CIS 285: Software Engineering Tools

University of Michigan – Dearborn

Erika Valle-Baird

Lab 9

Unit Testing

Due Date: Thursday 04/06 – Submission is due by 11:59 pm on Canvas

This laboratory exercise engages you in unit testing-based development of a simple Java method. You don’t need to be a Java expert to complete this exercise.

**Laboratory tasks**:

1. Unit testing with JUnit for the method “Largest” in Largest.java class
   1. Create a new Java Project in Eclipse
   2. Add a new Java class “Largest.java” class (Largest.java class source code is in next page) in the project
   3. Add a JUnit Test Case “testLargest.java” in the project. Initial testLargest.java source code is in the next page
   4. Develop and execute the following unit tests for Largest.java, Add the following methods in testLargest.java and write assertion to verify
      1. testPositives – testing a list of all positive integers
      2. testNegatives – testing a list of all negative integers
      3. testMixes – testing a list containing both positive and negative and zero values
   5. Some tests in (d) should fail. This indicates Largest.java class has defect. Fix the defect and re-run tests.

**Report submission**:

Create a MS Word or PDF document containing the following:

1. Unit testing with JUnit
   1. Outputs of unit tests. Include both failed and passed unit tests. In the case of failed unit tests, indicate what was wrong and what was done to fix the problem.
   2. Final outputs of all unit tests showing successful pass for all
   3. Source code of final testLargest.java class
   4. Source code of final Largest.java class after making correction

**Error**:

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Here is the Largest Class with the correction on the for loop:**

**package** Largest;

**public** **class** Largest {

**public** Largest() {

}

**public** **int** largest(**int**[] list){

**int** index;

**int** max =0;

**for** (index = 0; index < list.length; index++){

**if** (list[index] > max) {

max = list[index];

}

}

**return** max;

}

}

**ORIGINAL CODE THAT CAUSED ERROR:**

package Largest;

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

import org.junit.Test;

import junit.framework.\*;

import static org.junit.Assert.\*;

public class LargestTest {

private Largest temp1;

@Test

public void test() {

testPositive();

testNegative();

testMixes();

}

public void testPositive(){

int[] arr = new int[5];

arr[0] = 8;

arr[1] = 9;

arr[2] = 7;

arr[3] = 10;

arr[4] = 2;

Largest temp1 = new Largest();

int maxValue = temp1.largest(arr);

/\*\* add tests to check for this unit test \*\*/

assertEquals("TEST POSITIVE FAILED!", -1, maxValue);

}

public void testNegative(){

int[] arr = new int[5];

arr[0] = 8;

arr[1] = -9;

arr[2] = -7;

arr[3] = -10;

arr[4] = 2;

Largest temp1 = new Largest();

int maxValue = temp1.largest(arr);

assertEquals("TEST NEGATIVE FAILED!", -1, maxValue);

}

public void testMixes(){

int[] arr = new int[5];

arr[0] = 8;

arr[1] = 9;

arr[2] = -7;

arr[3] = -10;

arr[4] = 2;

Largest temp1 = new Largest();

int maxValue = temp1.largest(arr);

assertEquals("TEST MIXES FAILED!", -1, maxValue);

}

}

**\*\*\*I started testing using the assertEquals Junit test cause, which was not even remotely the correct test case to use. Each assert case needed to be uniquely identified to the function, which I realized as soon as I failed and edited the issue. Rather than attempting a blanket easy fix, I went through each, edited them to assertTrue and assertFalse causes depending on each of the function’s needs. Here is the corrected code that resulted in a passed Unit Test.**

**package** Largest;

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

**import** org.junit.Test;

**import** junit.framework.\*;

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

**public** **class** LargestTest {

**private** Largest temp1;

@Test

**public** **void** test() {

testPositive();

testNegative();

testMixes();

}

**public** **void** testPositive(){

**int**[] arr = **new** **int**[5];

arr[0] = 8;

arr[1] = 9;

arr[2] = 7;

arr[3] = 10;

arr[4] = 2;

Largest temp1 = **new** Largest();

**int** maxValue = temp1.largest(arr);

*assertTrue*("10 is the max value", 10 == maxValue);

*assertFalse*("2 is false", 2 == maxValue);

}

**public** **void** testNegative(){

**int**[] arr = **new** **int**[5];

arr[0] = 8;

arr[1] = -9;

arr[2] = -7;

arr[3] = -10;

arr[4] = 2;

Largest temp1 = **new** Largest();

**int** maxValue = temp1.largest(arr);

*assertTrue*("8 is the max value", 8 == maxValue);

*assertFalse*("-10 is false", -10 == maxValue);

}

**public** **void** testMixes(){

**int**[] arr = **new** **int**[5];

arr[0] = 8;

arr[1] = 9;

arr[2] = -7;

arr[3] = -10;

arr[4] = 2;

Largest temp1 = **new** Largest();

**int** maxValue = temp1.largest(arr);

*assertTrue*("9 is the max value", 9 == maxValue);

*assertFalse*("-10 is false", -10 == maxValue);

}

}