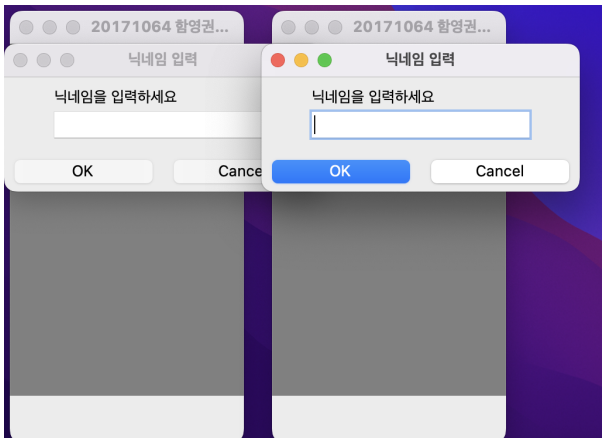
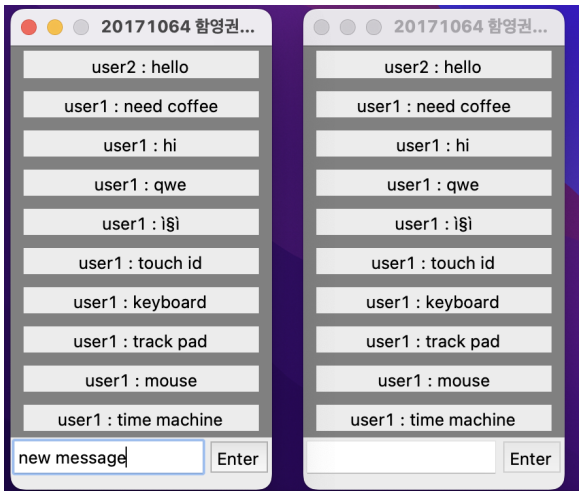
20171064 함영권 - 채팅 프로그램 만들기

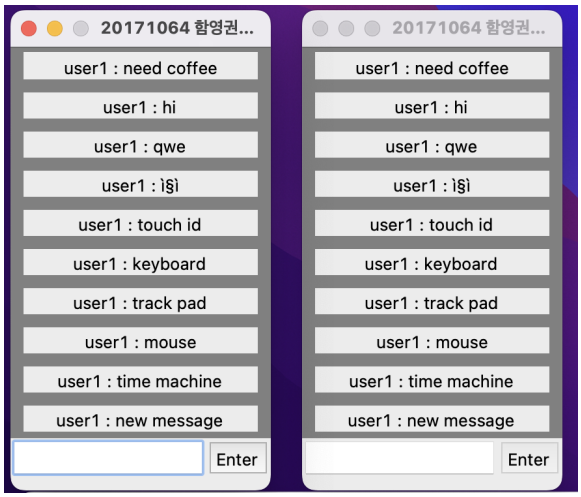
결과물 :



소스코드를 실행하면 닉네임을 입력받도록 함



메시지가 주고받기



메시지가 10개가 넘어가면, 가장 위의 메시지 삭제 후 새로운 메시지를 추가

결과물 : 서버 측 소스코드

```
from socket import *
from time import sleep
from multiprocessing import Process
from _thread import *
```

```
class User:
    def __init__(self, addr, name):
        self.addr = addr
        self.name = name


def recieve_from_client(sock):
    while True:

        msg = sock.recv(1024)
        # print(msg.decode('utf-8'))

        if len(msg) != 0:
            name = user_dict[sock].name
            print(name, '으로부터 받은 데이터 : ', msg.decode('utf-8'))
        else:
            msg = "check message".encode('utf-8')

        send_to_client(name, msg)
        sleep(1)

def send_to_client(nickname, message):
    message = nickname + " : " + message.decode('utf-8')
    client_sockets = user_dict.copy().keys()
    print(client_sockets)
    for client_socket in client_sockets:

        try:
            client_socket.send(message.encode('utf-8'))

        except Exception as e:
            print(e)
            if client_socket in user_dict:
                del user_dict[client_socket]


server_socket = socket(AF_INET, SOCK_STREAM)    # 서버의 소켓 생성 AF_INET = IPv4 의미
server_socket.bind(("", 8080))
server_socket.listen()

user_dict = dict()

while True:

    connected_socket, addr = server_socket.accept()    # accept() 가 실행될때까지 대기하게 됨

    print(connected_socket, addr)
    print()
    if connected_socket not in user_dict:
        nickname = connected_socket.recv(1024).decode('utf-8')
        user_dict[connected_socket] = User(addr, nickname)
        print(str(addr),'에서 접속이 확인되었습니다.', nickname)
    else:
        nickname = user_dict[connected_socket]

    # 서버 측 프로세스를 두개 이상 생성할 수 없어, 멀티 프로세싱을 사용하지 않고 멀티 스레딩을 사용합니다
    # p = Process(target=recieve_from_client, args=(connected_socket,))
    # p.start()
    start_new_thread(recieve_from_client, (connected_socket,))
```

결과물 : 클라이언트 측 소스코드

```
from socket import *
from tkinter import *
from _thread import *
from time import *
from collections import deque
from tkinter import simpledialog

def recieve_from_server(client_socket):
    # global entry
    while True:
        msg = client_socket.recv(1024)

        if len(msg) != 0:
            recieved_message = msg.decode('utf-8')
            print("기입창 내용 : ", entry.get())

            # 내가 전달한 메세지가 아니라면
            if entry.get() != recieved_message.split(" : ")[1]:
                if recieved_message == "check message":
                    pass
            # 내가 메시지를 전달했음이 확인되었으므로, entry의 문자열을 삭제한다.
            else:
                entry.delete(0, END)

            # 전체 메시지 갯수가 10개 이하인 경우
            if len(labels) < 10:
                label = Label(frame,text=msg,width=10, height=20)
                label.place(x=10, y=5 + (30 * len(labels) - 1), width=180, height=20)
                labels.append(label)

            # 메시지 갯수가 10개 이상인 경우, 가장 상단의 메시지는 없애고 새로운 메시지를 아래에 추가한다
            else:
                old_label = labels.popleft()
                print("old_label 삭제")
                print(labels)
                new_label = Label(frame,text=msg,width=10, height=20)

                labels.append(new_label)
                for i in range(len(labels)):
                    labels[i].place(x=10, y=5 + (30 * (i)), width=180, height=20)



        else:
            pass
            # label.insert(0, msg.decode('utf-8'))


clientSock = socket(AF_INET, SOCK_STREAM)
clientSock.connect(('127.0.0.1', 8080))



root = Tk()    # 메인 창인 Tk 객체의 인스턴스 생성
root.title("20171064 함영권 - 채팅 프로그램")
root.geometry("200x340")
root.resizable(False, False)

frame = Frame(root, background='gray')
frame.place(x = 0, y = 0, width=200, height=300)

nickname = simpledialog.askstring(title="닉네임 입력", prompt="닉네임을 입력하세요")
clientSock.send(nickname.encode('utf-8'))
```

```
# 가장 앞단의 요소들을 차례로 삭제할 것이므로 큐를 구현한 deque 사용
labels = deque()

entry = Entry(root)
entry.place(x=0, y=300,width=150, height=30)


enter_btn = Button(root, text="Enter", command=lambda: clientSock.send(entry.get().encode('utf-8')))
enter_btn.place(x=150, y=300, width=50, height=30)


start_new_thread(recieve_from_server, (clientSock,))

root.mainloop()
```

파이썬 싱글턴 객체 만들기

참고 : https://wikidocs.net/69361

아래처럼 클래스를 작성한 뒤, Server 클래스의 참조변수 s의 변수를 호출하려고 하면,

AttributeError 가 발생합니다.

```
class Server:
    def __init__(self):
        server_socket = socket(AF_INET, SOCK_STREAM)
        server_socket.bind(("", 8080))
        server_socket.listen()

s = Server()
print(s.server_socket)
```

`__init__` 메소드는 객체 생성 즉시 호출되는 것이 아니라고합니다. 자바의 생성자와는 조금 다른 것 같습니다.

아래와 같이 작성하여 싱글톤 객체를 생성합니다.

```
class Server:
    # 방법 1. 싱글톤 객체를 만들기에는 부적합
    #server_socket = socket(AF_INET, SOCK_STREAM)

    #방법 2.
    def __new__(cls, *args, **kwargs):
        if not hasattr(cls, "_instance"):
            cls._instance = super().__new__(cls)
            cls._instance.server_socket = socket(AF_INET, SOCK_STREAM)
        return cls._instance

    def __init__(self):
        self.server_socket.bind(("", 8080))
        self.server_socket.listen()

s = Server()
print(s.server_socket)
```

서버측 소켓을 포함하는 클래스를 싱글톤으로 하여, 문제를 해결하려고 했으나,

서버측 소켓은 하나의 프로세스에서만 실행 되어야 하는 것 같아 멀티 프로세싱을 사용하지 않고 멀티 스레딩을 사용했습니다.

사용자가 커넥션을 종료하게 되면, 서버측의 while 문이 불필요하게 수행되는 문제가 발생했습니다

커넥션을 종료한 소켓에 메시지를 전달하게 되면, 예외가 발생하는데

예외처리를 통해 클라이언트가 커넥션을 종료 했는지, 유지중인지 여부를 알 수 있습니다.

출처 : https://stackoverflow.com/questions/48024720/python-how-to-check-if-socket-is-still-connected

반복문을 통해 딕셔너리의 key를 순회하는 도중 key를 삭제했을 경우 발생하는 예외

copy() 명령어로 복사한 값을 순회하도록 합니다.

출처 : https://jeonghyeokpark.netlify.app/python/2020/12/17/python2.html