Progettazione di software sicuro
Esame del 23 settembre 2015 – (Prova di laboratorio)

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 Ora di punta [pt. 2]

Scrivere una classe *OraDiPunta* che modelli una versione semplificata del gioco *Ora di punta*. Su una griglia 6×6 sono disposte 6 macchine, numerate con i numeri da 1 a 6 (ogni macchina ha dimensione 1×1); celle vuote sono indicate con 0. Obiettivo del gioco è portare la macchina numero 1 (chiamata macchina *rossa*) all'*uscita* della griglia, che corrisponde alla cella con indici $(2, 5)$ (le coordinate iniziano da 0).

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	2
2	0	0	1	0	0	3
3	0	0	0	0	0	4
4	0	5	0	0	0	0
5	0	0	6	0	0	0

Table 1: Configurazione della griglia del gioco *Ora di punta* (in grigio viene mostrata l'uscita)

Tabella 1 mostra la configurazione iniziale del gioco; la cella grigia indica l'uscita.
Il costruttore della classe deve disporre le macchine sulla griglia come mostrato in Tabella 1.

Metodo spostaMacchina La classe deve disporre di un metodo booleano *spostaMacchina(int riga, int colonna, int dir)* che permetta di muovere una macchina sulla griglia; il metodo ritorna *true* se una macchina è stata spostata, *false* altrimenti. I parametri *riga* e *colonna* indicano una cella della griglia; *dir* indica una direzione nel seguente modo:

- 1: verso l'alto;
- 2: verso destra;
- 3: verso il basso;
- 4: verso sinistra.

Il metodo dove controllare che i valori passati siano corretti: *riga* e *colonna* sono indici corretti della griglia e *dir* è una delle quattro direzioni. Se in *riga* e *colonna* non c'è una macchina, niente deve accadere. Se una macchina non si può spostare in direzione *dir* perché ostruita da un'altra macchina o dai limiti della griglia, non deve essere spostata.

Metodo macchinaRossaUscita La classe dispone di un metodo booleano *macchinaRossaUscita* che dice se la macchina rossa ha raggiunto l'uscita; il metodo è mostrato in Codice 1.

```
public boolean macchinaRossaUscita() {
    return griglia[2][5] == 1;
}
```

Codice 1: Metodo *macchinaRossaUscita()*

2 Junit

In JUnit, scrivere i seguenti casi di test nella classe *OraDiPuntaTest*.

- Per il metodo *spostaMacchina(int riga, int colonna, int dir)*, scrivere:
  - un caso di test che mostri che, se si prova a spostare una macchina in una cella libera, lo spostamento viene eseguito (basandosi sul valore ritornato dal metodo); **[pt. 1]**
  - un caso di test che mostri che, se si prova a spostare una macchina in una cella occupata, lo spostamento non viene eseguito (basandosi sul valore ritornato dal metodo); **[pt. 1]**
- Scrivere un caso di test che mostri che è possibile portare la macchina rossa all'uscita. **[pt. 1]**

3 Copertura

Scrivere in JUnit una test suite per il metodo *spostaMacchina* che soddisfi la copertura delle istruzioni (creare una classe *SpostaMacchinaCoperturaIstruzioni*). Commentare in modo opportuno i singoli casi di test. **[pt. 1]**

4 JML

Scrivere in JML la seguente postcondizione al costruttore:

- nella griglia ci sono 6 celle non vuote. **[pt. 1.25]**

Scrivere in JML la seguente precondizione al metodo *spostaMacchina(int riga, int colonna, int dir)*:

- i parametri attuali identificano una cella della griglia e una delle quattro direzioni. **[pt. 0.25]**

Scrivere in JML le seguenti postcondizioni al metodo *spostaMacchina(int riga, int colonna, int dir)*:

- il prodotto di tutte le celle è 0; **[pt. 1.25]**
- non esiste una cella con valore minore di 0. **[pt. 1.25]**

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