



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر

تمرین دوم

آزمون نرم افزار

نگارش

محمد رضا اخگری زیری

محمد علی کشاورز

علی نظری

استاد درس

دکتر معصومه طارمی راد

بهار ۱۴۰۰

صفحه

فهرست مطالب

سوال اول.....	۳
سوال دوم.....	۴
گزارش انجام تمرین به صورت گروهی:.....	۱۲

سوال اول

در شکل ۱، نتایج اجرای این آزمون‌ها مشاهده می‌شود. موارد گفته شده در کلاس، در این آزمون‌ها پوشش داده شده است.

The screenshot shows an IDE with two tabs: `Min.java` and `DataDrivenMinTest.java`. The `DataDrivenMinTest.java` file contains the following code:

```

23 @
24 private static List NullListGenerator() {
25     ArrayList<Object> list = new ArrayList<>();
26     list.add(null);
27     return list;
28 }
29
30 @Parameters
31 public static Collection<Object[]> minValues() {
32     return Arrays.asList(
33         new Object[][]{
34             {Arrays.asList(null, "cat"), NullPointerException.class},
35             {NullListGenerator(), NullPointerException.class},
36             {null, NullPointerException.class},
37             {Arrays.asList("dog", "cat", 1), ClassCastException.class},
38             {Arrays.asList(), IllegalArgumentException.class},
39             {Arrays.asList("cat"), "cat"},
40             {Arrays.asList("dog", "cat"), "cat"},
41             {Arrays.asList(-2, 4, 10), -2},
42             {Arrays.asList(2, 2, 6), 2}
43         }
44     );
45 }
46
47 @Test

```

Below the code, the `Run` tab shows the execution of `DataDrivenMinTest`. The output indicates that all 9 tests passed in 8 ms. The command used to run the tests is `/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -ea`. The test results are as follows:

Test Name	Duration	Status
DataDrivenMinTest	8 ms	Passed
> [0]	4 ms	Passed
> [1]	1 ms	Passed
> [2]	1 ms	Passed
> [3]	0 ms	Passed
> [4]	1 ms	Passed
> [5]	0 ms	Passed
> [6]	0 ms	Passed
> [7]	1 ms	Passed
> [8]	0 ms	Passed
minTest[8]	0 ms	Passed

شکل ۱- نتایج اجرای آزمون‌ها

سوال دوم

(الف)

باید ببینیم مواردی که می‌خواهیم با هم مقایسه کنیم چه چیزی هستند و چه ویژگی‌هایی دارند و بعد تصمیم بگیریم که چگونه این تابع را پیاده‌سازی کنیم. در حقیقت پیاده‌سازی این تابع به اطلاعات ما از مساله دارد، یعنی مثلاً ما می‌دانیم که برابری دو نقطه، طبق اطلاعات ما، حالتی است که دو نقطه دارای مختصات یکسانی باشند. هرچند ممکن است به خاطر تعریف مساله این موضوع فرق کند ولی در اینجا باید دو نقطه را با هم مقایسه کنیم که دو ویژگی طول و عرض را دارد، پس برای پیاده‌سازی این تابع باید این مقادیر طول و عرض هر دو نقطه مدنظر را با هم مقایسه کنیم و نتیجه‌ی آن را به صورت خروجی تابع بیان کنیم.

(ب)

پیاده‌سازی این تابع در قطعه کد زیر قابل مشاهده است.

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Point point = (Point) o;
    return x == point.x && y == point.y;
}
```

در صورت ارث‌بری از کلاس دیگری، باید در پیاده‌سازی این تابع، تابع equals از کلاس پدر را نیز فراخوانی کنیم و در صورتی که هر دو تابع equals در کلاس پدر و فرزند اعلام برابری کنند، ما هم برابری را به عنوان خروجی اعلام می‌کنیم (البته این مساله باز به تعریف مساله بستگی دارد). در زیر، بروزرسانی تابع equals را مشاهده می‌کنیم.

```
return x == point.x && y == point.y && super.equals(o);
```

(ج)

در شکل ۲، چهار آزمون را مشاهده می‌کنیم که حالت‌های دو نقطه‌ی مساوی، دو نقطه‌ی نامساوی، مقایسه با نقطه‌ی Null و مقایسه دو جنس متفاوت را پوشش می‌دهد. نتیجه‌ی اجرای این آزمون‌ها نیز قابل مشاهده است.

(د)

در شکل ۳، آزمون‌های قسمت ج را به صورت داده‌رانه نوشته‌ایم و نتایج اجرای آن‌ها نشان داده‌ایم.

(ه)

رابطه‌ی تساوی ویژگی‌های «بازتابی»، «تقارنی» و «تعدی» را داراست. همچنین باید به Null نبودن مقادیر هم توجه کنیم. این ویژگی‌ها به صورت تئوری‌های Junit در شکل ۴ قابل مشاهده است. همانطور که قابل ملاحظه است، ما برای هر ویژگی تابع آزمونی نوشته‌ایم و در هر مورد هم به Null نبودن متغیرها توجه کرده‌ایم.

(و)

رابطه‌ی ریاضی بین این دو به این صورت است که اگر دو شی با هم برابر^۱ باشند الزاماً hashCodeهای یکسانی نیز دارند ولی برعکس این گزاره برقرار نیست یعنی اگر hashCodeها برابر باشند لزومی ندارد که این دو شی با هم equal باشند. برای مثال اگر ما تابع هش را به صورت زیر تعریف کنیم، دو نقطه‌ی (۷و۲) و (۲و۷) دارای هش برابر خواهند شد در حالی که دو نقطه برابر نیستند.

$$\text{hash}_p = p.x + p.y$$

^۱ equal

The screenshot shows an IDE with three tabs: Point.java, PointTest.java (active), and DataDrivenPointTest.java. The PointTest.java file contains four JUnit tests. Below the code, the Run window shows the execution results for the PointTest class, indicating that all four tests passed successfully.

```

6      @Test
7      public void testEqualPoints() {
8          Point p1 = new Point( x: 5, y: 2);
9          Point p2 = new Point( x: 5, y: 2);
10         assertTrue(p1.equals(p2));
11     }

12
13     @Test
14     public void testForNotEqualPoints() {
15         Point p1 = new Point( x: 6, y: 2);
16         Point p2 = new Point( x: 5, y: 2);
17         assertFalse(p1.equals(p2));
18     }

19
20     @Test
21     public void testForInconsistentTypeItems() {
22         Double p1 = 2.2;
23         Point p2 = new Point( x: 5, y: 2);
24         assertFalse(p2.equals(p1));
25     }

26
27     @Test
28     public void testForSoloNullItem() {
29         Point p = new Point( x: 5, y: 2);
30         assertFalse(p.equals(null));
31     }

```

Run: PointTest

Test Method	Duration	Status
PointTest	18 ms	Passed
testForSoloNullItem	18 ms	Passed
testForInconsistentTypeItems	0 ms	Passed
testForNotEqualPoints	0 ms	Passed
testEqualPoints	0 ms	Passed

Tests passed: 4 of 4

شکل ۲- چهار آزمون برای کلاس Point به همراه نتیجه‌ی اجرای آن‌ها

```

15     public Object b1, b2;
16     public Boolean isEqual;
17
18     public DataDrivenPointTest(Object b1, Object b2, Boolean isEqual) {
19         this.b1 = b1;
20         this.b2 = b2;
21         this.isEqual = isEqual;
22     }
23
24     @private static List NullListGenerator() {
25         ArrayList<Object> list = new ArrayList<>();
26         list.add(null);
27         return list;
28     }
29
30     @Parameters
31     @public static Collection<Object[]> equalValues() {
32         return Arrays.asList(
33             new Object[][]{
34                 {new Point(x: 5, y: 2), new Point(x: 5, y: 2), true},
35                 {new Point(x: 6, y: 2), new Point(x: 5, y: 2), false},
36                 {2.2, new Point(x: 5, y: 2), false},
37                 {null, new Point(x: 5, y: 2), false}
38             });
39     }
40
41     @Test
42     public void pointTest() {
43         assertTrue( condition: b2.equals(b1) == isEqual);
44     }
45 }

```

Run: DataDrivenPointTest x

Tests passed: 4 of 4 tests – 5 ms

Test Case	Duration	Details
✓ DataDrivenPointTest	5 ms	/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java
> ✓ [0]	4 ms	
> ✓ [1]	0 ms	Process finished with exit code 0
> ✓ [2]	0 ms	
> ✓ [3]	1 ms	
✓ pointTest[3]	1 ms	

شکل ۳- اجرای آزمون‌ها به صورت داده‌رانه

```

7 public class TheoryPointTest {
8     @DataPoints
9     @ public static Object[] objects() {
10         return new Object[]{new Point(x: 5, y: 2), new Point(x: 5, y: 2), new Point(x: 5, y: 2)};
11     }
12
13     @Theory
14     @ public void testEqualsReflexive(Object b1) {
15         assertTrue( condition: b1 != null);
16         assertEquals(b1.equals(b1));
17     }
18
19     @Theory
20     @ public void testEqualsSymmetric(Object b1, Object b2) {
21         assertTrue( condition: b1 != null);
22         assertTrue( condition: b2 != null);
23
24         assertEquals(b1.equals(b2));
25         assertEquals(b2.equals(b1));
26     }
27
28     @Theory
29     @ public void testEqualsTransitive(Object b1, Object b2, Object b3) {
30         assertTrue( condition: b1 != null);
31         assertTrue( condition: b2 != null);
32         assertTrue( condition: b3 != null);
33
34         assertEquals(b1.equals(b2));
35         assertEquals(b2.equals(b3));
36         assertEquals(b1.equals(b3));
37     }
38 }

```

Run: TheoryPointTest

Tests passed: 3 of 3 tests - 40 ms

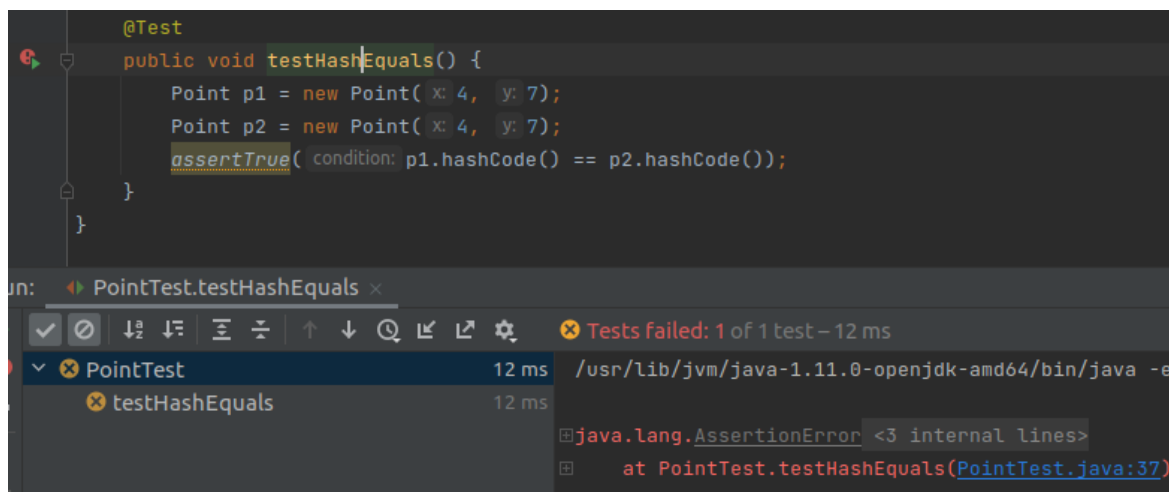
Test Name	Duration	Status
TheoryPointTest	40 ms	Passed
testEqualsReflexive	29 ms	Passed
testEqualsSymmetric	1 ms	Passed
testEqualsTransitive	10 ms	Passed

Process finished with exit code 0

شکل ۴- نمایش رابطه‌ی تساوی به صورت تئوری در junit

(ز)

در شکل ۵، مشاهده می‌شود که چون تابع equals را عوض کردیم ولی تابع hashCode تغییری نکرده است، این رابطه برقرار نیست.



شکل ۵- نتیجه‌ی اجرای تست برابری مقدار hashCode

(ح)

برای رفع این مشکل کافیست که تابع hashCode را بروز کنیم. در قطعه کد زیر این بروزرسانی قابل مشاهده است.

```

public class Point {
    private int x, y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Point point = (Point) o;
        return x == point.x && y == point.y;
    }

    @Override
    public int hashCode() {
        return Objects.hash(x, y);
    }
}

```

ط) همانطوریکه در قسمت‌های قبلی گفته شد، لازم است که رابطه‌ی تساوی در hashCodeها نیز برقرار باشد، برای همین تئوری‌های نوشته شده را برای hashCodeها نیز می‌نویسیم. در شکل ۶ تئوری‌های اضافه شده به تئوری‌های موجود را مشاهده می‌کنید.

The screenshot shows an IDE with a file named `TheoryPointTest.java`. The code defines three methods, each annotated with `@Theory` and `@` (likely `@Test`), using `assertTrue` to verify hash code properties.

```

37     }
38
39     @Theory
40     @
41     public void testEqualHashCodeReflexive(Object b1) {
42         assertTrue( condition: b1 != null);
43         assertTrue( condition: b1.hashCode() == b1.hashCode());
44     }
45
46     @Theory
47     @
48     public void testEqualsHashCodeSymmetric(Object b1, Object b2) {
49         assertTrue( condition: b1 != null);
50         assertTrue( condition: b2 != null);
51
52         assertTrue( condition: b1.hashCode() == b2.hashCode());
53         assertTrue( condition: b2.hashCode() == b1.hashCode());
54     }
55
56     @Theory
57     @
58     public void testEqualsHashCodeTransitive(Object b1, Object b2, Object b3) {
59         assertTrue( condition: b1 != null);
60         assertTrue( condition: b2 != null);
61         assertTrue( condition: b3 != null);
62
63         assertTrue( condition: b1.hashCode() == b2.hashCode());
64         assertTrue( condition: b2.hashCode() == b3.hashCode());
65         assertTrue( condition: b1.hashCode() == b3.hashCode());
66     }
67 }

```

Below the code, the `Run` tab shows the execution results for `TheoryPointTest`. All tests passed.

Test Name	Duration	Status
TheoryPointTest	95 ms	Passed
testEqualsHashCodeTransitive	30 ms	Passed
testEqualsSymmetric	2 ms	Passed
testEqualsTransitive	32 ms	Passed
testEqualHashCodeReflexive	3 ms	Passed
testEqualsHashCodeSymmetric	26 ms	Passed
testEqualReflexive	2 ms	Passed

Summary: Tests passed: 6 of 6 tests - 95 ms. Process finished with exit code 0.

شکل ۶- تئوری‌های اضافه شده برای کامل کردن رابطه‌ی تساوی

ی) در شکل ۷، تئوری‌های نوشته شده را با `datapoint`های مناسب ارزیابی کرده‌ایم. چون این نقاط تئوری‌های مشخص شده را ارضا نمی‌کنند، آزمون‌ها با شکست روبه رو شده‌اند (در شکل ۶ این تئوری‌ها را برای نقاط مساوی اجرا کرده‌ایم و همه‌ی آزمون‌ها قبول شدند).

The screenshot shows an IDE with three tabs: TheoryPointTest.java, Point.java, and AssertionError.class. The TheoryPointTest.java file contains the following code:

```

2  import org.junit.runner.RunWith;
3
4  import java.util.Arrays;
5  import java.util.HashSet;
6  import java.util.Set;
7
8  import static org.junit.Assert.assertTrue;
9
10 @RunWith(Theories.class)
11 public class TheoryPointTest {
12     @DataPoints
13     public static Set[] points = {
14         new HashSet(Arrays.asList(new Point( x: 3, y: 2), new Point( x: 3, y: 2), new Point( x: 4, y: 2))),
15         new HashSet(Arrays.asList(new Point( x: 5, y: 2), new Point( x: 3, y: 2), new Point( x: 3, y: 2))),
16         new HashSet(Arrays.asList(new Point( x: 3, y: 2), new Point( x: 3, y: 2), null)),
17     };
18
19
20     @Theory
21     @
22     public void testEqualReflexive(Object b1) {
23         assertTrue( condition: b1 != null);
24         assertTrue(b1.equals(b1));
25     }
26
27     @Theory
28     @
29     public void testEqualsSymmetric(Object b1, Object b2) {
30         assertTrue( condition: b1 != null);
31     }
32 }

```

The Run window shows the following test results:

Test Name	Duration	Status
TheoryPointTest	83 ms	Failed
testEqualsHashCodeTransitive	61 ms	Failed
testEqualsSymmetric	14 ms	Failed
testEqualsTransitive	4 ms	Failed
testEqualHashCodeReflexive	1 ms	Passed
testEqualsHashCodeSymmetric	2 ms	Failed
testEqualReflexive	1 ms	Passed

The summary bar indicates: Tests failed: 4, passed: 2 of 6 tests - 83 ms.

شکل ۷- نمایش اجرای تئوری‌ها برای datapointهای درون شکل

گزارش انجام تمرین به صورت گروهی:

برای اجرای گروهی این تمرین، سه جلسه در google meet به مدت مجموعاً ۵ ساعت تشکیل شد و تمامی سوال‌ها با همفکری و همچنین جستجو در اینترنت انجام شد. نهایتاً نسخه‌ی نهایی نوشته شد و به تایید تمامی اعضای گروه رسید.