

Progettazione di software sicuro

Esame del 24 gennaio 2013 - (Prova di laboratorio)

1 Produttore - consumatore [pt. 2]

Scrivere una classe *ProdCons* che simuli il problema del produttore - consumatore. La classe deve disporre di un buffer che contenga gli elementi prodotti dal produttore. Il buffer deve essere modellato con un array di interi: una cella vale 0 se non contiene il prodotto, 1 se lo contiene. Un opportuno indice deve indicare la prima cella senza prodotti.

La classe deve disporre di un metodo *produce* per produrre un prodotto ed un metodo *consume* per consumare un prodotto.

Il costruttore della classe deve inizializzare il buffer alla dimensione comunicata tramite parametro. All'inizio il buffer non contiene prodotti.

Metodo produce Il metodo *produce* aggiunge un prodotto al buffer solo se c'è spazio. Deve ritornare un booleano: true o false a seconda che sia stato inserito o meno il prodotto.

Metodo consume Il metodo *consume* elimina un prodotto dal buffer se questo contiene almeno un prodotto. Deve ritornare un booleano: true o false a seconda che sia stato eliminato o meno un prodotto.

2 Junit

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

- Per il metodo *produce*, scrivere:
 - un caso di test che mostri che, se il buffer non è pieno, l'inserimento di un prodotto viene eseguito (basandosi sul valore ritornato dal metodo); [pt. 1]
 - un caso di test che mostri che, se il buffer è pieno, l'inserimento di un prodotto non viene eseguito (basandosi sul valore ritornato dal metodo); [pt. 1]

- scrivere un caso di test che mostri che, appena dopo essere stato creato, il buffer non contiene prodotti (rendere il buffer pubblico per poterlo testare). **[pt. 1]**

3 Copertura

Scrivere in JUnit una test suite che soddisfi la copertura delle condizioni per il metodo *consume*; mettere i casi di test nella classe *ConsumeCoperturaCondizioni* e commentare in modo opportuno i singoli casi di test. **[pt. 1]**

4 JML

Scrivere in JML la seguente preconditione al costruttore:

- la capacità con cui deve essere inizializzato il buffer deve essere compresa tra 1 e 100; **[pt. 0.25]**

Scrivere in JML le seguenti postcondizioni al metodo *produce*:

- il buffer contiene solo 0 o 1; **[pt. 1]**
- nel buffer non c'è mai un 1 dopo uno 0; **[pt. 1]**
- il numero di 1 nel buffer dopo l'esecuzione del metodo è uguale o al massimo maggiore di un'unità al numero di 1 presenti nel buffer prima dell'esecuzione del metodo. **[pt. 1.75]**

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

package esami;

```
public class ProdCons_20130124 {
    public int[] buffer;
    private int free;

    //la capacita' con cui deve essere inizializzato il buffer deve essere compresa tra 1 e 100
    //@ requires capacity > 0 && capacity <= 100;
    public ProdCons_20130124(int capacity) {
        buffer = new int[capacity];
        free = 0;
    }

    //il buffer contiene solo 0 o 1
    //@ ensures (\forall int i; i >= 0 && i < buffer.length; buffer[i] == 0 || buffer[i] == 1);
    //nel buffer non c'e' mai un 1 dopo uno 0
    //@ ensures !(\exists int i; i > 0 && i < buffer.length; buffer[i-1] == 0 && buffer[i] == 1);
    //il numero di 1 nel buffer dopo l'esecuzione del metodo e' uguale o al massimo maggiore di
    //un'unita' al numero di 1 presenti nel buffer prima dell'esecuzione del metodo
    //@ ensures ((\num_of int i; i >= 0 && i < buffer.length; buffer[i] == 1) - (\num_of int i; i >= 0 && i <
buffer.length; \old(buffer[i] == 1)) <= 1;
    public boolean produce() {
        if(free < buffer.length) {
            buffer[free] = 1;
            free++;
            return true;
        }
        else {
            return false;
        }
    }

    public boolean consume() {
        if(free > 0) {
            free--;
            buffer[free] = 0;
            return true;
        }
        else {
            return false;
        }
    }

    public static void main(String[] args) {
        //ProdCons_20130124 p1 = new ProdCons_20130124(1000); //viola la precondizione

        ProdCons_20130124 p = new ProdCons_20130124(2);
        p.consume(); //non consuma nulla
        p.produce();
        p.produce();
        p.produce(); //non viene messo
        p.consume();
        p.consume();
    }
}
```

```
package esami;
```

```
import org.junit.Test;
```

```
//Scrivere in JUnit una test suite che soddisfi la copertura delle condizioni
```

```
//per il metodo consume; mettere i test case nella classe ConsumeCoperturaCondizioni
```

```
//e commentare in modo opportuno i singoli test case.
```

```
public class ConsumeCoperturaCondizioni_20130124 {
```

```
    // "free > 0" testato a true
```

```
    @Test
```

```
    public void testConsumeCoperturaCondizioni1() {
```

```
        ProdCons_20130124 p = new ProdCons_20130124(2);
```

```
        p.produce();
```

```
        p.consume();
```

```
    }
```

```
    // "free > 0" testato a false
```

```
    @Test
```

```
    public void testConsumeCoperturaCondizioni2() {
```

```
        ProdCons_20130124 p = new ProdCons_20130124(2);
```

```
        p.consume();
```

```
    }
```

```
}
```

```
package esami;
```

```
import static org.junit.Assert.assertTrue;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertEquals;
```

```
import org.junit.Test;
```

```
public class ProdConsTest_20130124 {
```

```
    //un caso di test che mostri che, se il buffer non e' pieno,
    //l'inserimento di un prodotto viene eseguito (basandosi sul valore ritornato dal metodo)
    @Test
```

```
    public void testProduceWithBufferNotFull() {
        ProdCons_20130124 p = new ProdCons_20130124(2);
        assertTrue(p.produce());
    }
```

```
    //un caso di test che mostri che, se il buffer e' pieno,
    //l'inserimento di un prodotto non viene eseguito (basandosi sul valore ritornato dal metodo)
    @Test
```

```
    public void testProduceWithBufferFull() {
        ProdCons_20130124 p = new ProdCons_20130124(2);
        p.produce();
        p.produce();
        assertFalse(p.produce());
    }
```

```
    //un caso di test che mostri che, appena dopo essere stato creato,
    //il buffer non contiene prodotti (rendere il buffer pubblico per poterlo testare)
    @Test
```

```
    public void testConstructorBufferIsEmpty() {
        int[] expected = new int[3];
        ProdCons_20130124 p = new ProdCons_20130124(3);
        assertEquals(expected, p.buffer);
    }
```

```
}
```