Dhirubhai Ambani
Institute of Information and Communication Technology

IT314 - Software Engineering

LAB - 8

Name - Prajapati Dhruv
ID - 202001219
Date - 20/04/23

Goal - To learn how to use JUnit to write unit tests for Java programs.

I made a project in Eclipse that first produced a testing package, followed by the production of a class for the boa.

Unit Testing with JUnit

Lab Exercises

1. Create a new Eclipse project, and within the project create a package.
2. Create a class for a Boa.

Boa.java file -

```
package Testing;
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;
	}

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

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).

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:

```
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?

```
@Test
	public void testIsHealthy_1() {
		boolean output = ken.isHealthy();
		assertEquals(output,true);
	}
	@Test
	public void testIsHealthy_2() {
		boolean output = jen.isHealthy();
		assertEquals(output,false);
	}
	@Test
	public void testFitsInCage_1() {

		int cage = 2;
		boolean output = jen.fitsInCage(cage);
		assertEquals(output,false);
```

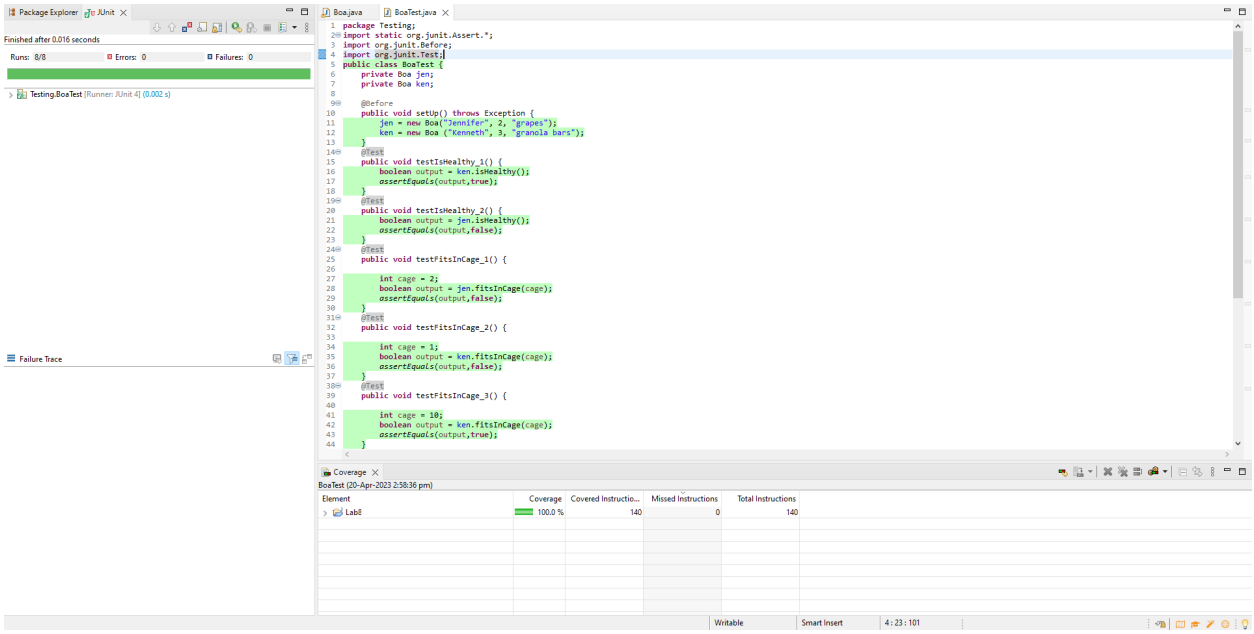
```
}
@Test
public void testFitsInCage_2() {

        int cage = 1;
        boolean output = ken.fitsInCage(cage);
        assertEquals(output,false);
}
@Test
public void testFitsInCage_3() {

        int cage = 10;
        boolean output = ken.fitsInCage(cage);
        assertEquals(output,true);
}
@Test
public void testFitsInCage_4() {

        int cage = 3;
        boolean output = ken.fitsInCage(cage);
        assertEquals(output,false);
}
```

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.



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.

```
@Test
	public void lengthInInches_1() {


		int output = jen.lengthInInches();
		assertEquals(output,24);
	}
	@Test
	public void lengthInInches_2() {
```

```
		int output = ken.lengthInInches();
		assertEquals(output,36);
	}
```

BoaTest.java file -

```
package Testing;
import static org.junit.Assert.*;
import org.junit.Before;
import org.junit.Test;
public class BoaTest {
	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 testIsHealthy_1() {
		boolean output = ken.isHealthy();
		assertEquals(output,true);
	}
	@Test
	public void testIsHealthy_2() {
		boolean output = jen.isHealthy();
		assertEquals(output,false);
	}
	@Test
	public void testFitsInCage_1() {

		int cage = 2;
		boolean output = jen.fitsInCage(cage);
		assertEquals(output,false);
	}
	@Test
	public void testFitsInCage_2() {

		int cage = 1;
		boolean output = ken.fitsInCage(cage);
		assertEquals(output,false);
```

```
	}
	@Test
	public void testFitsInCage_3() {

		int cage = 10;
		boolean output = ken.fitsInCage(cage);
		assertEquals(output,true);
	}

	@Test
	public void testFitsInCage_4() {

		int cage = 3;
		boolean output = ken.fitsInCage(cage);
		assertEquals(output,false);
	}
	@Test
	public void lengthInInches_1() {


		int output = jen.lengthInInches();
		assertEquals(output,24);
	}
	@Test
	public void lengthInInches_2() {


		int output = ken.lengthInInches();
		assertEquals(output,36);
	}
}
```