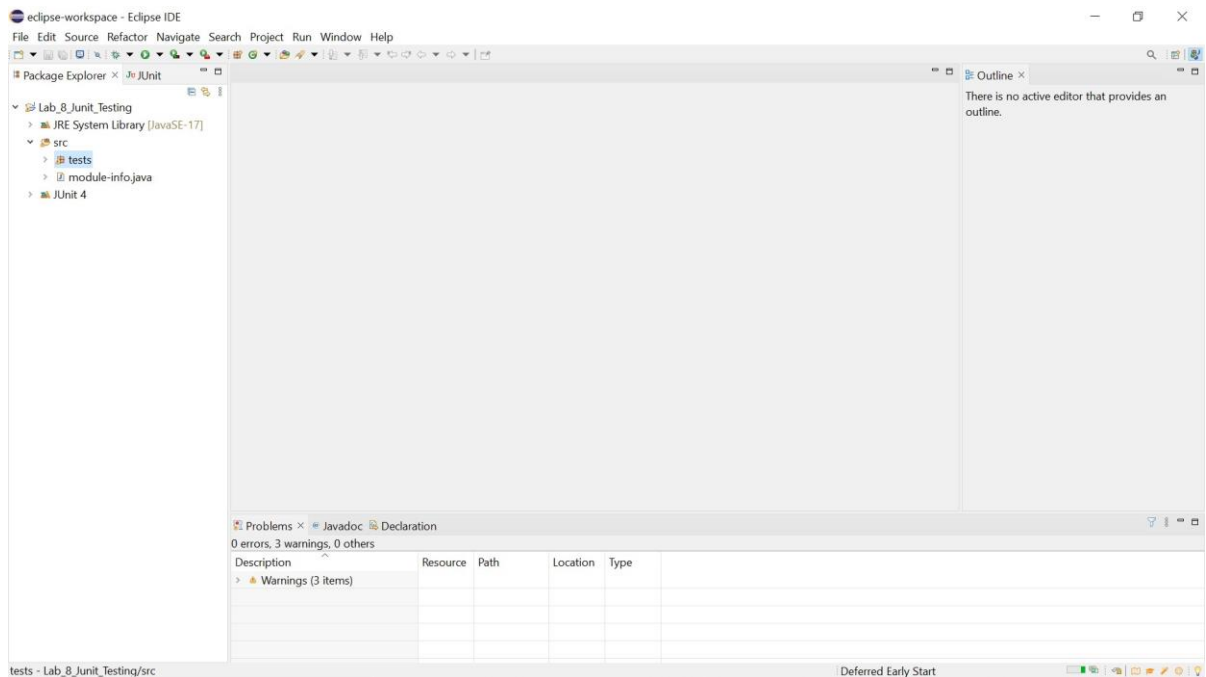
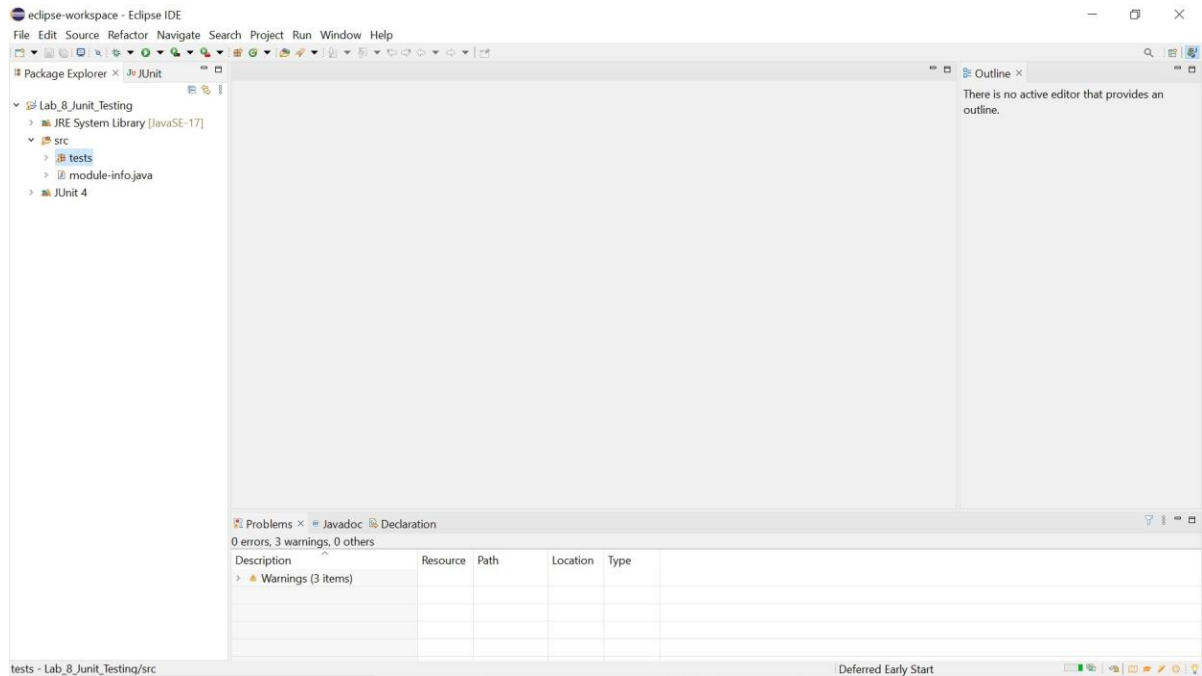

IT 314 *Software Engineering*
Lab 8 Unit Testing with JUnit
Name: Sangani Kishan
ID: 202001265

Lab Exercises : -

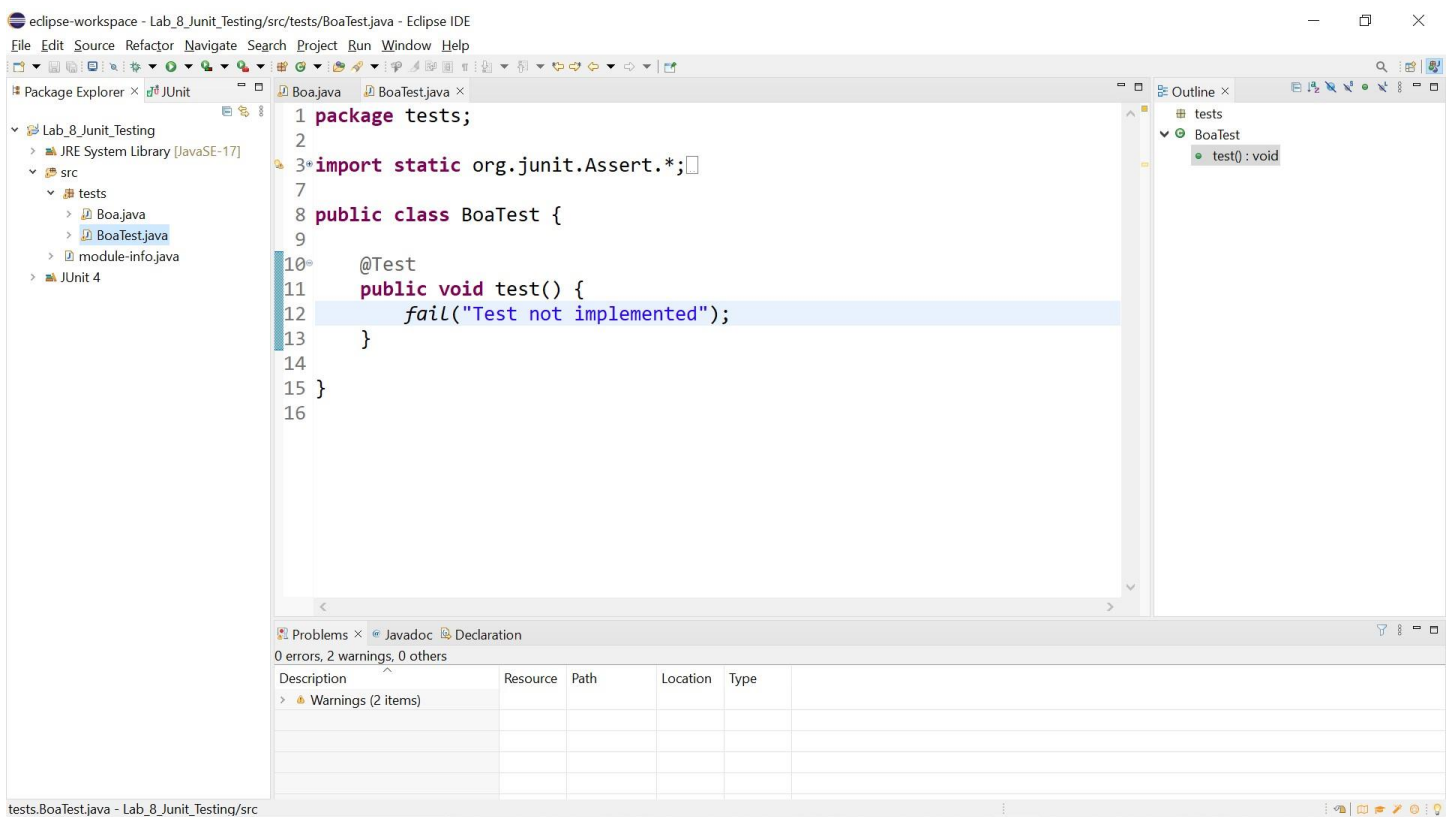
Create a new Eclipse project,



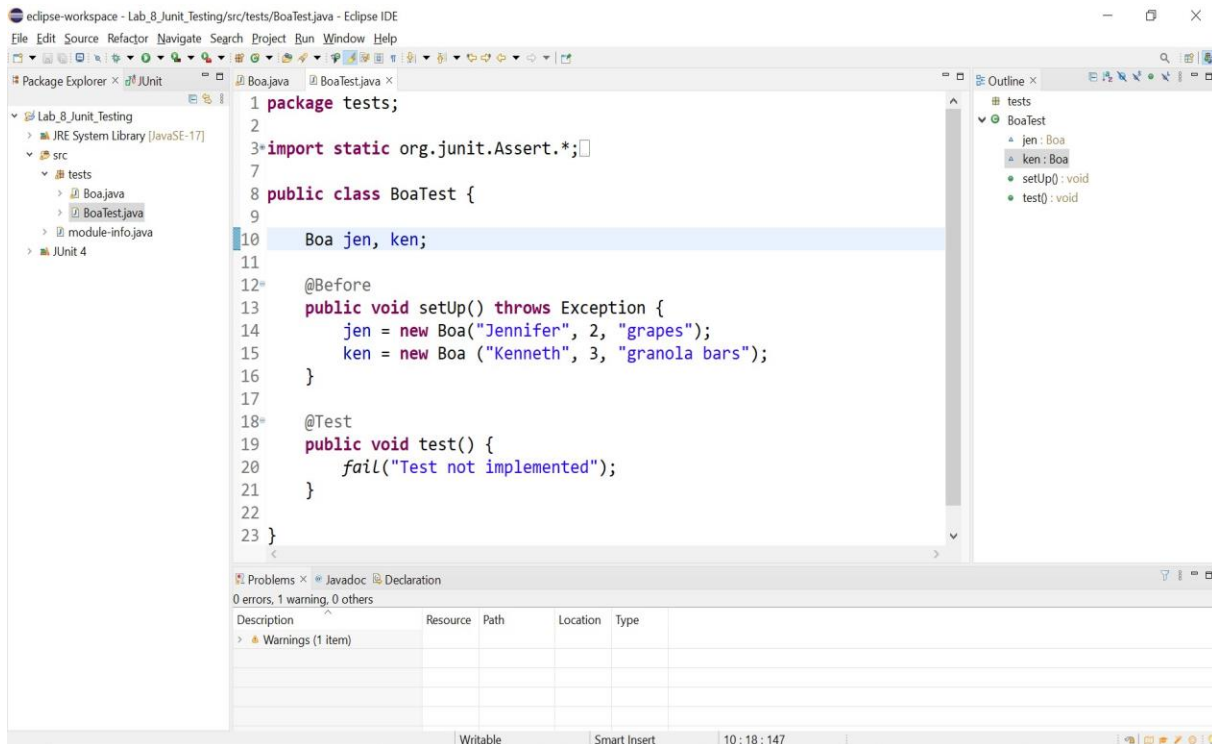
Create a new class named “Boa” with the following code,



Create a JUnit test case file named “BoaTest”,



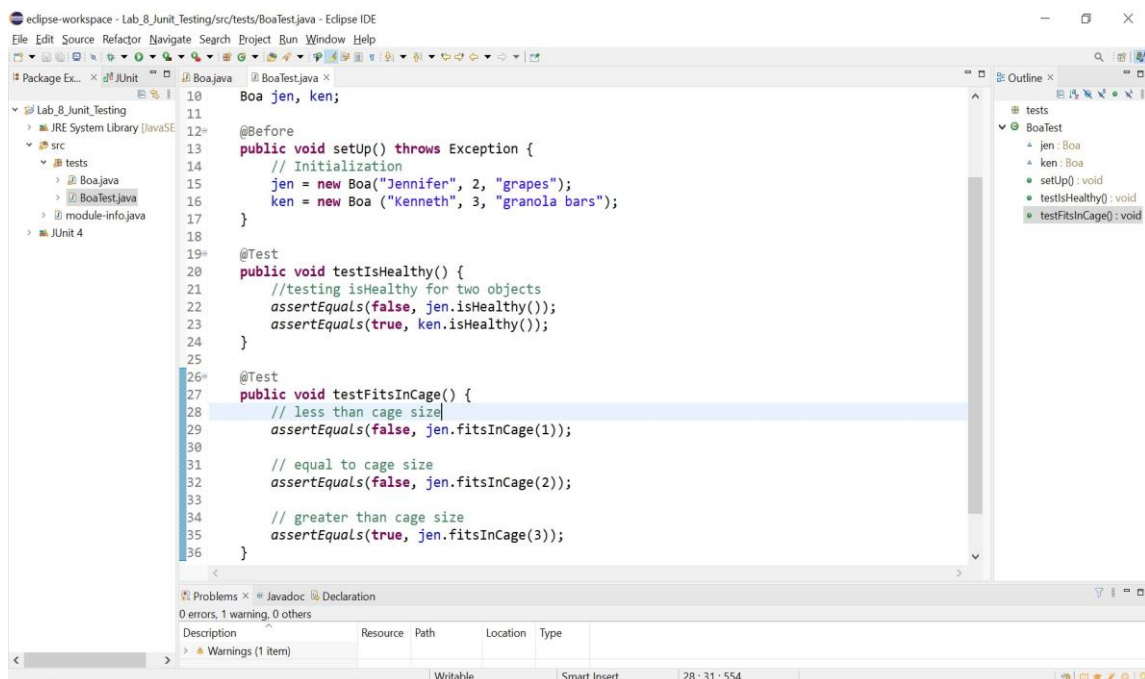
Create a setup function which initializes two new instances of `Boa` class named “jen” and “ken”,



The screenshot shows the Eclipse IDE with the `BoaTest.java` file open. The code defines a package `tests`, imports `org.junit.Assert.*`, and defines a class `BoaTest`. Inside `BoaTest`, there is a static field `Boa jen, ken;`. The `@Before` method `setUp()` initializes `jen` as a `Boa` object with name "Jennifer", age 2, and food "grapes", and `ken` as a `Boa` object with name "Kenneth", age 3, and food "granola bars". The `@Test` method `test()` currently calls `fail("Test not implemented");`. The Package Explorer on the left shows the project structure, and the Outline on the right shows the class and method declarations.

```
1 package tests;
2
3 import static org.junit.Assert.*;
4
5 public class BoaTest {
6
7     Boa jen, ken;
8
9     @Before
10    public void setUp() throws Exception {
11        jen = new Boa("Jennifer", 2, "grapes");
12        ken = new Boa("Kenneth", 3, "granola bars");
13    }
14
15    @Test
16    public void test() {
17        fail("Test not implemented");
18    }
19 }
```

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

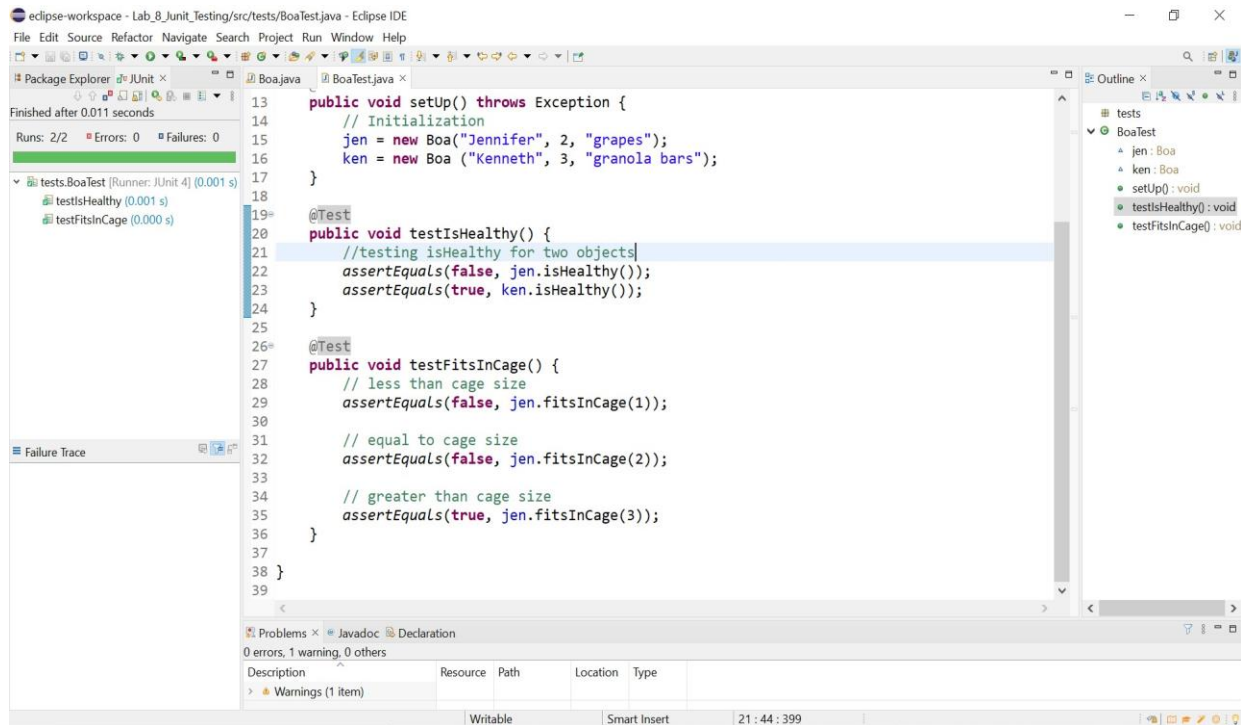


The screenshot shows the same Eclipse IDE with the `BoaTest.java` file, but now with two additional test methods implemented. The `testIsHealthy()` method tests the `isHealthy()` property of `jen` and `ken`, asserting that `jen` is not healthy and `ken` is healthy. The `testFitsInCage()` method tests the `fitsInCage()` property of `jen` for cage sizes 1, 2, and 3, asserting that `jen` does not fit in a cage of size 1 or 2, but does fit in a cage of size 3.

```
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 }
```

It is not necessary to develop tests for both ken and jen objects in order to test the `itsInCage()` 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 behaviour will be comparable

Running the testcases,

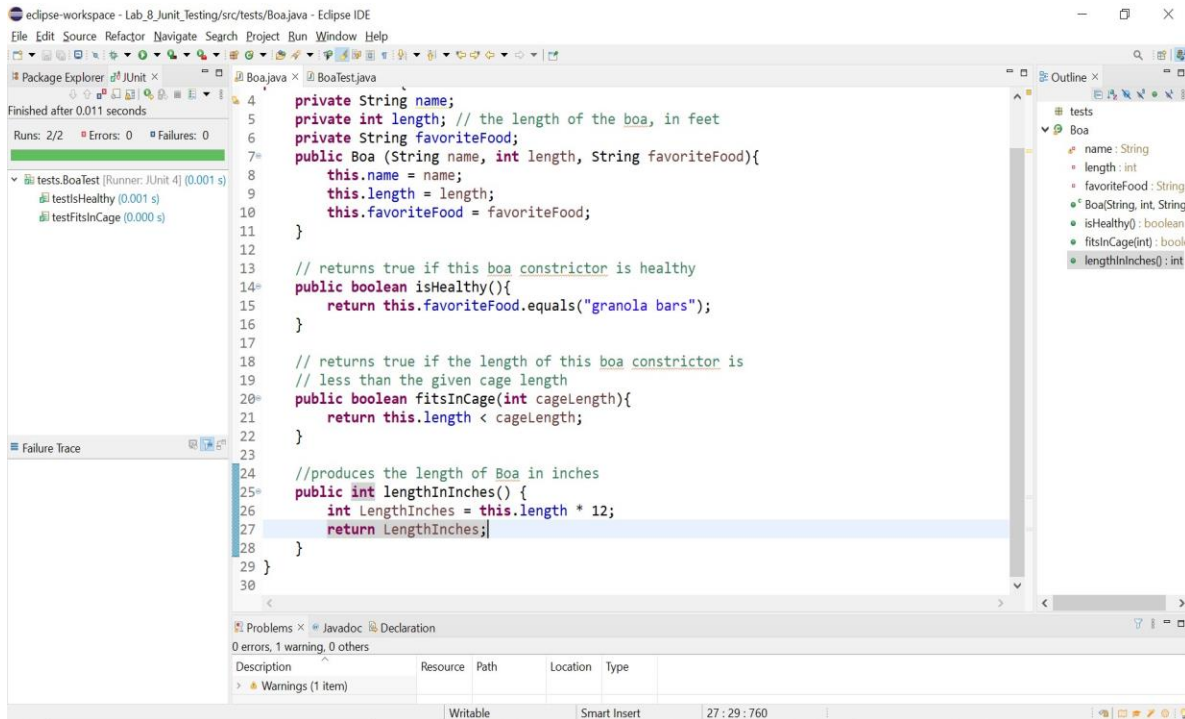


The screenshot shows the Eclipse IDE with a project named 'eclipse-workspace - Lab_8_Junit_Testing/src/tests/Boatest.java'. The main editor displays the code for `Boatest.java`, which includes a `setUp()` method for initialization, a `testIsHealthy()` method, and a `testFitsInCage()` method. The `testFitsInCage()` method tests the `fitsInCage()` method of the `Boa` class with different cage sizes (1, 2, 3) and compares the results to expected values. The test results are shown in the 'JUnit' view on the left, indicating that all tests passed (2/2 runs, 0 errors, 0 failures). The 'Outline' view on the right shows the structure of the test class, including the `setUp()` method and the two test methods. The 'Problems' view at the bottom shows 0 errors and 1 warning.

```
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 }
```

We can observe that both the tests ran successfully.

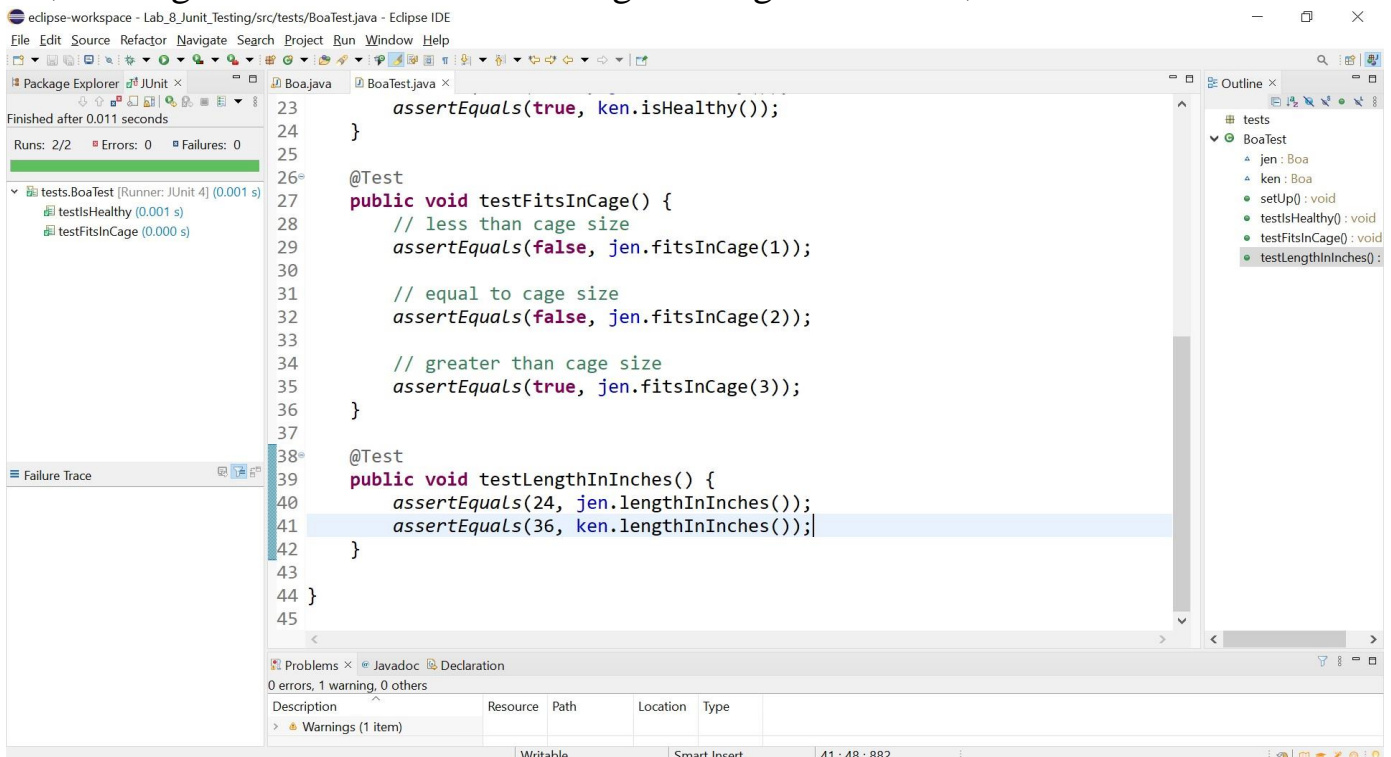
Now, we create a new method to the Boa class with name `lengthInInches()` to get the length in inches.



```
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
```

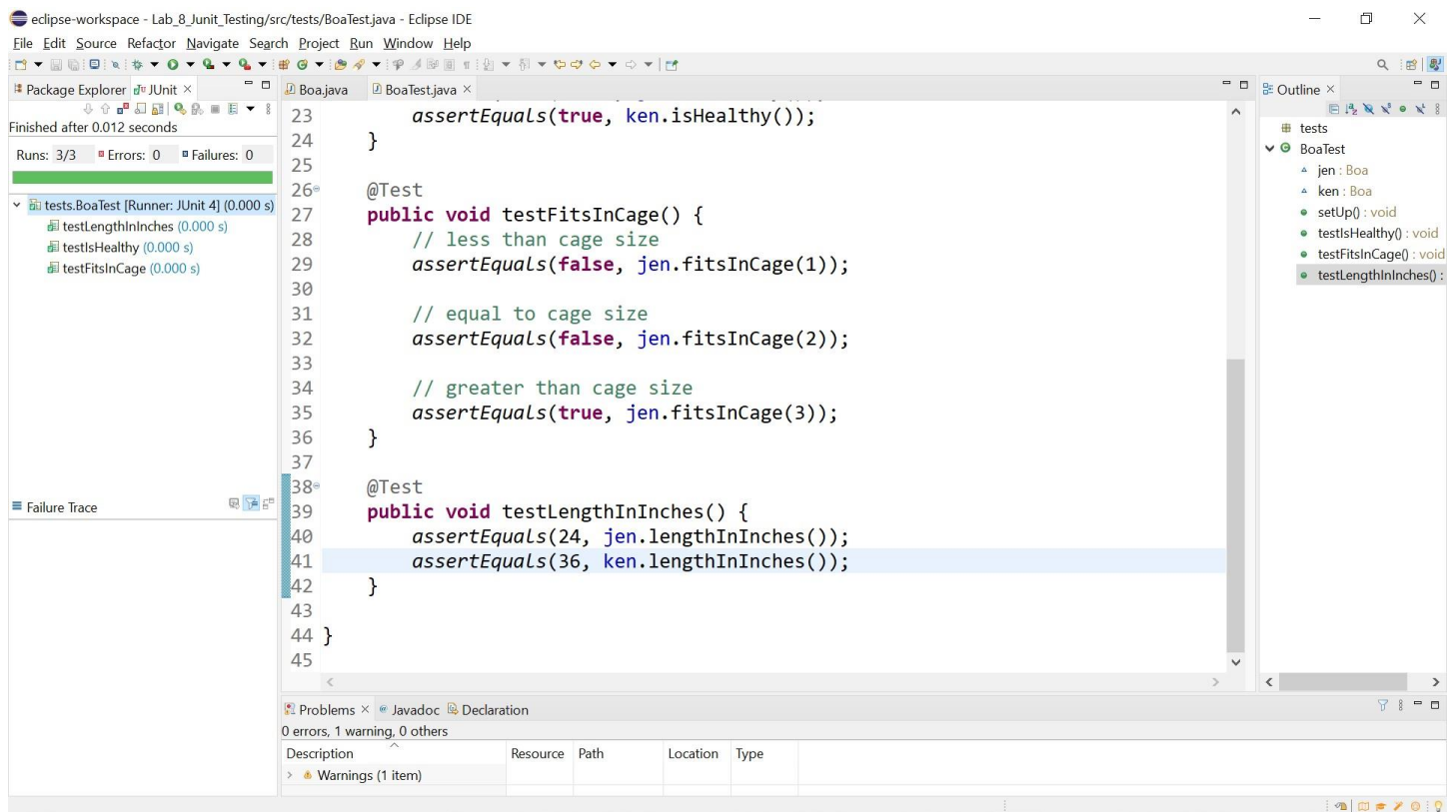
The Boa's length is specified in feet, so I multiplied length by 12 to convert it to inches and then returned the result.

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



```
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,



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.