

Reti di Calcolatori L-A

Appello del 21/12/2005

Compito 2

Cognome:
Nome:
Matricola:

Tempo a disposizione: 2h

E' obbligatorio mettere Cognome Nome Matricola e Numero Compito all'inizio di ogni file sorgente, pena la non valutazione del compito, che viene stampato in modo automatico, solo in caso siano presenti gli elementi detti sopra.

Si devono consegnare **tutti i file sorgente e tutti gli eseguibili prodotti singolarmente** (per favore, solo quelli relativi ai file sorgente consegnati!!).

La prova intende valutare le capacità progettuali e di programmazione sia in **ambiente Java** che in **ambiente C**, pertanto è consigliabile sviluppare, *almeno in parte*, **entrambe** le soluzioni richieste. In entrambi gli esercizi, sia in Java che in C, si effettuino gli opportuni controlli sui parametri della richiesta e si gestiscano le eccezioni, tenendo presente i criteri secondo cui si possa ripristinare il funzionamento del programma oppure si debba forzarne la terminazione.

Leggete con attenzione le specifiche del problema prima di impegnarvi "a testa bassa" nello sviluppo delle singole parti.
Naturalmente, i componenti da consegnare devono essere stati provati.

Si richiede il progetto della gestione dei servizi **Parla**, per utenti che vogliono comunicare attraverso l'ambiente realizzato. Obiettivo è l'insieme delle funzionalità di supporto per il provider dei servizi e non la realizzazione delle funzioni finali di scambio messaggi.

I servizi di Parla sono organizzati in **stanze** (con nomi che rappresentano l'argomento della stanza) e, per ciascuna stanza, si prevedono al massimo un certo numero di **utenti**. Ogni stanza prevede due modalità: scambi **broadcast di stanza o multicast (M)**, e scambi **punto-a-punto (P)** tra utenti. Un utente si può iscrivere ad una o più stanze e poi può inviare e ricevere i messaggi in transito sulle stanze, senza stato permanente dei messaggi stessi.

Si vogliono realizzare le funzionalità di gestione:

1. **aggiunta di una stanza**: questa operazione richiede il nome della stanza e il tipo di comunicazione da instaurare, e inizializza la prima stanza libera disponibile nella struttura dati;
2. **eliminazione di un utente**: questa operazione richiede il nome dell'utente, scorre le stanze liberandole dall'utente, e restituisce la lista di tutte (e sole) le stanze liberate;
3. **visualizzazione dello stato delle stanze**: questa operazione visualizza l'attuale stato di tutte le stanze, indicando anche, per ogni stanza, il tipo di operatività;
4. **sospensione del lavoro in una stanza**: questa operazione richiede il nome di una stanza e ne sospende la operatività producendo un cambiamento di stato della stanza stessa (stato S, aggiunto a P o M).

Si progetti con particolare attenzione la **struttura dati** che mantiene lo stato delle N stanze (L, per libero a default) per al massimo di K utenti, da implementare opportunamente nei diversi ambienti richiesti, Java e C. Ogni stanza prevede modalità o broadcast o punto-a-punto, e sospeso, e registra gli utenti presenti.

Nome	Stato	Utente 1	Utente 2	Utente 3	Utente 4	...	Utente K-1	Utente K
Stanza 1	P	Pippo	Minnie	Orazio	L	...	L	L
Stanza 2	SP	Pippo	Pluto	L	L	...	L	L
Stanza 3	M	Pluto	Paperino	Quo	Qui	...	L	L
L	L	L	L	L	L	...	L	L
Stanza N	SM	L	L	L	L	...	L	L

Parte C

Utilizzando RPC sviluppare un'applicazione C/S che consenta di effettuare le operazioni remote per:

- aggiungere una stanza alla struttura dati;
- eliminare un utente dalla struttura dati.

Il progetto RPC si basa su due procedure remote:

```
int aggiungi_stanza(Richiesta);  
Risultato elimina_utente(string);
```

La procedura **aggiungi_stanza** accetta come parametro d'ingresso la struttura dati Richiesta contenente il nome e il tipo di comunicazione per la stanza da aggiungere, e restituisce un intero che indica l'esito dell'operazione remota, 0 in caso di successo, -1 in caso di errore, ad esempio, se esiste già una stanza con lo stesso nome nella struttura dati o se la struttura dati è piena.

La procedura **elimina_utente** accetta come parametro d'ingresso il nome dell'utente da eliminare, e restituisce la struttura dati Risultato contenente la lista delle stanze liberate, e un intero che indica l'esito dell'operazione remota: 0 in caso di successo, -1 in caso di errore, ad esempio, se l'utente non è presente in nessuna delle stanze mantenute nella struttura dati.

Più in dettaglio:

- il **client** realizza l'interazione con l'utente proponendo ciclicamente i servizi che utilizzano le due procedure remote e stampa a video gli esiti delle chiamate, fino alla fine del file di input da tastiera.
- il **server** implementa le procedure invocabili in remoto restituendo l'esito delle operazioni come indicato sopra.

Parte Java

Sviluppare un'applicazione C/S che consenta di effettuare le operazioni remote per:

- visualizzare lo stato attuale di tutte le stanze;
- sospendere l'operatività di una stanza;

con un client e due server, che realizzano rispettivamente la prima e la seconda funzionalità. In particolare, il client viene invocato con 4 argomenti: l'host e la porta su cui sono rispettivamente in ascolto i due server:

```
java Client hostServerStream portServerStream hostServerDatagram portServerDatagram
```

Più in dettaglio:

- il **Client** chiede ciclicamente all'utente quale tipo di operazione effettuare, visualizzazione dello stato delle stanze o sospensione dell'operatività di una stanza, fino alla fine del file di input da tastiera.

La funzionalità di **visualizzazione dello stato delle stanze** viene realizzata utilizzando **socket stream**; per ogni richiesta il client apre la comunicazione col server, riceve la struttura dati, e la stampa a video.

La funzionalità di **sospensione dell'operatività di una stanza** viene realizzata utilizzando **socket datagram**; per ogni richiesta il client chiede all'utente il nome della stanza da sospendere, invia il dato letto, riceve l'esito, e lo stampa a video.

- il **Server_stream** gestisce in modo parallelo la funzionalità di **visualizzazione dello stato delle stanze**. Per ogni richiesta, il server genera un figlio che gestisce l'interazione col client fino alla chiusura della connessione da parte del client. Per ciascuna richiesta il figlio legge e invia la struttura dati.

- Il **Server_datagram** gestisce in modo seriale o parallelo la funzionalità di **sospensione dell'operatività di una stanza**. Per ogni richiesta, il server riceve il nome della stanza, ne cambia lo stato e invia la risposta al client: 0 in caso di successo, -1 in caso d'errore, ad esempio, se lo stato della stanza era già sospeso.