

Reti di Calcolatori T

Prova d'esame

23/01/2019 - Compito 1

Tempo a disposizione: 2h

Cognome:
Nome:
Matricola:

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 **RentAcaR**, per la gestione delle prenotazioni di autovetture condivise. I servizi di **RentAcaR** mantengono, per ogni prenotazione, le seguenti informazioni: la **targa del veicolo** (unico all'interno del sistema), la **patente di guida** dell'utente (una stringa di 5 interi), il **tipo di veicolo prenotato** (una stringa che può assumere uno dei seguenti valori: **camper**, **auto**), le **immagini del veicolo**. I file con le immagini relative ad un veicolo sono mantenuti in una cartella denominata '<targa>_img' situata nel direttorio corrente dove vengono lanciati server e client. In particolare, si richiede di realizzare le seguenti funzionalità:

1. **download delle foto delle vetture in base al numero di targa**: questa operazione richiede il numero di targa e scarica le immagini presenti nella cartella relativa al veicolo;
2. **aggiornamento della patente di guida**: nel caso in cui cambi il guidatore, questa operazione richiede il numero di targa del veicolo e il numero di patente del nuovo guidatore, quindi aggiorna la struttura dati;
3. **eliminazione di una prenotazione**: questa operazione richiede il numero di targa di un veicolo quindi elimina la prenotazione e il file immagine relativi, aggiornando la struttura dati e il file system;
4. **visualizzazione di tutte le prenotazioni di un certo tipo di veicolo** (camper, auto), **immatricolato successivamente al 2011**: questa operazione richiede il tipo di veicolo e stampa a video la lista delle prenotazioni richieste. Le targhe vengono generate in successione cambiando le prime due lettere (es. **AAxxxXX**, **ABxxxXX**, **ACxxxXX**, ..., **EDxxxXX**), consideriamo targhe del 2011 quelle che iniziano con le lettere **'ED'**.

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

Targa del veicolo	Patente di guida	Tipo del veicolo	Folder immagini
AN745NL	00003	auto	AN745NL_img/
FE457GF	50006	camper	FE457GF_img/
L	0	L	L
...
NU547PL	40063	auto	NU547PL_img/
LR897AH	56832	camper	LR897AH_img/
MD506DW	00100	camper	MD506DW_img/

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

Parte Java

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

- **eliminare una prenotazione**;
- **visualizzare tutte le prenotazioni di un certo tipo di veicolo** (camper o auto), **immatricolato successivamente al 2011**.

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 **elimina_prenotazione** accetta come parametro d'ingresso il **numero di targa di un veicolo**, e restituisce un intero con l'esito dell'operazione.
- Il metodo **visualizza_prenotazioni** accetta come parametro d'ingresso il **tipo di veicolo**, e restituisce la lista (array) delle prenotazioni del tipo richiesto, e di veicoli la cui immatricolazione è avvenuta successivamente all'anno 2011, cioè aventi come prime due lettere della targa un valore maggiore di **'ED'**.

Si progettino inoltre 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 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.

Parte C

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

- **download delle foto di un veicolo richiesto**,
 - **aggiornamento del numero di patente**,
- utilizzando solamente un'unica connessione per ogni sessione cliente.

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 **download delle foto del veicolo** utilizzando **socket stream** e **un'unica socket**.

Per ogni richiesta, il client chiede all'utente e invia al server il numero di targa, quindi riceve i nomi e i file delle immagini e li salva sul file system locale (multiple-get dal folder immagini mantenuto dal server ad una cartella locale al client).

- Il **client_datagram** è organizzato come un **processo filtro ciclico fino alla fine del file di input** e realizza la funzionalità di **aggiornamento del numero di patente** utilizzando **socket datagram**.

Per ogni richiesta, il client chiede all'utente e invia al server il numero di targa e il nuovo numero di patente; quindi riceve l'esito dell'operazione stampandolo a video.

- Il **server** principale unico discrimina le richieste utilizzando la primitiva **select**: il **server gestisce in modo parallelo** la funzionalità di download delle foto; mentre la funzionalità di aggiornamento del numero di patente è **realizzata in modo seriale**.

Per ogni richiesta di **download delle foto**, il figlio riceve il numero di targa, quindi invia nome e contenuto di tutti i file delle immagini richieste usando l'unica connessione della sessione.

Per ogni richiesta di **aggiornamento del numero di patente** il server riceve il numero di targa e il nuovo numero di patente, quindi aggiorna la struttura dati e restituisce un intero positivo che indica l'esito dell'operazione: 0 in caso di successo, -1 in caso di problemi.