Let's implement a network chat program that supports multiple clients using UDP. First, in the given server/client code, the server supports only one client. Let's modify it to support multiple clients. Through this, we will learn the basics of how to actually implement UDP-based network communication. It provides fast real-time message transmission by utilizing the characteristics of connectionless UDP, and if we apply it in addition to message transmission, we can also add simple quiz or game functions.

**Overview**:

* Server: Receives UDP packets from multiple clients, processes the received data, and responds to all clients
* Client: Sends packets periodically to the server, receives responses from the server, and processes them
* Network structure: Multiple clients communicate with the same server using the connectionless UDP protocol

**Objectives**:

* Implements connectionless communication between multiple clients and servers using the UDP protocol.
* Establishes a basic network game environment where multiple clients communicate data with the server.

-------------------------------------------------------------------------------------------------------------------

.**Step0**.[5pt] After the instructor's explanation, implement and run the attached code as is.

* Single client support (no chat)

.**Step1**.[5pt] Modify the code to support multiple clients. [10:50 Sample code released]

* To track multiple clients, the server needs to store the IP address and port of each client
* Broadcast: When the server receives a message from a specific client, it needs to broadcast it to other stored clients
* Broadcast messages to clients asynchronously. This allows sending messages to multiple clients simultaneously, and while sending messages to each client, the server receives messages from other clients
* No need to change the client code
* Example server code:
  + <https://kmuackr-my.sharepoint.com/:u:/g/personal/102113_ms_kmu_ac_kr/ET_VrS-4gNlCrhsGUBHU8-cBhYImWF4xLeD1RlGhFtzqGw?e=z35W0O>

.**Step2**.[5pt] Add a function to set a nickname when a client connects and send chat messages through the nickname [11:20 Sample code released]

* Client nickname setting:
  + The client enters a nickname when the program starts. This nickname is sent as the first message to the server, and subsequent messages are processed using the nickname.
  + If a nickname is not entered, the nickname is set to "Anonymous" by default.
* Nickname processing on the server:
  + The server stores the message received when the client first connects as a nickname.
  + When receiving a message from that client, it broadcasts the message along with the nickname to other clients.
  + When a new client connects, the nickname of that client is sent to all other clients. For example, broadcasting a message such as "John has joined the chat"
  + Exame for Server side:
    - <https://kmuackr-my.sharepoint.com/:u:/g/personal/102113_ms_kmu_ac_kr/ESeMYe41ss1DqibKam1QltIBR_Op4evRmEghcUHN-t_J_Q?e=1lAeQp>
  + Exame for Client side:
    - <https://kmuackr-my.sharepoint.com/:u:/g/personal/102113_ms_kmu_ac_kr/EVADIdL6JI9Fh1MXDWA8Cf0B-UcoIES2ynlq-sN-jDNc8w?e=d5fRZe>

.**Step3**.[5pt] Add a feature to notify all other clients when a client leaves a chat room.

* Client exit handling:
  + When a client types the */exit* command, it sends a special string (e.g. *disconnect*) to the server and exits the chat room.
  + The client calls *udpClient.Close*() to disconnect from the server and terminate the program.
* Server exit handling:
  + When the server receives a disconnect message from a client, it removes the client from the list of connected clients.
  + It broadcasts a "*has left the chat*" message along with the client's nickname to all other clients.

-------------------------------------------------------------------------------------------------------------------

**Submission Guidlines:**

* Students who complete Step 1,2,3 within the recitation class will be checked and may leave
* Submission deadline after the recit class: By October 19 (Sat)
* Submission items: A single Word (docx) or pdf file containing the following two items
  + Final source code (text format, not image)
  + Result: Image capture of both running server and clients

■