Reti di Calcolatori T
Reti di Calcolatori L-A
Appello del 10/01/2014
Compito 2

Cognome: ……
Nome: ……
Matricola: ……

Tempo a disposizione: 2h

È obbligatorio inserire Cognome, Nome, e numero di Matricola 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 consigliato sviluppare **entrambe** le soluzioni richieste al meglio.
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, ci aspettiamo che i componenti da consegnare siano stati provati e siano funzionanti.

****

Si richiede il progetto della gestione dei servizi **TicketTwo**, per la gestione della compravendita di biglietti per eventi.
I servizi di TicketTwo mantengono, per ogni evento, le seguenti informazioni: la **descrizione** dell'evento, unico all'interno del sistema; il **tipo** di evento che può essere Concerto, Calcio, Formula1; la **data** dell'evento e il **luogo** dell'evento; la **disponibilità**, intesa come i biglietti ancora disponibili; e il **prezzo**, cioè un intero che indica il prezzo unitario per biglietto.
Si vogliono realizzare le funzionalità di gestione:

1. **inserimento di un nuovo evento**: questa operazione richiede la descrizione dell'evento, il tipo, la data e il luogo, la disponibilità e il prezzo, ed inserisce il prodotto nella struttura dati;
2. **visualizzazione di tutti gli eventi di un certo tipo in un determinato luogo**: questa operazione richiede tipo di evento e luogo e restituisce l'elenco di tutti gli eventi di quel tipo il cui luogo è uguale a quello indicato;
3. **visualizzazione di tutti gli eventi disponibili e con prezzo inferiore o uguale a un prezzo dato**: questa operazione richiede il prezzo massimo per biglietto, valuta gli eventi con prezzo accettabile e la disponibilità e li restituisce all'utente;
4. **acquisto di uno o più biglietti per un determinato evento**: questa operazione richiede la descrizione dell'evento e il numero di biglietti da acquistare, valuta se i biglietti sono disponibili e quindi aggiorna la struttura dati.

Si progetti con particolare attenzione la **struttura dati** che mantiene lo stato, fino ad un massimo di N prodotti (L, per libero a default), da implementare opportunamente nei diversi ambienti richiesti, Java e C.

Descrizione	Tipo	Data	Luogo	Disponibilità	Prezzo
L	**L**	**L**	**L**	**L**	**L**
String	**Concerto**	**11/01/2014**	**Verona**	**40**	**40**
Junentus-Inger	**Calcio**	**03/05/2014**	**Torino**	**21**	**150**
…	**…**	**…**		**…**	**…**
GP Bologna	**Formula1**	**07/09/2014**	**Bologna**	**10**	**200**
L	**L**	**L**	**L**	**L**	**L**

Si considerino e si segnalino le possibilità di interferenze fra le operazioni, evitandole dove necessario.

Parte C

Progettare un servitore multiservizio (usando obbligatoriamente una select) che consenta di effettuare le operazioni remote per:

- visualizzazione di tutti gli eventi di un certo tipo in un determinato luogo;
- visualizzazione di tutti gli eventi disponibili e con prezzo inferiore o uguale a un prezzo dato.

Più in dettaglio:

- Il **client_stream** è organizzato come un **processo filtro ciclico fino alla fine del file di input** e realizza la funzionalità di **visualizzazione di tutti gli eventi di un certo tipo in un determinato luogo** utilizzando **un'unica socket stream**.
  Per ogni richiesta, il client richiede all'utente i parametri e li invia al server, quindi riceve la lista di degli eventi corrispondenti stampandoli a video.
- Il **client_datagram** è organizzato come un **processo filtro ciclico fino alla fine del file di input** e realizza la funzionalità di **visualizzazione di tutti gli eventi disponibili e con prezzo inferiore o uguale a un prezzo dato** utilizzando **socket datagram**.
  Per ogni richiesta, il client richiede all'utente e invia al server l'identificatore; quindi riceve l'esito dell'operazione e lo stampa a video.
- Il **server** principale unico discrimina le richieste utilizzando la primitiva select: il **server gestisce in modo parallelo** la funzionalità di visualizzazione di tutti gli eventi di un certo tipo in un determinato luogo; mentre la funzionalità di visualizzazione di tutti gli eventi disponibili e con prezzo inferiore a P **può essere realizzata in modo o seriale o parallelo**: si motivi la scelta fatta**.**
  Per ogni richiesta di **visualizzazione di tutti gli eventi di un certo tipo in un determinato luogo**, il figlio riceve i parametri e trasmette al client la lista degli eventi che soddisfano i requisiti.
  Per ogni richiesta di **visualizzazione di tutti gli eventi disponibili e con prezzo inferiore o uguale a un prezzo dato**, il server riceve il prezzo massimo e restituisce al client la lista di tutti gli eventi con biglietti disponibili con costo inferiore a quello richiesto, oppure un esito negativo, ad esempio se non sono presenti eventi che soddisfano la richiesta.

Parte Java

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

- inserimento di un nuovo evento;
- acquisto di uno o più biglietti per un determinato evento.

Il progetto RMI si basa su un'interfaccia (contenuta nel file *RMI_interfaceFile.java*) in cui vengono definiti i metodi invocabili in remoto dal client:

- Il metodo **`inserimento_evento`** accetta come parametro d'ingresso la struttura dati di inserimento contenente la descrizione dell'evento, il tipo, la data, il luogo, la sua disponibilità, il suo prezzo, e aggiunge l'evento alla struttura dati; quindi restituisce l'esito dell'operazione, 0 se la registrazione è andata a buon fine, -1 altrimenti, ad esempio, se la struttura dati è piena
- Il metodo **`acquista_biglietti`** accetta come parametro d'ingresso una struttura che indica contenente la stringa che indica la descrizione dell'evento e un intero che indica il numero di biglietti da acquistare; quindi il server valuta la richiesta, controlla se esiste l'evento e il numero di biglietti richiesti e aggiorna la struttura restituendo l'esito dell'operazione, 0 se l'acquisto è andato a buon fine, -1 altrimenti, ad esempio, se l'evento non è presente nella struttura dati.
- Si progettino le classi:
- ***RMI_Server*** (contenuta nel file *RMI_Server.java*), che implementa i metodi del server invocabili in remoto;
- ***RMI_Client*** (contenuta nel file *RMI_Client.java*), il processo filtro che realizza l'interazione con l'utente **proponendo ciclicamente i servizi** che utilizzano i due metodi remoti, e stampa a video i risultati, fino alla fine del file di input da tastiera.