IRC-chatt via telnet

Instruktioner

Anropa main-funktionen i Server-klassen (utan arguments) för att starta servern. Efter det kan du starta main-funktionen i Client (utan argument) en gång för varje användare du vill emulera. De kommer att kopplas till servern automatiskt via localhost. För varje klient får du nu skriva in serverns lösenord, ”yeet” för att logga in på servern. På varje konsol kan du nu chatta och och interagera med de andra konsolerna.

Servern består av olika rum, där användare endast kan interagera med andra användare i samma rum. Som default skapas rummet ”Lobby” när servern startas där alla användare anländer först.

Om du inputtar en rad text utan ’/’ i början till terminalen skickar du helt enkelt det du skrev som meddelande till alla andra användare i samma rum.
https://github.com/lumpor/IRC.git
Annars finns dessa kommandon att välja mellan:

/nick *param*
Ändrar ditt användarnamn till *param*.

/rooms
Skriver ut vilket rum du är i och vilka rum som finns tillgängliga.

/create *param*
Skapar ett rum med namnet *param*.

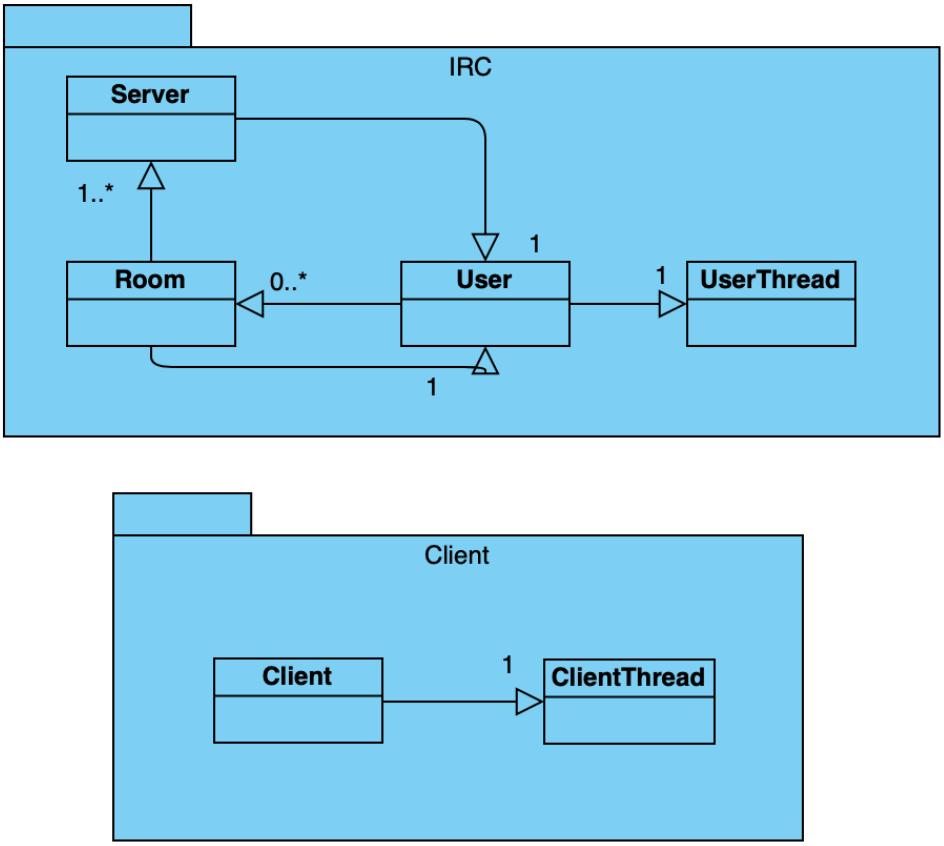
/users
Listar alla användare i rummet du befinner dig i.

/kill *param*
Bannar användaren *param*

/nuke *param*
Förstör rummet *param*. Alla användare i rummet notifieras och skickas till lobbyn.
/join *param*
Får dig att joina rummet *param*.

/exit
Loggar ut dig.

Uppbyggnad



Server skapar instanser av User och Room, och innehåller ett visst antal Room. Room innehåller ett visst antal users. Ett User-objekt har reda på rummet de befinner sig i och i vilken Server de är skapade.

En UserThread skapas när ett User-objekt skapas. Den innehåller en User och sköter den användarens server-interaktion.

Client startar en ClientThread som sköter data som användaren får från servern medans Clients egna main-metod lyssnar på input från användaren och skickar det till servern.

Kommentarer

För att kunna koppla mellan olika hosts krävs endast små ändringar i main-metoderna på Server och Client.

Koden har tyvärr cykliska beroenden mellan Server, Room och Client. Dessutom har Room en funktion, getUsers(), som returnerar dess inbördes lista av users, vilket bryter mot lokalitetsprincipen. Båda dessa problem beror främst på att jag ville lokalisera alla anrop till write() i en klass (User). Annars hade jag behövt skicka en Socket eller en User som parameter till servern/rummen för att de skulle kunna skriva till klienten.

Många exception-fall är omhändertagna. Så vitt jag vet finns det inget sätt att krascha servern från klient-sidan.

I och med att performCommand, write, logout och alla funktioner i server är synchronized borde programmet vara thread-safe.

Lambdas används i user-klassen för att skapa anonyma funktioner för att representera olika commands.

Hoppas ni har fina meningsfulla konversationer på servern ☺

//Fabian