1. Introduzione e formulazione del problema:

L'obiettivo di questo esercizio è realizzare una chat punto-punto che sfrutti il protocollo TCP per la comunicazione. Sia client che server devono poter mostrare sullo standard output i propri messaggi e quelli dell'host con cui comunicano e devono poter chiudere la comunicazione in qualsiasi momento.
Si deve quindi scegliere un prompt chiaro per comprendere chi sta parlando e si deve mostrare l'output del client e del server in due colori diversi in entrambi gli host.
Nella comunicazione, oltre al testo libero, si possono scambiare dei messaggi speciali noti ad entrambi gli host. I tipi di messaggio che possono essere inviati devono essere visualizzabili sullo standard output all'instaurazione della connessione e ogni volta che ce ne sia bisogno.

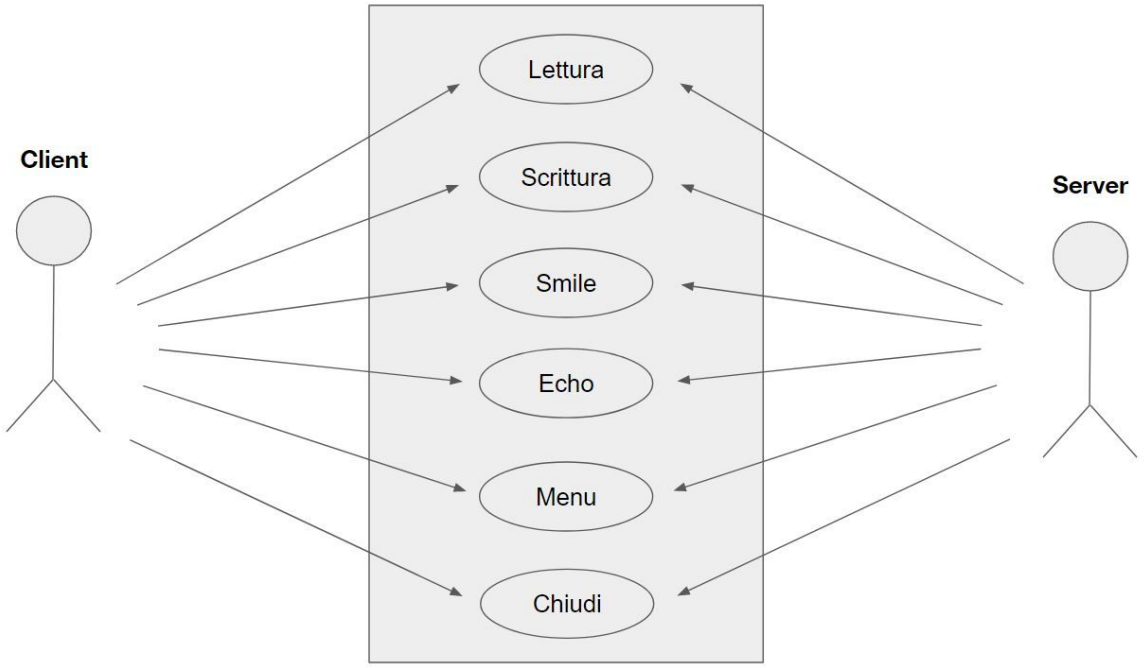
2.Descrizione dell'architettura dell'applicazione (componenti hardware e software)

Per la realizzazione di questo esercizio abbiamo utilizzato l'ambiente di sviluppo Netbeans installato su una macchina con sistema operativo Ubuntu 16.04 LTS (Elmi Christian) e una macchina con sistema operativo Windows 10 (Mele stavri). Per entrambe le macchina la versione di Netbeans utilizzata è la 8.2.

2.1 Attori

Gli attori principali dell'applicazione sono il Server e il Client

2.2 Diagramma dei casi d'uso



2.3 Descrizione dei casi d'uso

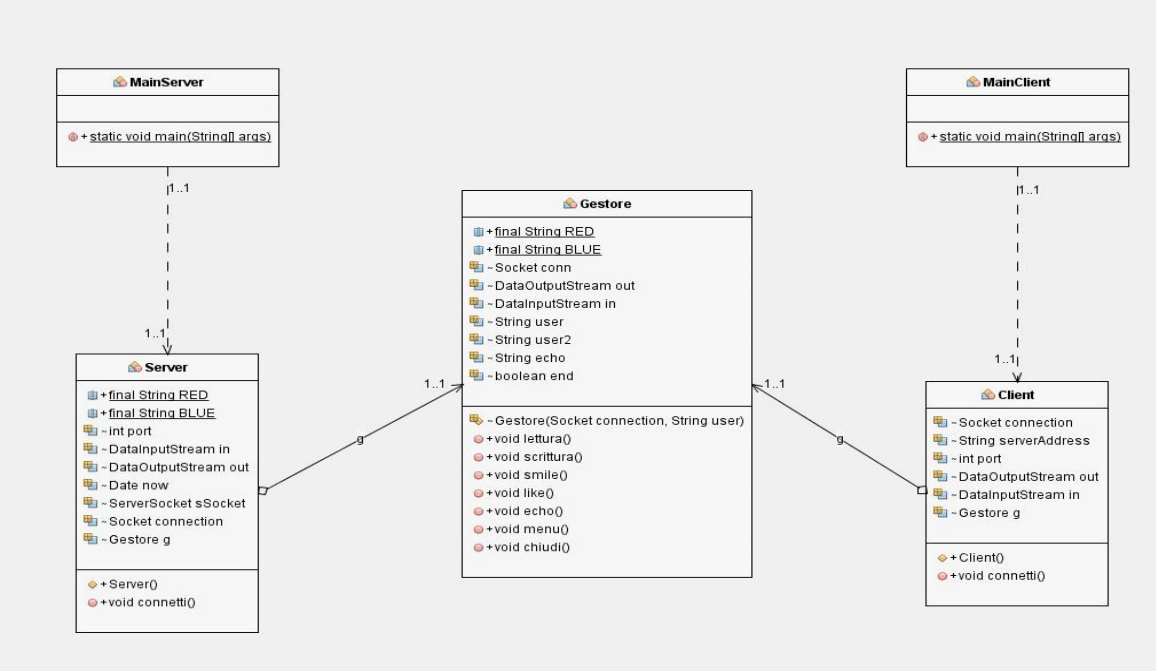
Caso d'uso: Scrittura
Id:
Attori: Client, Server
Scenario Principale: Scrittura
Precondizioni: 1. é stata instaurata la connessione 2. è stato istanziato un oggetto DataOutputStream
Sequenza degli eventi: 1. viene letto il messaggio da inviare da tastiera 2. viene inviato il messaggio
Postcondzioni: 1. si passa al messaggio successivo
Scenari alternativi: Errore di I/O o connessione chiusa

Caso d'uso: Lettura
Id:
Attori: Client, Server
Scenario Principale: Lettura
Precondizioni: 1. é stata instaurata la connessione 2. è stato istanziato un oggetto DataInputStream
Sequenza degli eventi: 1. viene letto il messaggio 2. viene stampato il messaggio
Postcondzioni: 1. si passa al messaggio successivo
Scenari alternativi: Errore di I/O o connessione chiusa

2.4 Vincoli e tecnologie usate

L'applicazione non ha vincoli relativi al tipo di sitema operativo o componenti hardware. È stata testata sia su macchina windows che su macchina linux e risponde allo stesso modo su entrambe.

3. Diagramma delle classi



4. Descrizione e test dell’applicazione

L’applicazione simula una chat ed è scritta in linguaggio Java. é possibile scambiare messaggi testuali e smile e in qualsiasi momento è possibile abbandonare la chat chiudendo la connessione. Nel nostro caso, sia il server che il client risiedono nella stessa macchina, ma è possibile far risiedere essi su due macchine diverse. L’applicazione è stata realizzata sfruttando gli oggetti Socket di java e gli Stream.
L’applicazione è stata testata eseguendo più volte e in ordine diverso tutti i suoi possibili metodi, in modo tale da poter scovare eventuali errori e correggerli. Questa versione, la più aggiornata sembra non presentare evidenti problemi. Il funzionamento è molto semplice,il server si mette in ascolto e il client si connette così si è instaurata una connessione. Ora il server può dar via alla chat vera e propria inviando reciprocamente messaggi.

4.1 Server in ascolto e instaurazione della connessione

```
In attesa di connessioni!
Connessione stabilita!
Socket server: /127.0.0.1:2000
Socket client: /127.0.0.1:54988

Premi 2 per chiudere la connessione
Premi 3 per inviare uno smile
Premi 4 per inviare un like (beta)
Premi 5 per inviare lo stesso messaggio ricevuto
Premi invio per digitare un messaggio
```

4.2 Connessione aperta

```
Connessione aperta
```

4.3 Invio di uno smile

```
In attesa di connessioni!
Connessione stabilita!
Socket server: /127.0.0.1:2000
Socket client: /127.0.0.1:54988

Premi 2 per chiudere la connessione
Premi 3 per inviare uno smile
Premi 4 per inviare un like (beta)
Premi 5 per inviare lo stesso messaggio ricevuto
Premi invio per digitare un messaggio
3
```

4.4 Ricezione

```
Connessione aperta
Server: ☺

Premi 2 per chiudere la connessione
Premi 3 per inviare uno smile
Premi 4 per inviare un like (beta)
Premi 5 per inviare lo stesso messaggio ricevuto
Premi invio per digitare un messaggio
```

4.5 Invio dello stesso messaggio ricevuto

```
Connessione aperta
Server: ☺

Premi 2 per chiudere la connessione
Premi 3 per inviare uno smile
Premi 4 per inviare un like (beta)
Premi 5 per inviare lo stesso messaggio ricevuto
Premi invio per digitare un messaggio
5
Client: ☺
```

4.6 Ricezione

```
In attesa di connessioni!
Connessione stabilita!
Socket server: /127.0.0.1:2000
Socket client: /127.0.0.1:54988

Premi 2 per chiudere la connessione
Premi 3 per inviare uno smile
Premi 4 per inviare un like (beta)
Premi 5 per inviare lo stesso messaggio ricevuto
Premi invio per digitare un messaggio
3
Client: ☺

Premi 2 per chiudere la connessione
Premi 3 per inviare uno smile
Premi 4 per inviare un like (beta)
Premi 5 per inviare lo stesso messaggio ricevuto
Premi invio per digitare un messaggio
```

4.7 Invio di un messaggio testuale

```
Premi 2 per chiudere la connessione
Premi 3 per inviare uno smile
Premi 4 per inviare un like (beta)
Premi 5 per inviare lo stesso messaggio ricevuto
Premi invio per digitare un messaggio
3
Client: ☺

Premi 2 per chiudere la connessione
Premi 3 per inviare uno smile
Premi 4 per inviare un like (beta)
Premi 5 per inviare lo stesso messaggio ricevuto
Premi invio per digitare un messaggio

Server: ciao client, sono un server che sta mandando un messaggio!
```

4.8 Ricezione

```
Premi 2 per chiudere la connessione
Premi 3 per inviare uno smile
Premi 4 per inviare un like (beta)
Premi 5 per inviare lo stesso messaggio ricevuto
Premi invio per digitare un messaggio
5
Client: ☺Server: ciao client, sono un server che sta mandando un messaggio!

Premi 2 per chiudere la connessione
Premi 3 per inviare uno smile
Premi 4 per inviare un like (beta)
Premi 5 per inviare lo stesso messaggio ricevuto
Premi invio per digitare un messaggio
```

4.9 Chiusura della connessione

```
Premi 2 per chiudere la connessione
Premi 3 per inviare uno smile
Premi 4 per inviare un like (beta)
Premi 5 per inviare lo stesso messaggio ricevuto
Premi invio per digitare un messaggio

Server: ciao client, sono un server che sta mandando un messaggio!
Client HA CHIUSO LA CONNESSIONE
BUILD SUCCESSFUL (total time: 3 minutes 12 seconds)
```

```
Premi 2 per chiudere la connessione
Premi 3 per inviare uno smile
Premi 4 per inviare un like (beta)
Premi 5 per inviare lo stesso messaggio ricevuto
Premi invio per digitare un messaggio
2
HAI CHIUSO LA CONNESSIONE
BUILD SUCCESSFUL (total time: 2 minutes 58 seconds)
```

5.0 Collaborazione

L'applicazione è stata svolta in collaborazione tra Elmi Christian e Mele Stavri. Per lo svolgimento del lavoro non abbiamo adottato tecniche di suddivisione del lavoro ma di confronto. Perciò il lavoro è stato eseguito insieme, sia per il codice sia per la relazione. Tutto l'avanzamento del lavoro è stato approvato da entrambi, data anche la triennale collaborazione.