28-02-2014

Fondamenti di Informatica 2 – Prova Scritta

Riportare su Eventuale Foglio : Nome, Cognome, Matricola e Numero di Postazione

[Nota: ad inizio esame ad ogni studente è assegnato il tema, composto da due caratteri, il primo essere A,B C e il secondo può essere 1,2,3. Il testo del tema cambia a seconda del carattere indicato].

Si vuole progettare il software di una centralina usata per il controllo delle apparecchiature in casa. Il proprietario della casa riesce, attraverso una interfaccia web e connessioni di rete a mandare dei comandi sotto forma di stringa. La centralina, interpreta le stringhe e le utilizza per governare i dispositivi, restituendo una stringa di risposta.

Esistono 2 specifiche e distinte categorie di dispositivi (gli elenchi tra parentesi quadre indicano valori alternativi):

- **Controllo on off (esempio : luci)**
  - **Funzionalità : accende/spegne il dispositivo**
  - **Formato Richiesta : *nome_dispotivo [on,off,reset*]***
  - **Formato Risposta : *nome_dispotivo [done,error]***
  - ***reset è sinonimo di on, ma implica un riavvio forzato del dispositivo***
- **Controlli percentuali (esempio : persiane, riscaldamento)**
  - **Funzionalità : imposta il valore di un dispositivo**
  - **Formato Richiesta : *nome_dispotivo set=valore***
  - **Formato Risposta : *nome_dispotivo [done,set_with valore,error]****
  - ***con set_with=valore ritorna il valore effettivamente impostato quando è essere diverso dall'originale***

Non si conoscono ancora gli strumenti necessari per mandare gli effettivi ordini ai dispositivi. Il sistema memorizza in un file di Log tutti i messaggi in ingresso ed in uscita, anticipati da Data e ora:minuti:secondi (vedi java.text.SimpleDateFormat). Es.:

- **(08,04,14) 9:10:00 Esame on**
- **(08,04,14) 9:10:20 Esame done**
- **(08,04,14) 13:10:15 Esame off**
- **(08,04,14) 13:10:47 Esame error**

Specifiche Aggiuntive Tema A[[[

Inoltre, i messaggi mandati dall'interfaccia grafica Web possono contenere opzionalmente un orario di avvio (formato: ore:minuti in coda al messaggio). L'esecuzione del comando in questo caso non deve essere istantanea, ma deve avvenire esattamente ad una distanza temporale indicata dai ore e minuti nel messaggio. Utilizzare java.util.Timer per implementare questa caratteristica. Sfruttare eventualmente java.util.Calendar e java.util.Date. In questo caso, il sistema ritorna il messaggio di default '*planned*'.
Se esiste un evento pianificato per un dispositivo non ancora eseguito, il sistema rifiuterà ulteriori chiamate, restituendo il comando 'busy'.
]]]

Specifiche Aggiuntive Tema B[[[

Il sistema, inoltre, supporta un comando addizionale (per tutti i dispositivi) chiamato 'check' che verifica la presenza in rete di tutti i dispositivi. I Dispositivi non in rete sono considerati 'missing', mentre i Dispositivi in rete sono considerati 'working'. Se in questa fase un dispositivo working diventa missing, o viceversa, nel log deve comparire un messaggio così formato:

- **(Data) Orario nome_dispositivo [up,down]**

Quando un dispositivo risulta missing e giungono al sistema comandi per quel dispositivo, il sistema ritorna il messaggio di default '*missing*'. Se il valore di 'missing/working' per quel dispositivo non è aggiornato da più di 10 minuti (vedi java.util.Calendar e java.util.Date), viene effettuato un check per quel solo dispositivo prima di elaborare la risposta.
]]]

L'obiettivo è quello di sviluppare i moduli principali del sistema, effettuando opportuni test. Al fine di effettuare i test, vengono fornite 3 classi di Test chiamate 'FrigoDiTest', 'LuceDiTest' e 'RiscaldamentoDiTest'. E' obbligatorio che i test vengano effettuati facendo uso di queste 3 classi e SENZA MODIFICARNE IL CODICE.

1. Creare un Workspace **Eclipse**. Creare un Progetto **esame**. Dopo aver studiato il problema, implementare in **Java** una possibile soluzione modulare e ad oggetti alle richieste del Tema, ed effettuare sui componenti realizzati opportuni Tests.
2. Su foglio protocollo, **a titolo di documentazione e ai fini della valutazione**, si realizzi uno schema UML **sintetico** che metta in luce le relazioni che intercorrono tra i moduli implementati. E' possibile utilizzare ObjectAID UML, ma in quel caso è obbligatorio esportare gli schemi UML in formato immagine png. E' inoltre necessario aver chiaro che non basta buttare qualsiasi cosa dentro al diagramma solo perché è comodo e semplice fare così **.** ***E' necessario*** **inoltre utilizzare la documentazione Javadoc nel codice dove lo si ritenga opportuno**.
3. Lo studente può accedere al percorso **/home/etc/FDI2** per recuperare la documentazione **Javadoc**, i cosiddetti **esempi forniti** e altro materiale utile. E' inoltre possibile consultare qualsiasi testo scritto.
4. Alla fine dell'esame, esportare un file zip attraverso la funzionalità **Export...** di eclipse (vedi le **istruzioni di salvataggio dati**) e salvarlo come **/home/esm/esame_N/esame_N.zip** (ad esempio **/home/esm/esame_20/esame_20.zip**)

Punteggio (Totale 15+ punti)

- **6+ punti** per **l'architettura** del progetto, con particolare enfasi alla struttura del Modello.
- **3 punti** per la corretta **implementazione** in **Java** delle funzionalità del programma.
- **3 punti** in merito alla **validità** di implementazione interna ad **ogni singola classe**.
- **3 punti** sono assegnati sulla base :
  - TEMI 1 : ai test che devono effettuare un utilizzo quanto più possibile esaustivo dei componenti messi a disposizione dell'esame.
  - TEMI 2 : ai test E ad un utilizzo quanto più possibile dei componenti delle librerie Java .
  - TEMI 3 : alla documentazione Javadoc e ai Design Pattern utilizzati.