

Linguaggi di Programmazione per la Sicurezza

Esame del 18 aprile 2011 - (Prova di laboratorio)

1 Rompicapo delle otto regine semplificato

Il *problema delle otto regine semplificato*¹ consiste nel posizionare otto regine su una scacchiera 8×8 in modo tale che nessuna di esse possa catturare un'altra regina muovendosi in orizzontale o in verticale (in questo problema la regina non può muoversi in diagonale). Quindi, una soluzione prevede che nessuna delle otto regine abbia la colonna o la riga in comune con un'altra regina.

1.1 Classe Java [pt. 2]

Scrivere una classe *OttoRegineSimpl* che rappresenti una scacchiera 8×8 di booleani: la scacchiera deve memorizzare se in una cella è presente una regina. La classe deve disporre di un metodo che permetta di posizionare una regina sulla scacchiera.

All'inizio la scacchiera non deve contenere regine.

Metodo place La classe deve disporre di un metodo *place(int i, int j)* che permetta di posizionare in (i, j) una regina. Non bisogna fare alcun controllo sul fatto che i, j siano indici validi e che la regina non possa essere catturata.

Metodo isSolved La classe deve disporre di un metodo booleano *isSolved()* che dica se il rompicapo è stato risolto, cioè se sono state posizionate otto regine sulla scacchiera ed il posizionamento è tale per cui le regine non possono catturarsi muovendosi in orizzontale o in verticale (come in tabella 1).









	a	b	c	d	e	f	g	h
8								
7								
6								
5								
4								
3								
2								
1								

Table 1: Possibile soluzione del rompicapo.

¹Versione semplificata del *rompicapo delle otto regine*: http://en.wikipedia.org/wiki/Eight_queens_puzzle

2 JUnit

- In JUnit, per il metodo `place(int i, int j)`, scrivere:
 - un caso di test in cui si mostri che il posizionamento di due regine, la prima in (i, j) e la seconda in (h, k) (con (i, j) ed (h, k) scelti da voi in modo che le regine non possano catturarsi), avviene in modo corretto; [pt. 1]
 - un caso di test in cui si mostri che il posizionamento di una regina in una cella in cui può essere catturata viene eseguito portando la scacchiera in una configurazione errata. Il caso di test si aspetta che, se si prova a posizionare una regina in una cella in cui può essere catturata, il posizionamento non dovrebbe essere eseguito: invece il metodo esegue il posizionamento. [pt. 1]
- scrivere in JUnit un caso di test che evidenzi che la scacchiera impostata dal costruttore non contiene la soluzione del rompicapo; [pt. 1]
- scrivere in JUnit un caso di test che evidenzi che è possibile eseguire una serie di 8 posizionamenti che portano alla soluzione del rompicapo; nella tabella 1 potete trovare un possibile soluzione del rompicapo e, quindi, ricavare i parametri attuali delle otto chiamate del metodo `place(int i, int j)`. [pt. 1]

3 JML

Scrivere in JML le seguenti precondizioni al metodo `place(int i, int j)`:

- gli indici i, j sono indici validi della scacchiera; [pt. 0.5]
- la riga identificata da i non contiene nessuna regina; [pt. 0.75]
- la colonna identificata da j non contiene nessuna regina; [pt. 0.75]

Scrivere in JML la seguente postcondizione al metodo `place(int i, int j)`:

- il numero di regine sulla scacchiera è minore od uguale ad 8. [pt. 2]

Totale punti = 2 + 4 + 4 = 10

package esami;

```
public class OttoRegineSimpl_20110418 {
    /*@ spec_public @*/ private boolean[][] board;

    public OttoRegineSimpl_20110418() {
        board = new boolean[8][8];
    }

    //Scrivere in JML le seguenti precondizioni al metodo place
    //gli indici i e j sono indici validi della scacchiera
    //@ requires i >= 0 && i < 8 && j >= 0 && j < 8;
    //la riga identificata da i non contiene nessuna regina
    //@ requires (\forall int k; k >= 0 && k < 8; !board[i][k]);
    //la colonna identificata da j non contiene nessuna regina
    //@ requires (\forall int k; k >= 0 && k < 8; !board[k][j]);

    //Scrivere in JML la seguente postcondizione al metodo place
    //il numero di regine sulla scacchiera e' minore od uguale ad 8
    //@ensures (\num_of int h, k; h >= 0 && h < 8 && k >= 0 && k < 8; board[h][k]) <= 8;
    public void place(int i, int j) {
        board[i][j] = true;
    }

    public boolean isSolved() {
        int numQueens = 0;
        boolean queenFound;
        for(int i = 0; i < 8; i++) {
            queenFound = false;
            for(int j = 0; j < 8; j++) {
                if(board[i][j]) {
                    if(queenFound) {
                        return false;
                    }
                    else {
                        queenFound = true;
                        numQueens++;
                    }
                }
            }
        }
        if(numQueens < 8) {
            return false;
        }
        else {
            for(int j = 0; j < 8; j++) {
                queenFound = false;
                for(int i = 0; i < 8; i++) {
                    if(board[i][j]) {
                        if(queenFound) {
                            return false;
                        }
                        else {
                            queenFound = true;
                        }
                    }
                }
            }
        }
    }
}
```

```

        }
        return true;
    }
}

public boolean getCell(int i, int j) {
    return board[i][j];
}

public static void main(String[] args) {
    OttoRegineSimpl_20110418 or = new OttoRegineSimpl_20110418();
    //or.place(8, 0); //violato invariante su indici corretti
    //or.place(0, -1); //violato invariante su indici corretti

    /*or.place(0, 0);
    or.place(0, 1); //violato invariante su riga vuota*/

    /*or.place(4, 5);
    or.place(7, 5); //violato invariante su colonna vuota*/

    //posizionamento corretto
    or.place(0, 0);
    or.place(1, 2);
    or.place(2, 5);
    or.place(3, 6);
    or.place(4, 3);
    or.place(5, 4);
    or.place(6, 7);
    or.place(7, 1);
}
}

```

```
package esami;
```

```
import static org.junit.Assert.*;  
import org.junit.Test;
```

```
public class Test_OttoRegineSimpl_20110418 {
```

```
    //Un caso di test in cui si mostri che il posizionamento di due regine,  
    //la prima in (i, j) e la seconda in (h, k) (con (i, j) ed (h, k) scelti da voi  
    //in modo che le regine non possano catturarsi), avviene in modo corretto.  
    @Test
```

```
    public void testPlace1() {  
        OttoRegineSimpl_20110418 or = new OttoRegineSimpl_20110418();  
        or.place(0, 0);  
        assertTrue(or.getCell(0, 0));  
        or.place(1, 7);  
        assertTrue(or.getCell(1, 7));  
    }
```

```
    //Un caso di test in cui si mostri che il posizionamento di una regina in una cella  
    //in cui puo' essere catturata viene eseguito portando la scacchiera in una configurazione errata.  
    //Il caso di test si aspetta che, se si prova a posizionare una regina in una cella in cui puo' essere  
    //catturata, il posizionamento non dovrebbe essere eseguito: invece il metodo esegue il posizionamento.  
    @Test
```

```
    public void testPlace2() {  
        OttoRegineSimpl_20110418 or = new OttoRegineSimpl_20110418();  
        or.place(0, 0);  
        assertTrue(or.getCell(0, 0));  
        or.place(0, 1);  
        assertFalse(or.getCell(0, 1));  
    }
```

```
    //Un caso di test che evidenzi che la scacchiera impostata dal costruttore  
    //non contiene la soluzione del rompicapo.  
    @Test
```

```
    public void testIsSolved1() {  
        OttoRegineSimpl_20110418 or = new OttoRegineSimpl_20110418();  
        assertFalse(or.isSolved());  
    }
```

```
    //Un caso di test che evidenzi che e' possibile eseguire una serie di 8  
    //posizionamenti che portano alla soluzione del rompicapo.  
    @Test
```

```
    public void testIsSolved2() {  
        OttoRegineSimpl_20110418 or = new OttoRegineSimpl_20110418();  
        or.place(0, 0);  
        or.place(1, 2);  
        or.place(2, 5);  
        or.place(3, 6);  
        or.place(4, 3);  
        or.place(5, 4);  
        or.place(6, 7);  
        or.place(7, 1);  
        assertTrue(or.isSolved());  
    }  
}
```