

تمرین امتیازی دوم درس آزمون نرم افزار

دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)

بردیا اردکانیان
۹۸۳۱۰۷۲

استاد درس: دکتر گوهری

بهار ۱۴۰۲

Ch6.4-q1: The restriction on interleaving next() and remove() calls is quite complex. The JUnit tests in IteratorTest.java only devote one test for this situation, which may not be enough. Refine the input domain model with one or more additional characteristics to probe this behavior, and implement these tests in JUnit.

برای اصلاح مدل دامنه ورودی و بررسی این رفتار، ویژگی های اضافی زیر را اضافه می کنیم:

- Iterator with multiple elements: این مشخصه رفتار در هم آمیختن فراخوانی های «next» و «remove» را روی یک تکرار کننده با عناصر متعدد آزمایش کند.
- Iterator with a single element: این مشخصه رفتار interleaving "next" و "remove" را روی یک تکرار کننده تنها با یک عنصر آزمایش کند.
- Empty iterator: این مشخصه رفتار در هم آمیختن تماس های «next» و «remove» را روی یک تکرار کننده خالی آزمایش کند.
- Concurrent modifications: این مشخصه رفتار درهم آمیزی تماس های «next» و «remove» را زمانی که مجموعه زیربنایی به طور همزمان اصلاح شده است، آزمایش کند.

برای پیاده سازی این تست ها در JUnit، می توان موارد آزمایشی را نوشت که اشیاء iterator با ویژگی های مختلف ایجاد کند و رفتار iterator را زمانی که «next» و «remove» به ترتیب مختلف فراخوانی می شوند، آزمایش کند. کد ارائه شده در زیر می تواند از اظهارات برای تأیید اینکه تکرار کننده مطابق انتظار رفتار می کند استفاده کند، برای مثال، با بررسی اینکه فراخوانی «remove» پس از فراخوانی «next» عنصر صحیح را از مجموعه اصلی حذف می کند.

```
package com.company;

import static org.junit.Assert.*;
import java.util.ArrayList;
import java.util.ConcurrentModificationException;
import java.util.Iterator;
import org.junit.Test;

public class IteratorTest {
    //...

    @Test
    public void testMultipleElements() {
        ArrayList<Integer> list = new ArrayList<>();

        list.add(1);
        list.add(2);
        list.add(3);
```

```

        Iterator<Integer> it = list.iterator();
        it.next();
        it.remove();
        assertEquals(2, (int) it.next());
    }

    @Test
    public void testSingleElement() {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(1);

        Iterator<Integer> it = list.iterator();
        it.next();
        it.remove();
        assertFalse(it.hasNext());
    }

    @Test
    public void testEmptyIterator() {
        ArrayList<Integer> list = new ArrayList<>();

        Iterator<Integer> it = list.iterator();
        assertFalse(it.hasNext());
    }
































    @Test(expected = ConcurrentModificationException.class)
    public void testConcurrentModifications() {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(1);

        Iterator<Integer> it = list.iterator();
        list.add(2); // modify collection
        it.next(); // should throw ConcurrentModificationException
    }
}

```

خروجی

همانطور که در عکس زیر مشخص است تست‌های نوشته شده بعضی پاس و بعضی فیل شدند.

Run:  IteratorTest x		
            		
	IteratorTest (com.company)	20 ms
	testSingleElement	14 ms
	testRemove_BaseCase	0 ms
	testConcurrentModifications	0 ms
	testHasNext_BaseCase	0 ms
	testHasNext_C1	0 ms
	testHasNext_C5	1 ms
	testEmptyIterator	1 ms
	testNext_BaseCase	0 ms
	testMultipleElements	1 ms
	testNext_C1	1 ms
	testNext_C2	0 ms
	testNext_C5	0 ms
	testRemove_C1	1 ms
	testRemove_C2	0 ms
	testRemove_C3	0 ms
	testRemove_C4	1 ms