IT314-Software Engineering

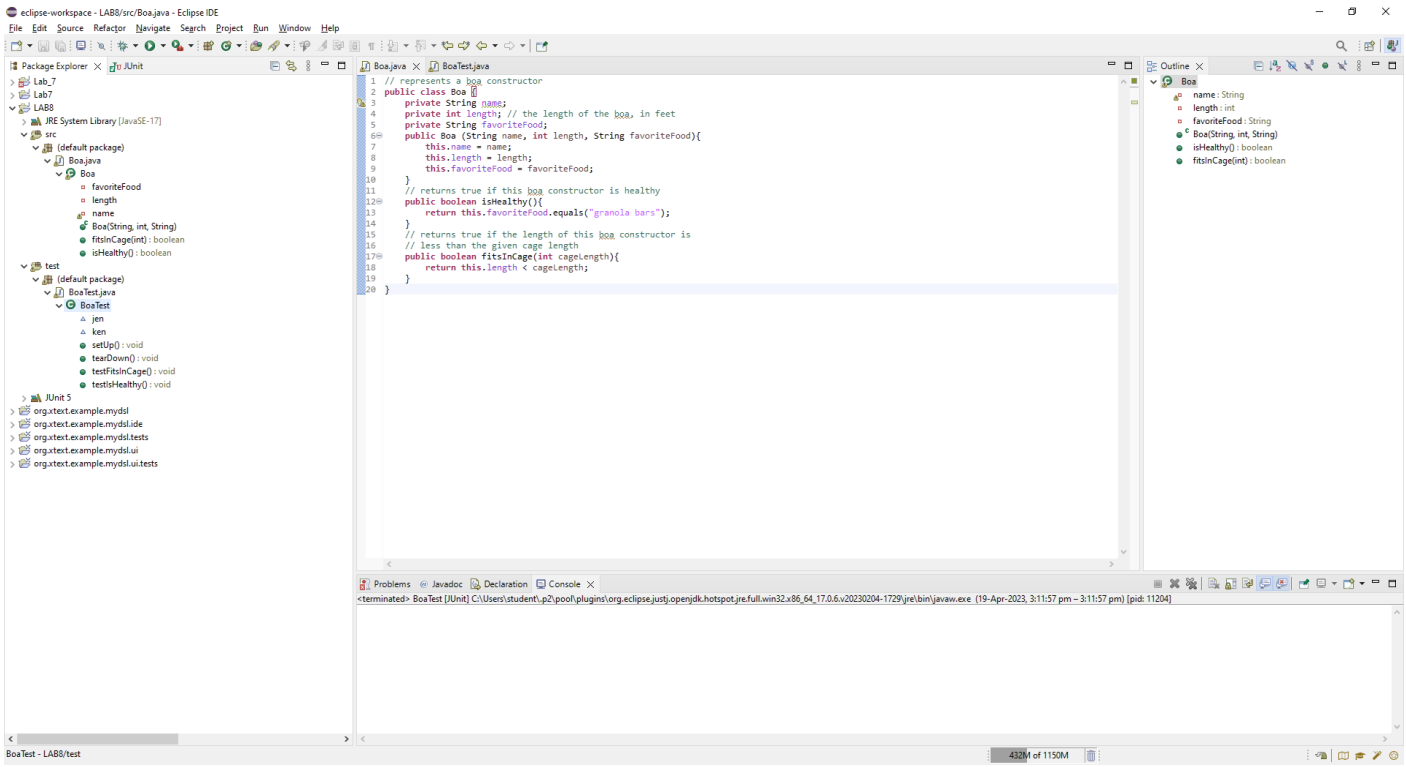
Lab 8 - JUnit Testing

Lab Group - 3

Name: Ramoliya Harsh Madhukantbhai

Student ID: 202001155

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



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).

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

public class BoaTest {

    @Test
    public void testIsHealthy() {
        Boa boa1 = new Boa("Sneaky", 6, "granola bars");
        assertTrue(boa1.isHealthy());

        Boa boa2 = new Boa("Slithery", 8, "pizza");
        assertFalse(boa2.isHealthy());
```

```
    }

    @Test
    public void testFitsInCage() {
        Boa boa1 = new Boa("Slinky", 4, "mice");
        assertTrue(boa1.fitsInCage(5));
        assertFalse(boa1.fitsInCage(3));

        Boa boa2 = new Boa("Curly", 10, "rabbits");
        assertTrue(boa2.fitsInCage(12));
        assertFalse(boa2.fitsInCage(9));
    }
}
```

In the first test case, I am testing the isHealthy method of the Boa class. I created two Boa objects with different favorite foods, and verify that isHealthy returns true for the first object and false for the second object.

In the second test case, I am testing the fitsInCage method of the Boa class. I created two Boa objects with different lengths, and test whether they fit in cages of different lengths.

(Expect the first boa to fit in a cage of length 5 but not 3, and the second boa to fit in a cage of length 12 but not 9.)

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");

}

Ans :

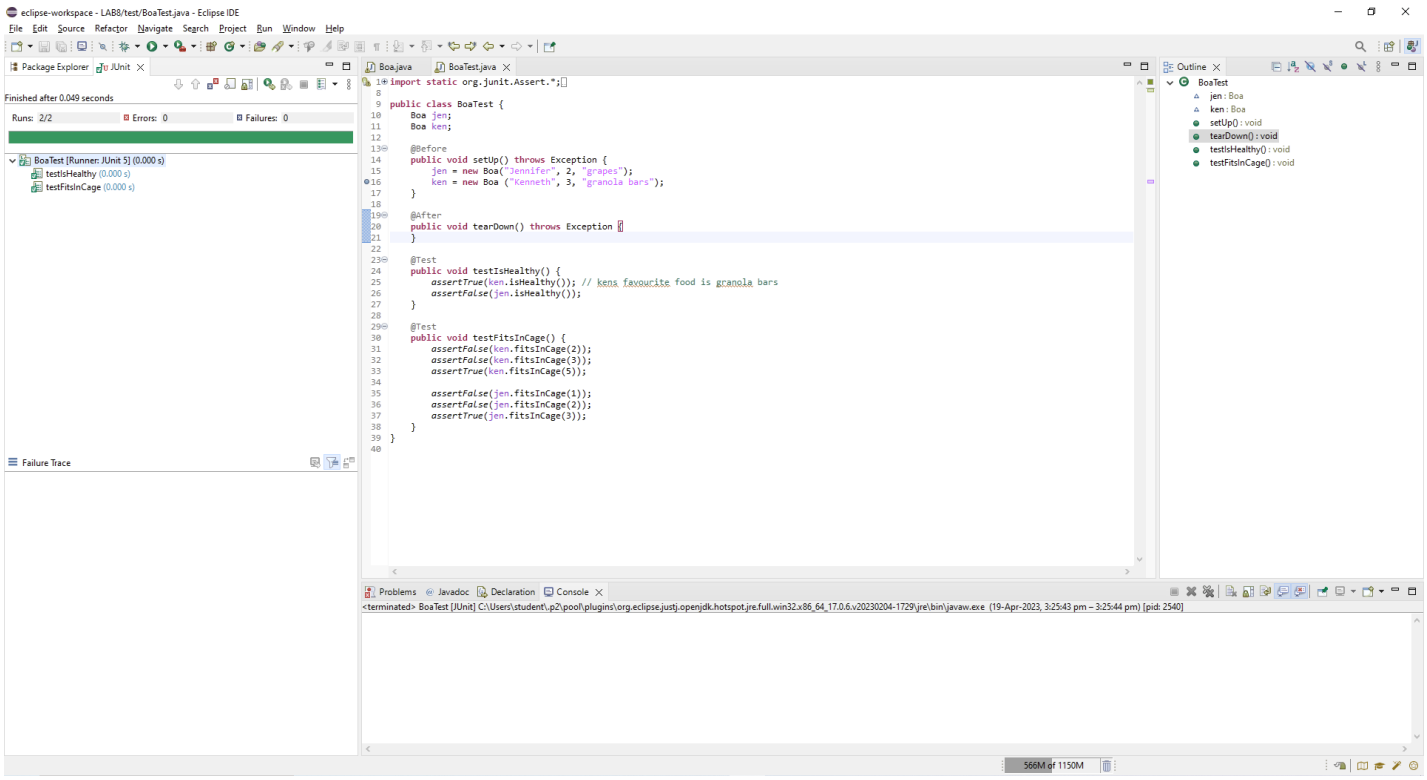
```
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;
public class BoaTest {
    private Boa jen;
    private Boa ken;
    @Before
    public void setUp() throws Exception {
        jen = new Boa("Jennifer", 2, "grapes");
        ken = new Boa("Kenneth", 3, "granola bars");
    }
    @Test
    public void testIsHealthy() {
        assertTrue(jen.isHealthy());
        assertTrue(ken.isHealthy());
    }
    @Test
    public void testFitsInCage() {
        assertTrue(jen.fitsInCage(3));
```

```
        assertFalse(ken.fitsInCage(2));

    }

}
```

Added private fields for jen and ken to the BoaTest class, and initialized them in the setUp method. I also updated the testIsHealthy and testFitsInCage methods to use these Boa objects for testing.

Note that the setUp method will be executed before each test method, so any changes made to the Boa objects during a test will not affect the objects used in other tests.

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?



testIsHealthy() method:

```
import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.*;

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

    @Test
    public void testFitsInCage() {
        assertTrue(jen.fitsInCage(3));
        assertFalse(ken.fitsInCage(2));
    }
}
```

Added assertions to the testIsHealthy method to check that the isHealthy method returns true for both Boa objects. We also added assertions to the testFitsInCage method to check that the fitsInCage method returns the expected results for the Boa objects.

testFitsInCage() method:

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

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

    @Test
    public void testFitsInCage() {
        assertFalse(jen.fitsInCage(1)); // cage length less than boa
length
        assertTrue(jen.fitsInCage(2)); // cage length equal to boa
length
        assertTrue(jen.fitsInCage(3)); // cage length greater than boa
length

        assertFalse(ken.fitsInCage(2)); // cage length less than boa
length
        assertTrue(ken.fitsInCage(3)); // cage length equal to boa
length
        assertTrue(ken.fitsInCage(4)); // cage length greater than boa
length
    }
}
```

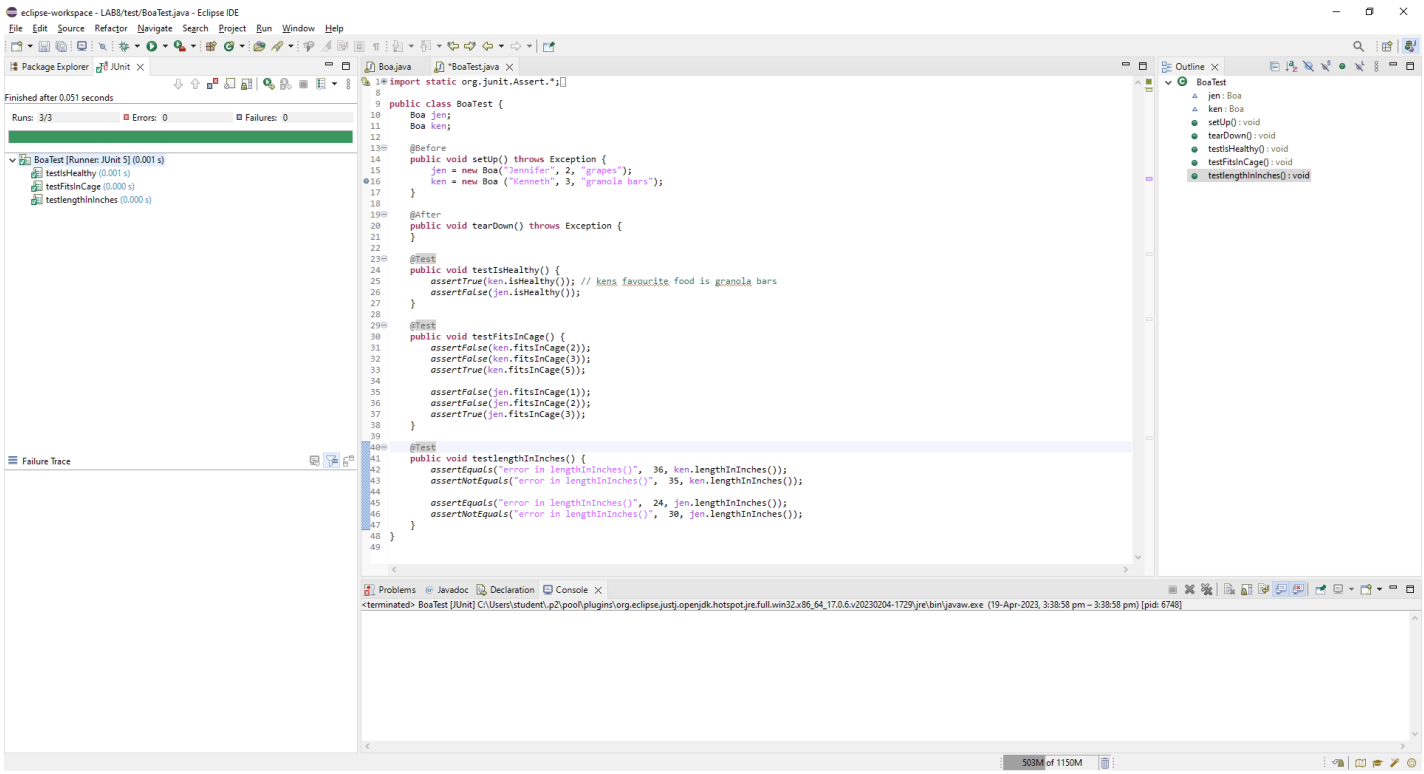
Added assertions to the testFitsInCage method to check that the fitsInCage method returns the expected results for both jen and ken when the cage length is less than, equal to, and greater than the length of the boa.

Note that the testFitsInCage method is now more robust, as it checks the behavior of the fitsInCage method for different input values.

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.

The updated code for the Boa class with the new lengthInInches() method:

```
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;
    }

    // produces the length of the Boa in inches
    public int lengthInInches() {
        return this.length * 12;
    }
}
```

The updated code for the BoaTest class with the new testLengthInInches() method:

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

    @Test
    public void testFitsInCage() {
        assertTrue(jen.fitsInCage(24));
        assertFalse(jen.fitsInCage(16));
        assertTrue(ken.fitsInCage(36));
        assertTrue(ken.fitsInCage(24));
        assertFalse(ken.fitsInCage(18));
    }

    @Test
    public void testLengthInInches() {
        assertEquals(24, jen.lengthInInches());
        assertEquals(36, ken.lengthInInches());
    }
}
```