JavaIM

Dario Andrea Raimondi
706021
carotinobg@yahoo.it

JavaIM è un programma di Instant Messaging realizzato in Java, da presentare come progetto per il corso di **Sistemi di Elaborazione dell'Informazione** tenuto dal professor Damiani presso il DTI di Crema.

Visione d'insieme

JavaIM è un'applicazione multithreaded di instant messaging, realizzata tramite un server aperto su di una porta non nota (8080 di default) il quale accetta connessioni TCP da utenti remoti, ed è anche in grado di effettuare chiamate verso server remoti.

È stato realizzato un protocollo minimale per gestire la chiamata e l'accettazione, e un'interfaccia grafica per organizzare il tutto.

Il programma è scritto in Java; il codice è stato steso con NetBeans 5.5.

Istruzioni per l'uso

Per lanciare JavaIM da console:

```
java –jar javaim2 [porta]
```

Il parametro **porta** è opzionale. Se non indicato, il server viene aperto sulla porta 8080.

La finestra principale permette di impostare il nome dell'utente, che viene stabilito all'avvio del programma tramite il riconoscimento del nome dell'utente attuale del sistema operativo, e i parametri di chiamata. Il nome viene usato per l'identificazione nella conversazione.

L'area in basso nella finestra serve per notificare all'utente eventuali messaggi di errore.

La chiusura della finestra principale termina il programma.

La chiamata avviene specificando un'indirizzo IP ed una porta. Il programma provvederà a tentare di stabilire una connessione col server in ascolto su quell'IP ed a quella porta, e a negoziare una conversazione.

In caso invece di ricezione di richieste di conversazione, all'utente verrà chiesto se accettare o no.

La finestra di chat è composta da tre aree di testo. Nell'area superiore vengono visualizzati i messaggi locali e remoti degli utenti impegnati nella conversazione. Al centro si scrivono i messaggi che si vogliono inviare (premere Invio durante la digitazione o il pulsante Invia hanno lo stesso effetto di inviare il messaggio sul socket). In basso sono indicati eventuali messaggi di stato o di errore.

Chiudendo la finestra di chat si interrompe la conversazione. È anche possibile interromperla scrivendo il messaggio **/quit**. In questo caso la finestra non verrà chiusa. La finestra non verrà chiusa nemmeno se l'utente remoto terminerà la conversazione.

Dettagli tecnici

La classe principale si chiama **Main**. Il metodo statico **public static void main(String[] args)** crea un nuovo thread contenente un'istanza di **Main**. Questa a sua volta crea la finestra principale ed un altro thread che gestisce il server.

Il server, quando riceve una chiamata, istanzia una nuova classe **Ricezione** in un nuovo thread. Questa si occuperà di gestire il protocollo, il socket e la finestra di chat relativa a quella conversazione.

L'attivazione del bottone **Chiama** dalla finestra principale istanzia la classe **Chiamata** in un nuovo thread. Questa si occuperà di chiamare il server remoto, negoaziare la conversazione e gestire una finestra di chat.

La comunicazione tra i vari thread è affidata alle interfacce **MainInterface**, **FinestraChatInterface**, **FinestraPrincipaleInterface**, **ComunicazioneInterface**. Esse espongono i metodi necessari ad un thread per comunicare con gli altri. Alla creazione di una nuova istanza di una classe viene passata un'istanza di una delle interfacce qui menzionate, per garantire la comunicazione.

Per conoscere i dettagli implementativi si può dare un'occhiata al codice, che è stato commentato, e alla documentazione JavaDOC generata automaticamente per il progetto.

Il protocollo

Il protocollo è molto semplice. Tutti i messaggi sono seguiti da un indicazione di nuova riga, per identificarli in modo semplice.

Quando viene inviata una richiesta di conversazione, dopo l'accettazione e il passaggio del nuovo

socket a **Ricezione**, il ricevente rimane in ascolto . Il chiamante invia il proprio nome utente. Il ricevente chiede all'utente se vuole conversare con l'utente remoto. In caso di risposta affermativa, il ricevente invia sul socket il messaggio **OK** seguito da un nuovo messaggio contenente il proprio nome. A quel punto la conversazione è iniziata. Se l'utente rifiuta la conversazione, viene inviato il messaggio **NO**.

Durante la conversazione, l'unico comando riconosciuto è **/quit**, che serve per interromperla.