

Assignment 7

Rasika Mohod

rmohod@gmu.edu

G01044774

Report:

I generally use Eclipse for java development and thus have used **EclEmma**, a free Java code coverage tool in Eclipse for this assignment. I did my code coverage on **java.util.Stack** class.

Given below is the code for **Stack.java**:

```
package com.gmu.rmohod;

import java.util.EmptyStackException;
import java.util.Vector;

public class Stack<E> extends Vector<E> {

    public Stack() {

    }

    public E push(E item) {
        addElement(item);

        return item;
    }

    public synchronized E pop() {
        E      obj;
        int     len = size();

        obj = peek();
        removeElementAt(len - 1);

        return obj;
    }

    public synchronized E peek() {
        int     len = size();

        if (len == 0)
            throw new EmptyStackException();
        return elementAt(len - 1);
    }

    public boolean empty() {
        return size() == 0;
    }
}
```

```

        public synchronized int search(Object o) {
            int i = lastIndexOf(o);

            if (i >= 0) {
                return size() - i;
            }

            return -1;
        }
    }
}

```

Given below is the code for JUnit test class **StackTest.java**:

```

package com.gmu.rmohod;

import static org.junit.Assert.*;

public class StackTest {

    private Stack<String> stackObject;

    @Before
    public void setUp() {
        stackObject = new Stack<String>();

        stackObject.push("Rasika");
        stackObject.push("Mohod");
    }

    @Test
    public void test() {

        assertEquals(2, stackObject.search("Rasika"));

        assertEquals("Mohod", stackObject.peek());

        stackObject.pop();
        assertEquals(false, stackObject.isEmpty());

        assertEquals("Rasika", stackObject.peek());

        stackObject.pop();
        assertEquals(true, stackObject.isEmpty());
















        assertEquals(-1, stackObject.search("Rasika"));

        assertEquals(true, stackObject.empty());
    }
}

```

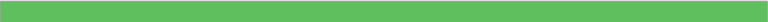
Here Red color shows the code that is being missed and Green shows the code being covered.

Given below is the **output of coverage tool** on my code:

StackTest (Mar 20, 2017 2:18:09 AM)					
Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions	
▼ SWE637Assignment7	 95.3 %	123	6	129	
▼ src	 95.3 %	123	6	129	
▼ com.gmu.rmohod	 95.3 %	123	6	129	
▼ Stack.java	 89.3 %	50	6	56	
▼ Stack<E>	 89.3 %	50	6	56	
peek()	 73.3 %	11	4	15	
empty()	 71.4 %	5	2	7	
Stack()	 100.0 %	3	0	3	
pop()	 100.0 %	13	0	13	
push(E)	 100.0 %	5	0	5	
search(Object)	 100.0 %	13	0	13	
▼ StackTest.java	 100.0 %	73	0	73	
▼ StackTest	 100.0 %	73	0	73	
setUp()	 100.0 %	16	0	16	
test()	 100.0 %	54	0	54	

I did not include the test case for stack `EmptyStackException()` ;

JUnit Output:

Package Explorer JUnit	
Finished after 0.016 seconds	
Runs: 1/1	Errors: 0 Failures: 0
	
▼ com.gmu.rmohod.StackTest [Runner: JUnit 4] (0.000 s)	
test (0.000 s)	

Technical difficulties:

I did not face any severe technical twist or difficulty apart from deciding on the tool to be used and how to use.

Once I could run the tool on my code it was very satisfying and encouraging to see which tests I have missed or which part of code was not at all tested through my (seemed to be complete) test sets. This assignment really helped me to leverage code coverage in everyday code testing.