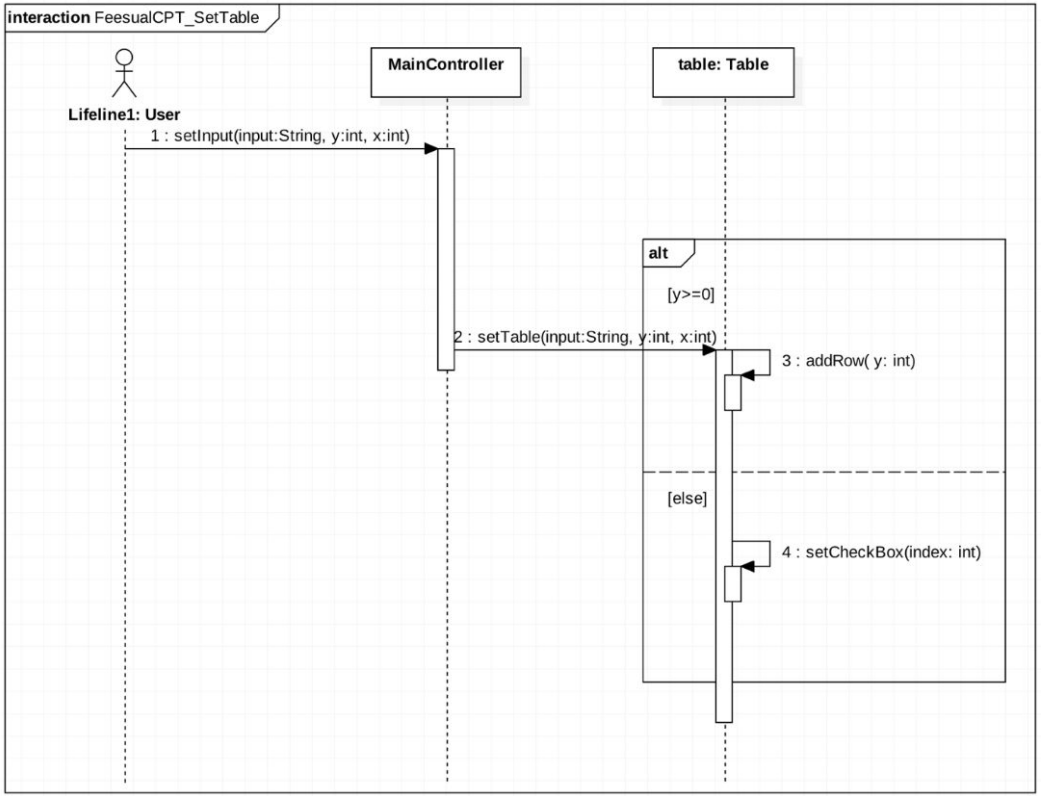
Feesual CPT Tool

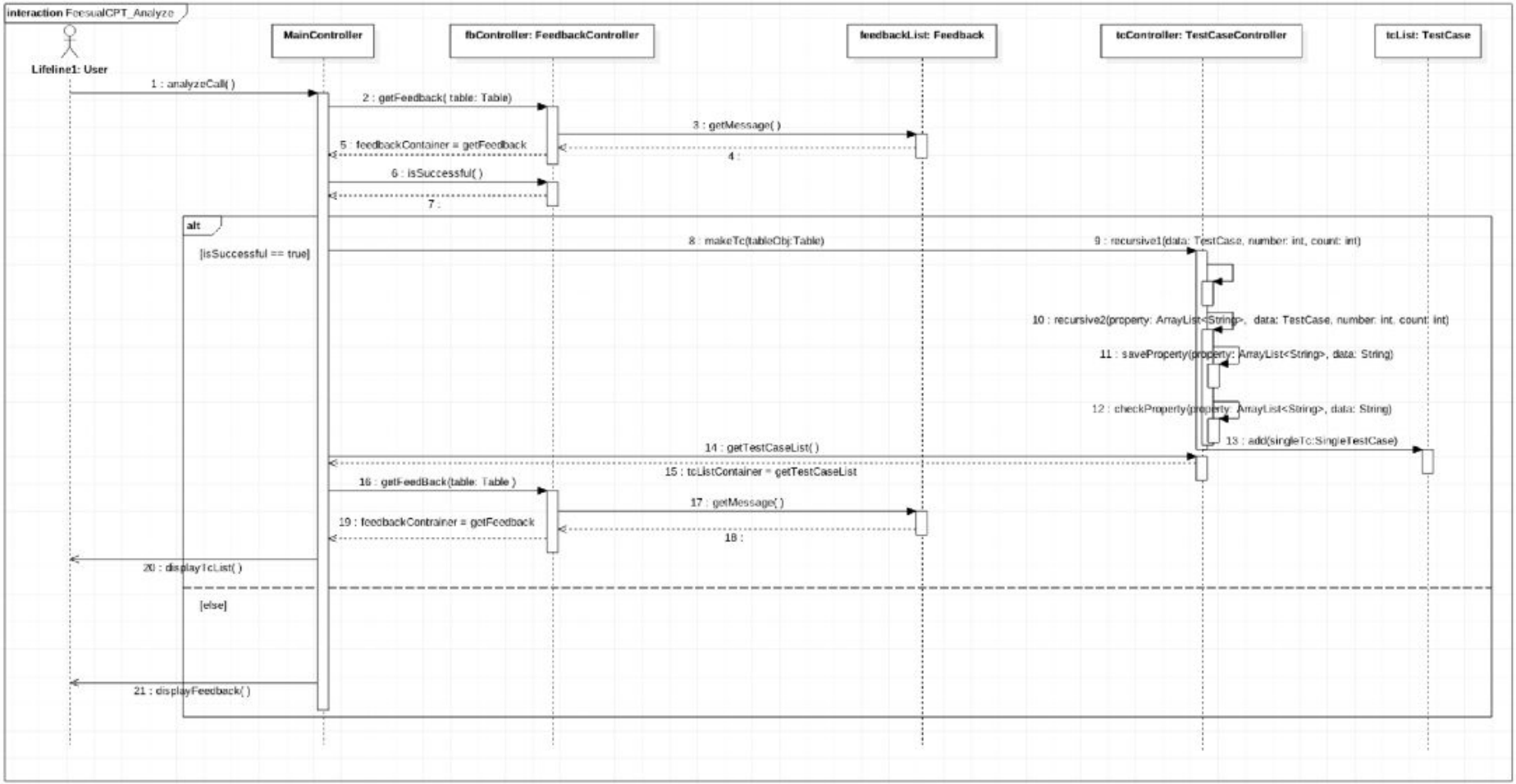
OOPT Stage 2050, 2060

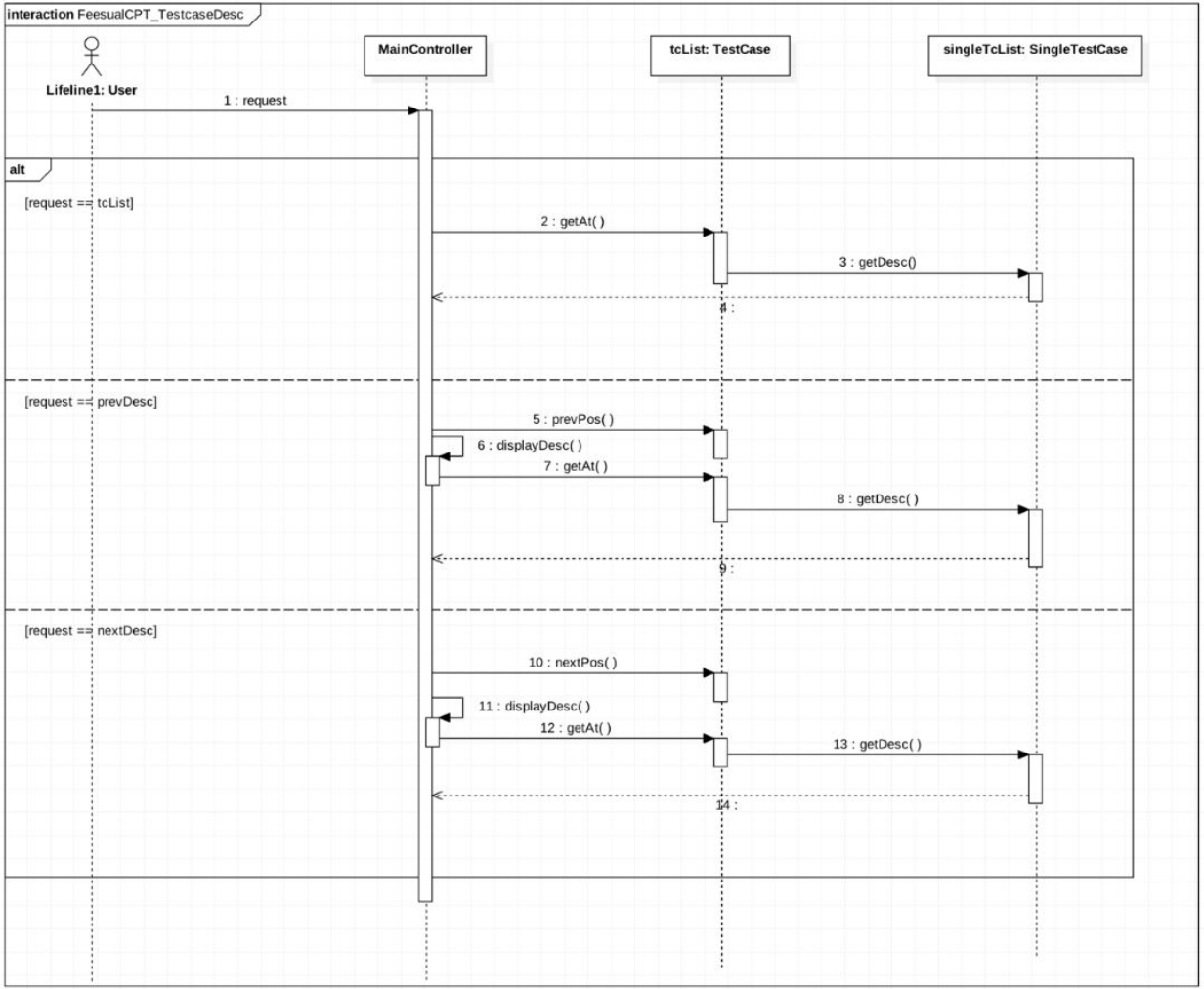
TEAM 8
박성근 (201211347)
임제현 (201211376)
김태홍 (201411270)

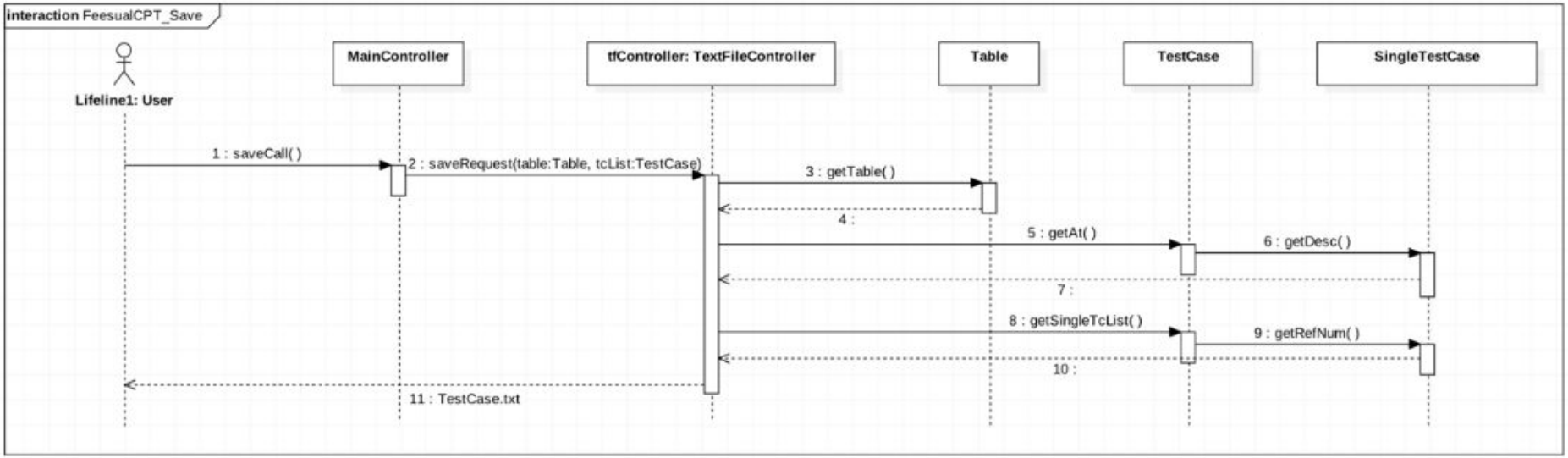
Table of Contents

Revised Plan




interaction FeesualCPT_Analyze
Lifeline1: User
MainController
fbController: FeedbackController
feedbackList: Feedback
tcController: TestCaseController
tcList: TestCase
1 : analyzeCall( )
2 : getFeedback(table: Table)
3 : getMessage( )
4 :
5 : feedbackContainer = getFeedback
6 : isSuccessful( )
7 :
alt
[isSuccessful == true]
8 : makeTc(tableObj:Table)
9 : recursive1(data: TestCase, number: int, count: int)
10 : recursive2(property: ArrayList<String>, data: TestCase, number: int, count: int)
11 : saveProperty(property: ArrayList<String>, data: String)
12 : checkProperty(property: ArrayList<String>, data: String)
13 : add(singleTc:SingleTestCase)
14 : getTestCaseList( )
15 : tcListContainer = getTestCaseList
16 : getFeedBack(table: Table)
17 : getMessage( )
18 :
19 : feedbackContrainer = getFeedback
20 : displayTcList( )
[else]
21 : displayFeedback( )


interaction FeesualCPT_TestcaseDesc
Lifeline1: User
MainController
tcList: TestCase
singleTcList: SingleTestCase
1 : request
alt
[request == tcList]
2 : getAt( )
3 : getDesc()
4 :
[request == prevDesc]
5 : prevPos( )
6 : displayDesc( )
7 : getAt( )
8 : getDesc( )
9 :
[request == nextDesc]
10 : nextPos( )
11 : displayDesc( )
12 : getAt( )
13 : getDesc( )
14 :


interaction FeesualCPT_Save
Lifeline1: User
MainController
tfController: TextFileController
Table
TestCase
SingleTestCase
1 : saveCall( )
2 : saveRequest(table:Table, tcList:TestCase)
3 : getTable( )
4 :
5 : getAt( )
6 : getDesc( )
7 :
8 : getSingleTcList( )
9 : getRefNum( )
10 :
11 : TestCase.txt

Implement Class

Type	Class
Name	MainController
Purpose	GUI로 통해 받은 사용자의 요청을 처리하고 관리하기 위해
Overview	GUI로 통해 받은 사용자의 요청을 처리하고 관리하는 class
Cross Reference	Function: All UseCase: All
Exceptional Courses of Events	N/A

Type	Class
Name	FeedbackController
Purpose	사용자에게 Feedback을 알려주기 위해
Overview	사용자에게 알려줄 Feedback을 분석한다.
Cross Reference	Function: R2, R2.1 UseCase: Analyze, Mk Feedback
Exceptional Courses of Events	N/A

Implement Class

Type	Class
Name	Main
Purpose	GUI 담당
Overview	사용자의 입출력 관리를 담당하는 class
Cross Reference	Function: All UseCase: All
Exceptional Courses of Events	N/A

Type	Class
Name	Table
Purpose	사용자가 TestCase를 생성하기 위해 입력한 값들을 저장하기 위해
Overview	사용자가 TestCase를 생성하기 위해 입력한 값들을 저장하는 class
Cross Reference	Function: R1, R2, R2.1, R2.2, R4 UseCase: Set Table, Analyze, Mk Feedback, Mk Test case, Save File
Exceptional Courses of Events	N/A

Implement Class

Type	Class
Name	Feedback
Purpose	사용자에게 알려줄 메세지를 저장하기 위해
Overview	사용자에게 알려줄 메세지가 저장된 class
Cross Reference	Function: R2, R2.1 UseCase: Analyze, Mk Feedback
Exceptional Courses of Events	N/A

Type	Class
Name	SingleTestCase
Purpose	TestCase를 구성하는 가장 낮은 단위의 TestCase로서, 해당 Reference Number와 상세 설명을 저장하기 위해
Overview	TestCase를 구성하는 가장 낮은 단위의 TestCase로서, 해당 Reference Number와 상세 설명을 저장한다.
Cross Reference	Function: R2, R2.1, R2.2, R3, R4 UseCase: Analyze, Mk Feedback, Mk Test Case, Test Case Desc, Save File
Exceptional Courses of Events	N/A

Implement Class

Type	Class
Name	MainController
Purpose	GUI로 통해 받은 사용자의 요청을 처리하고 관리하기 위해
Overview	GUI로 통해 받은 사용자의 요청을 처리하고 관리하는 class
Cross Reference	Function: All UseCase: All
Exceptional Courses of Events	N/A

Type	Class
Name	FeedbackController
Purpose	사용자에게 Feedback을 알려주기 위해
Overview	사용자에게 알려줄 Feedback을 분석한다.
Cross Reference	Function: R2, R2.1 UseCase: Analyze, Mk Feedback
Exceptional Courses of Events	N/A

Methods Definitions

Type	Method
Name	getFeedback
Purpose	입력받은 Table에 맞는 feedback을 분석하기 위해
Cross Reference	Function: R2, R2.1 UseCase: Analyze, Mk Feedback
Input	table: Table
Output	feedbackList: Feedback
Abstract operation	분석한 feedback을 반환한다.
Exceptional Courses of Events	N/A

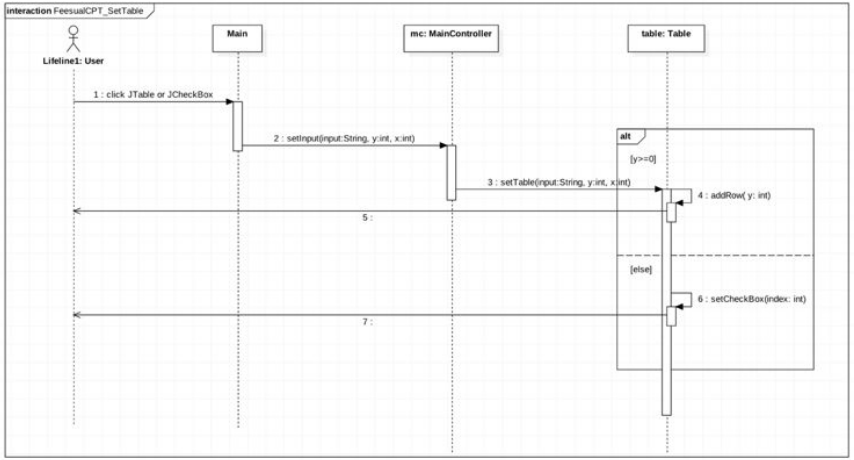
Methods Definitions

Type	Method
Name	setConstraints
Purpose	Constraints: int[]를 설정하기 위해
Cross Reference	Function: R2 UseCase: Analyze
Input	tf0: boolean, tf1: boolean, tf2: boolean
Output	N/A
Abstract operation	입력받은 값으로 Constraints: int[]를 설정한다.
Exceptional Courses of Events	N/A

Methods Definitions

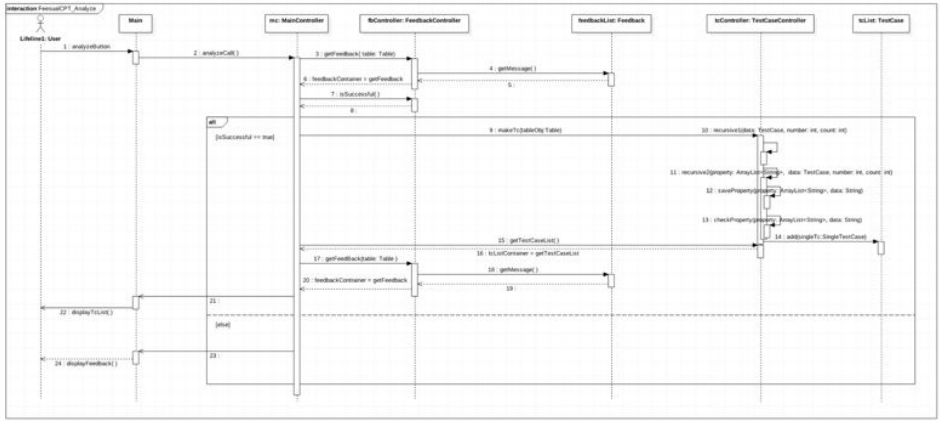
Type	Method
Name	makeTc
Purpose	TestCase를 생성하여 tcList:ArrayList<TestCase>에 추가한다.
Cross Reference	Function: R2, R2.2 UseCase: Analyze, Mk Test Case
Input	Table: Table
Output	N/A
Abstract operation	Single과 error를 먼저 검사하여 TestCaseList에 추가하고, if-property가 있는 subCategory와 없는 subCategory를 분리한다. propertySub끼리 조합하여 propertyTc를 만들고,nonPropertySub끼리 조합하여 nonPropertyTc를 만든다. nonPropertyTc와 propertyTc를 조합하여 TestCase를 만들어내서 tcList에 추가한다.
Exceptional Courses of Events	N/A

Implement Windows



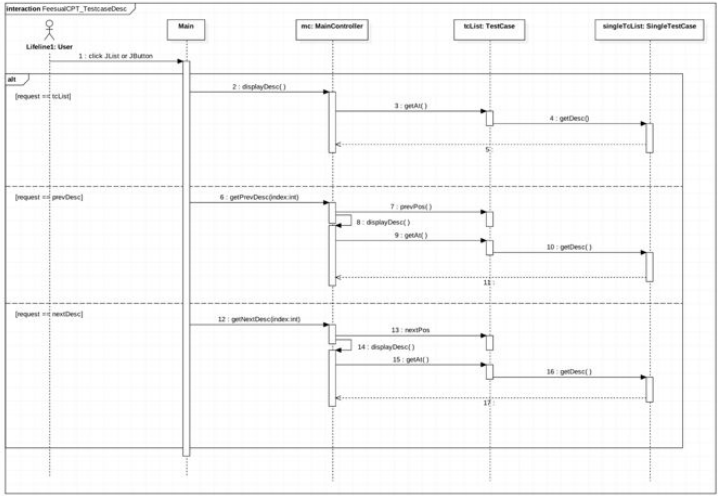
Name	click JTable or JCheckBox
Responsibilities	GUI에서 Table이나 CheckBox를 누른다.
Type	GUI
Cross Reference	R1
Notes	GUI에서 Table이나 CheckBox를 누른다.
Pre-Conditions	GUI 창이 뜬다.
Post-Conditions	유저의 입력에 따라 값을 유지하고 보여준다.

Implement Windows



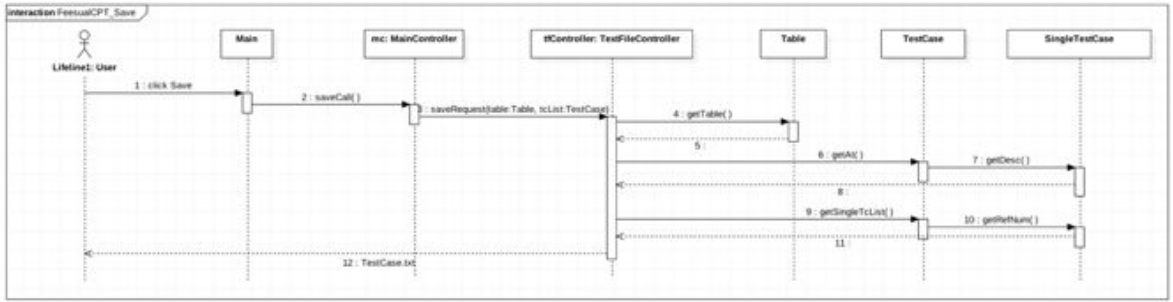
Name	analyzeButton
Responsibilities	GUI에서 analyzeButton을 누른다.
Type	GUI
Cross Reference	R2, R2.1, R2.2
Notes	GUI에서 analyzeButton을 누른다.
Pre-Conditions	Table에 값이 저장되어 있다.
Post-Conditions	저장된 Table을 통해 Feedback을 얻고, TestCase를 생성할 수 있다면 생성한다.

Implement Windows



Name	click JList or JButton
Responsibilities	GUI에서 JList나 prevButton, nextButton을 누른다.
Type	GUI
Cross Reference	R3
Notes	GUI에서 JList나 prevButton, nextButton을 누른다.
Pre-Conditions	Analyze를 통해 얻은 Test Case 리스트를 보유한다.
Post-Conditions	유저의 액션에 따라 Test Case의 상세 정보들 중 하나를 띄운다.

Implement Windows



Name	click Save
Responsibilities	GUI에서 Save 버튼을 누른다.
Type	GUI
Cross Reference	R4
Notes	GUI에서 Save 버튼을 누른다.
Pre-Conditions	N/A
Post-Conditions	Table 입력값과 모든 Test Case들을 저장한 TestCase.txt를 생성한다.

Unit Test Code

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

import java.util.ArrayList;

import org.junit.Test;

public class FeedbackControllerTest {

  FeedbackController fbc;

  @Test
  public void testGetFeedbackTable() {
    fbc = new FeedbackController();
    assertNotSame(fbc.getFeedback(new Table()), new Feedback(
        "Table에 해당하는 피드백이 없습니다."));
  }

  @Test
  public void testGetFeedbackTable1() {
    fbc = new FeedbackController();
    assertNotNull(fbc.getFeedback(new Table()));
  }

  @Test
  public void testGetFeedbackTable2() {
    ///returns false b/c they are not referring to the same object
    fbc = new FeedbackController();
    assertSame(fbc.getFeedback(new Table()), new Feedback("고객센터로 연락 ㅠㅠ"));
  }

  @Test
  public void testGetFeedbackArrayListOfTestCase() {
    fbc = new FeedbackController();
    assertNotSame(fbc.getFeedback(new ArrayList()), new Feedback(
        "Table에 해당하는 피드백이 없습니다."));
  }
  @Test
  public void testGetFeedbackArrayListOfTestCase1() {
    fbc = new FeedbackController();
    assertNotNull(fbc.getFeedback(new ArrayList()));
  }
  @Test
  public void testGetFeedbackArrayListOfTestCase2() {
    ///returns false b/c they are not referring to the same object
        fbc = new FeedbackController();
        assertSame(fbc.getFeedback(new ArrayList()), new Feedback("고객센터로 연락 ㅠㅠ"));
  }

}
```

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

import org.junit.Test;

public class FeedbackTest {

  Feedback fbt;

  @Test
  public void testGetMessage() {
    String str = "RIGHT";
    fbt = new Feedback(str);
    assertEquals(fbt.getMessage(), str);
  }

  @Test
  public void testGetMessage1() {
    String str1 = "RIGHT";
    String str2 = "WRONG";
    fbt = new Feedback(str1);
    assertEquals(fbt.getMessage(), str2);
  }

}
```

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

import java.util.ArrayList;

import org.junit.Test;

public class TestCaseTest {

  TestCase tc;

  @Test
  public void testNextPos() {
    tc = new TestCase();
    assertEquals(0, tc.pos);
    assertNotSame(-1, tc.pos);
  }

  @Test
  public void testNextPos1() {
    tc = new TestCase();
    assertEquals(-1, tc.pos);
  }

  @Test
  public void testPrevPos() {
    tc = new TestCase();
    tc.prevPos();
    assertEquals(-1, tc.pos);
    assertNotSame(0, tc.pos);
  }

  @Test
  public void testPrevPos1() {
    tc = new TestCase();
    tc.prevPos();
    assertEquals(0, tc.pos);
  }

  @Test
  public void getSingTcListTest() {
    tc = new TestCase();
    ArrayList<SingleTestCase> stlist = new ArrayList<SingleTestCase>();
    assertArrayEquals(stlist.toArray(), tc.getSingleTcList().toArray());
  }

  @Test
  public void getSingTcListTest1() {
    tc = new TestCase();
    ArrayList<String> stlist = new ArrayList<String>();
    assertTrue(stlist.equals(tc.getSingleTcList()));
  }

  @Test
  public void testGetAt() {
    tc = new TestCase();
    tc.add(null, "a", null, null);
    assertEquals("a", tc.getAt());
  }

}
```

Unit Test Code

```java
import static org.junit.Assert.*;
import java.util.ArrayList;
import org.junit.Test;

public class TestCaseControllerTest {

  TestCaseController tcc;

  @Test
  public void testGetTestCaseList() {
    tcc = new TestCaseController();

    assertNull(tcc.getTestCaseList());
  }

  @Test
  public void testGetTc() {
    MainController mc = new MainController();

    mc.setInput("1.1.1", 0, 0);
    mc.setInput("시스템 액션", 0, 1);
    mc.setInput("장애조작", 0, 2);
    mc.setInput("화재", 0, 3);
    mc.setInput("OBSFR", 0, 5);

    mc.setInput("1.1.2", 1, 0);
    mc.setInput("시스템 액션", 1, 1);
    mc.setInput("장애조작", 1, 2);
    mc.setInput("수해", 1, 3);
    mc.setInput("OBSFD", 1, 5);

    mc.setInput("1.1.3", 2, 0);
    mc.setInput("시스템 액션", 2, 1);
    mc.setInput("장애조작", 2, 2);
    mc.setInput("지진", 2, 3);
    mc.setInput("OBSEQ", 2, 5);

    mc.setInput("1.1.4", 3, 0);
    mc.setInput("시스템 액션", 3, 1);
    mc.setInput("장애조작", 3, 2);
    mc.setInput("추락", 3, 3);
    mc.setInput("OBSCR", 3, 5);

    mc.setInput("1.2.1", 4, 0);
    mc.setInput("시스템 액션", 4, 1);
    mc.setInput("시뮬레이션", 4, 2);
    mc.setInput("시작", 4, 3);

    mc.setInput("1.2.2", 5, 0);
    mc.setInput("시스템 액션", 5, 1);
    mc.setInput("시뮬레이션", 5, 2);
    mc.setInput("정지", 5, 3);
    mc.setInput("error", 5, 7);

    mc.setInput("3.1.1", 6, 0);
    mc.setInput("계기판 display", 6, 1);
    mc.setInput("표시", 6, 2);
    mc.setInput("캐빈 무게", 6, 3);
    mc.setInput("WEI", 6, 5);
```

```java
    mc.setInput("3.1.2", 7, 0);
    mc.setInput("계기판 display", 7, 1);
    mc.setInput("표시", 7, 2);
    mc.setInput("운행 속도", 7, 3);
    mc.setInput("single", 7, 6);

    mc.setInput("3.1.3", 8, 0);
    mc.setInput("계기판 display", 8, 1);
    mc.setInput("표시", 8, 2);
    mc.setInput("대기 인원", 8, 3);

    mc.setInput("3.1.4", 9, 0);
    mc.setInput("계기판 display", 9, 1);
    mc.setInput("표시", 9, 2);
    mc.setInput("모터의 힘", 9, 3);

    mc.setInput("3.1.5", 10, 0);
    mc.setInput("계기판 display", 10, 1);
    mc.setInput("표시", 10, 2);
    mc.setInput("만원", 10, 3);

    mc.setInput("3.1.6", 11, 0);
    mc.setInput("계기판 display", 11, 1);
    mc.setInput("표시", 11, 2);
    mc.setInput("탑승 인원", 11, 3);

    mc.setInput("2.1.1", 12, 0);
    mc.setInput("설정", 12, 1);
    mc.setInput("승객 생성", 12, 2);
    mc.setInput(">= 0 인 정수", 12, 3);

    mc.setInput("2.1.2", 13, 0);
    mc.setInput("설정", 13, 1);
    mc.setInput("승객 생성", 13, 2);
    mc.setInput("그 외", 13, 3);
    mc.setInput("error", 13, 7);

    mc.setInput("2.2.1", 14, 0);
    mc.setInput("설정", 14, 1);
    mc.setInput("모터 출력", 14, 2);
    mc.setInput(">= 0 인 정수", 14, 3);

    mc.setInput("2.2.2", 15, 0);
    mc.setInput("설정", 15, 1);
    mc.setInput("모터 출력", 15, 2);
    mc.setInput("그 외", 15, 3);
    mc.setInput("error", 15, 7);

    mc.setInput("2.3.1", 16, 0);
    mc.setInput("설정", 16, 1);
    mc.setInput("전체 승객 수", 16, 2);
    mc.setInput(">= 0 인 정수", 16, 3);

    mc.setInput("2.3.2", 17, 0);
    mc.setInput("설정", 17, 1);
    mc.setInput("전체 승객 수", 17, 2);
    mc.setInput("그 외", 17, 3);
    mc.setInput("error", 17, 7);
```

```java
    mc.setInput("2.4.1", 18, 0);
    mc.setInput("설정", 18, 1);
    mc.setInput("캐빈 정원", 18, 2);
    mc.setInput(">= 0 인 정수", 18, 3);

    mc.setInput("2.4.2", 19, 0);
    mc.setInput("설정", 19, 1);
    mc.setInput("캐빈 정원", 19, 2);
    mc.setInput("그 외", 19, 3);
    mc.setInput("error", 19, 7);

    mc.setInput("2.5.1", 20, 0);
    mc.setInput("설정", 20, 1);
    mc.setInput("캐빈 한계 무게", 20, 2);
    mc.setInput("<= 현재 총 무게", 20, 3);
    mc.setInput("WEI", 20, 4);
    mc.setInput("WEIBELW", 20, 5);

    mc.setInput("2.5.2", 21, 0);
    mc.setInput("설정", 21, 1);
    mc.setInput("캐빈 한계 무게", 21, 2);
    mc.setInput("> 현재 총 무게", 21, 3);
    mc.setInput("WEI", 21, 4);
    mc.setInput("WEIOVER", 21, 5);

    mc.setInput("4.1.1", 22, 0);
    mc.setInput("장애대응", 22, 1);
    mc.setInput("화재대응", 22, 2);
    mc.setInput("인명피해 발생", 22, 3);
    mc.setInput("OBSFR", 22, 4);

    mc.setInput("4.1.2", 23, 0);
    mc.setInput("장애대응", 23, 1);
    mc.setInput("화재대응", 23, 2);
    mc.setInput("캐빈 우선동작", 23, 3);
    mc.setInput("OBSFR", 23, 4);

    mc.setInput("4.1.3", 24, 0);
    mc.setInput("장애대응", 24, 1);
    mc.setInput("화재대응", 24, 2);
    mc.setInput("화재 진압", 24, 3);
    mc.setInput("OBSFR", 24, 4);

    mc.setInput("4.2.1", 25, 0);
    mc.setInput("장애대응", 25, 1);
    mc.setInput("수해대응", 25, 2);
    mc.setInput("1층 인명피해 발생", 25, 3);
    mc.setInput("OBSFD", 25, 4);

    mc.setInput("4.2.2", 26, 0);
    mc.setInput("장애대응", 26, 1);
    mc.setInput("수해대응", 26, 2);
    mc.setInput("캐빈 우선동작", 26, 3);
    mc.setInput("OBSFD", 26, 4);
```

```java
    mc.setInput("4.3.1", 27, 0);
    mc.setInput("장애대응", 27, 1);
    mc.setInput("지진대응", 27, 2);
    mc.setInput("인명피해 발생", 27, 3);
    mc.setInput("OBSEQ", 27, 4);

    mc.setInput("4.3.2", 28, 0);
    mc.setInput("장애대응", 28, 1);
    mc.setInput("지진대응", 28, 2);
    mc.setInput("캐빈 동작", 28, 3);
    mc.setInput("OBSEQ", 28, 4);

    mc.setInput("4.4.1", 29, 0);
    mc.setInput("장애대응", 29, 1);
    mc.setInput("추락대응", 29, 2);
    mc.setInput("인명피해 발생", 29, 3);
    mc.setInput("OBSCR/WEIOVER", 29, 4);

    mc.setInput("4.4.2", 30, 0);
    mc.setInput("장애대응", 30, 1);
    mc.setInput("추락대응", 30, 2);
    mc.setInput("인명피해 발생", 30, 3);
    mc.setInput("OBSCR/WEIOVER", 30, 4);

    mc.tableObj.setConstraints(true, true, true);

    mc.tcController.makeTc(mc.tableObj);
    mc.tcController.getTestCaseList().size();
    assertEquals(mc.tcController.getTestCaseList().size(), 55);
  }
```

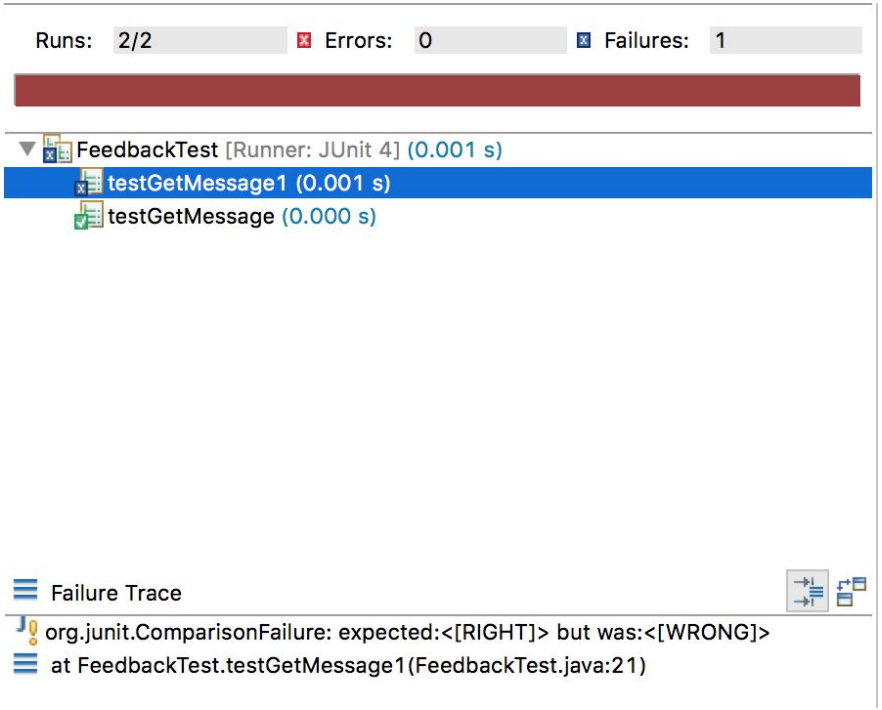
Unit Test Code

```
@Test
public void testGetTc2() {
  MainController mc = new MainController();

  mc.setInput("1.1.1", 0, 0);
  mc.setInput("시스템 액션", 0, 1);
  mc.setInput("장애조작", 0, 2);
  mc.setInput("화재", 0, 3);
  mc.setInput("OBSFR", 0, 5);

  mc.setInput("1.1.2", 1, 0);
  mc.setInput("시스템 액션", 1, 1);
  mc.setInput("장애조작", 1, 2);
  mc.setInput("수해", 1, 3);
  mc.setInput("OBSFD", 1, 5);

  mc.setInput("1.1.3", 2, 0);
  mc.setInput("시스템 액션", 2, 1);
  mc.setInput("장애조작", 2, 2);
  mc.setInput("지진", 2, 3);
  mc.setInput("OBSEQ", 2, 5);

  mc.setInput("1.1.4", 3, 0);
  mc.setInput("시스템 액션", 3, 1);
  mc.setInput("장애조작", 3, 2);
  mc.setInput("추락", 3, 3);
  mc.setInput("OBSCR", 3, 5);

  mc.setInput("1.2.1", 4, 0);
  mc.setInput("시스템 액션", 4, 1);
  mc.setInput("시뮬레이션", 4, 2);
  mc.setInput("시작", 4, 3);

  mc.setInput("1.2.2", 5, 0);
  mc.setInput("시스템 액션", 5, 1);
  mc.setInput("시뮬레이션", 5, 2);
  mc.setInput("정지", 5, 3);
  mc.setInput("error", 5, 7);

  mc.setInput("3.1.1", 6, 0);
  mc.setInput("계기판 display", 6, 1);
  mc.setInput("표시", 6, 2);
  mc.setInput("캐빈 무게", 6, 3);
  mc.setInput("WEI", 6, 5);
```

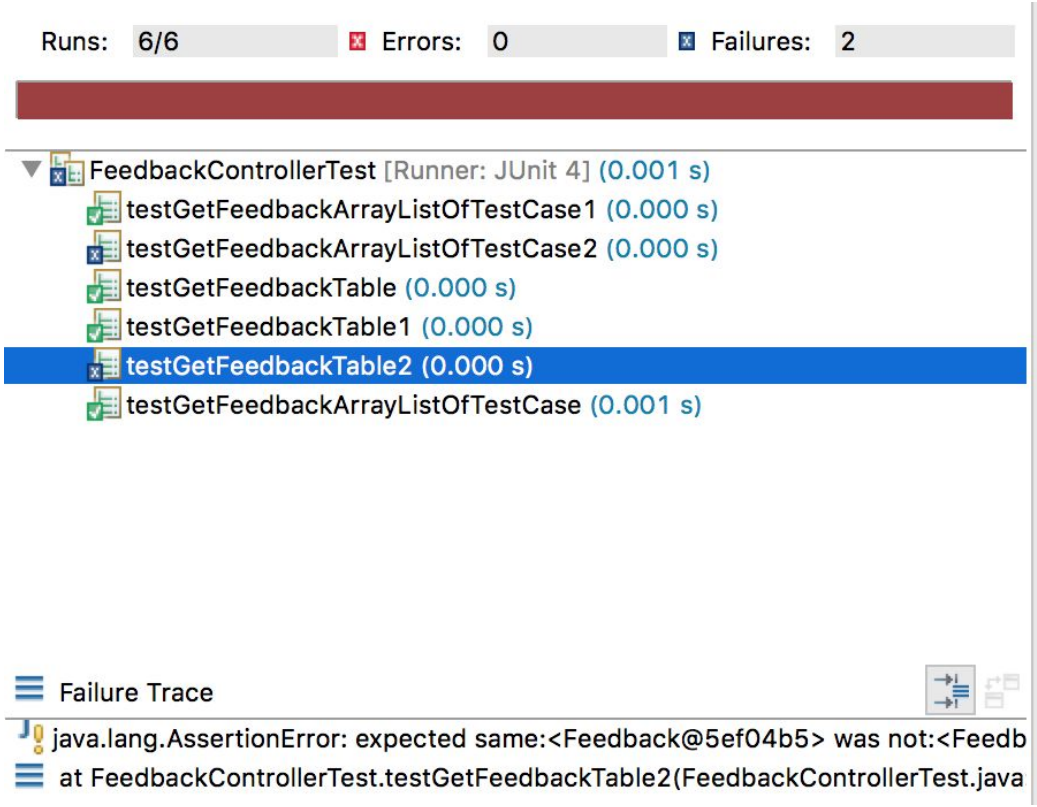
```
  mc.setInput("4.3.1", 27, 0);
  mc.setInput("장애대응", 27, 1);
  mc.setInput("지진대응", 27, 2);
  mc.setInput("인명피해 발생", 27, 3);
  mc.setInput("OBSEQ", 27, 4);

  mc.setInput("4.3.2", 28, 0);
  mc.setInput("장애대응", 28, 1);
  mc.setInput("지진대응", 28, 2);
  mc.setInput("캐빈 동작", 28, 3);
  mc.setInput("OBSEQ", 28, 4);

  mc.setInput("4.4.1", 29, 0);
  mc.setInput("장애대응", 29, 1);
  mc.setInput("추락대응", 29, 2);
  mc.setInput("인명피해 발생", 29, 3);
  mc.setInput("OBSCR/WEIOVER", 29, 4);

  mc.setInput("4.4.2", 30, 0);
  mc.setInput("장애대응", 30, 1);
  mc.setInput("추락대응", 30, 2);
  mc.setInput("인명피해 발생", 30, 3);
  mc.setInput("OBSCR/WEIOVER", 30, 4);

  mc.tableObj.setConstraints(false, false, false);

  mc.tcController.makeTc(mc.tableObj);
  mc.tcController.getTestCaseList().size();
  assertEquals(mc.tcController.getTestCaseList().size(), 960);
}
```

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

import java.util.ArrayList;

import org.junit.Test;

public class TestCaseTest {

  TestCase tc;

  @Test
  public void testNextPos() {
    tc = new TestCase();
    assertEquals(0, tc.pos);
    assertNotSame(-1, tc.pos);
  }

  @Test
  public void testNextPos1() {
    tc = new TestCase();
    assertEquals(-1, tc.pos);
  }

  @Test
  public void testPrevPos() {
    tc = new TestCase();
    tc.prevPos();
    assertEquals(-1, tc.pos);
    assertNotSame(0, tc.pos);
  }

  @Test
  public void testPrevPos1() {
    tc = new TestCase();
    tc.prevPos();
    assertEquals(0, tc.pos);
  }

  @Test
  public void getSingTcListTest() {
    tc = new TestCase();
    ArrayList<SingleTestCase> stlist = new ArrayList<SingleTestCase>();
    assertArrayEquals(stlist.toArray(), tc.getSingleTcList().toArray());
  }

  @Test
  public void getSingTcListTest1() {
    tc = new TestCase();
    ArrayList<String> stlist = new ArrayList<String>();
    assertTrue(stlist.equals(tc.getSingleTcList()));
  }

  @Test
  public void testGetAt() {
    tc = new TestCase();
    tc.add(null, "a", null, null);
    assertEquals("a", tc.getAt());
  }

}
```

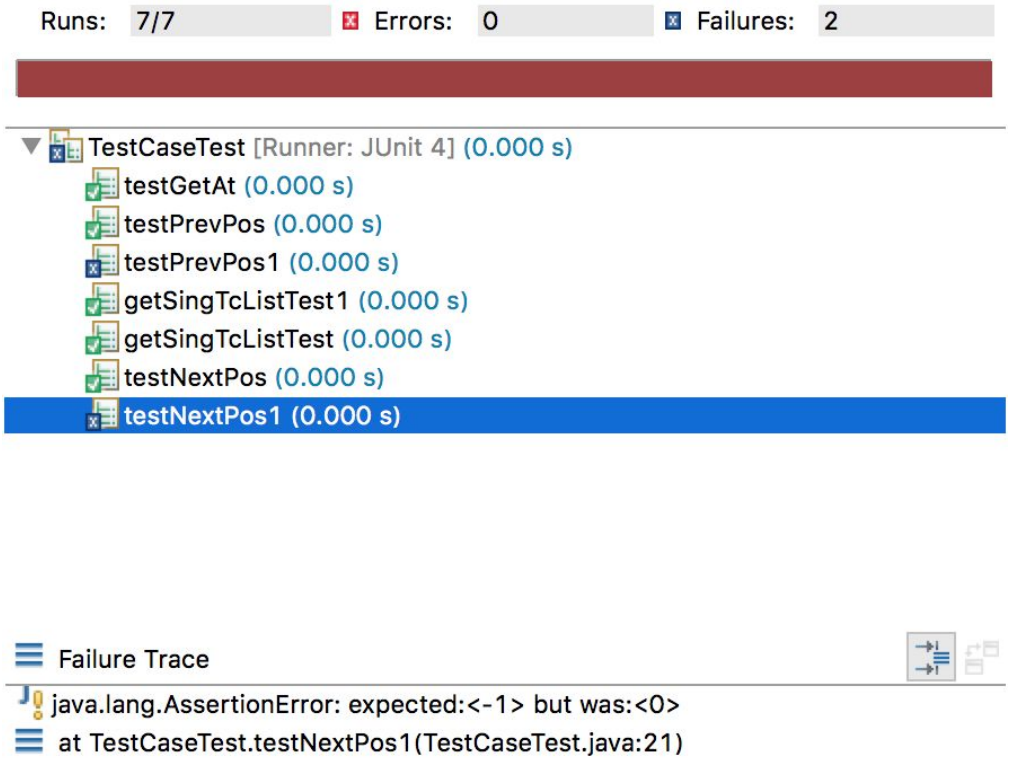
Unit Testing



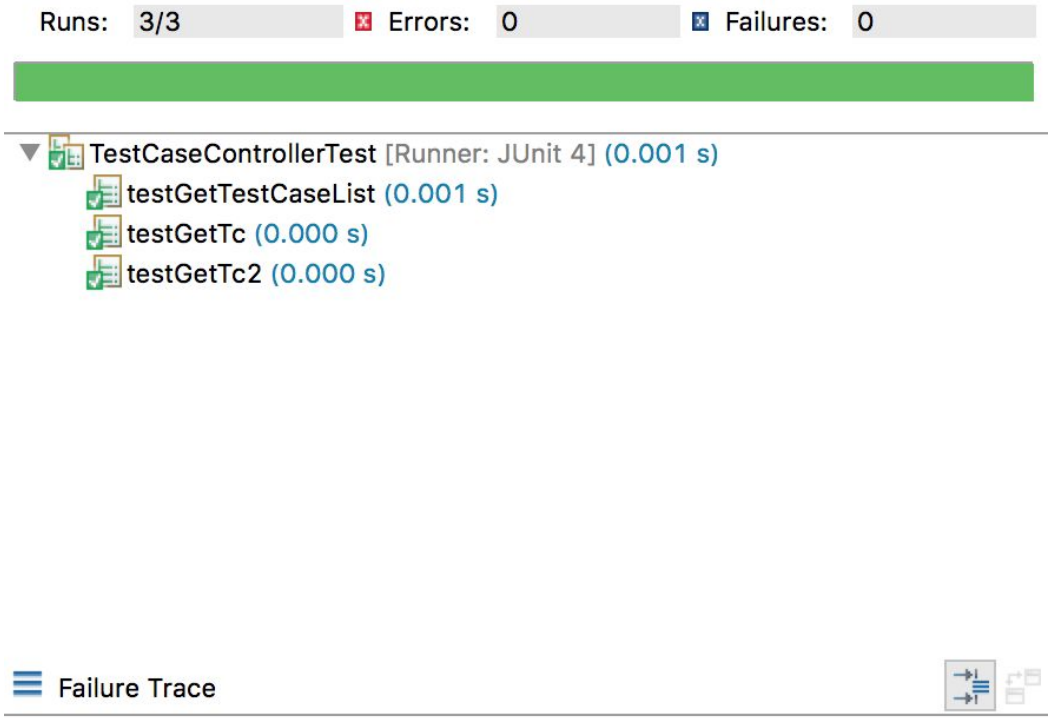
Unit Testing



Unit Testing



Unit Testing



System Testing (Functional)

Number	Test 항목	Test Desc	UseCase	Function
1-1	Input Test	표에 값을 입력한다.	Set Table	R1
1-2	Input Test	표에 값을 입력하고 다시 지운다.	Set Table	R1
2-1	Anaylze Test	빈 표일 때 Analyze 버튼을 누른다.	Analyze	R2
2-2	Analyze Test	표에 알맞은 값을 채우고 Analyze 버튼을 누른다.	Analze	R2
2-3	Analyze Test	표에 부적절한 값을 채우고 Analyze 버튼을 누른다.	Analyze	R2
3-1	Feedback Test	빈 표일 때 Analyze버튼을 누른 뒤 Feedback을 확인한다.	Mk Feedback	R2.1
3-2	Feedback Test	Table의 Single칸에 error를 입력하고 Analyze버튼을 누른 뒤 Feedback을 확인한다.	Mk Feedback	R2.1
3-3	Feedback Test	Property칸에 ABC를 입력하고 If칸에 입력하지 않은 채로 Analyze버튼을 누른 뒤 Feedback을 확인한다.	Mk Feedback	R2.1
3-4	Feedback Test	성공적으로 TestCase를 생성한 뒤 Feedback을 확인한다.	Mk Feedback	R2.1
4-1	Make TestCase Test	빈 표일 때 Analyze버튼을 누른 뒤 All Steps List를 확인한다.	Mk Test case	R2.2
4-2	Make TestCase Test	Table의 Single칸에 error를 입력하고 Analyze버튼을 누른 뒤 All Steps List를 확인한다.	Mk Test case	R2.2
4-3	Make TestCase Test	Property칸에 ABC를 입력하고 If칸에 입력하지 않은 채로 Analyze버튼을 누른 뒤 All Steps List를 확인한다..	Mk Test case	R2.2

System Testing (Functional)

4-4	Make TestCase Test	성공적으로 TestCase를 생성한 뒤 All Steps List를 확인하고 개수를 확인한다.	Mk Test case	R2.2
5-1	TestCaseDesc Test	성공적으로 TestCase를 생성한 뒤 All Steps List에서 2번째 줄을 누른다.	Test case Desc	R3
5-2	TestCaseDesc Test	성공적으로 TestCase를 생성한 뒤 All Steps List에서 2번째 줄을 누른 뒤 < 버튼을 누른다.	Test case Desc	R3
5-3	TestCaseDesc Test	성공적으로 TestCase를 생성한 뒤 All Steps List에서 2번째 줄을 누른 뒤 > 버튼을 누른다.	Test case Desc	R3
6-1	Save Test	Save 버튼을 누른다.	Save File	R4
6-2	Save Test	성공적으로 TestCase를 생성한 뒤 Save 버튼을 누른다.	Save File	R4

System Testing (Non-Functional)

Number	Test 항목	Non-Functional Requirements
NF_1	컴퓨터에 능숙하지 않은 자가 Table에 입력을 불편없이 입력할 수 있는지 확인한다.	사용자가 Table에 값을 쉽게 입력할 수 있어야 한다.
NF_2	Category Partition Test에 대한 기본적인 지식이 있는 사람들에게 메뉴얼을 보여주지 않고 Table을 보여 줬을 때, 해당 Test에 대해 똑같이 이해하고 있는지 확인한다.	사용자가 Table에 입력된 값을 직관적으로 볼 수 있게 한다.
NF_3	Category Partition Test에 대한 기본적인 지식이 있는 사람들에게 Feedback을 주었을 때 올바르게 이해하는지 확인한다.	다양한 Feedback들을 사용자가 알고 수정하기 쉽게 알려준다.
NF_4	Category Partition Test에 대한 기본적인 지식이 있는 사람들이 다른 사람이 만든 txt파일을 보고 결과를 올바르게 이해하는지 확인한다.	저장하는 txt파일은 알아보기 쉬운 형식으로 저장되어야 한다.