

**به نام خدا**

**تمرین دوم درس آزمون نرم افزار**

**آریا خلیق**

**9524014**

**حسین محمدی**

**9533081**

**آریا وارسته نژاد**

**9531345**

در این قسمت تمرین برای دو حالت می‌شد تست غیر داده‌رانه نوشت که در کلاس جدا در همین کد نوشته شده‌اند ولی مابقی تست‌ها که چندین ورودی داشتند به صورت مدل‌رانه توسعه یافته‌اند.

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

import org.junit.experimental.runners.Enclosed;

import org.junit.runner.RunWith;

import org.junit.runners.Parameterized;

import org.junit.runners.Parameterized.Parameters;


import java.util.*;


import static org.junit.Assert.assertSame;
import static org.junit.Assert.assertTrue;


@RunWith(Enclosed.class)

public class MinTest {

    @RunWith(Parameterized.class)

    public static class DataParametrized<T extends Comparable<? super T>> {

        public List<T> list;

        public T min;


        public DataParametrized(List<T> list, T min) {

            this.list = list;
```

```
    this.min = min;  
}
```

@Parameters

```
public static Collection<Object[]> listValues() {  
  
    List<String> l1 = new ArrayList<String>();  
  
    l1.add("a");  
  
    l1.add("A");  
  
    l1.add("b");  
  
  
    List<String> l2 = new ArrayList<String>();  
  
    l2.add("a");  
  
    l2.add("b");  
  
    l2.add("A");  
  
  
    List<String> l3 = new ArrayList<String>();  
  
    l3.add("a");  
  
  
  
    List<Integer> l4 = new ArrayList<Integer>();  
  
    l4.add(1);  
  
  
  
    List<Integer> l5 = new ArrayList<Integer>();  
  
    l5.add(1);  
  
    l5.add(-1);  
  
    l5.add(2);  
}
```

```

List<Integer> l6 = new ArrayList<Integer>();

l6.add(1);

l6.add(2);

l6.add(-1);


return Arrays.asList(new Object[][]{{l1, "A"}, {l2, "A"}, {l3, "a"}, {l4, 1}, {l5,
-1}, {l6, -1}}});

}


@Test

public void testDataValues() {

    assertEquals("Min right value test", Min.min(list), min);

}

}


@RunWith(Parameterized.class)

public static class NullPointerParametrized<T extends Comparable<? super T>> {

    public List<T> list;


    public NullPointerParametrized(List<T> list) {

        this.list = list;

    }
}

```

@Parameters

```
public static Collection<Object[]> listValues() {  
  
    List<String> l1 = null;  
  
    List<String> l2 = new ArrayList<>();  
  
    l2.add(null);  
  
    l2.add("Karim");  
  
    List<Integer> l3 = new ArrayList<>();  
  
    l3.add(null);  
  
    l3.add(1);  
  
    return Arrays.asList(new Object[][]{{l1}, {l2}, {l3}});  
}
```

@Test(expected = NullPointerException.class)

```
public void testNullPointerException() {  
  
    Min.min(list);  
  
}  
}
```

```
public static class NotParameterizedPart<T extends Comparable<? super T>> {  
  
    private List<T> list;
```

@Before

```
public void setUp() {  
  
    list = new ArrayList<T>();  
  
}  
  
@After  
  
public void tearDown() {  
  
    list = null;  
  
}  
  
@Test(expected = ClassCastException.class)  
  
public void testMutuallyIncomparable() {  
  
    List list = new ArrayList();  
  
    list.add("Arya1");  
  
    list.add(2);  
  
    list.add("Hosseini");  
  
    list.add("Arya2");  
  
    Min.min(list);  
  
}  
  
@Test(expected = IllegalArgumentException.class)  
  
public void testEmptyList() {  
  
    Min.min(list);  
  
}  
  
@Test(expected = NullPointerException.class)
```

```
public void testForSoloNullElement() {  
    list.add(null);  
    Min.min(list);  
}  
}  
}
```

(۲)

الف) باید از لحاظ منطقی دو شیء Point را بررسی کنیم. در ابتدا باید چک کنیم که شیء O داده شده آیا از جنس Point است که اکسپشن تولید نکند. پس از آن باید x , y دو شیء را با هم بررسی کنیم. در این حالت دیگر ملاک انتخاب ما مکان اشیا در حافظه نمی باشد.

(ب)

بدون در نظر گرفتن ارث‌بری:

```
@Override
public boolean equals(Object o) {
    if (o == this) {
        return true;
    }

    if (!(o instanceof Point)) {
        return false;
    }

    Point point = (Point) o;

    return x == point.x && y == point.y;
}
```



با در نظر گرفتن ارث‌بری باید تساوی پدرها را چک کنیم (فرض شده که equals پدر پیاده شده است):

```
}

@Override
public boolean equals(Object o) {
    if (o == this) {
        return true;
    }

    if (!(o instanceof Point)) {
        return false;
    }

    Point point = (Point) o;

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



```
import org.junit.*;

import org.junit.Assert.*;

import org.junit.experimental.runners.Enclosed;

import org.junit.runner.RunWith;

import org.junit.runners.Parameterized;

import org.junit.runners.Parameterized.Parameters;


import java.util.*;


import static org.junit.Assert.*;


public class PointTest {

    private Point point;


    @Before

    public void setUp() {

        point = new Point(1, 2);

    }


    @After

    public void tearDown() {

        point = null;

    }

}
```

@Test

```
public void testEqualsWithOtherClass() {  
    assertFalse(point.equals("Foo"));  
}
```

@Test

```
public void testEqualsWithSamePoint() {  
    assertTrue(point.equals(point));  
}
```

@Test

```
public void testEqualsWithOtherPointTrue() {  
    Point point2 = new Point(1, 2);  
    point.equals(point2);  
    assertTrue(point.equals(point2));  
}
```

@Test

```
public void testEqualsWithOtherPointFalse() {  
    Point point2 = new Point(2, 2);  
    assertFalse(point.equals(point2));  
}  
}
```

نتائج:



(د)

```
@RunWith(Parameterized.class)

public class PointParametrizedTest {

    Point point;

    Object compareWith;

    boolean result;

    public PointParametrizedTest(Point point, Object compareWith, boolean result) {

        this.point = point;

        this.compareWith = compareWith;

        this.result = result;

    }

    @Parameters

    public static Collection<Object[]> pointValues() {

        return Arrays.asList(

            new Object[][]{

                {new Point(1, 2), "Foo", false},
```

```

        {new Point(1, 2), null, false},

        {new Point(1, 2), new Point(1, 3), false},

        {new Point(1, 2), new Point(2, 2), false},

        {new Point(1, 2), new Point(1, 2), true}

    }

);

}

@Test

public void equalsTest() {

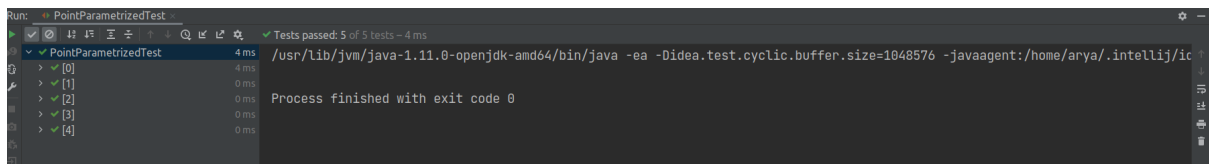
    assertEquals(point.equals(compareWith), result);

}

}

```

نتایج:



(۵)

۳ ویژگی زیر را باید داشته باشد:

- خاصیت بازتابی یعنی:  $a=a$
- خاصیت تقارنی یعنی اگر  $a=b$  باشد  $b=a$  نیز باشد.

- خاصیت تراییبی یعنی اگر  $a=b$  و  $b=c$  آنگاه  $a=c$  باشد.

```
@RunWith(Theories.class)

public class PointTestTheory {

    @DataPoints

    public static Object[] points() {

        return (

            new Object[]{

                new Point(1, 2),

                new Point(2, 1),

                new Point(0, 0),

                new Point(0, 0),

                new Point(0, 0)

            }

        );

    }

    @Theory

    public void testReflexivity(Point p) {

        assertTrue(p.equals(p));

    }

    @Theory

    public void testSymmetry(Point p1, Point p2) {

        assumeTrue(p1.equals(p2));

    }

}
```

```
    assertTrue(p2.equals(p1));  
}
```

@Theory

```
public void testTransitivity(Point p1, Point p2, Point p3) {  
    assertTrue(p1.equals(p2));  
    assertTrue(p2.equals(p3));  
    assertTrue(p1.equals(p3));  
}  
  
}
```

نتایج:



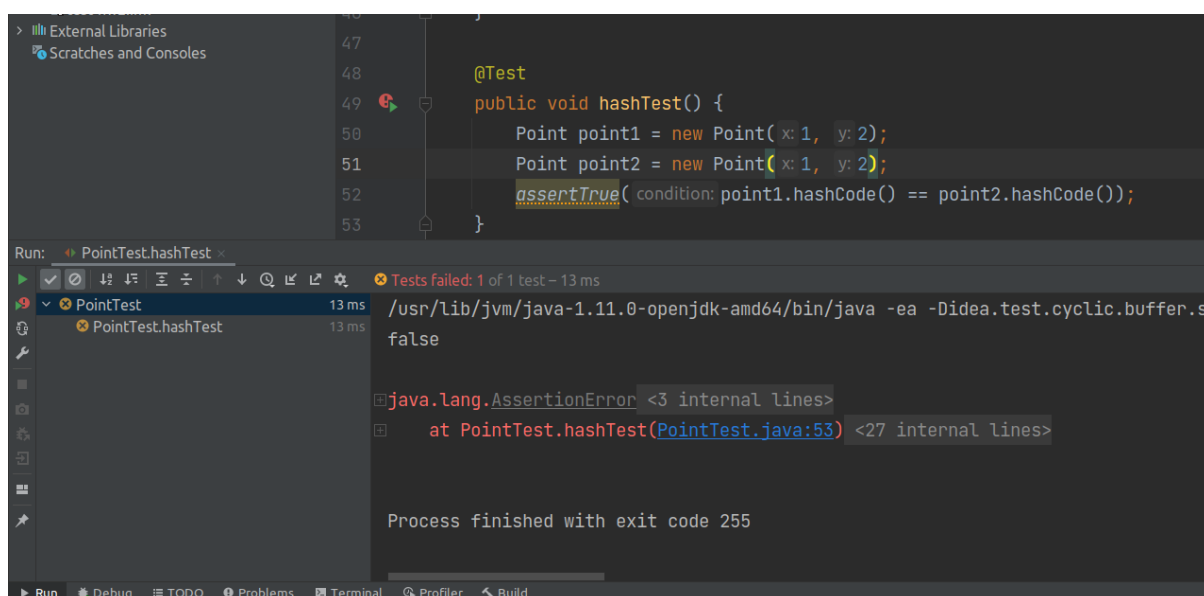
Tests passed: 3 of 3 tests - 30 ms	
PointTestTheory	30 ms
testSymmetry	16 ms
testReflexivity	2 ms
testTransitivity	12 ms

(و)

مانند تئوری در کتاب:

$a.equals(b) \Rightarrow a.hashCode() == b.hashCode()$

(ج)



```
47
48
49 @Test
50 public void hashTest() {
51     Point point1 = new Point(x: 1, y: 2);
52     Point point2 = new Point(x: 1, y: 2);
53     assertTrue(condition: point1.hashCode() == point2.hashCode());
54 }
```

Run: PointTest.hashTest x

Tests failed: 1 of 1 test - 13 ms

PointTest 13 ms

PointTest.hashTest 13 ms

false

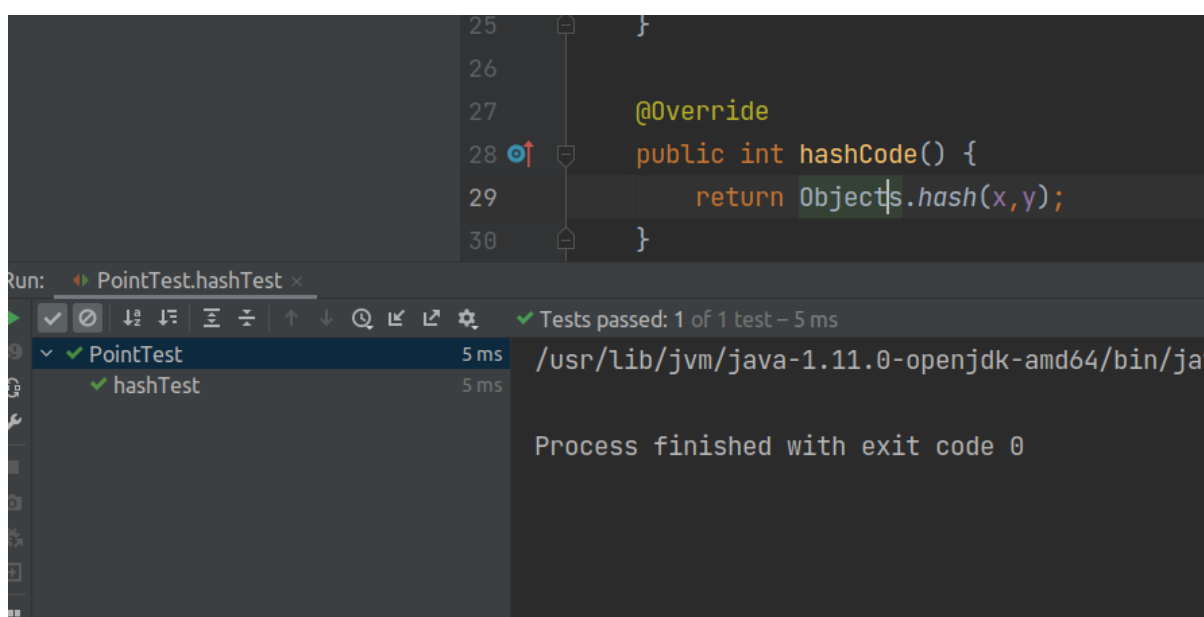
java.lang.AssertionError <3 internal lines>

at PointTest.hashTest(PointTest.java:53) <27 internal lines>

Process finished with exit code 255

(ح)

شیوه هشینگ خودمان را بر اساس فیلدها پیاده می‌کنی و تست پاس می‌شود:



```
25
26
27 @Override
28 public int hashCode() {
29     return Objects.hash(x, y);
30 }
```

Run: PointTest.hashTest x

Tests passed: 1 of 1 test - 5 ms

PointTest 5 ms

hashTest 5 ms

Process finished with exit code 0

(ط)



```
@RunWith(Theories.class)

public class PointTestTheory {

    @DataPoints

    public static Object[] points() {

        return (

            new Object[]{

                new Point(1, 2),

                new Point(2, 1),

                new Point(0, 0),

                new Point(0, 0),

                new Point(0, 0)

            }

        );

    }


    @Theory

    public void testReflexivity(Point p) {

        assertTrue(p.equals(p));

    }


    @Theory

    public void testSymmetry(Point p1, Point p2) {

        assumeTrue(p1.equals(p2));

        assertTrue(p2.equals(p1));

    }

}
```

@Theory

```
public void testTransitivity(Point p1, Point p2, Point p3) {  
  
    assertTrue(p1.equals(p2));  
  
    assertTrue(p2.equals(p3));  
  
    assertTrue(p1.equals(p3));  
  
}
```

@Theory

```
public void testHashCode(Point p1, Point p2) {  
  
    assertTrue(p1.equals(p2));  
  
  
    assertTrue(p1.hashCode() == p2.hashCode());  
  
}  
  
}
```

نتائج:

✓ Tests passed: 4 of 4 tests – 48 ms		/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -ea -Didea.test.cyclic.buffer.size=10485	
✓ PointTestTheory	48 ms	Process finished with exit code 0	
✓ testSymmetry	21 ms		
✓ testReflexivity	2 ms		
✓ testHashCode	8 ms		
✓ testTransitivity	17 ms		

(٥)

```

@DataPoints
public static Object[] points() {
    return (
        new Object[]{
            new Point( x: 1, y: 2),
            new Point( x: 2, y: 1),
            new Point( x: 0, y: 0),
            new Point( x: 0, y: 0),
            new Point( x: 0, y: 0)
        }
    );
}

```

نتایج:

The screenshot shows the IntelliJ IDEA test runner interface. At the top, it states "Tests passed: 4 of 4 tests - 43 ms". Below this, a list of tests is shown with green checkmarks indicating they all passed:

- ✓ PointTestTheory (43 ms)
- ✓ testSymmetry (20 ms)
- ✓ testReflexivity (1 ms)
- ✓ testHashCode (6 ms)
- ✓ testTransitivity (16 ms)

At the bottom, it says "Process finished with exit code 0". The background of the terminal window shows the Java command used to run the tests: `/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -ea -Didea.test.cyclic.buffer.size=1048576 -javaagent:/home/arya/.intelli/...`