Progettazione di software sicuro
Esame del 3 luglio 2015 – (Prova di laboratorio) – Corso on-line

Istruzioni

- Create una cartella sul desktop con il vostro cognome.
- Aprite eclipse e selezionate come workspace la cartella creata sul Desktop. Se eclipse si avvia senza farvi scegliere il workspace, riavviatelo (File/Switch Workspace/Other...): a questo punto dovreste essere in grado di selezionare il workspace.
- Create un progetto Java e chiamatelo con il vostro cognome.
- Quando avete terminato il compito, chiudete eclipse e consegnate solo la cartella del progetto Java. Dentro la cartella creata sul desktop (quella con il vostro cognome) trovate un'altra cartella (ancora con il vostro cognome) che corrisponde al progetto Java: dovete consegnare questa cartella.

1 Ostello [pt. 2]

Scrivere una classe *Ostello* per modellare un ostello. L'ostello ha 50 letti in totale, distribuiti in 10 stanze da 5 letti ciascuna: ogni letto può essere libero o occupato.

Lo stato dei letti nelle stanze deve essere modellato con una matrice di booleani (le righe sono le stanze e le colonne i letti).

All'inizio tutti i letti sono liberi.

Metodo checkin La classe dispone di un metodo `checkin(int stanza, int letto)`, mostrato in Fig. 1, che permette di occupare un determinato `letto` in una determinata `stanza`.

```
public boolean checkin(int stanza, int letto) {
    if(stanza >= 0 & stanza < 10 & letto >= 0 & letto < 5) {
        if(lettoLibero[stanza][letto]) {
            lettoLibero[stanza][letto] = false;
            return true;
        }
    }
    return false;
}
```

[10][50]

Figure 1: Metodo *parcheggia*

Il metodo controlla che i valori `letto` e `stanza` siano corretti e che il letto selezionato sia libero. Il metodo ritorna *true* se il letto viene occupato, *false* altrimenti.

Metodo ottimizzaOstello L'ostello vuole minimizzare il numero di stanze usate. A tal fine la classe deve disporre di un metodo void `ottimizzaOstello()` che sposti tutti i clienti nelle prime stanze libere (a partire dalla prima stanza e dal primo letto). Per esempio, se c'è un solo cliente per ogni stanza (quindi in totale ci sono 10 clienti), il metodo `ottimizzaOstello()` sposta tutti i clienti nelle prime due stanze.

2 Junit

In JUnit, scrivere:

- un caso di test che mostri che è possibile trovare un letto libero ed occuparlo (basandosi sul valore ritornato dal metodo); **[pt. 1]**
- un caso di test che mostri che non è possibile occupare un letto già occupato (basandosi sul valore ritornato dal metodo); **[pt. 1]**
- un caso di test che mostri che se si mettono 5 persone nell'ultima stanza e poi si ottimizza l'occupazione delle stanze dell'ostello, gli ospiti sono spostati tutti nella prima stanza; rendere pubblico il campo *lettoLibero* per facilitare la scrittura del test. **[pt. 1]**

3 Copertura

Scrivere in JUnit una test suite per il metodo `checkin` che soddisfi la copertura delle decisioni ma che non soddisfi anche la copertura delle condizioni. Creare una classe `CheckinCoperturaDecisioni` e commentare in modo opportuno i singoli casi di test. **[pt. 1]**

4 JML

Scrivere in JML la seguente postcondizione al costruttore:

- all'inizio tutti i letti sono liberi. **[pt. 1]**

Scrivere in JML le seguenti postcondizioni al metodo `checkin(int stanza, int letto)`:

- tutte i letti diversi dal letto indentificato dai parametri `stanza` e `letto` non hanno cambiato il loro stato; **[pt. 1]**
- il numero di letti occupati è minore o uguale a 50; **[pt. 1]**
- se `stanza` è negativo, il metodo ha ritornato *false*. **[pt. 1]**

Totale punti = 2 + 3 + 1 + 4 = 10