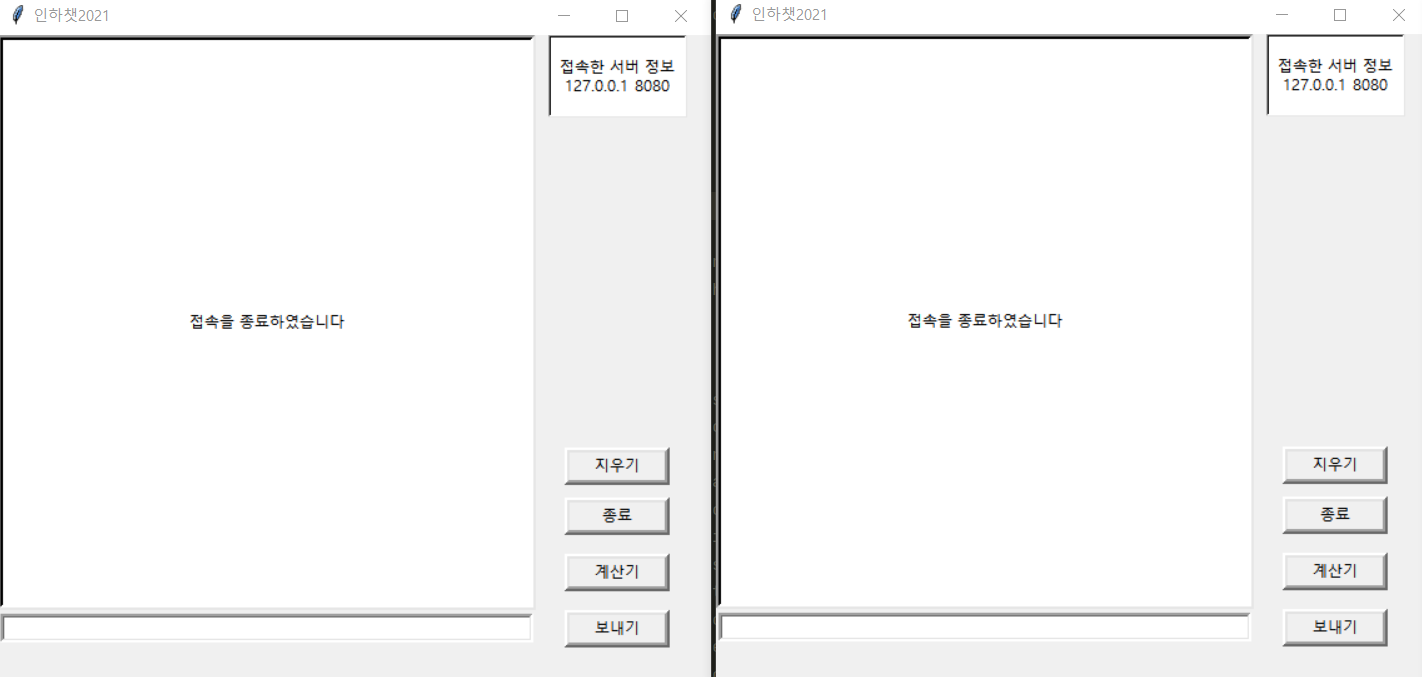
(클라이언트 두 명 / 주소 입력 후 엔터버튼 누르면 채팅방이 열림)

(클라이언트 2명 입장)

(동시 채팅)

(클라이언트3 까지 합류한 뒤 채팅)

(계산기 기능)

(종료버튼을 눌렀을 시)

(대화내용이 기록된 모습)

