

SOFTWARE ENGINEERING

LAB:8

TANK NANDANI JAGDISHBHAI

STUDENT ID:202001201

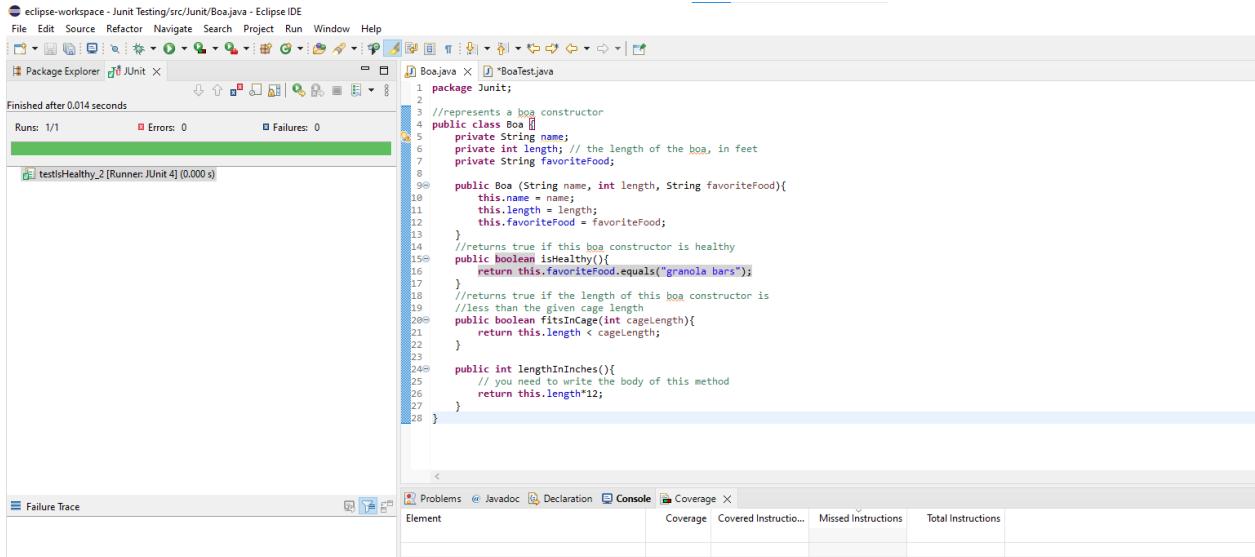
3.Boa.java file:

```
package Junit;

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

public class BoaTest {

    private Boa jen;
    private Boa ken;
    @Before
    public void setUp() throws Exception {
        jen = new Boa("Jennifer", 2, "grapes");
        ken = new Boa ("Kenneth", 3, "granola bars");
    }
    @Test
    public void testIsHealthy_1() {
        boolean output = jen.isHealthy();
        assertEquals(output, false);
    }
    @Test
    public void testFitsInCage_1() {
        boolean output = jen.fitsInCage(5);
        assertEquals(output, true);
    }
}
```



4. Modify Setup method:

BoaTest.java file:

```
package JUnit;

import static org.junit.Assert.*;

import org.junit.Before;
import org.junit.Test;

public class BoaTest {

    private Boa jen;
    private Boa ken;

    @Before
    public void setUp() throws Exception {
        jen = new Boa("Jennifer", 2, "grapes");
        ken = new Boa ("Kenneth", 3, "granola bars");
    }
}
```

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - JUnit Testing/src/Junit/BoaTest.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar items.
- Left Sidebar:** Package Explorer (JUnit), JUnit (Run, Stop, Run All).
- Center Area:** A code editor window titled "Boajava" containing the following Java code:

```
1 package Junit;
2
3 import static org.junit.Assert.*;
4
5 public class BoaTest {
6
7     private Boa jen;
8     private Boa ken;
9
10    @Before
11    public void setUp() throws Exception {
12        jen = new Boa("Jennifer", 2, "grapes");
13        ken = new Boa ("Kenneth", 3, "granola bars");
14    }
15
16    @Test
17    public void testIsHealthy_1() {
18        boolean output = jen.isHealthy();
19        assertEquals(output, false);
20    }
21
22    @Test
23    public void testIsHealthy_2() {
24        boolean output = ken.isHealthy();
25        assertEquals(output, true);
26    }
27
28    @Test
29    public void testFitsInCage_1() {
30        boolean output = jen.fitsInCage(5);
31        assertEquals(output, true);
32    }
33
34    @Test
35    public void testFitsInCage_2() {
36    }
37}
```

- Bottom Status Bar:** Problems, Javadoc, Declaration, Console, Coverage.

5. @Test stubs

A. Testing for testIshealthy() function:

Code:

```
@Test
public void testIsHealthy() {
    boolean output = jen.isHealthy();
    assertEquals(output, false);

}
```

I created two test case for testIshealthy function and test as below.now we test using junit testing method.

1. testIshealthy1()

The screenshot shows the Eclipse IDE interface. On the left, the 'Package Explorer' view displays a green bar indicating 'Finished after 0.011 seconds' with 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'. Below it, the 'JUnit' view shows a single test named 'testIsHealthy_1' with a duration of '(0.001 s)'. On the right, the 'BoaTest.java' code editor is open, showing Java code for testing boas. A blue rectangular selection highlights the first test method:

```
1 package Junit;
2
3 import static org.junit.Assert.*;
4
5 public class BoaTest {
6
7     private Boa jen;
8     private Boa ken;
9
10    @Before
11    public void setUp() throws Exception {
12        jen = new Boa("Jennifer", 2, "grapes");
13        ken = new Boa ("Kenneth", 3, "granola bars");
14    }
15
16    @Test
17    public void testIsHealthy_1() {
18        boolean output = jen.isHealthy();
19        assertEquals(output, false);
20    }
21
22    @Test
23    public void testFitsInCage_1() {
24        boolean output = jen.fitsInCage(5);
25        assertEquals(output, true);
26    }
27
28
29
30
31
32 }
```

2. testIshealthy2()

The screenshot shows the Eclipse IDE interface. On the left, the 'Package Explorer' view displays a green bar indicating 'Finished after 0.012 seconds' with 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'. Below it, the 'JUnit' view shows a single test named 'testIsHealthy_2' with a duration of '(0.000 s)'. On the right, the 'BoaTest.java' code editor is open, showing Java code for testing boas. A blue rectangular selection highlights the second test method:

```
1 package Junit;
2
3 import static org.junit.Assert.*;
4
5 public class BoaTest {
6
7     private Boa jen;
8     private Boa ken;
9
10    @Before
11    public void setUp() throws Exception {
12        jen = new Boa("Jennifer", 2, "grapes");
13        ken = new Boa ("Kenneth", 3, "granola bars");
14    }
15
16    @Test
17    public void testIsHealthy_1() {
18        boolean output = jen.isHealthy();
19        assertEquals(output, false);
20    }
21
22    @Test
23    public void testIsHealthy_2() {
24        boolean output = ken.isHealthy();
25        assertEquals(output, true);
26    }
27
28    @Test
29    public void testFitsInCage_1() {
30        boolean output = jen.fitsInCage(5);
31        assertEquals(output, true);
32    }
33
34
35 }
```

B. Testing for testFitsInCage()

Code:

```
@Test
public void testFitsInCage() {
    boolean output = jen.fitsInCage(5);
    assertEquals(output, true);
```

```
}
```

I created three test cases for function `testFitsInCage()` and test as below.mow we test using junit testing method.

1. `testFitsInCage_1()`

The screenshot shows the Eclipse IDE interface. On the left, the 'JUnit' view in the 'Package Explorer' shows 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'. Below it, a green bar indicates the test was completed successfully. On the right, the code editor displays `BoaTest.java`. The line `public void testFitsInCage_1() {` is highlighted with a blue selection bar. The code defines two boas, `jen` and `ken`, and tests their `isHealthy` and `fitsInCage` methods.

```
1 package Junit;
2
3 import static org.junit.Assert.*;
4
5 public class BoaTest {
6
7     private Boa jen;
8     private Boa ken;
9
10    @Before
11    public void setUp() throws Exception {
12        jen = new Boa("Jennifer", 2, "grapes");
13        ken = new Boa ("Kenneth", 3, "granola bars");
14    }
15
16    @Test
17    public void testIsHealthy_1() {
18        boolean output = jen.isHealthy();
19        assertEquals(output, false);
20    }
21
22    @Test
23    public void testIsHealthy_2() {
24        boolean output = ken.isHealthy();
25        assertEquals(output, true);
26    }
27
28    @Test
29    public void testFitsInCage_1() {
30        boolean output = jen.fitsInCage(5);
31        assertEquals(output, true);
32    }
33
34}
```

2. `testFitsInCage_2()`

The screenshot shows the Eclipse IDE interface. On the left, the 'JUnit' view in the 'Package Explorer' shows 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'. Below it, a green bar indicates the test was completed successfully. On the right, the code editor displays `BoaTest.java`. The line `public void testFitsInCage_2() {` is highlighted with a blue selection bar. The code is identical to the first test, but the cage size is set to 1.

```
8 public class BoaTest {
9
10    private Boa jen;
11    private Boa ken;
12
13    @Before
14    public void setUp() throws Exception {
15        jen = new Boa("Jennifer", 2, "grapes");
16        ken = new Boa ("Kenneth", 3, "granola bars");
17    }
18
19    @Test
20    public void testIsHealthy_1() {
21        boolean output = jen.isHealthy();
22        assertEquals(output, false);
23    }
24
25    @Test
26    public void testIsHealthy_2() {
27        boolean output = ken.isHealthy();
28        assertEquals(output, true);
29    }
30
31    @Test
32    public void testFitsInCage_1() {
33        boolean output = jen.fitsInCage(5);
34        assertEquals(output, true);
35    }
36
37    @Test
38    public void testFitsInCage_2() {
39        boolean output = jen.fitsInCage(1);
40        assertEquals(output, false);
41    }
42
43}
```

3. testFitsCage_3()

The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The left sidebar has a Package Explorer tab and a JUnit tab showing 'Finished after 0.012 seconds' with 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'. The main editor window contains the code for BoaTest.java:

```
15    ken = new Boa ("Kenneth", 3, "granola bars");
16}
17@Test
18public void testIsHealthy_1() {
19    boolean output = jen.isHealthy();
20    assertEquals(output, false);
21}
22@Test
23public void testIsHealthy_2() {
24    boolean output = ken.isHealthy();
25    assertEquals(output, true);
26}
27@Test
28public void testFitsInCage_1() {
29    boolean output = jen.fitsInCage(5);
30    assertEquals(output, true);
31}
32@Test
33public void testFitsInCage_2() {
34    boolean output = jen.fitsInCage(1);
35    assertEquals(output, false);
36}
37@Test
38public void testFitsInCage_3() {
39    boolean output = jen.fitsInCage(2);
40    assertEquals(output, false);
41}
42
43
44
45}
```

The code editor has a blue selection bar around the last three test methods (37-45). Below the editor is a 'Failure Trace' view. At the bottom is a 'Coverage' view with tabs for Problems, Javadoc, Declaration, Console, and Coverage. The Coverage tab shows a table with columns: Element, Coverage, Covered Instructions, Missed Instructions, and Total Instructions. The table is currently empty.

7. Add a new method to the Boa class, with this purpose and signature:

```
// produces the length of the Boa in inches
public int lengthInInches() {
// you need to write the body of this method
}
```

Adding new method:

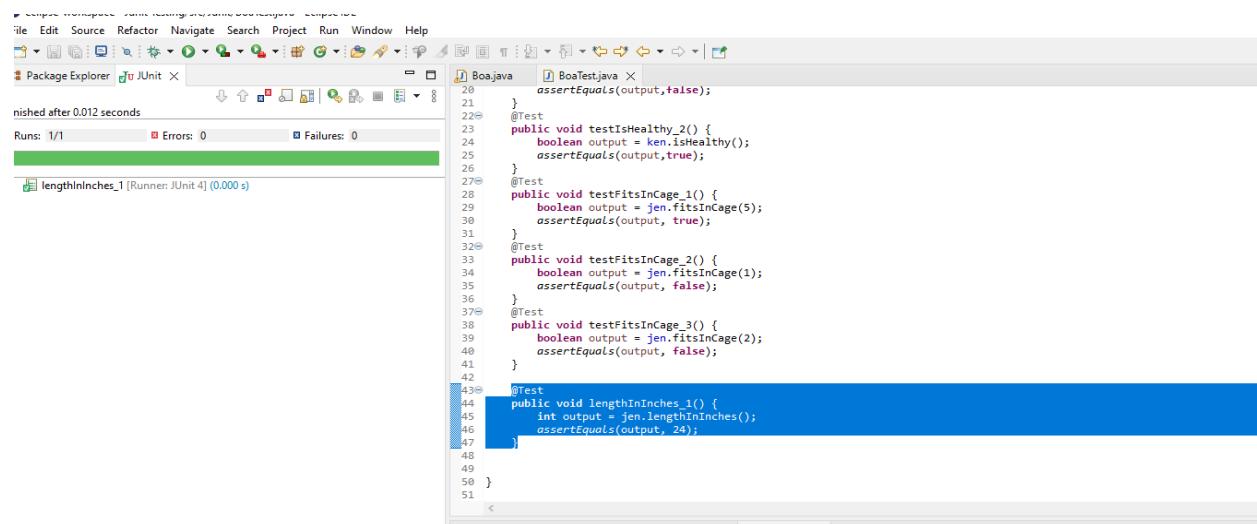
Testing for lengthInInches()

I created the two test case for lengthInInches() function and then using junit testing method.

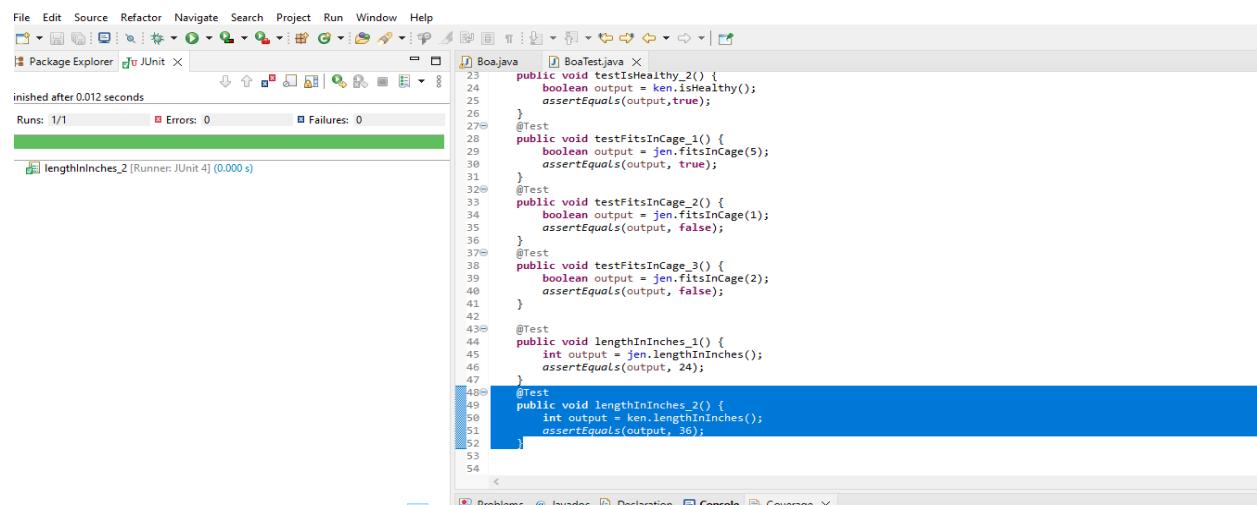
Code:

```
@Test  
public void lengthInInches() {  
    int output = jen.lengthInInches();  
    assertEquals(output, 24);  
}
```

`1.lengthInInches_1()`



2.lengthInInches_2()



The modified `testFitsInCage()` method tests the results of the `fitsInCage()` method when the cage length is less than, equal to, and greater than the length of the boa for both `healthyBoa` and `unhealthyBoa` objects.

Since the `setUp()` method initializes two different `Boa` objects, `healthyBoa` and `unhealthyBoa`, there is no need to write separate tests for "jen" and "ken". The tests should cover both objects as specified in the `setUp()` method.