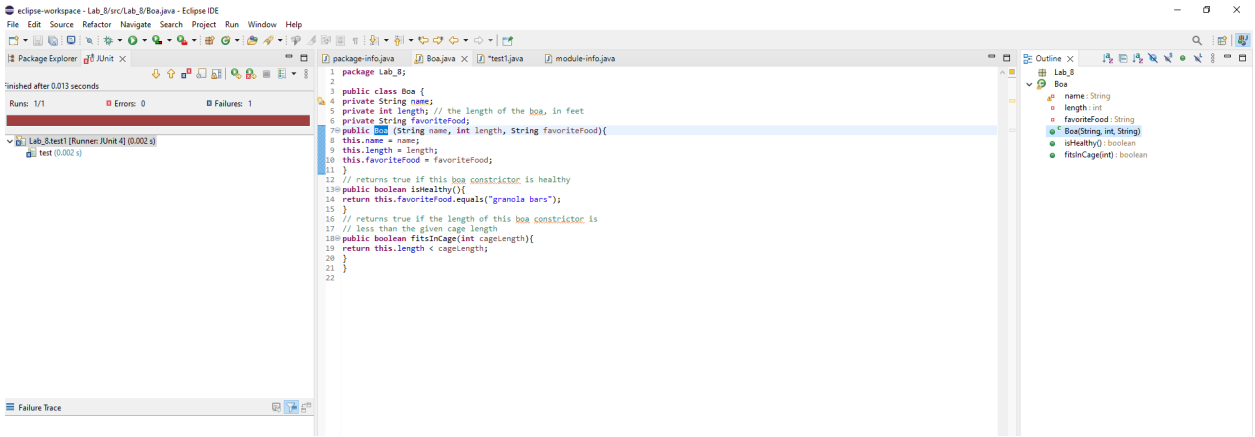
IT 314
LAB 8
Junit Testing Framework

Name: Isha Shah
ID: 202001163
Date: 20 April, 2023

1. Create a package named MyPackage and a class Boo under eclipse IDE.



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

// 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;
}
}
```

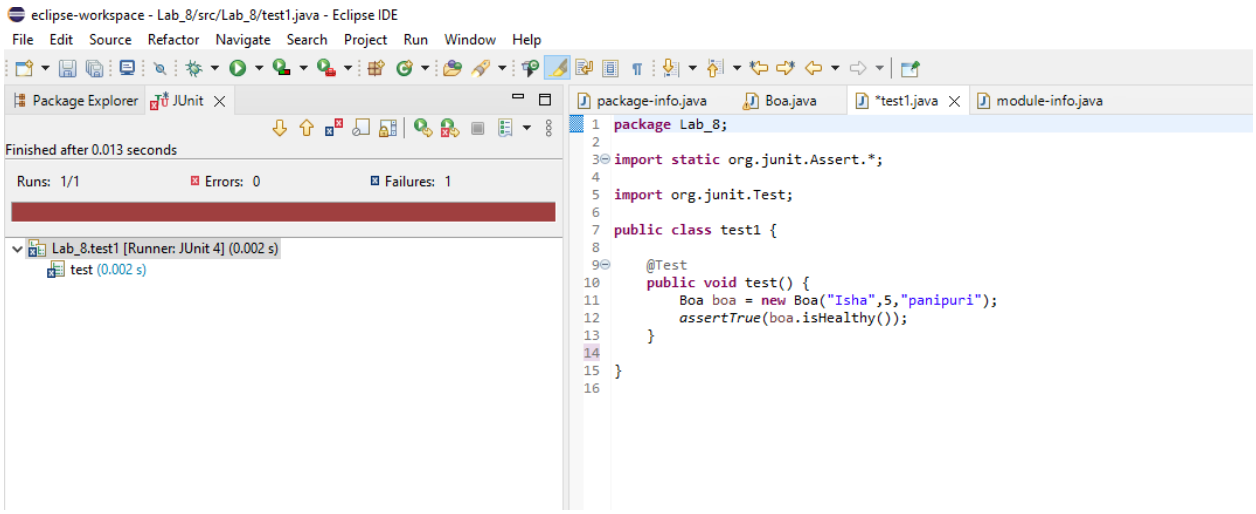
3. Follow the instructions in the JUnit tutorial in the section “Creating a JUnit Test Case in Eclipse”. You’ll be creating a test case for the class Boa. When you’re asked to select test method stubs, select both isHealthy() and fitsInCage(int).

All test codes:

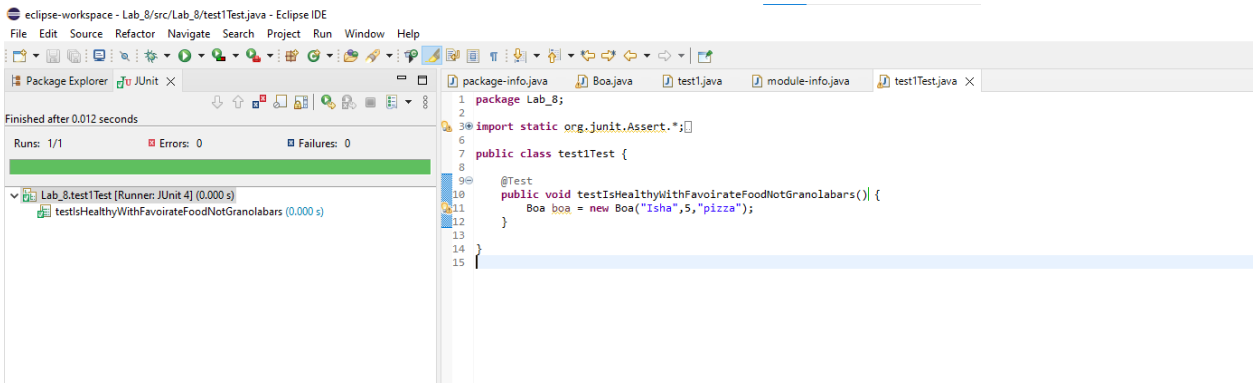
```
package Lab_8;
import static org.junit.Assert.*;
import org.junit.Test;
public class test1 {
	@Test
	public void isHealthyforgranolabarsfood() {
		Boa boa = new Boa("Isha",5,"granola bars");
		assertTrue(boa.isHealthy());
	}


	public void isHealthyforgranolabarsnotasfavoiratefood() {
		Boa boa = new Boa("Isha",5,"panipuri");
		assertFalse(boa.isHealthy());
	}

	public void testFitsInCagewhenLengthLessThanCageLength() {
		Boa boa = new Boa("Isha",5,"granola bars");
		assertTrue(boa.fitsInCage(10));
	}
```

```
	public void testFitsInCagewhenLengthGreaterThanCageLength() {
		Boa boa = new Boa("Isha",5,"granola bars");
		assertFalse(boa.fitsInCage(3));
	}
}
```

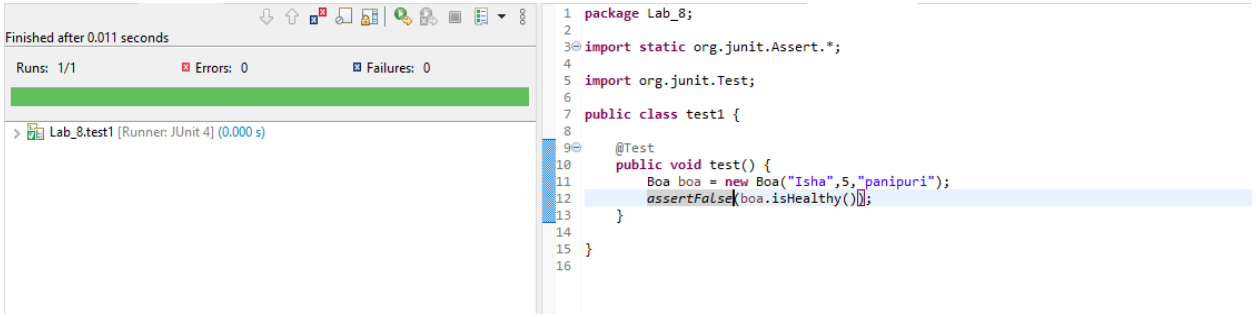
Test 1:

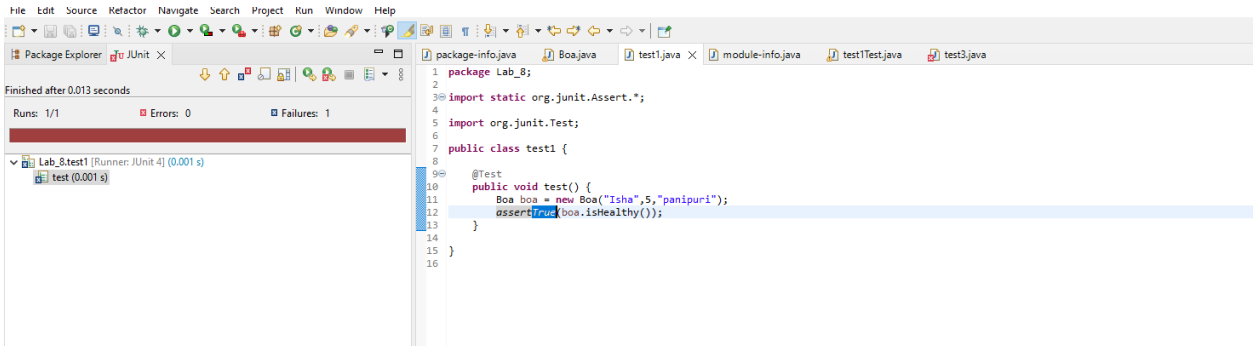


Test2:

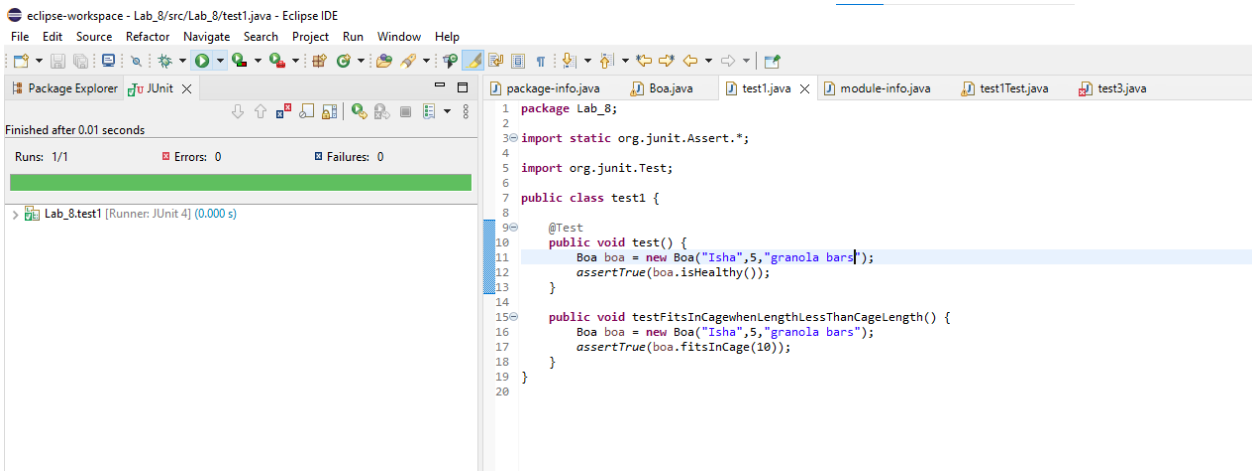


Test3:

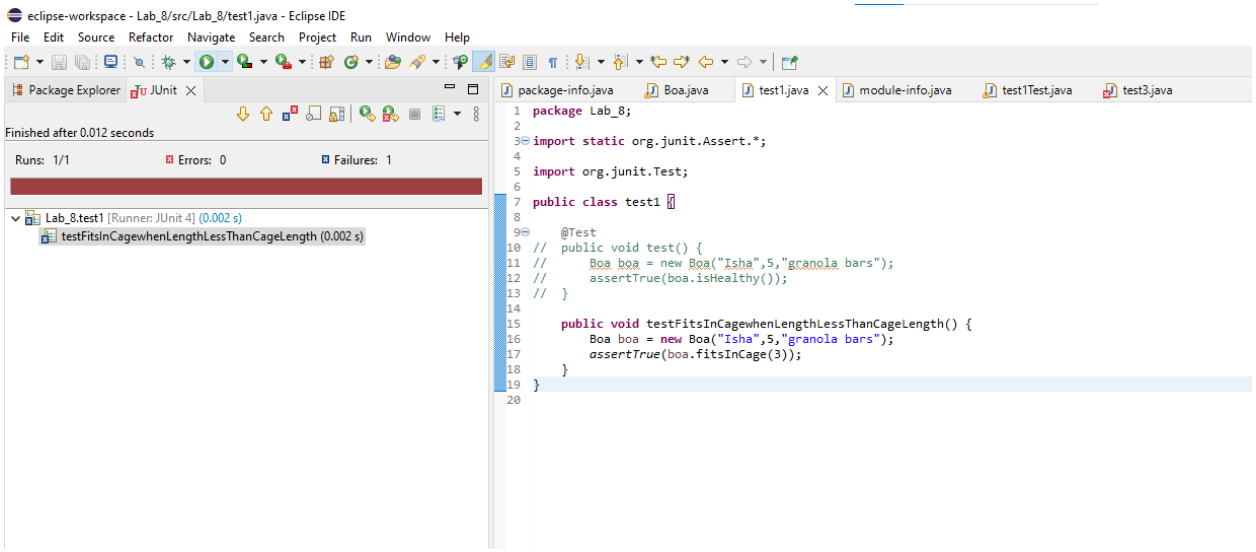




Test 4:



→ This is true as length(5) is less that cage length defined (10) hence it fits.

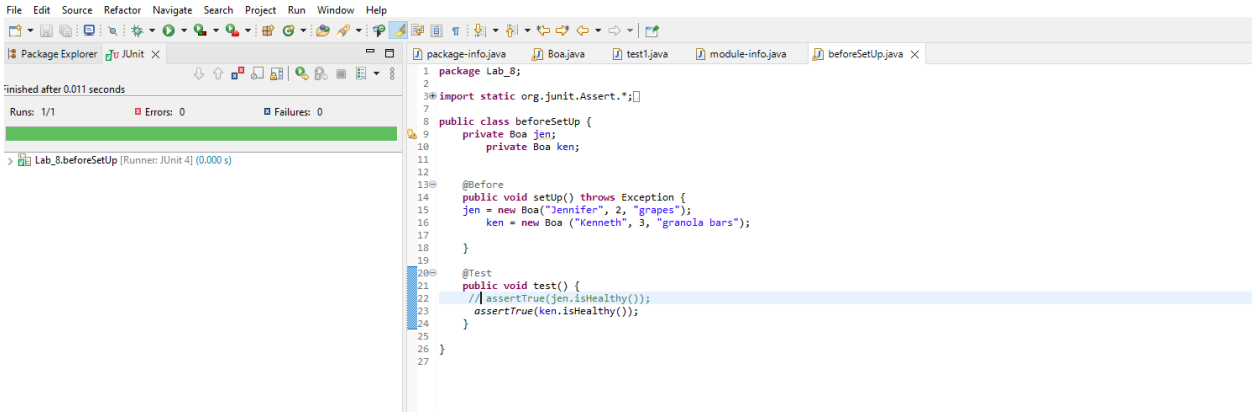


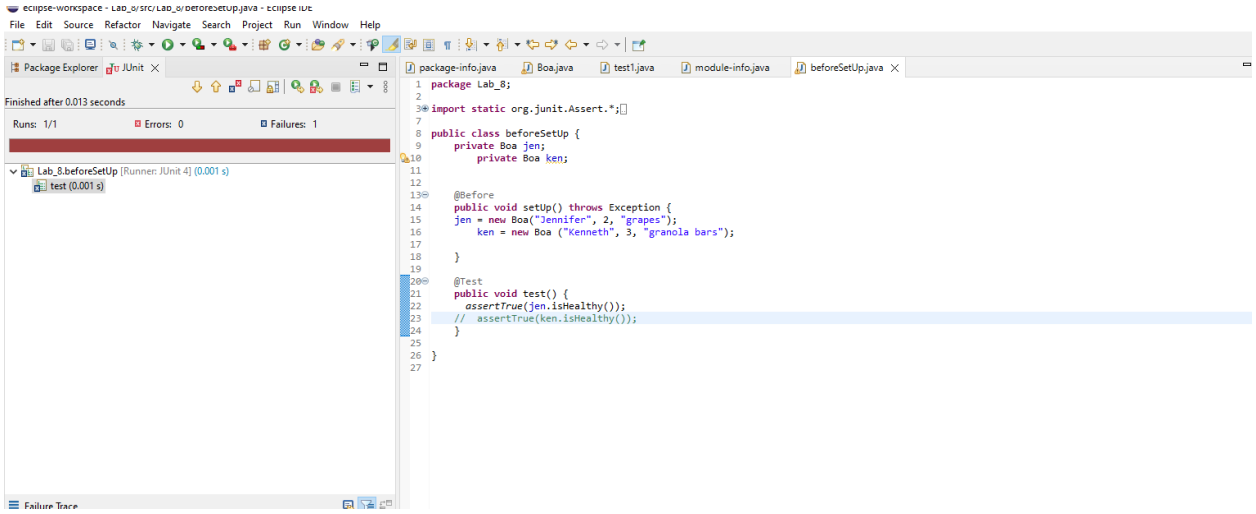
→ This fails as length (5) is less than cage lengthy defined (3) in this case.

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, as follows:

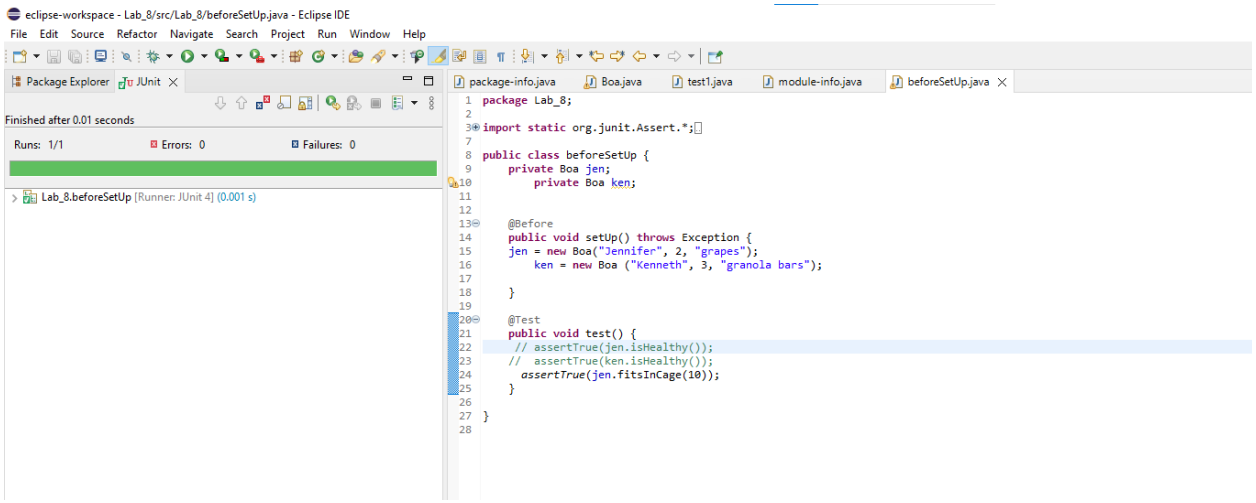
```
@Before
public void setUp() throws Exception {
jen = new Boa("Jennifer", 2, "grapes");
ken = new Boa ("Kenneth", 3, "granola bars");
}
```

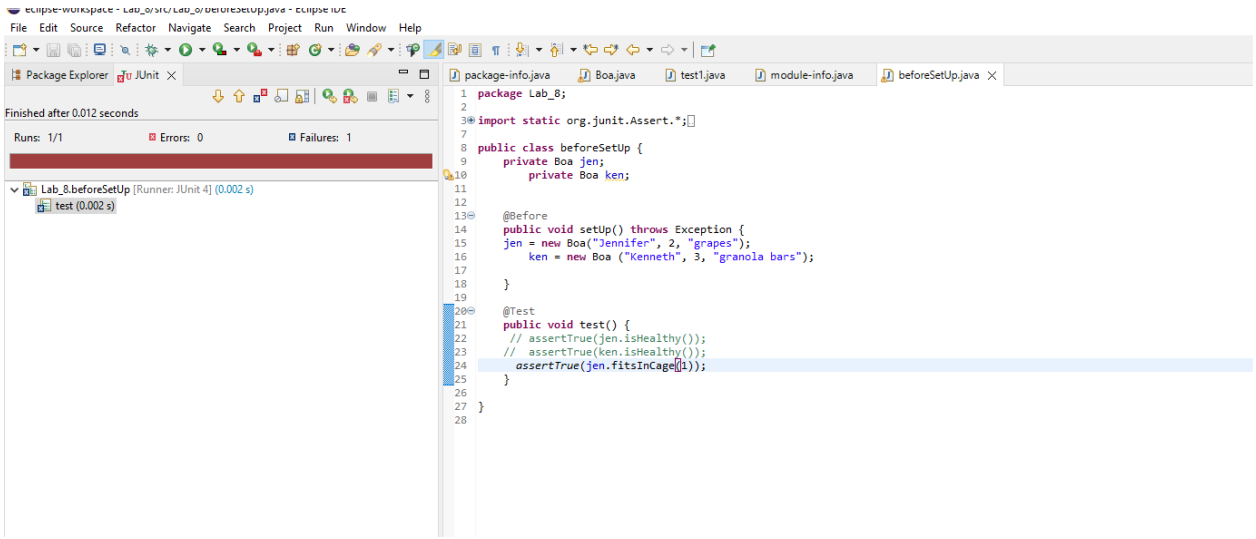
(You will need to add private fields for jen and ben to the BoaTest class, otherwise the
compiler will complain that there are no variables with those names.)





→ We can see that for ken it is healthy but for jen as favorite food is grapes, it is not healthy which is shown with red bar.





→ For cage length of jen, if cage length is 10 greater than 3 than it passes but for smaller cage length like 1, it fails.

→ Similarly, for ken with smaller or equal cage length than 3 it fails and with larger cage length than 3 it passes.

Code:

```
import static org.junit.Assert.*;

import org.junit.Before;
```

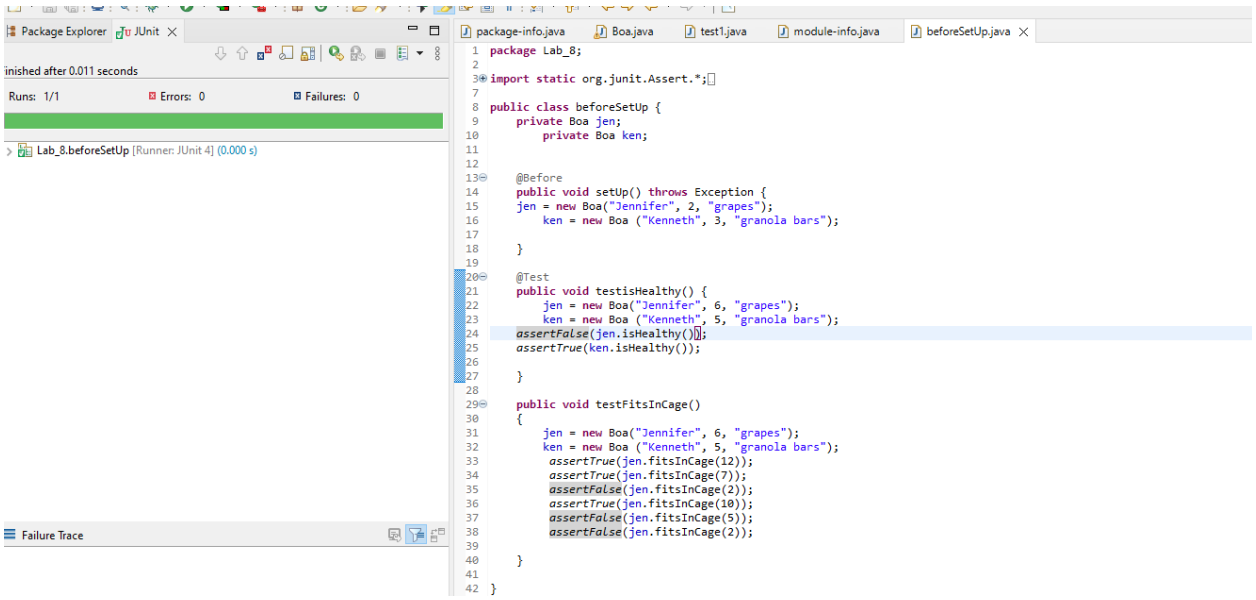
```
import org.junit.Test;

public class beforeSetUp {
    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 test() {
      assertTrue(jen.isHealthy());
    assertTrue(ken.isHealthy());
      assertTrue(jen.fitsInCage(2));
      assertTrue(ken.fitsInCage(10));
    }



}
```

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?
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 if the error is in the test case itself or in the code it’s testing.

Code:

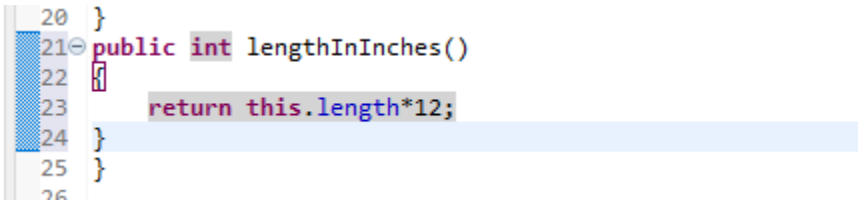
```
@Test
	public void testisHealthy() {
		jen = new Boa("Jennifer", 6, "grapes");
		ken = new Boa ("Kenneth", 5, "granola bars");
	assertFalse(jen.isHealthy());
	assertTrue(ken.isHealthy());

	}

	public void testFitsInCage()
	{
		jen = new Boa("Jennifer", 6, "grapes");
		ken = new Boa ("Kenneth", 5, "granola bars");
		 assertTrue(jen.fitsInCage(12));
		 assertTrue(jen.fitsInCage(7));
		 assertFalse(jen.fitsInCage(2));
		 assertTrue(jen.fitsInCage(10));
		 assertFalse(jen.fitsInCage(5));
		 assertFalse(jen.fitsInCage(2));

	}
```



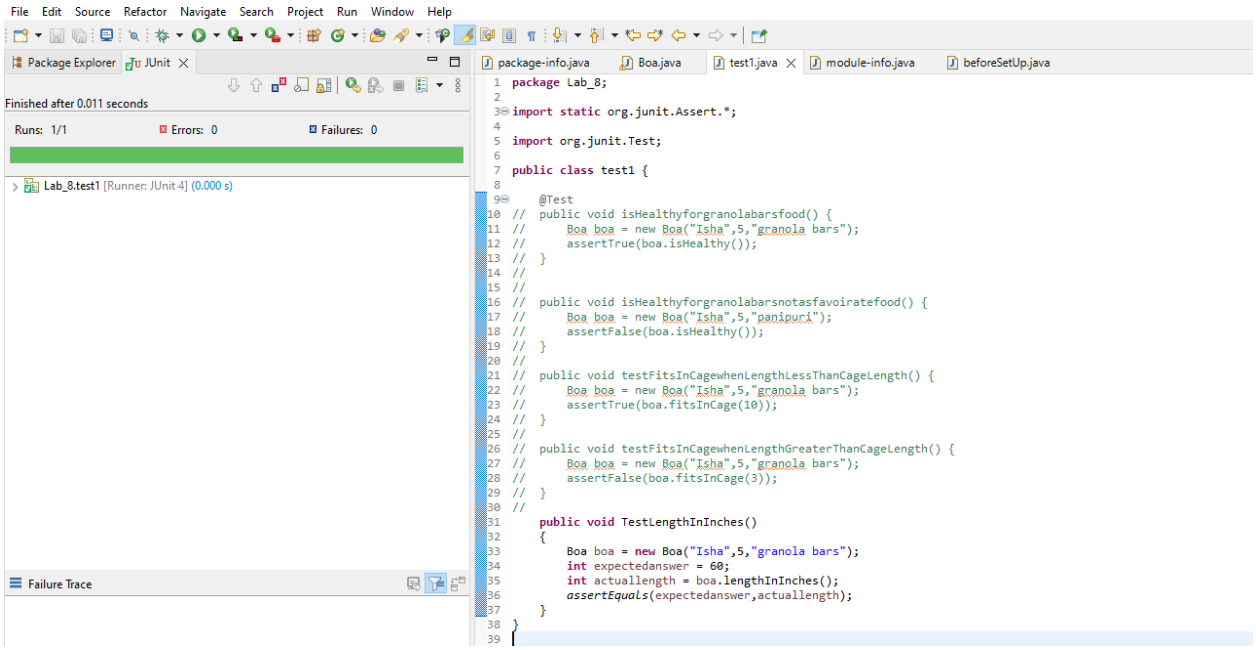
All possible test cases related to isHealthy and FitsInCage.

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
}
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.

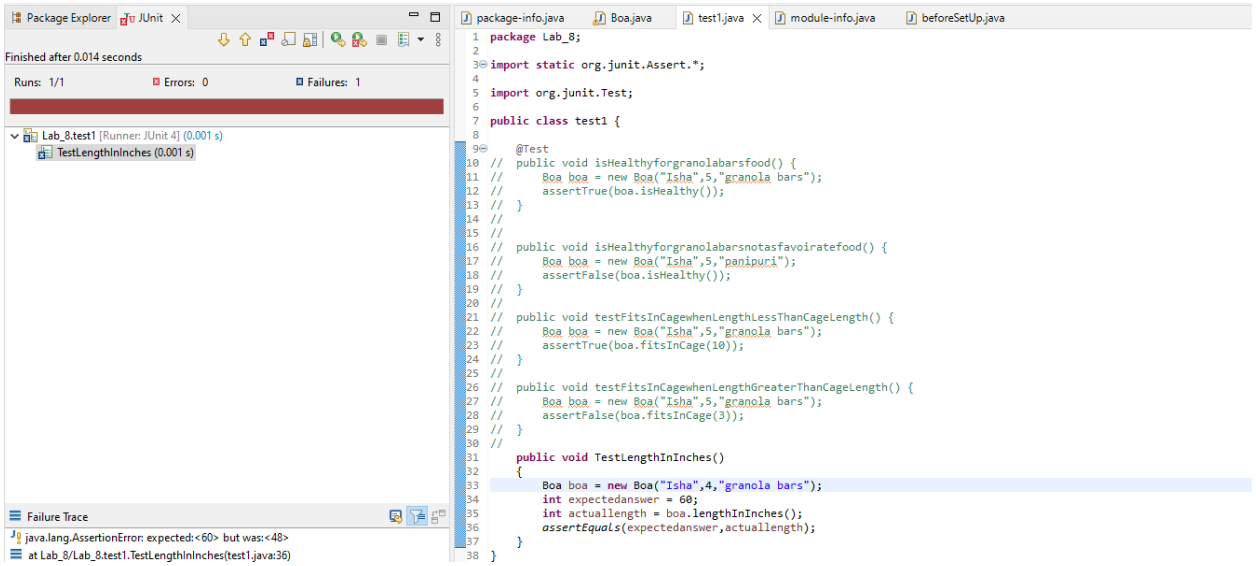
Added function Code:



TestCase for this function:



→ This runs true as both expected and actual answer are same.



→ But for different length, this doesn't returns true.

Code for test:

```
@Test

public void TestLengthInInches()
      {
            Boa boa = new Boa("Isha",4,"granola bars");
            int expectedanswer = 60;
            int actuallength = boa.lengthInInches();
            assertEquals(expectedanswer,actuallength);
      }
```