|  |  |
| --- | --- |
| 과목 | 소프트웨어 공학 |
| 학과 | 컴퓨터공학부 |
| 학번 | 09 |
| 이름 | 백인혁 20141266  백찬희 20142503  이동규 20140490  함지훈 20146512 |

EMB00000aec676b

SE Project Document

**목차**

1. **Domain model**
2. **Software Architecture, Design Model**

가. Class Diagram

나. Sequence diagram

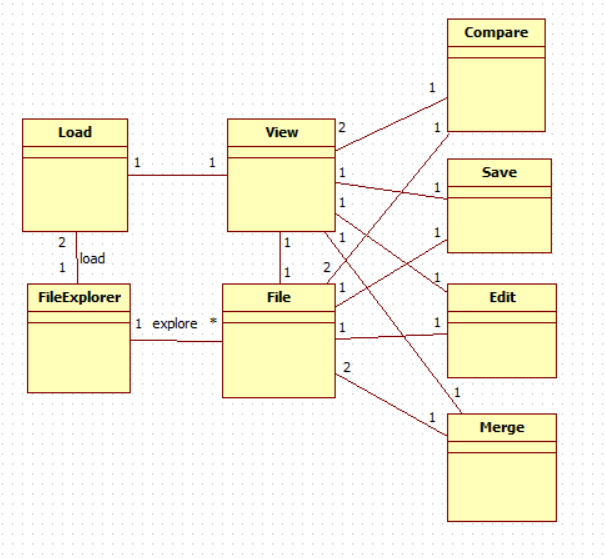
다. MVC

라. Unit-test

마. OOP

1. **Screen Shots of examples**
2. **Functional unit test cases and result**
3. **System test cases and result**
4. **Program Limitation**

**1. Domain model**

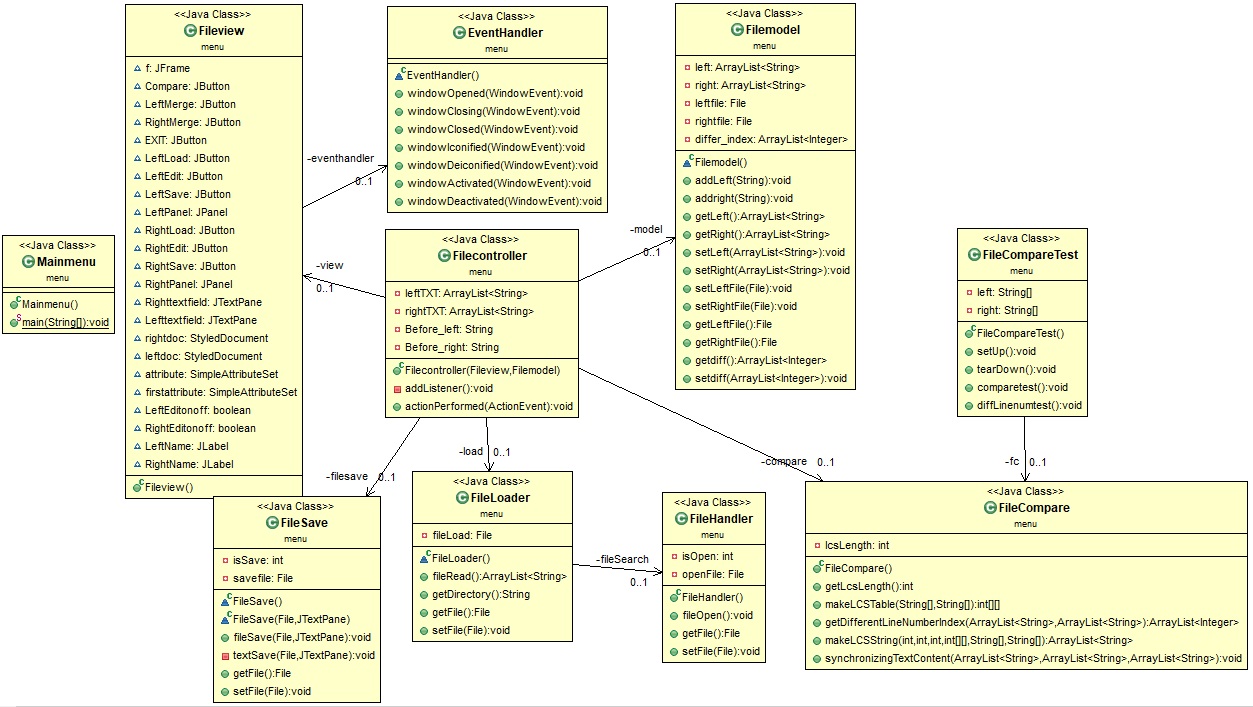


1개의 파일당 1개의 뷰를 가집니다. 또한 뷰를 통해서 파일을 수정, 병합, 비교를 할 수 있고 각 뷰마다 EDIT 버튼이 독립적으로 있습니다. EDIT 버튼은 뷰와 1대1 대응합니다. MERGE와 COMPARE는 2개의 뷰에 있는 텍스트를 비교하고 병합하는 데 사용되어집니다. 따라서 뷰 2개와 대응됩니다.

**2. Software Architecture, Design Model**

**가. Class Diagram**

실제 프로그램 구현에서는 MVC모델을 고려하여 구현하였습니다. FileLoader class를 통해서 텍스트 파일을 입력 받은 뒤 내용을 FIlemodel class에 저장합니다. Filecontroller class는 파일을 로드, 세이브, 에딧 등 컨트롤 할 수 있는 컨트롤러 입니다. 내부에는 해당 메소드들이 구현되어 있으며 파일을 보여주는 Fileview class에 사용되고 파일을 조작하기 때문에 Filemodel class에 데이터를 전달합니다. Fileview class는 유저한테 파일을 보여주고 입력을 받는 곳이기 때문에 이벤트 헨들러를 사용합니다.



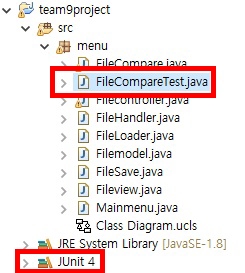
**나. Sequence Diagram**



비교를 원하는 파일을 2개 등록할 수 있다. 이때 등록은 LOAD를 통해서 파일 탐색기를 호출하여 파일을 선택한다. 등록한 파일 2개를 비교할 수 있는 데 COMPARE버튼을 통해 COMPARE함수에서 비교를 하여 결과를 뷰에서 보여줍니다. 뷰에 왼쪽병합과 오른쪽병합을 통해서 원하는 방향으로 파일에 내용을 병합할 수 있습니다. 병합시 해당 뷰의 내용이 변경됩니다. 수정의 경우 각 뷰에 1개씩 있으며 해당하는 텍스트파일을 수정할 수 있다. 마지막으로 저장도 뷰마다 1개씩 존재하며 텍스트 파일을 현재 상태로 저장합니다.

**다. MVC 개념**

File model은 로드한 파일에 내용을 저장합니다. File view는 사용자 인터페이스를 제공하고 model에 있는 저장한 내용을 사용자에게 보이는 화면 뷰를 출력합니다. File controller 는 사용자가 직접 연관이 있는 작업으로 model에 저장된 파일 내용을 이용해 저장 수정 비교 병합등의 기능을 수행할 수 있습니다. 사용자가 명령을 하면 그 내용을 기반으로 데이터가 모여있는 model로 보내고 모델에 저장되는 데이터에 상태를 변경 시키고 그 결과를 view에 출력합니다.

**라. Unit Test**

Eclipse에서 기본적으로 제공하는JUNIT 4를 사용해서 프로젝트를 테스트할 수 있도록 설계 되었습니다. 사진을 보면 JUnit4가 있는 것을 볼 수 있으며 FileCompare에 대한 내용을 테스트를 하기 위해 FileCompareTest.java를 추가해서 테스트를 진행하였습니다.

**마. Oop**

MVC모델을 기반으로 크게 파일을 수정할 수 있는 컨트롤러, 모델, 뷰로 나누었으며 이외에 탐색기를 불러오는 FileHandler와 FileLoader 클래스가 있으며 프로그램의 핵심기능인 Compare기능은 FileCompare 클래스로 별도 구현하였습니다. 각 클래스의 변수들은 다른 객체에서 접근하면 오류가 발생할 수 있기 때문에 private로 감춰주었고 접근 소요 발생시 각 클래스에 주어진getter, setter 메소드로 접근이 가능 하도록 해서 Data Encpasulation 했습니다. Filecontroller에서는 사용자에게 입력을 받기 위해서 ActionListener를 구현하였고 이를 위해서 ActionListener 인터페이스를 implements 해서 오버라이딩을 통해 구현했습니다.



데이터를 저장해서 모아두는 filemodel 클래스에 경우 변수를 마음대로 접근할 경우 문제가 될 수 있기에 위에 코드와 같이 private로 설정해서 외부에서 함부로 접근을 못하도록 구현했습니다. 이 내용을 변경하기 위해서 setter와 getter에 대한 내용들을 구현해서 이 부분을 통해서만 접근할 수 있도록 filemodel 클래스를 구현했습니다.

**3. Screen Shots of examples**

**가. Main**

**나. LOAD**

|  |  |
| --- | --- |
|  | 프로그램 실행화면이다. 맨 처음 실행시켰을 때 떠있는 메인 뷰이며 필요에 따라 LOAD, EDIT, SAVE, COMPARE, MERGE, EXIT 기능을 사용할 수 있다. 텍스트 파일을 등록하면 각 왼쪽, 오른쪽 뷰에 내용이 표기된다. |

|  |  |
| --- | --- |
|  | 불러오고 싶은 쪽의 LOAD를 누르면 파일 탐색기 창이 실행된다. 사용자가 등록하고 싶은 파일을 등록하면 해당 뷰에 표시가 된다. |
|  | 왼쪽은 텍스트 파일을 양쪽 뷰에 불러와 등록한 모습이다. 줄 수가 많은 경우 스크롤바를 통해 아래의 내용도 확인 할 수 있다. |

**다. COMPARE**

|  |  |
| --- | --- |
|  | COMPARE 버튼을 누르면 두 개의 텍스트를 비교하여 다른 부분을 노란색 배경과 밑줄, 굵은 글꼴로 표시를 해줍니다. 개행문자를 사용하여 줄 맞춤기능까지 제공하고 있습니다. |

**라. MERGE**

|  |  |
| --- | --- |
|  | 왼쪽으로 합치는 Leftmerge, 오른쪽으로 합치는 Rightmerge 기능을 제공하고 있으며 각 버튼을 누르면 비교했을 때 다른 부분을 추가/삭제 하여 텍스트를 변경해줍니다. |

**마. EDIT**

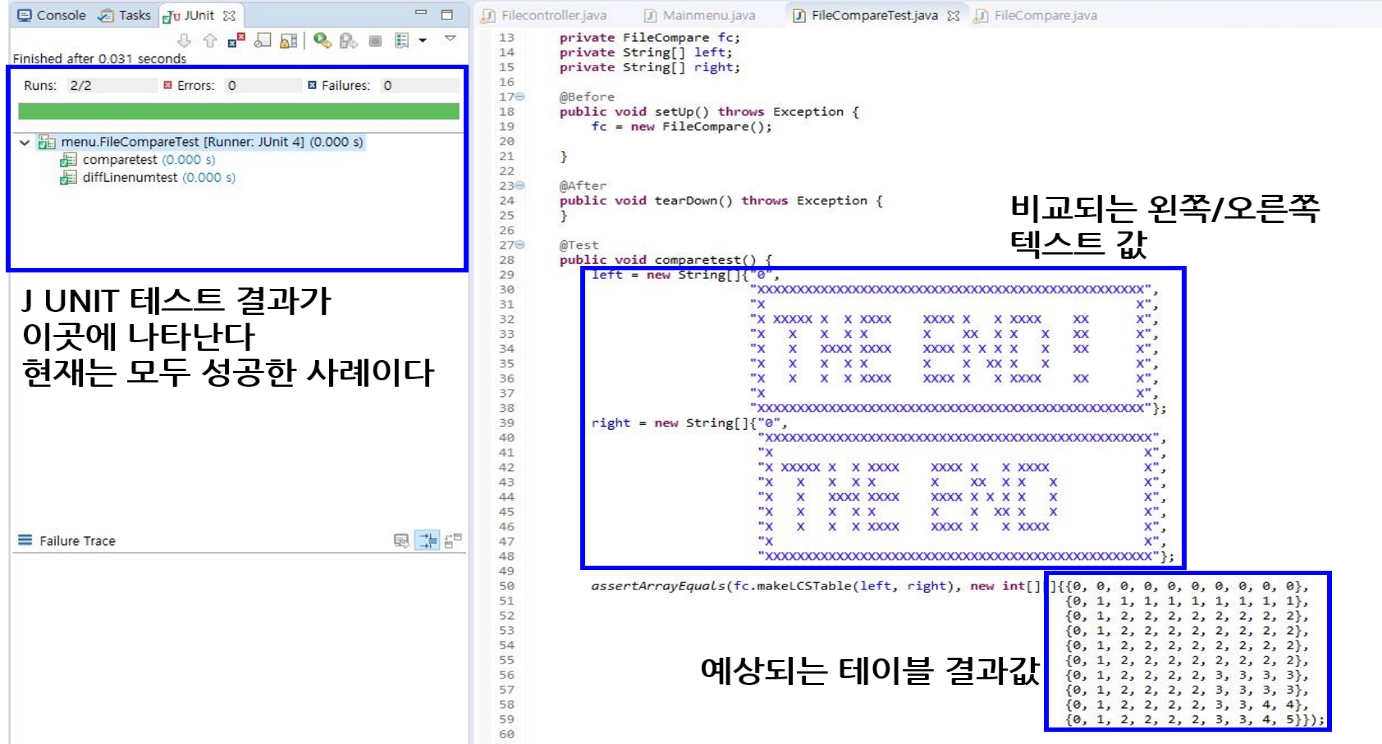
|  |  |
| --- | --- |
|  | 사용자가 원한다면 프로그램에서 텍스트를 수정할 수 있습니다. LOAD와 SAVE 가운데에 있는 EDIT 버튼을 누르면 뷰가 수정할 수 있게 설정됩니다. 이를 통해 텍스트를 수정할 수 있으며 저장하고 싶다면 SAVE 버튼을 통해서 저장할 수 있습니다. |

**바. SAVE**

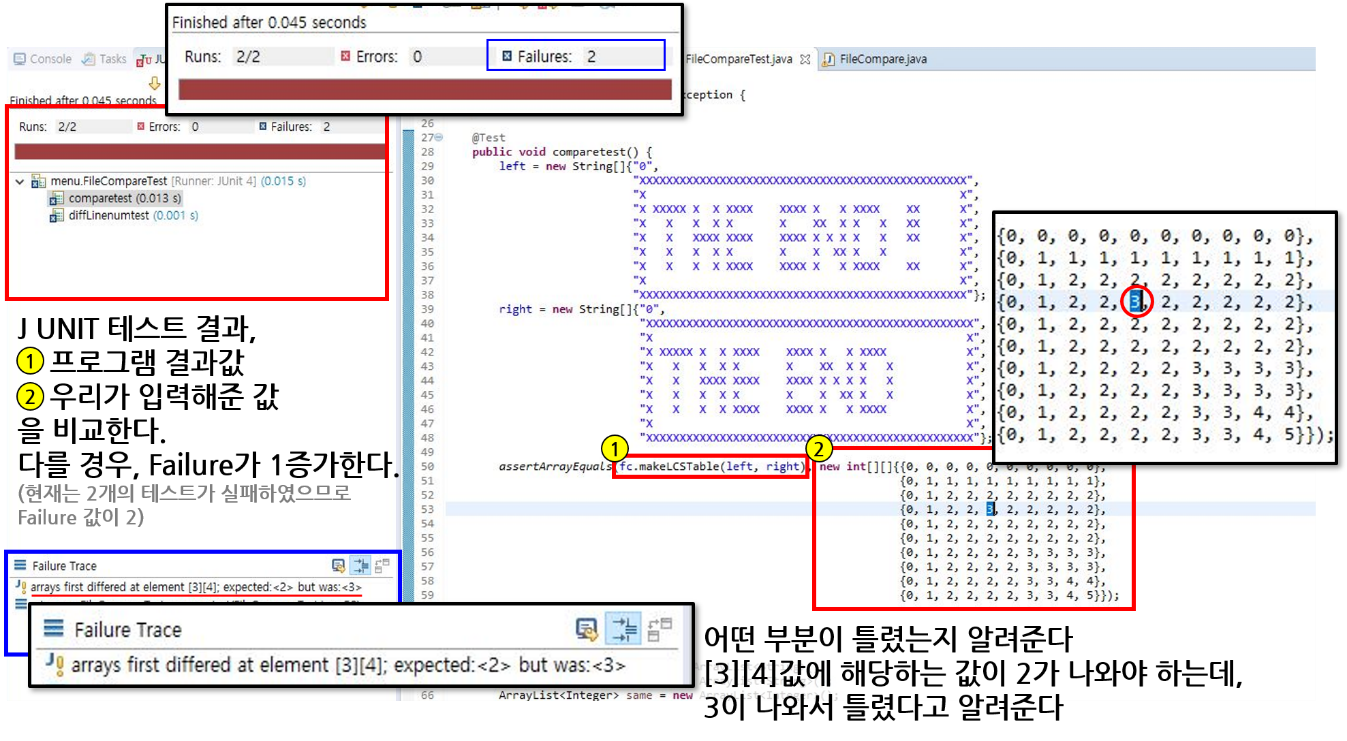
|  |  |
| --- | --- |
|  | COMPARE나 EDIT 등을 통해 변경된 내용을 저장하고 싶다면 SAVE버튼을 통해서 저장 할 수 있다. SAVE 버튼을 누르면 탐색기 창이 뜨며 원하는 경로와 파일명으로 저장할 수 있다. |
|  | 만약 원본파일에 저장하려고 하면 덮어쓰기를 묻는 메시지를 띄운다. 만약 ‘아니오’를 선택하면 저장 명령이 취소가 된다. 덮어쓰기를 하면 해당 파일에 내용이 덮어씌어진다. |
|  | EDIT으로 수정한 내용을 원본파일에 덮어씌운 저장한 결과이다. 정상적으로 값이 저장된 것을 확인 할 수 있다. |

**4. Functional unit test cases and result**

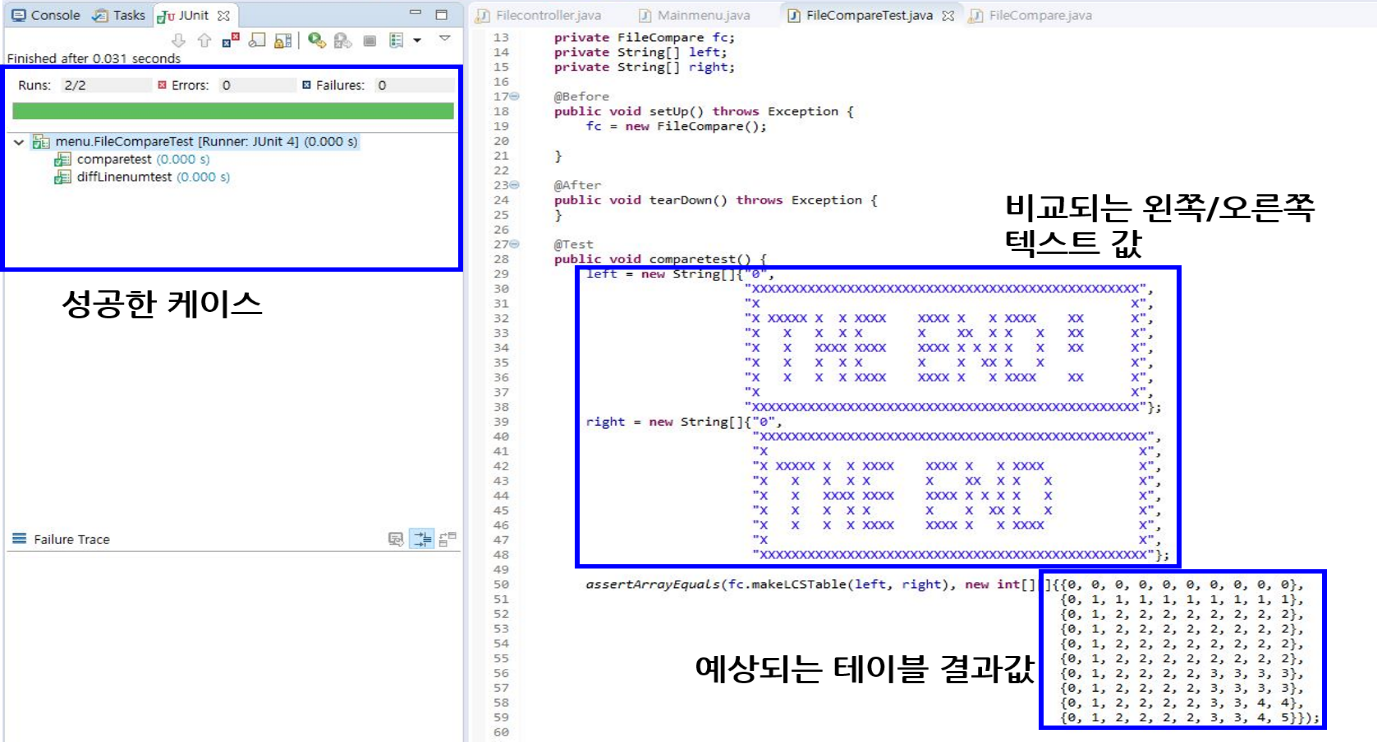
Junit4를 이용해서 테스트를 진행하였는데 S14left.txt와 S14right.txt를 비교하는 내용을 집중적으로 테스트하였습니다.



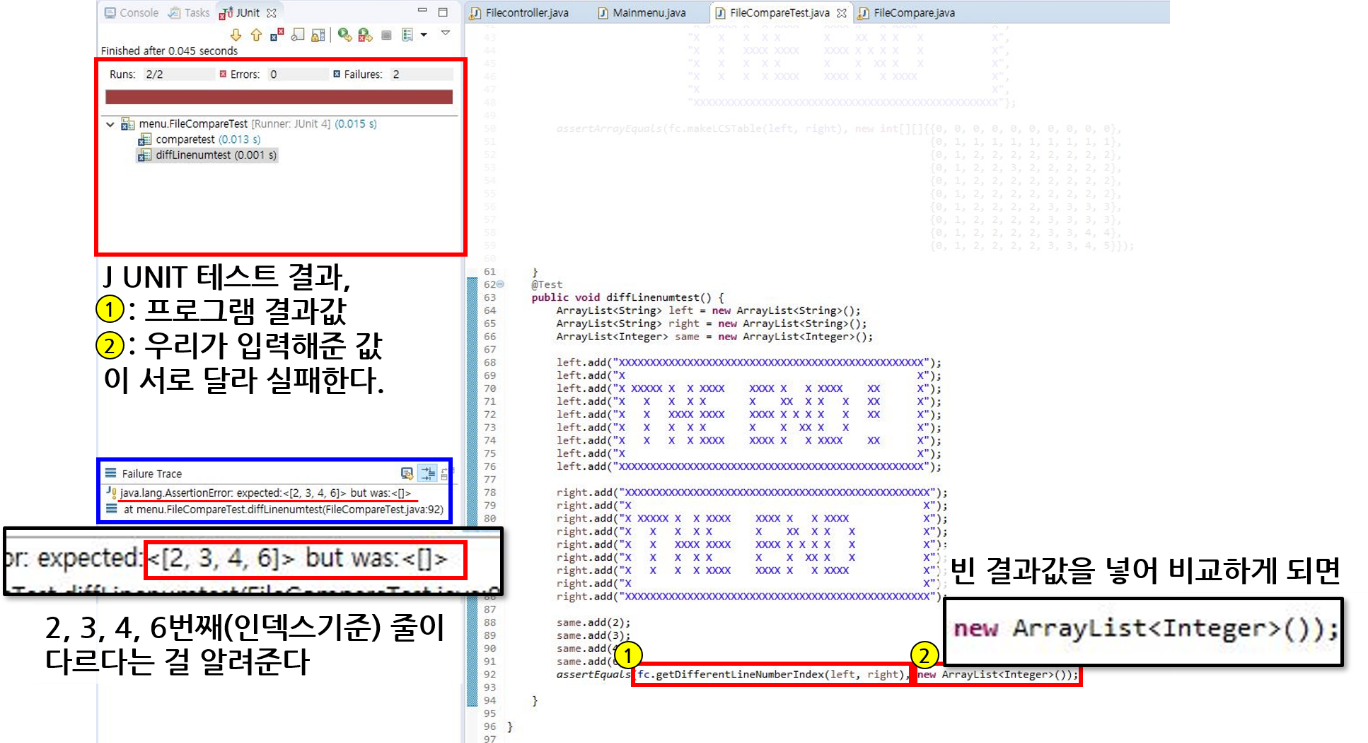
비교를 하기 위해 해당하는 테이블을 만드는 함수 makeLCSTable를 테스트를 했는데 S14left.txt에대한 내용과 S14right.txt에 대한 내용을 실험하기 위해 아래의 그림과 같이 설정을 하였습니다. 처음에 잘못된 결과를 테스트해보기 위해 도중에 3이라는 오류 값을 집어넣어 비교 시켜주면, 왼쪽하단 부분같이 예상 값과 다른 부분을 가르쳐 줍니다.



정확한 예상결과를 추가해서 테스트를 해보았는데 아래 사진처럼 오류 내용 없이 정상적인 결과가 나왔다는 것을 알 수 있었습니다.



그 다음 테스트로 getDifferentLineNumberIndex함수를 테스트해보았는데 이 테스트를 통해서 서로 다른 줄이 몇 번째 줄인지를 테스트를 통해서 원하는 값이랑 맞는지 테스트를 해보았습니다.



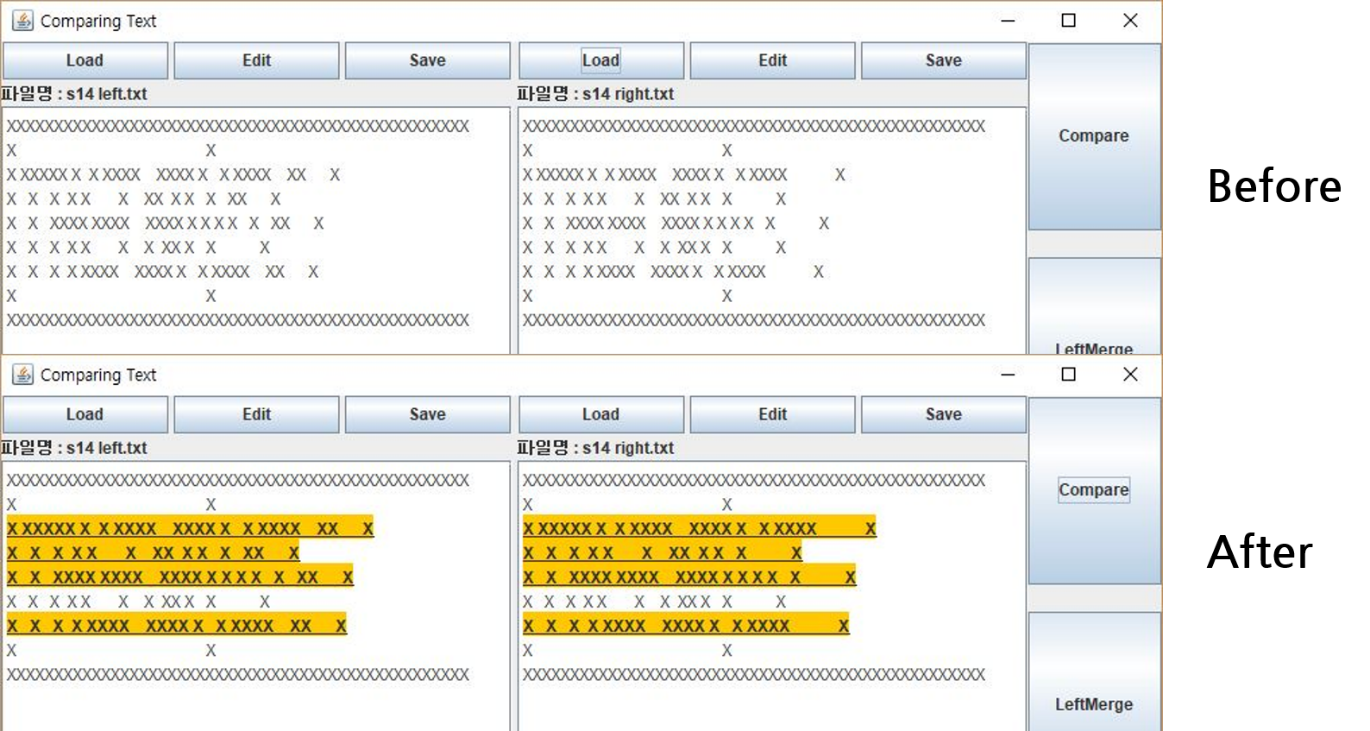
일단 예상 값을 2,3,4,6번째 줄이 다르므로 [2,3,4,6]으로 생각을 하고 맨 처음에는 비어있는 값을 넣어서 예상 값이 뭐라고 나오는지 실험을 해보았습니다. 그 결과 이 부분이 오류라고 나오고 예상결과로 [2,3,4,6]을 알려주었습니다.



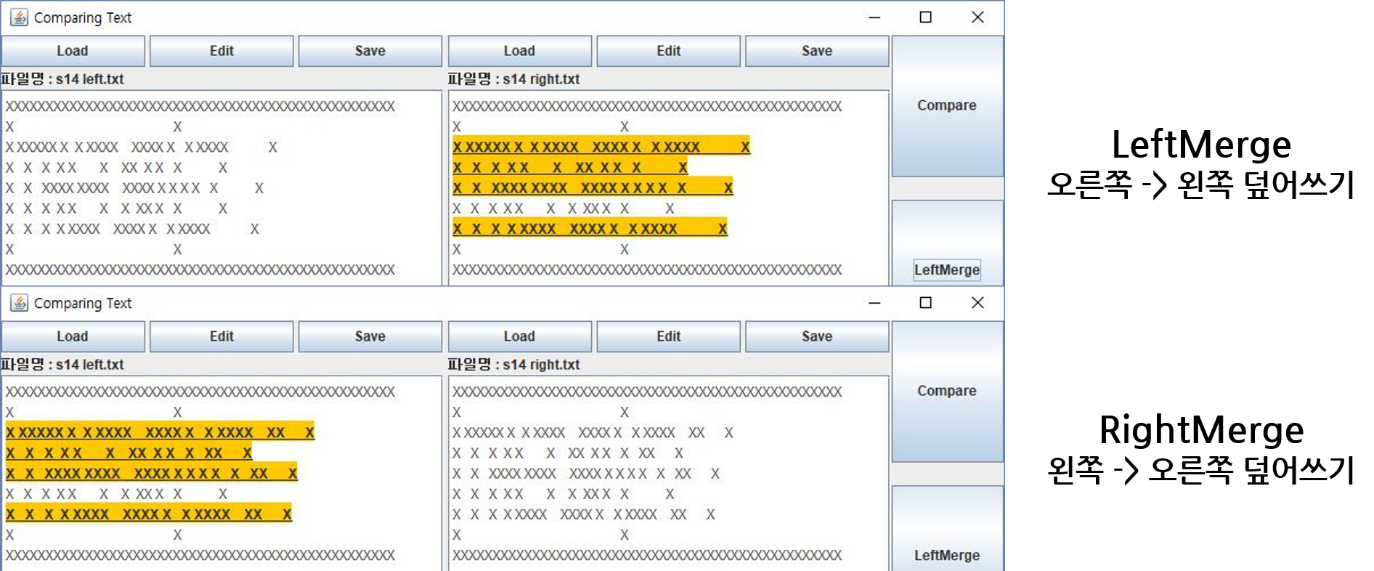
그 후 정확한 결과 값을 넣고 실험을 했는데 Junit test에서 오류가 없고 정확한 결과값이 나왔다고 출력했습니다.

**5. System test cases and result**

JUNIT을 통해 유닛이 잘 돌아가는지 확인을 완료하고, 본격적으로 실제 프로그램을 실행시켜보았습니다. 실행결과는 아래 사진과 같습니다.



예제 14번을 테스트한 결과를 스크린샷으로 찍어 비교한 사진입니다. 보다시피 3, 4, 5, 7번 라인이 서로 다르기 때문에 색으로 구별된 모습을 확인하실 수 있습니다.



Merge기능도 각각 눌렀을 때, LeftMerge는 오른쪽을 기준으로, 왼쪽으로 덮어쓰고, RightMerge는 왼쪽을 기준으로, 오른쪽으로 덮어쓰는 걸 확인할 수 있었습니다.

**6. Program Limitation**

프로그램의 비정상 종료 시 기존의 작업 내용이 저장되지 않습니다.