

Unit Testing Report

Software Modeling & Analysis

소프트웨어 모델링 및 분석

보고서 Version. 1

Team. T1

201111388 조연호

201211374 이창오

201211379 장종훈

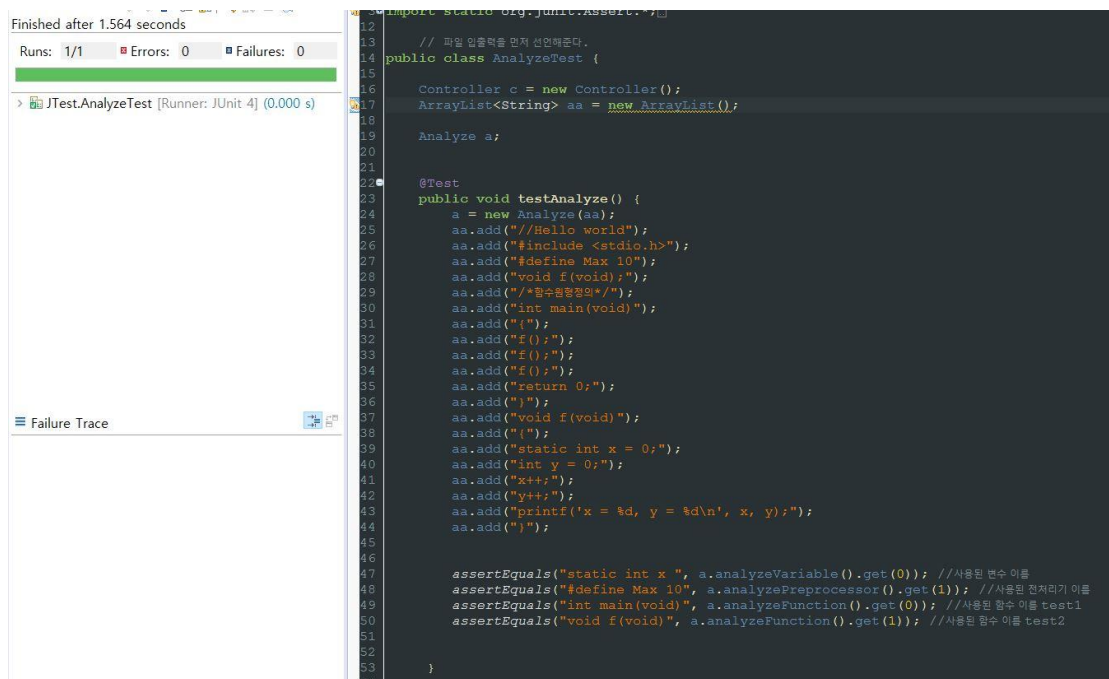
201314196 양동혁

System Testing Report

1. Test Environment

- A. 테스트 진행 : Team. T1
- B. 날짜 : 2015. 5. 16
- C. OS : Windows 10 (64 bit)
- D. Test 제외 항목
 - i. GUI 관련 메소드
 - ii. Listeners (Interface 내 메소드)
 - iii. Setters, Getters

2. Test Result



The screenshot displays a Java IDE with two panels. The left panel shows the test execution results, and the right panel shows the source code of the test class.

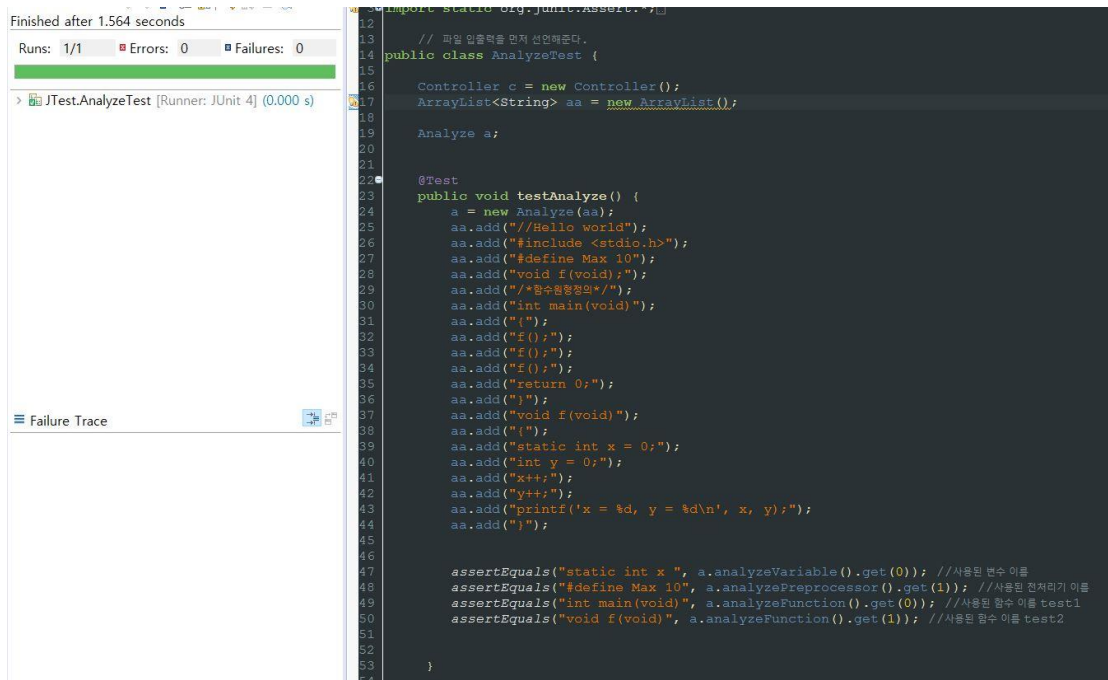
Test Execution Results (Left Panel):

- Finished after 1.564 seconds
- Runs: 1/1
- Errors: 0
- Failures: 0
- Tested: JTest.AnalyzeTest [Runner: JUnit 4] (0.000 s)
- Failure Trace: (Empty)

Source Code (Right Panel):

```
12 import static org.junit.Assert.*;
13
14 // 파일 인출자를 먼저 선언해준다.
15 public class AnalyzeTest {
16
17     Controller c = new Controller();
18     ArrayList<String> aa = new ArrayList();
19
20     Analyze a;
21
22     @Test
23     public void testAnalyze() {
24         a = new Analyze(aa);
25         aa.add("//Hello world");
26         aa.add("#include <stdio.h>");
27         aa.add("#define Max 10");
28         aa.add("void f(void);");
29         aa.add("/*함수원형정의*/");
30         aa.add("int main(void)");
31         aa.add("f");
32         aa.add("f()");
33         aa.add("f()");
34         aa.add("f()");
35         aa.add("return 0;");
36         aa.add("");
37         aa.add("void f(void)");
38         aa.add("");
39         aa.add("static int x = 0;");
40         aa.add("int y = 0;");
41         aa.add("x++");
42         aa.add("y++");
43         aa.add("printf('x = %d, y = %d\\n', x, y);");
44         aa.add("");
45
46         assertEquals("static int x ", a.analyzeVariable().get(0)); //사용된 변수 이름
47         assertEquals("#define Max 10", a.analyzePreprocessor().get(1)); //사용된 전처리기 이름
48         assertEquals("int main(void)", a.analyzeFunction().get(0)); //사용된 함수 이름 test1
49         assertEquals("void f(void)", a.analyzeFunction().get(1)); //사용된 함수 이름 test2
50
51     }
52
53 }
```

System Testing Report



1.1 Analyze – void analyzeLine

1	Name = "analyzeLine"	Input	Output	Result
1.1	assertEquals(19, result); 19줄짜리 코드를 입력 시 올바른 값을 출력	19줄짜리 코드	19	PASS

1.2 Analyze – void analyzeFunction

2	Name = "analyzeFunction"	Input	Output	Result
1.2.1	assertEquals(2, result);	함수가 2개 인 소스코드	2	PASS
	함수 2개가 포함된 소스코드 입력 시 올바른 값을 출력			
1.2.2	assertEquals("int main(void)", result.get(0));	함수 이름	함수 이름	PASS
	analyzeFunction이 함수가 포함된 소스코드 입력 시 배열 순서에 따라 함수를 순서적으로 저장하고 있는지를 출력			
1.2.3	assertEquals("void f(void)", result.get(1));	함수 이름	함수 이름	PASS
	1.2.2 와 같음. 단지 순서가 2번째로 오는 함수가 출력			

1.3 Analyze – void analyzeVariable

3	Name = "analyzeVariable"	Input	Output	Result
1.3.1	assertEquals(2, result);	static int x, int y	2	PASS
	변수 개수가 2개인 소스코드를 입력 할 시 그 개수를 출력			
1.3.2	assertEquals("static int x", result.get(0));	static int x	Static int x	PASS
	소스코드 안에 있는 변수들을 배열에 저장하여, 순서대로 출력			

1.4 Analyze – void analyzePreprocessor

4	Name = "analyzePreprocessor"	Input	Output	Result
1.4.1	assertEquals(1, result);	#include <stdio.h>	1	PASS
	전처리가 1개인 소스 코드 입력 시 그 개수를 출력			
1.4.2	assertEquals("#include<stdio.h>", result);	#include <stdio.h>	#include <stdio.h>	PASS
	소스코드에 포함된 전처리를 순서대로 배열에 저장하여, 배열 값에 따라 출력			

System Testing Report

1.5 Analyze – void analyzeAnnotation

5	Name = "analyzeAnnotation"	Input	Output	Result
1.5.1	assertEquals(2, result);	주석 2개 소스 코드	2	PASS
	주석 개수가 2개인 소스코드 입력 시 그 개수를 출력			

The screenshot shows an IDE interface. On the left, a test runner window displays the following information:

- Finished after 0.02 seconds
- Runs: 5/5
- Errors: 0
- Failures: 0
- Test suite: JTest.TestCalculate [Runner: JUnit 4] (0.000 s)
 - testCalAnnotation (0.000 s)
 - testCalPreprocessor (0.000 s)
 - testCalFunction (0.000 s)
 - testCalLine (0.000 s)
 - testCalVariable (0.000 s)

On the right, the source code for the test is shown:

```

1 package JTest;
2
3 import static org.junit.Assert.*;
4
5 import java.util.ArrayList;
6
7 import org.junit.Test;
8
9 import cloneChecker.Analyze;
10 import cloneChecker.Calculate;
11 import cloneChecker.Files;
12
13 public class TestCalculate {
14     Analyze a;
15     Analyze b;
16     Calculate c;
17     Files f1 = new Files("jang");
18     Files f2 = new Files("dong");
19
20     ArrayList<String> a1 = new ArrayList();
21     ArrayList<String> b1 = new ArrayList();
22     ArrayList<Files> files = new ArrayList();
23
24     @Test
25     public void testCalLine() {
26
27         /*f1과 f2의 갯수의 예상 일치율은 40*/
28         f1.setNumOfLine(100); //f1은 100줄
29         f2.setNumOfLine(90); //f2는 90줄
30
31         files.add(f1);
32         files.add(f2);
33
34         c = new Calculate(files);
35
36         assertEquals(40,c.calLine());
37     }
38 }

```

2.1 Calculate – void CalLine

1	Name = "CalLine"	Input	Output	Result
2.1	assertEquals(40, result);	두개의 소스 코드	40	PASS
	코드의 라인 개수가 각각 100개, 90개인 두 개의 소스코드 입력 시 그 일치율을 출력			

2.2 Calculate – void CalFunction

2	Name = "CalFunction"	Input	Output	Result
2.2	assertEquals(60, result);	두 개의 소스코드	60	PASS
	CalFuntion에서 확인하고자 하는 함수의 개수의 차이에 따른 일치율과 함수 이름에 따른 일치율의 SyncRate를 출력한다.			

```

@Test
public void testCalFunction() {
    a1.add("void f(void)");
    b1.add("void f(void)");
    f1.setListFunction(a1);
    f2.setListFunction(b1);
    /*f1의 함수이름과 f2의 함수이름이 같다고 가정*/
    /*함수갯수 일치율 에서 +20이 가산되어진다.*/

    /*f1과 f2의 함수갯수의 예상 일치율은 40*/
    f1.setNumOfFunction(100); //f1은 100개의 함수
    f2.setNumOfFunction(90); //f2는 90개의 함수

    files.add(f1);
    files.add(f2);

    c = new Calculate(files);

    assertEquals(60, c.calFunction());
}

```

2.3 Calculate – void CalVariable

3	Name = "CalVariable"	Input	Output	Result
2.3	assertEquals(60, result);	두 개의 소스코드	60	PASS
	CalVariable에서 확인하고자 하는 변수의 개수의 차이에 따른 일치율과 변수 이름에 따른 일치율의 SyncRate를 출력한다.			

System Testing Report

```

@Test
public void testCalVariable() {
    a1.add("int a");
    b1.add("int a");
    f1.setListVariable(a1);
    f2.setListVariable(b1);
    /*f1의 변수이름과 f2의 변수이름이 같다고 가정*/
    /*변수갯수 일치율 에서 +20이 가산되어진다.*/

    /*f1과 f2의 변수 갯수의 예상 일치율은 40*/
    f1.setNumOfVariable(100); //f1은 100개의 변수
    f2.setNumOfVariable(90); //f2는 90개의 변수

    files.add(f1);
    files.add(f2);

    c = new Calculate(files);

    assertEquals(60,c.calVariable());
}

```

2.4 Calculate – void CalPreprocessor

4	Name = "CalPreprocessor"	Input	Output	Result
2.4	assertEquals(60, result); CalPreprocessor에서 확인하고자 하는 전처리기의 개수의 차이에 따른 일치 율과 전처리기 이름에 따른 일치율의 SyncRate를 출력한다.	두 개의 소 스코드	60	PASS

```

@Test
public void testCalPreprocessor() {
    a1.add("#include");
    b1.add("#include");
    f1.setListPreprocessor(a1);
    f2.setListPreprocessor(b1);
    /*f1의 전처리기이름과 f2의 전처리기이름이 같다고 가정*/
    /*전처리기갯수 일치율 에서 +20이 가산되어진다.*/

    /*f1과 f2의 전처리기 갯수의 예상 일치율은 40*/
    f1.setNumOfPreprocessor(100); //f1은 100개의 전처리기
    f2.setNumOfPreprocessor(90); //f2는 90개의 전처리기

    files.add(f1);
    files.add(f2);

    c = new Calculate(files);

    assertEquals(60,c.calPreprocessor());
}

```

System Testing Report

2.5 Calculate – void CalAnnotation

5	Name = "CalAnnotation"	Input	Output	Result
2.5	assertEquals(60, result);	두 개의 소스코드	40	PASS
	코드의 주식 개수가 각각 100개, 90개인 두 개의 소스코드 입력 시 그 일치율을 출력			