Ovviamente, a differenza di quanto accade per **TCP**, non assoceremo ad un **thread** la connessione dato che qui si parla di uno scambio di messaggi tra processo **Client** (o più client) **– processo Server** (unico).

**Immagine che contiene testo, diagramma, schermata, Piano

Descrizione generata automaticamente**

Esiste la possibilità di creare un Server UDP che lavori con più thread, però non ce ne fotte niente.

Sì, è possibile creare un ServerUDP multithreaded. Tuttavia, a differenza di un server TCP, non è necessario creare thread multipli per gestire più client contemporaneamente. Ciò è dovuto al fatto che la comunicazione con i client avviene tramite datagrammi UDP inviati su un canale condiviso. [Pertanto, un singolo thread può gestire la ricezione dei datagrammi da tutti i client e decidere come gestirli](https://stackoverflow.com/questions/33984719/why-a-udp-server-need-not-be-multi-threaded-to-serve-several-clients-concurrentl)

**UDP Server**

Ascolta i **pacchetti** **UDP** in arrivo sulla porta 8050, e quando ricevuto viene letto e stampato per poi inviare un messaggio di Ok al Client.

Si sfrutta la **classe**  **DatagramSocket** per creare una **socket** per l’invio e ricezione dei datagram.

Dichiaro la struttura del packet da ricevere dalla socket, e poi ne ricavo i **dati** con la lunghezza con i metodi **packet**.**getData** e **getLenght**.

Mandiamo un messaggio di risposta, definendo una stringa di OK, creando un **DatagramPacket** basandoci sui dati di questa stringa, la lunghezza e ovviamente necessitando di **indirizzo** e **port** del **Client** sfruttiamo il **pacchetto** che è stato ricevuto in precedenza per ricavare i dati.

**UDP Client**

Sarà l’inverso, e ovviamente per il primo invio dobbiamo necessariamente conoscere l’**indirizzo** e il **port** su cui è in ascolto la **socket Server**.