IT 314 - Software Engineering

Date - Apr 20, 2023

Name - Chhagani Krunal Ajaybhai

Id - 202001158

Lab Exercises :

1. Create a new Eclipse project, and within the project create a package.

→ First of all , created an eclipse project and then created a new package.

2. Create a class for a Boa. Here's the code you can use (you may copy/paste):

→ Below is the provided class for testing "Boa".

Code :

```
// represents a boa constrictor
public class Boa {
private String name;
private int length; // the length of the boa, in feet
private String favoriteFood;
public Boa (String name, int length, String favoriteFood){
this.name = name;
this.length = length;
this.favoriteFood = favoriteFood;
}
// returns true if this boa constrictor is healthy
public boolean isHealthy(){
return this.favoriteFood.equals("granola bars");
}
// returns true if the length of this boa constrictor is
// less than the given cage length
public boolean fitsInCage(int cageLength){
return this.length < cageLength;
}
}
```

3. Follow the instructions in the JUnit tutorial in the section “Creating a JUnit Test Case in Eclipse”. You’ll be creating a test case for the class Boa. When you’re asked to select test method stubs, select both isHealthy() and fitsInCage(int).

→ After that a Junit test is created for testing the “Boa” class named “Boa_test”.

→ This is the test code for running initial test cases.

Code :

```
package test_lab8;
import static org.junit.jupiter.api.Assertions.*;
import org.junit.Before;
import org.junit.jupiter.api.Test;
class Boa_test {
  @Test
  public void testIsHealthyWithFavoriteFoodGranolaBars() {
     Boa boa = new Boa("Benny", 5, "granola bars");
     assertTrue(boa.isHealthy());
  }

  @Test
  public void testIsHealthyWithFavoriteFoodNotGranolaBars() {
     Boa boa = new Boa("Benny", 5, "mice");
     assertFalse(boa.isHealthy());
  }

  @Test
  public void testFitsInCageWhenLengthLessThanCageLength() {
     Boa boa = new Boa("Benny", 5, "granola bars");
     assertTrue(boa.fitsInCage(10));
  }

  @Test
  public void testFitsInCageWhenLengthGreaterThanCageLength() {
     Boa boa = new Boa("Benny", 20, "granola bars");
     assertFalse(boa.fitsInCage(10));

 }
}
```

→ Now we can run our test cases successfully.

4. Now it’s time to write some unit tests. Notice that the BoaTest class that JUnit created for you contains stubs for several methods. The first stub (for the method setUp()) is annotated with @Before. The @Before annotation denotes that the method setUp() will be run prior to the execution of each test method. setUp() is typically used to initialize data needed by each test. Modify the setUp() method so that it creates a couple of Boa objects,

→ Here , I have used @Before tag to run another test case. This @Before tag will run before the particular test case.

Code :

```
   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");
```

```
}
```

**5. JUnit also provided stubs for two test methods, each annotated with @Test. Work on the testIsHealthy() method first. The purpose of this method is to check that the isHealthy() method in the Boa class behaves the way it's supposed to. In the JUnit tutorial, read the section on "Writing Tests". Modify the testIsHealthy() method so that it checks the results of activating the isHealthy() method on the two Boa objects you
created in setup(). Likewise, modify the testFitsInCage() method to test the results of that method. Make sure your test is robust; it should check the results when the cage length is less than the length of the boa, when the cage length is equal to the length of the boa, and when the cage length is greater than the length of the boa. Should you write tests for both jen
and ken?**

Code :

```
 @Test
  public void testIsHealthy() {
         Boa jen = new Boa("Jen", 5, "granola bars");
      Boa ken = new Boa("Ken", 6, "mice");


    assertTrue(jen.isHealthy());
    assertFalse(ken.isHealthy());
  }

  @Test
 public void testFitsInCage() {
       Boa jen = new Boa("Jen", 5, "granola bars");
     Boa ken = new Boa("Ken", 6, "mice");

    assertFalse(jen.fitsInCage(2));
    assertTrue(jen.fitsInCage(10));
    assertTrue(jen.fitsInCage(15));
    assertFalse(ken.fitsInCage(4));
    assertTrue(ken.fitsInCage(15));
    assertTrue(ken.fitsInCage(20));
 }
```

6. Now you can run your tests. Read the section “Running Your Test Case” in the tutorial. Did you get a green bar in the JUnit pane? If you got a red bar, use the output in the JUnit pane to determine which test(s) failed. Fix your tests, and try running the test case again. It’s important to note that a red bar doesn’t necessarily mean that the test case is written incorrectly; it could be that the method that’s being tested isn’t correct. In fact, that’s what unit testing is supposed to do – help us find errors in our code. When a test fails, you need to determine

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

Add a new test case to the BoaTest class that tests the lengthInInches() method. Make
sure you annotate the new test method with @Test. Run your tests.

→ Creating a new method with signature lengthInInches in the “Boa” class.

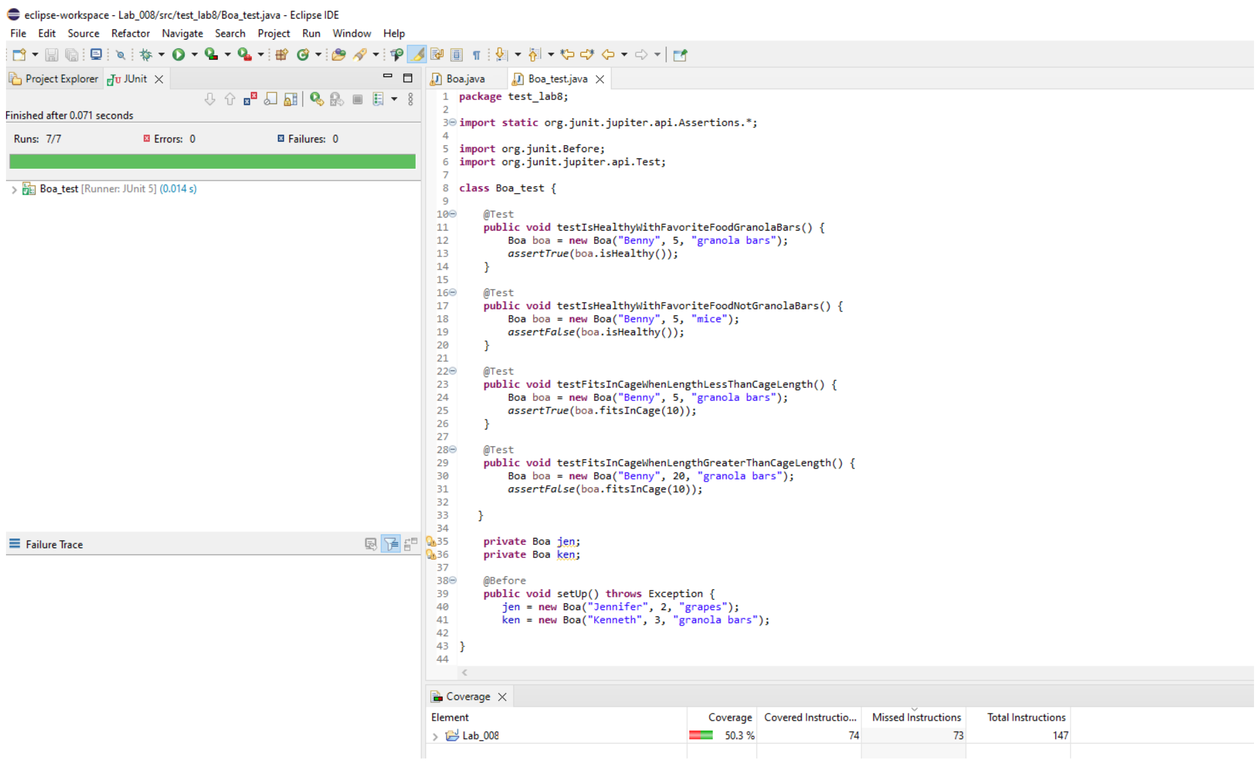
Method Code :

```
public int lengthInInches() {
  return this.length * 12;
}
```

Testing code:

```
@Test
  public void testLengthInInches() {
     Boa boa = new Boa("John", 5, "grapes");
     int expectedLengthInInches = 60;
     int actualLengthInInches = boa.lengthInInches();
     assertEquals(expectedLengthInInches, actualLengthInInches);
  }
```

→ Following are screenshots for testing the code :


eclipse-workspace - Lab_008/src/test_lab8/Boa_test.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
JUnit
Boa.java
Boa_test.java
Failure Trace
Coverage

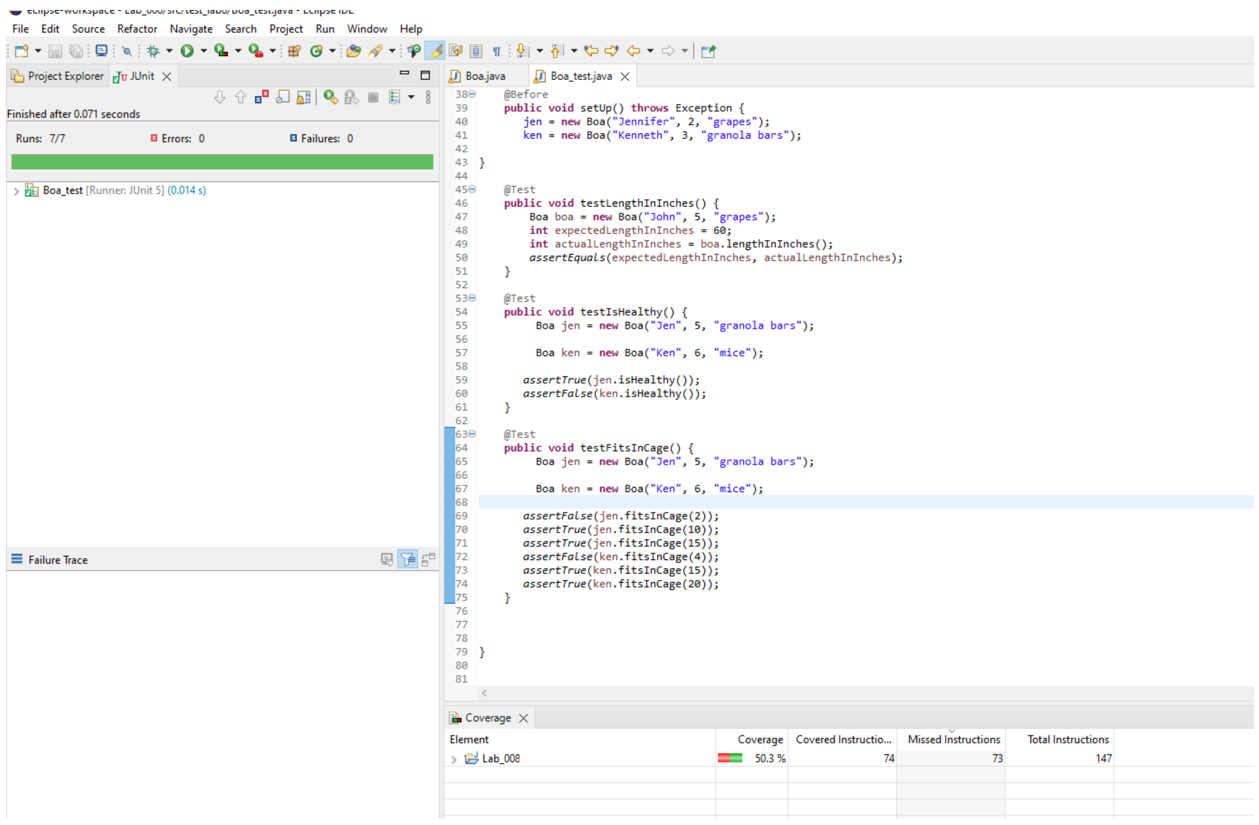
Finished after 0.071 seconds

Runs: 7/7 Errors: 0 Failures: 0

Boa_test [Runner: JUnit 5] (0.014 s)

```java
package test_lab8;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.Before;
import org.junit.jupiter.api.Test;

class Boa_test {

    @Test
    public void testIsHealthyWithFavoriteFoodGranolaBars() {
        Boa boa = new Boa("Benny", 5, "granola bars");
        assertTrue(boa.isHealthy());
    }

    @Test
    public void testIsHealthyWithFavoriteFoodNotGranolaBars() {
        Boa boa = new Boa("Benny", 5, "mice");
        assertFalse(boa.isHealthy());
    }

    @Test
    public void testFitsInCageWhenLengthLessThanCageLength() {
        Boa boa = new Boa("Benny", 5, "granola bars");
        assertTrue(boa.fitsInCage(10));
    }

    @Test
    public void testFitsInCageWhenLengthGreaterThanCageLength() {
        Boa boa = new Boa("Benny", 20, "granola bars");
        assertFalse(boa.fitsInCage(10));

   }

    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");

}
```

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
Lab_008	50.3 %	74	73	147


eclipse-workspace - Lab_008/src/test_lab8/Boa_test.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
JUnit
Boa.java
Boa_test.java
Failure Trace
Coverage

Finished after 0.071 seconds

Runs: 7/7 Errors: 0 Failures: 0

Boa_test [Runner: JUnit 5] (0.014 s)

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

}

    @Test
    public void testLengthInInches() {
        Boa boa = new Boa("John", 5, "grapes");
        int expectedLengthInInches = 60;
        int actualLengthInInches = boa.lengthInInches();
        assertEquals(expectedLengthInInches, actualLengthInInches);
    }

    @Test
    public void testIsHealthy() {
         Boa jen = new Boa("Jen", 5, "granola bars");

         Boa ken = new Boa("Ken", 6, "mice");

       assertTrue(jen.isHealthy());
       assertFalse(ken.isHealthy());
    }

    @Test
    public void testFitsInCage() {
         Boa jen = new Boa("Jen", 5, "granola bars");

         Boa ken = new Boa("Ken", 6, "mice");

       assertFalse(jen.fitsInCage(2));
       assertTrue(jen.fitsInCage(10));
       assertTrue(jen.fitsInCage(15));
       assertFalse(ken.fitsInCage(4));
       assertTrue(ken.fitsInCage(15));
       assertTrue(ken.fitsInCage(20));
    }



}
```

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
Lab_008	50.3 %	74	73	147

8. Here are some other things you should know about unit testing and JUnit:

→ Each method annotated with @Test will be run, but the order of the tests is not guaranteed.
→ Any method annotated with @Before will be run before each test executes.
→ Any method annotated with @After will be run after each test executes.