Immagine che contiene testo, Biglietto Post-it, Carattere, schermata

Descrizione generata automaticamente

Partiamo col definire l’**interfaccia** del **servizio**, dunque **ICounter**.

Poi lavoriamo lato **Client**: il **proxy** va implementato in modo che **CounterProxy** implementi l’interfaccia **ICounter** gestendo la **comunicazione**, e dando un oggetto **CounterProxy** al **Client** che andrà ad invocarne i metodi. In questo modo avremo come l’impressione che i metodi siano locali al Client.

Per dare al **Client** l’oggetto **counterProxy** facciamo:

ICounter counter = new CounterProxy(host, 2500);

Questo perché?   è una buona pratica di programmazione perché consente di scrivere codice più modulare e flessibile. In questo modo, se in futuro si desidera utilizzare un’implementazione diversa dell’interfaccia ICounter, sarà sufficiente modificare la riga 4 del codice senza dover modificare il resto del codice che utilizza l’interfaccia.

A questo punto il **Client** può invocare i metodi forniti dall’interfaccia.  
Nell’esempio si fa vedere come la funzione può essere passata da terminale, abbuò nun serv è sul p’allungà o bror.

**CounterProxy**

Implementa l’interfaccia ICounter (servizio).

**Costruttore**: ovviamente necessita dell’indirizzo del server composto da **host** e **2500**.

**Metodi**, in cui si apre la **socket** verso quel determinato host e si apre una connessione TCP dichiarando **DataOutputStream** e **DataInputStream** sfruttando:

  DataOutputStream ostream= new DataOutputStream (new BufferedOutputStream(soc.getOutputStream()));

            DataInputStream istream = new DataInputStream(new BufferedInputStream(soc.getInputStream()));

[L’oggetto BufferedOutputStream può scrivere più byte alla volta, riducendo il numero di chiamate al sistema sottostante e migliorando le prestazioni](https://docs.oracle.com/javase/8/docs/api/java/io/BufferedOutputStream.html).

Il parametro che si dà a **DataOutputStream** e **Input** sono principalmente degli oggetti di classi che estendono la **classe astratta OutputStream**, come appunto **Buffered**,**File**,**ByteArray** che semplicemente specificano dove viene scritto il **dato** /letto.

A questo punto si invia il **messaggio** tramite questo **stream di dati** da utilizzare.  
Ovviamente in questo caso, dato che si sfrutta il concetto di **buffered**, bisogno andare poi a fare un **flush**.

E alla fine ovviamente chiudere la socket.

**CounterSkel**  
In questo caso lo skeleton è implementato per **ereditarietà**, dunque semplicemente questa sarà una classe **abstract** che implementa l’**interfaccia ICounter** che all’interno ha la funzione **runSkeleton** in cui si **ricevono** le **richieste di connessione** dai **Client** e assegnando ai **SkeletonThread** poi l’onere di effettuare l’ **unmarshalling**.

**Counter Worker**

Foss o Thread, implementato per ereditarietà.  
Sarà questo a collegarsi allo **stream** associato alla socket che gli viene passata dal **CounterSkel** dal costruttore.