

---

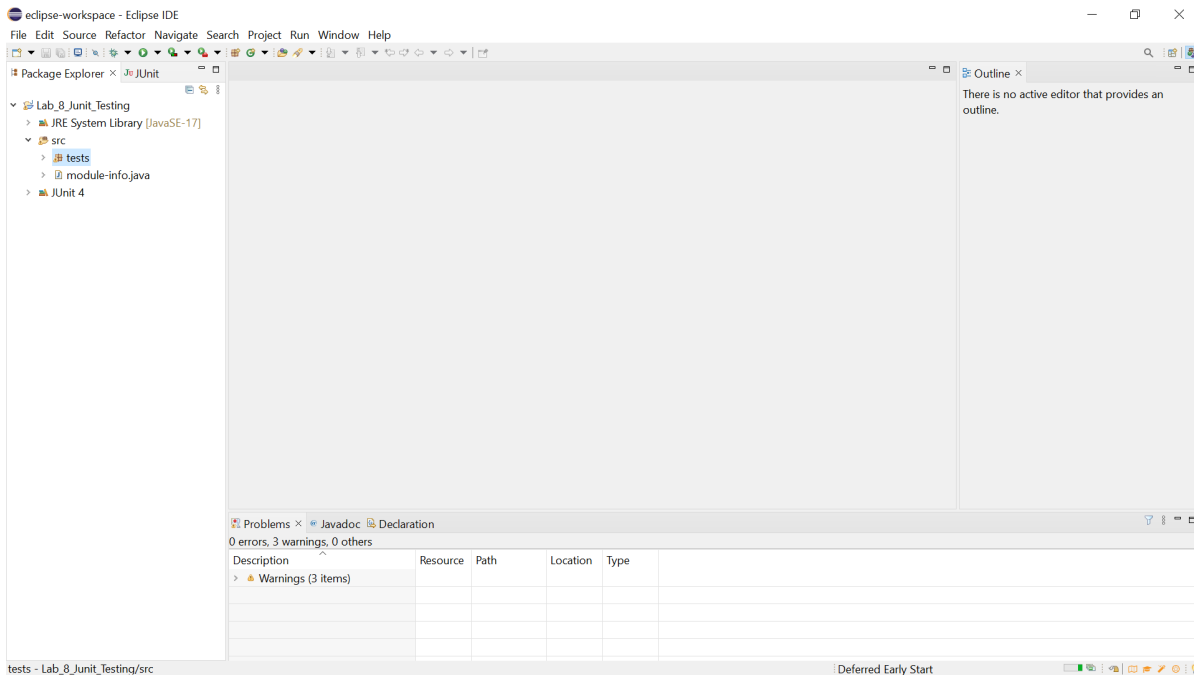
# IT 314 Software Engineering

## Lab 8

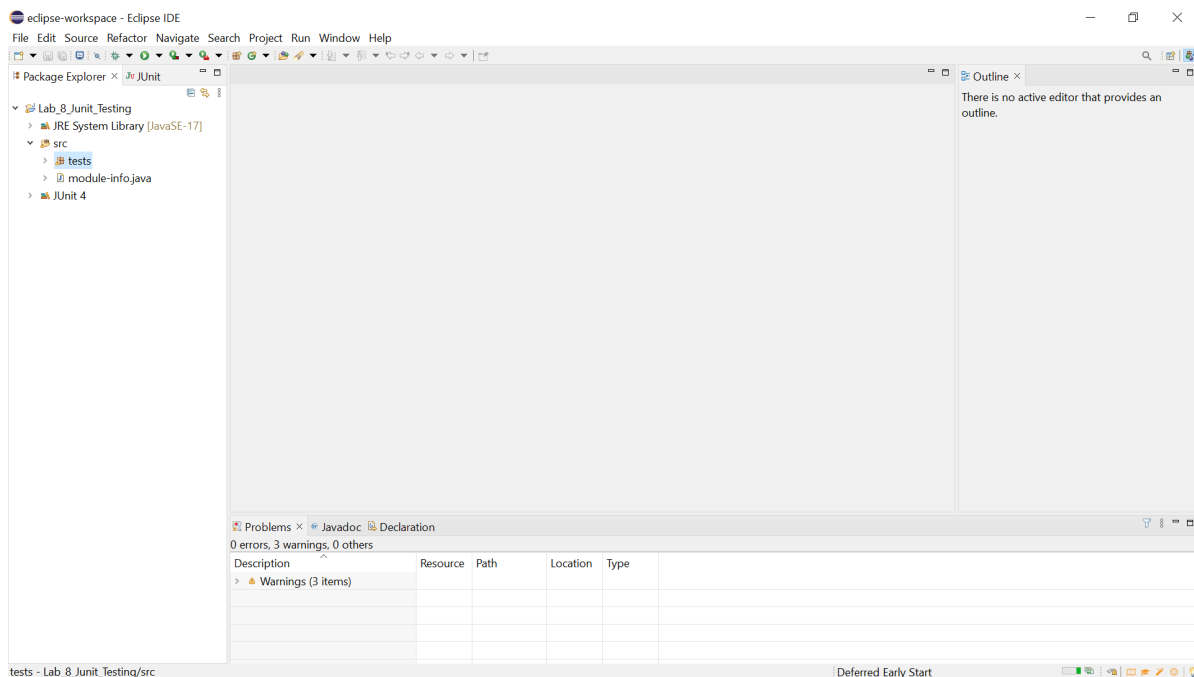
**Name:** Poriya Neel Dharmeshbhai  
**ID:** 202001139

---

Create a new Eclipse project,



Create a new class named “Boa” with the following code,



Create a JUnit test case file named “BoaTest”,

The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows the project structure with 'Lab\_8\_Junit\_Testing' containing 'src' and 'tests' folders. 'BoaTest.java' is selected in the 'tests' folder.
- Editor:** Displays the code for 'BoaTest.java':

```
1 package tests;
2
3 import static org.junit.Assert.*;
4
5 public class BoaTest {
6
7     @Test
8     public void test() {
9         fail("Test not implemented");
10    }
11 }
```
- Outline:** Shows the class 'BoaTest' with a single method 'test() : void'.
- Problems:** Shows 0 errors and 2 warnings.

Create a setup function which initializes two new instances of Bos class named “jen” and “ken”,

The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Same as the previous screenshot, with 'BoaTest.java' selected.
- Editor:** Displays the updated code for 'BoaTest.java':

```
1 package tests;
2
3 import static org.junit.Assert.*;
4
5 public class BoaTest {
6
7     @Before
8     public void setUp() throws Exception {
9         jen = new Boa("Jennifer", 2, "grapes");
10        ken = new Boa ("Kenneth", 3, "granola bars");
11    }
12
13     @Test
14     public void test() {
15         fail("Test not implemented");
16    }
17 }
```
- Outline:** Shows the class 'BoaTest' with two methods: 'setUp() : void' and 'test() : void'.
- Problems:** Shows 0 errors and 1 warning.

Now we have to implement the testIsHealthy() and testFitsInCage() functions in the “BoaTest” class.

```
10  Boa jen, ken;
11
12  @Before
13  public void setUp() throws Exception {
14      // Initialization
15      jen = new Boa("Jennifer", 2, "grapes");
16      ken = new Boa ("Kenneth", 3, "granola bars");
17  }
18
19  @Test
20  public void testIsHealthy() {
21      //testing isHealthy for two objects
22      assertEquals(false, jen.isHealthy());
23      assertEquals(true, ken.isHealthy());
24  }
25
26  @Test
27  public void testFitsInCage() {
28      // less than cage size
29      assertEquals(false, jen.fitsInCage(1));
30
31      // equal to cage size
32      assertEquals(false, jen.fitsInCage(2));
33
34      // greater than cage size
35      assertEquals(true, jen.fitsInCage(3));
36  }
```

Outline:

- tests
  - BoaTest
    - jen : Boa
    - ken : Boa
    - setUp() : void
    - testIsHealthy() : void
    - testFitsInCage() : void

Problems: 0 errors, 1 warning, 0 others

Description	Resource	Path	Location	Type
Warnings (1 item)				

It is not necessary to develop tests for both ken and jen objects in order to test the fitsInCage() method because the function is the same for both, and the results of test cases depend only on whether the specified length is greater than, less than, or equal to the actual length of the object. In both situations, the behavior will be comparable.

## Running the testcases,

```
13  public void setUp() throws Exception {
14      // Initialization
15      jen = new Boa("Jennifer", 2, "grapes");
16      ken = new Boa ("Kenneth", 3, "granola bars");
17  }
18
19  @Test
20  public void testIsHealthy() {
21      //testing isHealthy for two objects
22      assertEquals(false, jen.isHealthy());
23      assertEquals(true, ken.isHealthy());
24  }
25
26  @Test
27  public void testFitsInCage() {
28      // less than cage size
29      assertEquals(false, jen.fitsInCage(1));
30
31      // equal to cage size
32      assertEquals(false, jen.fitsInCage(2));
33
34      // greater than cage size
35      assertEquals(true, jen.fitsInCage(3));
36  }
37
38  }
39
```

Package Explorer:

- tests.BoaTest [Runner: JUnit 4] (0.001 s)
  - testIsHealthy (0.001 s)
  - testFitsInCage (0.000 s)

Failure Trace:

Finished after 0.011 seconds

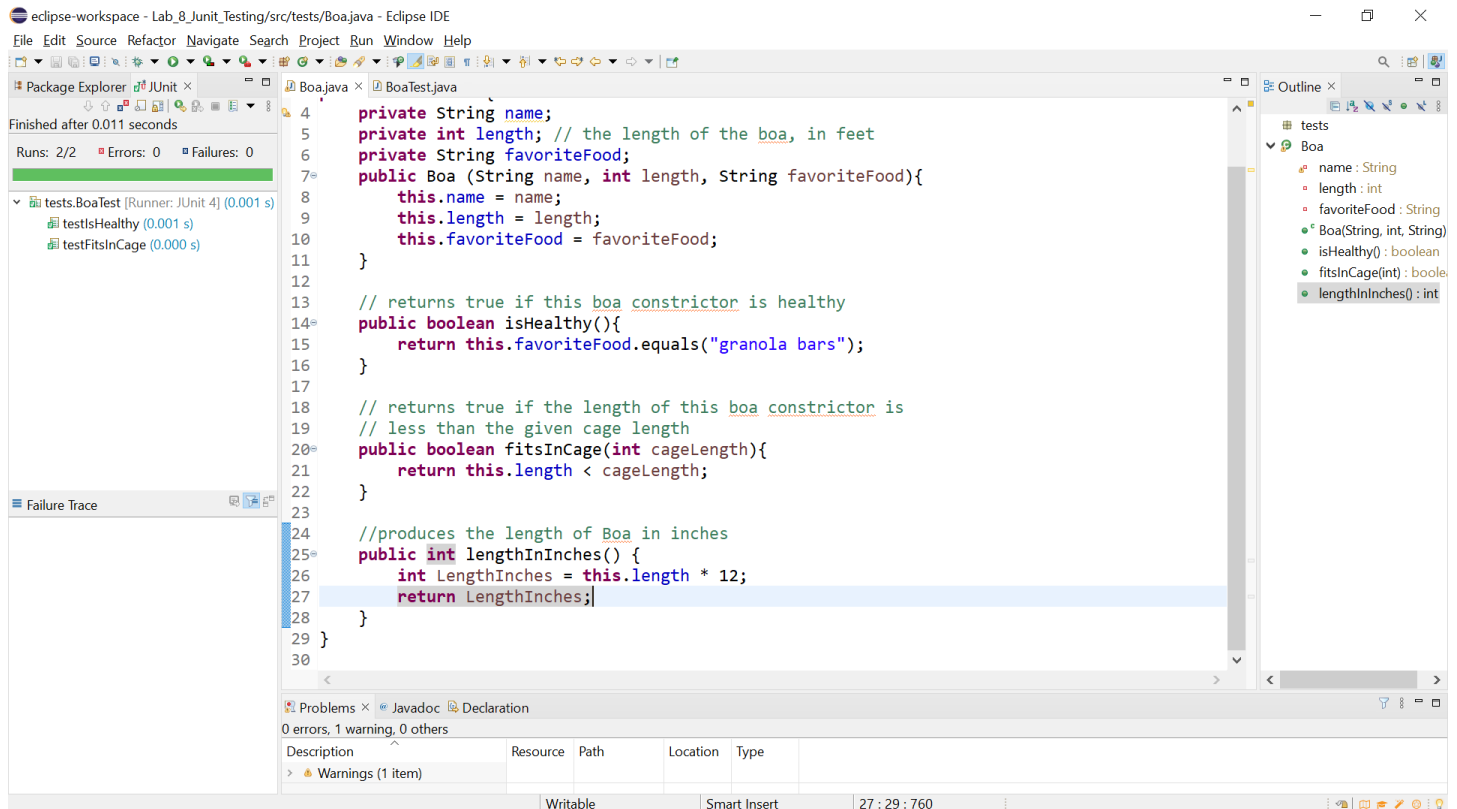
Runs: 2/2 Errors: 0 Failures: 0

Problems: 0 errors, 1 warning, 0 others

Description	Resource	Path	Location	Type
Warnings (1 item)				

We can observe that both the tests ran successfully.

Now, we create a new method to the Boa class with name `lengthInInches()` to get the length in inches.

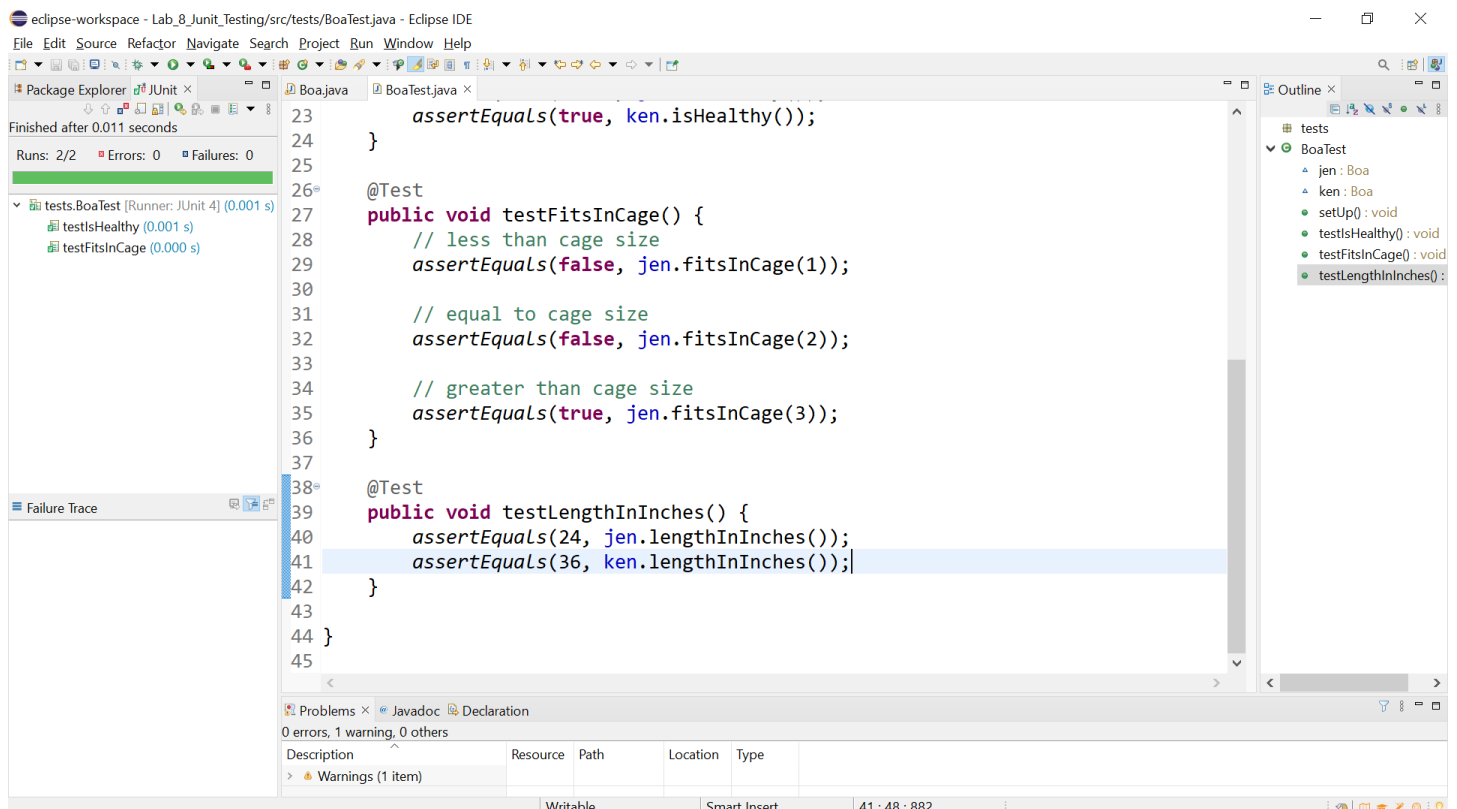


The screenshot shows the Eclipse IDE with the `Boa.java` file open. The code defines a `Boa` class with attributes `name`, `length`, and `favoriteFood`. It includes a constructor and two methods: `isHealthy()` and `fitsInCage()`. A new method, `lengthInInches()`, has been added, which calculates the length in inches by multiplying the current length by 12. The IDE's Package Explorer on the left shows the test results for `BoaTest`, indicating that all tests passed. The Outline view on the right lists the methods of the `Boa` class, including the newly added `lengthInInches()`.

```
4 private String name;
5 private int length; // the length of the boa, in feet
6 private String favoriteFood;
7 public Boa (String name, int length, String favoriteFood){
8     this.name = name;
9     this.length = length;
10    this.favoriteFood = favoriteFood;
11 }
12
13 // returns true if this boa constructor is healthy
14 public boolean isHealthy(){
15     return this.favoriteFood.equals("granola bars");
16 }
17
18 // returns true if the length of this boa constructor is
19 // less than the given cage length
20 public boolean fitsInCage(int cageLength){
21     return this.length < cageLength;
22 }
23
24 //produces the length of Boa in inches
25 public int lengthInInches() {
26     int LengthInches = this.length * 12;
27     return LengthInches;
28 }
29 }
30
```

The Boa's length is specified in feet, so I multiplied length by 12 to convert it to inches and then returned the result.

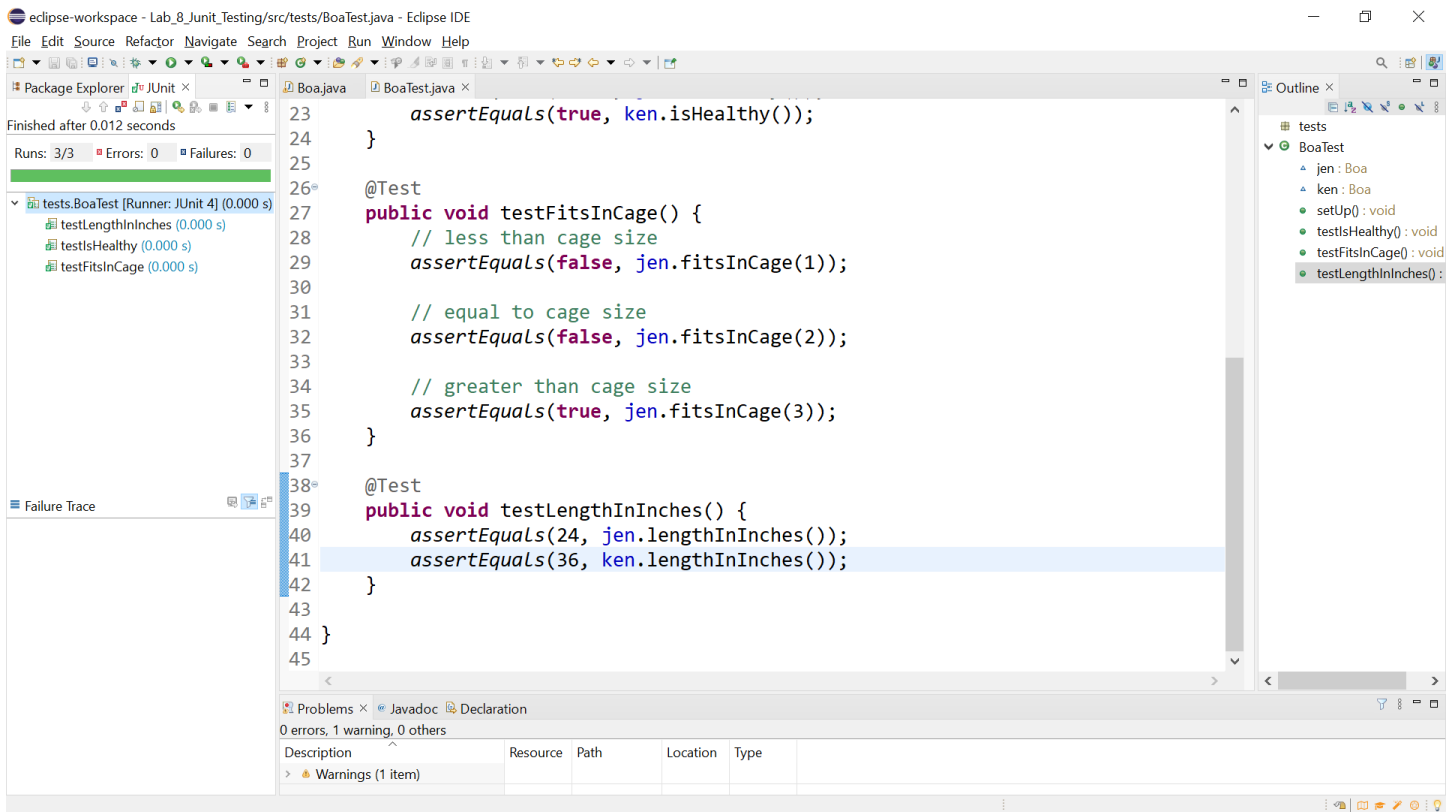
Now, writing a new test case for testing the length in inches,



The screenshot shows the Eclipse IDE with the `BoaTest.java` file open. The code contains two test methods: `testFitsInCage()` and `testLengthInInches()`. The `testLengthInInches()` method has been added, which asserts that the length of a `Boa` object is 24 inches when the length is 2 feet, and 36 inches when the length is 3 feet. The IDE's Package Explorer on the left shows the test results for `BoaTest`, indicating that all tests passed. The Outline view on the right lists the methods of the `BoaTest` class, including the newly added `testLengthInInches()`.

```
23 assertEquals(true, ken.isHealthy());
24 }
25
26 @Test
27 public void testFitsInCage() {
28     // less than cage size
29     assertEquals(false, jen.fitsInCage(1));
30
31     // equal to cage size
32     assertEquals(false, jen.fitsInCage(2));
33
34     // greater than cage size
35     assertEquals(true, jen.fitsInCage(3));
36 }
37
38 @Test
39 public void testLengthInInches() {
40     assertEquals(24, jen.lengthInInches());
41     assertEquals(36, ken.lengthInInches());
42 }
43
44 }
45
```

Running the newly created test cases,



The screenshot shows the Eclipse IDE with a Java project named 'Lab\_8\_Junit\_Testing'. The main editor displays the file 'BoaTest.java'. The code includes a package declaration, imports for 'Boa' and 'JUnit4', and two test methods. The first method, 'testFitsInCage()', uses 'assertEquals' to verify the 'fitsInCage' method of a 'Boa' object named 'jen' with values 1, 2, and 3. The second method, 'testLengthInInches()', uses 'assertEquals' to verify the 'lengthInInches' method of 'jen' with the value 24, and 'ken' with the value 36. The left sidebar shows the 'Package Explorer' with 'tests.BoaTest' selected, and the 'JUnit' runner showing 'testLengthInInches' and 'testFitsInCage' as passed tests. The right sidebar shows the 'Outline' view with the same test methods listed. The bottom status bar indicates '0 errors, 1 warning, 0 others'.

```
23     assertEquals(true, ken.isHealthy());
24 }
25
26 @Test
27 public void testFitsInCage() {
28     // less than cage size
29     assertEquals(false, jen.fitsInCage(1));
30
31     // equal to cage size
32     assertEquals(false, jen.fitsInCage(2));
33
34     // greater than cage size
35     assertEquals(true, jen.fitsInCage(3));
36 }
37
38 @Test
39 public void testLengthInInches() {
40     assertEquals(24, jen.lengthInInches());
41     assertEquals(36, ken.lengthInInches());
42 }
43
44 }
45
```

As a result, test cases have been created for the specified Boa class, and the necessary Junit test cases have been used to test all three methods.

**\*\*\*END OF ASSIGNMENT\*\*\***