Homework #1

CSE 565: Software Verification/ Validation/ Test

(2015 FALL)

Software Unit Testing Frameworks

Submitted By:

Madhu Meghana Talasila (1207740881)

PART 1

What is a unit testing framework? Unit testing is a process in which individual units of application are tested independently. Automated Testing is using a special software tools to control the execution of pre-scripted test. Test automation framework is an integrated system which sets rules for automation of a test case. Unit Testing Framework is a type of test automation framework (Data-driven) in which unit tests are executed to test if they are working correctly as expected. **How would a developer utilize a framework?** Developer utilizes the predefined libraries in the test framework to write test cases for happy paths as well as unhappy paths. "Developer uses the framework to formalize requirements, clarify architecture, write code, debug code, integrate code, release, optimize, and test" [1]. In test driven development, developer first writes the test cases and then actually start developing the code. In general, developer first writes the code and use any of the framework to write test cases on that code and tests (white box testing). **What benefits does a framework provide?** "The framework provides the basis of test automation and simplifies the automation effort" [2]. The main advantage of test automation framework is low maintenance cost. Using unit testing frameworks, problems can be detected early in the software development lifecycle. Software development process becomes more flexible and frameworks takes care of time and space and reduces need for manual testing. Also developers can change existing code with little effort" [3]. **Identify 2 frameworks for further consideration. Compare the frameworks in terms of common capabilities as well as any differences.** The frameworks that I am considering are CPPUnit and JUnit. Both of them are unit testing frameworks and instances of XUnit. They are cross platform and uses graphical and textual test drivers. Both frameworks are structured similarly for example they make use of a test runner program to run test cases. CPPUnit is used in C++ whereas JUnit is used in Java. JUnit uses annotations like @Test/ @before/ @after to define test cases whereas CPPUnit does not. Generic functions can be used in CPPUnit. JUnit does not

support generic functions. Helpers Macros automatically implement the test suite method in CPPUnit.

No such macros exist in JUnit.

PART 2. Binary Search Algorithm

Copy of Code

BinarySearch.java

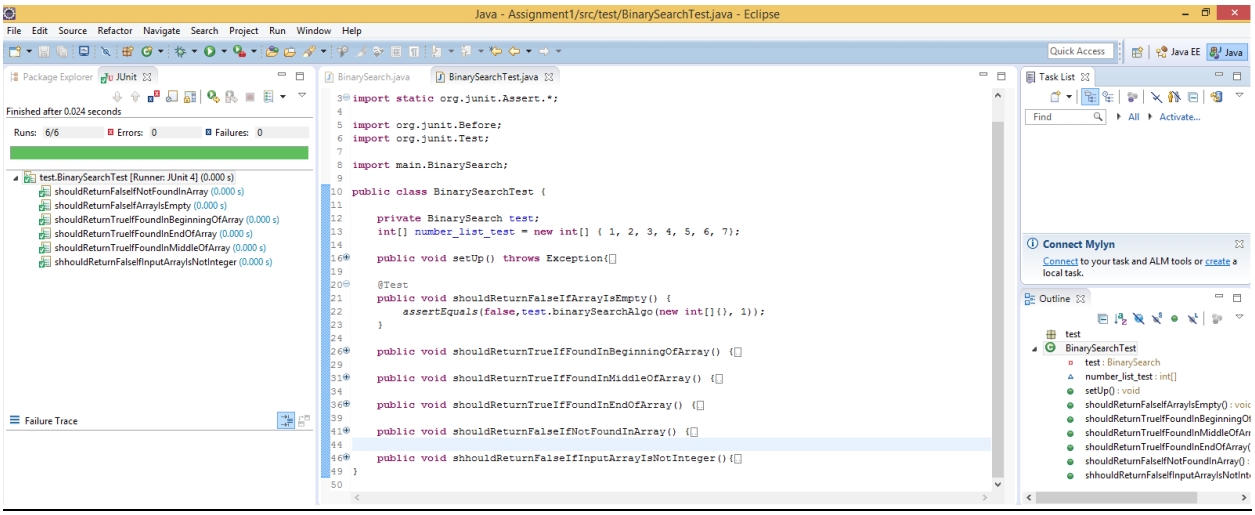
```
public class BinarySearch {
	/* Input should be in ascending order*/
	public boolean binarySearchAlgo(int[] number_list, int key){
		return search(number_list,key,0,number_list.length-1);
	}

	public boolean search(int[] number_list, int key, int lowerIndex, int upperIndex)
	{

		if(number_list.length == 0){
			return false;
		}

		if (lowerIndex > upperIndex) {
	  return false;
	}

		int middleIndex = (lowerIndex + upperIndex ) / 2;

		if (number_list[middleIndex] > key) {
	  return search(number_list, key, lowerIndex, middleIndex - 1);
	} else if (number_list[middleIndex] < key) {
	  return search(number_list, key, middleIndex + 1, upperIndex);
	} else {
	  return true;
	}
	  }
```

BinarySearchTest.java

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

import org.junit.Before;
import org.junit.Test;

import main.BinarySearch;

public class BinarySearchTest {

	private BinarySearch test;
```

```
        int[] number_list_test = new int[] { 1, 2, 3, 4, 5, 6, 7};

        @Before
        public void setUp() throws Exception{
                test = new BinarySearch();
        }

        @Test
        public void shouldReturnFalseIfArrayIsEmpty() {
                assertEquals(false,test.binarySearchAlgo(new int[]{}, 1));
        }

        @Test
        public void shouldReturnTrueIfFoundInBeginningOfArray() {
                 assertEquals(true, test.binarySearchAlgo(number_list_test, 1));
         }

        @Test
         public void shouldReturnTrueIfFoundInMiddleOfArray() {
                 assertEquals(true, test.binarySearchAlgo(number_list_test, 7));
         }

         @Test
         public void shouldReturnTrueIfFoundInEndOfArray() {
                 assertEquals(true, test.binarySearchAlgo(number_list_test, 4));
          }

         @Test
          public void shouldReturnFalseIfNotFoundInArray() {
                  assertEquals(false, test.binarySearchAlgo(number_list_test, 10));
          }

         @Test
          public void shhouldReturnFalseIfInputArrayIsNotInteger(){
                  assertEquals(false,test.binarySearchAlgo(new int[]{'1','2','3','4','5'}, 1));
          }
}
```

Output Report – Screenshot from tool



Test Cases

1. public void shouldReturnFalseIfArrayIsEmpty()
    a. This is unhappy path. Test case is working if code returns false when input Array is empty
2. public void shouldReturnTrueIfFoundInBeginningOfArray()
    a. This is happy path. Test case is working if code returns true when key is found in the input array (Beginning of the input array).
3. public void shouldReturnTrueIfFoundInMiddleOfArray()
    a. This is happy path. Test case is working if code returns true when key is found in the input array (Middle of the input array).
4. public void shouldReturnTrueIfFoundInEndOfArray()
    a. This is happy path. Test case is working if code returns true when key is found in the input array (End of the input array).
5. public void shouldReturnFalseIfNotFoundInArray()
    a. This is unhappy path. Test case is working if code returns false when key is not found in the input Array
6. public void shhouldReturnFalseIfInputArrayIsNotInteger()
    a. This is unhappy path. Test case is working if code returns false when input array is not integer but key is integer.

References

1. http://www.extremeprogramming.org/rules/unittestframework.html
2. https://en.wikipedia.org/wiki/Test_automation
3. http://www.codeproject.com/Articles/5404/The-benefits-of-automated-unit-testing
4. http://cppunit.sourceforge.net/doc/cvs/hierarchy.html
5. http://junit.org/javadoc/latest/index.html
6. http://www.digizol.com/2013/08/java-binary-search-recursive-testcases.html