Documentazione del progetto:

Chat Server-Client

con la possibilità di avere più Client

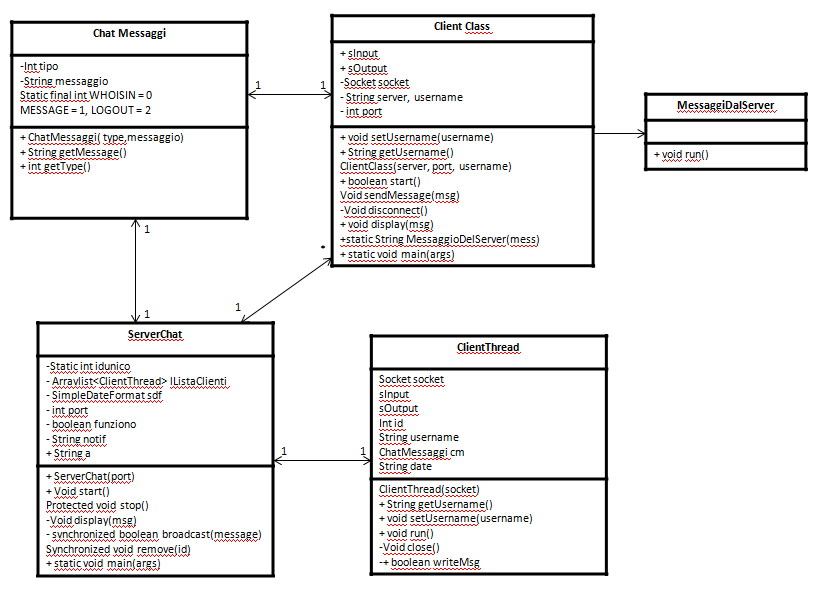
(Multi-Thread)

Mirco Fossi 5BIA

1. **Introduzione**

Il progetto descritto è la creazione, attraverso java, di un programma che sia composto da un server unico e da diversi client che hanno la possibilità di connettersi al server e potersi scambiare messaggi. Tutti i messaggi che i client connessi si scambiano vengono prima inviati al Server, che ha il compito di rinviare i messaggi ai client, tranne a colui che lo ha inviato.

1. **Diagramma delle classi**



* 1. **Classe ServerChat**

La classe ServerChat è la classe del server, cha ha il compito di gestire tutti i collegamenti con i client, indirizzare i messaggi verso i client desiderati e tenere uno schema con tutti i clienti connessi alla chat. E’ composto da diverse variabili, come l’idunico, che viene assegnato ad ogni cliente connesso, così da distinguerli, la porta che indica la porta sul quale il server sta girando, e dove i client devono connettersi, la ListaClienti che è un array nel quale vengono inseriti tutti i clienti che partecipano alla chat. Oltre al compito di dover inviare i messaggi ai client, controllo anche la connessione che ogni client effettua, ovvero controlla che un client, periodicamente, sia connesso alla chat. Ex(Se un client invece di disconnettersi chiude la connessione, il server, attraverso un controllo, invia un messaggio a tutti i client connessi, e se il messaggio da esito negativo, ciò significa che il client si è disconnesso, avvisa gli altri client che quello specifico client ha abbandonato la chat, e lo rimuove dalla lista dei clienti. Quando riceve un messaggio da inviare ad un solo client, va a fare il controllo sul nome del client destinatario con la lista dei client presenti nella chat; se trova il destinatario, gli invia il messaggio, altrimenti invia un messaggio di errore al client mittente.

* 1. **ClientThread**

La classe ClientThread viene creata quando un client si connette al server, e prende tutti i valori del client connesso. La classe viene utilizzata cosi da poter creare più client e poterli connettere allo stesso server, creando così una comunicazione server-Multiclient, e quindi multi-thread. La classe si occupa anche di inviare i messaggi del client al server, e di ricevere messaggi dal server, sia messaggi che sono stati inviati direttamente dal server, sia inviati da altri client. La classe fa anche un controllo sul diverso tipo di messaggio, ovvero se il messaggio è un messaggio per tutti i partecipanti alla chat, oppure se è un messaggio per un solo client, oppure se sono richieste da parte del client. I tipi di messaggio sono spiegati nella classe: ChatMessaggi.

* 1. **ClientClass**

La classe ClientClass serve per gestire il singolo cliente, contiene gli stream di input e di output, che servono per vedere i messaggi dal server o inviare messaggi al server, inoltre contiene l’username, ovvero il nome del cliente che viene chiesto all’avvio della classe, e infine la porta del server al quale deve collegarsi.

* 1. **MessaggiDalServer**

Serve per far visualizzare al cliente i messaggi che provengono direttamente dal server.

* 1. **ChatMessaggi**

La classe ChatMessaggi ha il compito di gestire il messaggio in se per se, ovvero verificare se è composto da un codice, oppure no. Infatti ci sono 3 categorie di messaggio:

1 WHOISIN: che serve per far visualizzare a schermo i partecipanti alla chat

2 LOGOUT: che serve per disconnettersi dalla chat

3 MESSAGE: che contiene il messaggio in se per se da inviare agli altri clienti. Il messaggio può essere per tutti, allora viene scritto normalmente, se invece un cliente deve scrivere un messaggio ad un singolo cliente, prima del messaggio deve scrivere @Nome\_Del\_Client MESSAGGIO, così il messaggio arriverà solamente al destinatario scritto accanto alla @. Se il destinatario non esiste si riceve un codice di errore da parte del server, che ci avviserà dell’assenza del client alla chat.

1. **Javadoc**

Il javadoc è una documentazione che viene effettuata attraverso un inizio ed una fine. L’inizio è scaturito dalla seguente sintassi “ /\*\* ”, e la conclusione dalla seguente “ \*/ ”. Se nel programma sono inseriti documenti a lato, o fra linee di codice, segnati attraverso “ // ”, vengono ignorati dal javadoc. All’interno del javadoc vengono inserite le caratteristiche di un metodo o di una classe. Le caratteristiche vengono accompagnate da un intersezione, ad esempio per indicare un parametro c’è scritto @param, alla quale accanto scriveremo una variabile, o un metodo, e spiegheremo il suo funzionamento; oppure @return, dove dobbiamo associare la variabile, se c’è, che il metodo ritorna.