



Dhirubhai Ambani
Institute of Information and Communication Technology

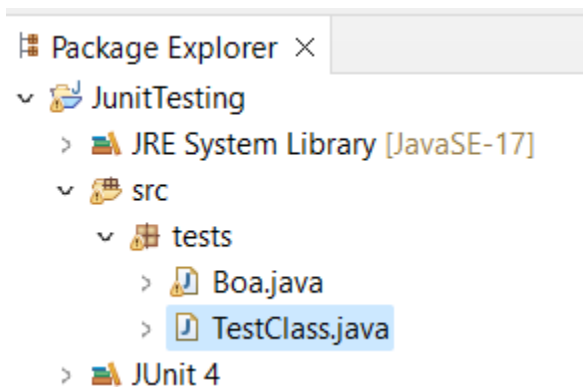
Name - **Kavish Shah**

Student ID - **202001114**

Course - **Software Engineering (IT314)**

Lab 8

1. Creating a new eclipse project and a package within the project



2. Creating a class for Boa

```

Boa.java × TestClass.java
1 package tests;
2
3 // represents a boa constrictor
4 public class Boa {
5     private String name;
6     private int length; // the length of the boa, in feet
7     private String favoriteFood;
8
9     public Boa (String name, int length, String favoriteFood){
10         this.name = name;
11         this.length = length;
12         this.favoriteFood = favoriteFood;
13     }
14     // returns true if this boa constrictor is healthy
15     public boolean isHealthy(){
16         return this.favoriteFood.equals("granola bars");
17     }
18     // returns true if the length of this boa constrictor is
19     // less than the given cage length
20     public boolean fitsInCage(int cageLength){
21         return this.length < cageLength;
22     }
23 }

```

3. Creating a JUnit test case with methods isHealthy and fitsInCage

```
Boa.java TestClass.java ×
1 package tests;
2
3 import static org.junit.Assert.*;
4 import org.junit.Before;
5 import org.junit.Test;
6
7 public class TestClass {
8     private Boa jen, 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 isHealthy() {
18        fail("Not yet implemented");
19    }
20
21    @Test
22    public void fitsInCage() {
23    }
24
25 }
```

4. Creating setup method and annotating with @before and creating jen and ken objects of boa class

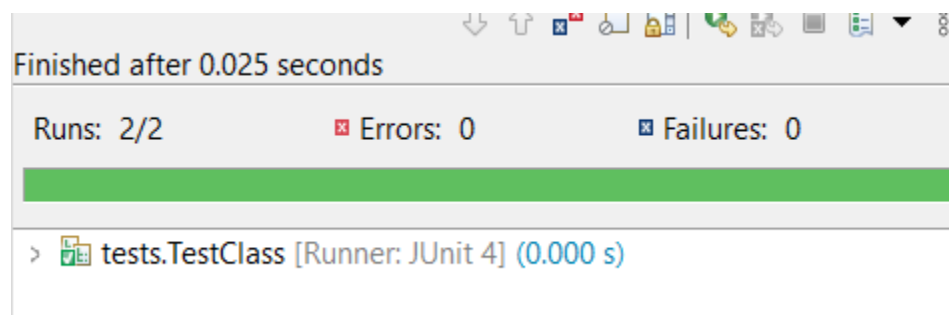
```
public class TestClass {
    private Boa jen, ken;

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

5. Writing tests for fitsInCage and isHealthy method

```
15
16 @Test
17 public void testIsHealthy() {
18     assertFalse(jen.isHealthy());
19     assertTrue(ken.isHealthy());
20 }
21
22 @Test
23 public void testFitsInCage() {
24     assertTrue(jen.fitsInCage(3));
25     assertFalse(jen.fitsInCage(2));
26     assertFalse(jen.fitsInCage(1));
27     assertFalse(jen.fitsInCage(0));
28     assertFalse(jen.fitsInCage(-1));
29     assertTrue(ken.fitsInCage(10));
30     assertFalse(ken.fitsInCage(3));
31     assertFalse(ken.fitsInCage(0));
32     assertFalse(ken.fitsInCage(-1));
33 }
34 }
```

6. Running both tests



```

24 // produces the length of the Boa in inches
25 public int lengthInInches() {
26     return 12*this.length;
27 }
28 }
29 }

```

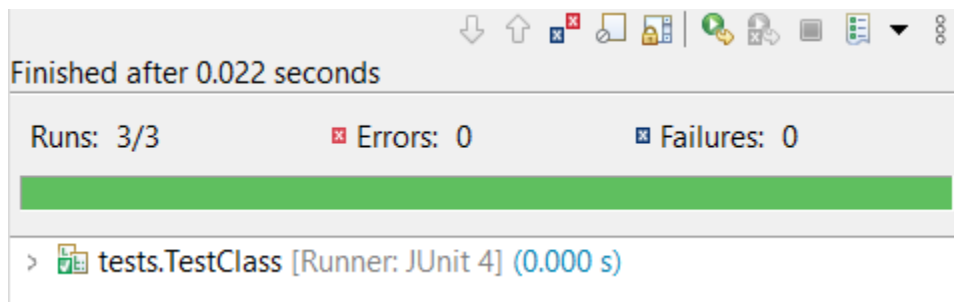
7. Writing the tests for length in inches

```

35 |
36 @Test
37 public void testLengthInInches() {
38     assertEquals(24, jen.lengthInInches());
39     assertEquals(36, ken.lengthInInches());
40 }
41 }

```

8. Running the tests



The screenshot shows a test runner interface with a toolbar at the top containing icons for navigation and actions. Below the toolbar, it states "Finished after 0.022 seconds". A summary bar shows "Runs: 3/3", "Errors: 0", and "Failures: 0". A green progress bar is visible below the summary. At the bottom, a list shows the test class "tests.TestClass [Runner: JUnit 4] (0.000 s)".

Finished after 0.022 seconds

Runs: 3/3 Errors: 0 Failures: 0

> tests.TestClass [Runner: JUnit 4] (0.000 s)