Analysis & Design, Implementation, and Test

Document

개체이(가) 표시된 사진

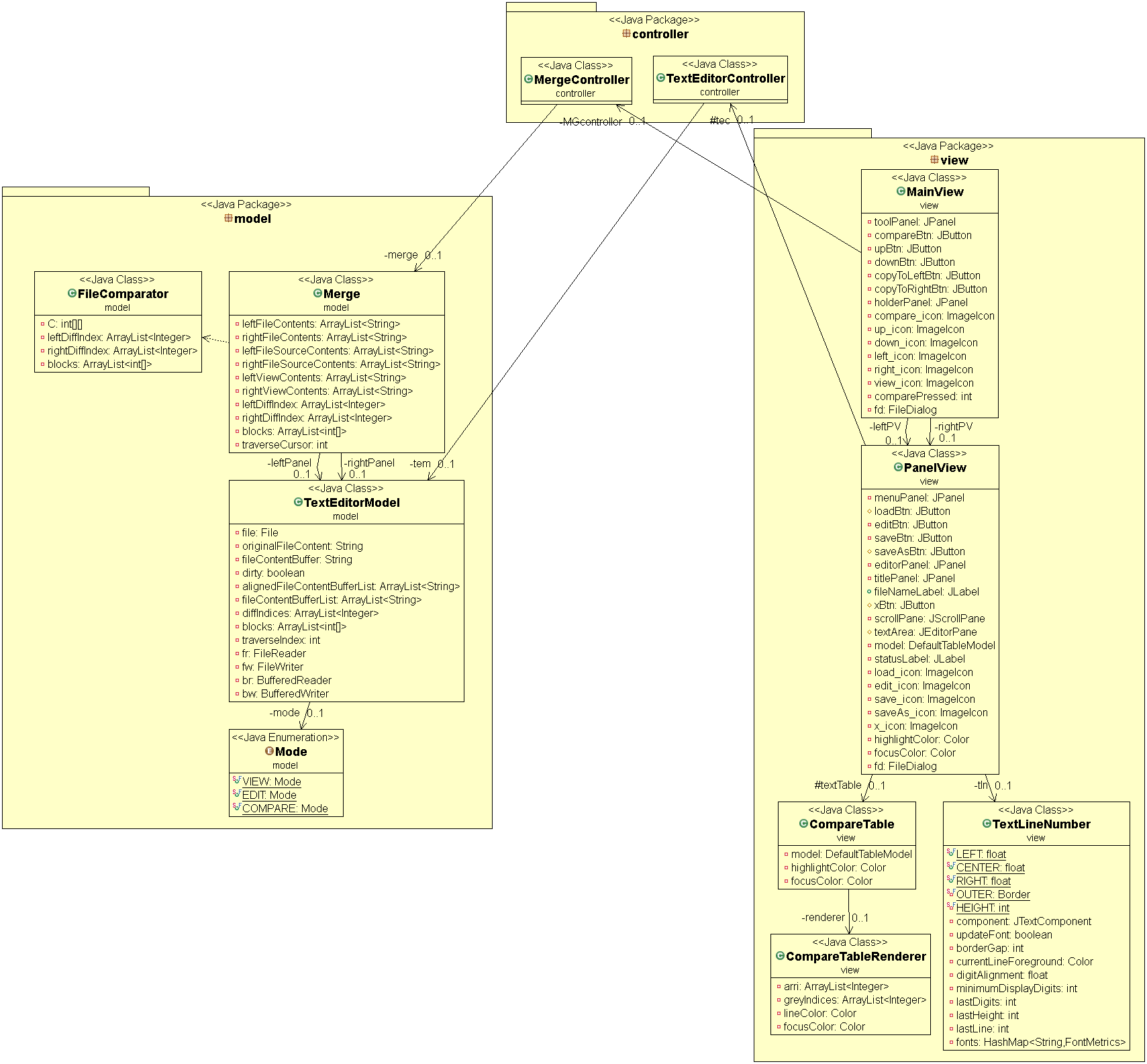
매우 높은 신뢰도로 생성된 설명

TEAM 16

**김은경 (20163927) 김소연 (20165417) 김수진 (20160342) 남유선 (20163228)**

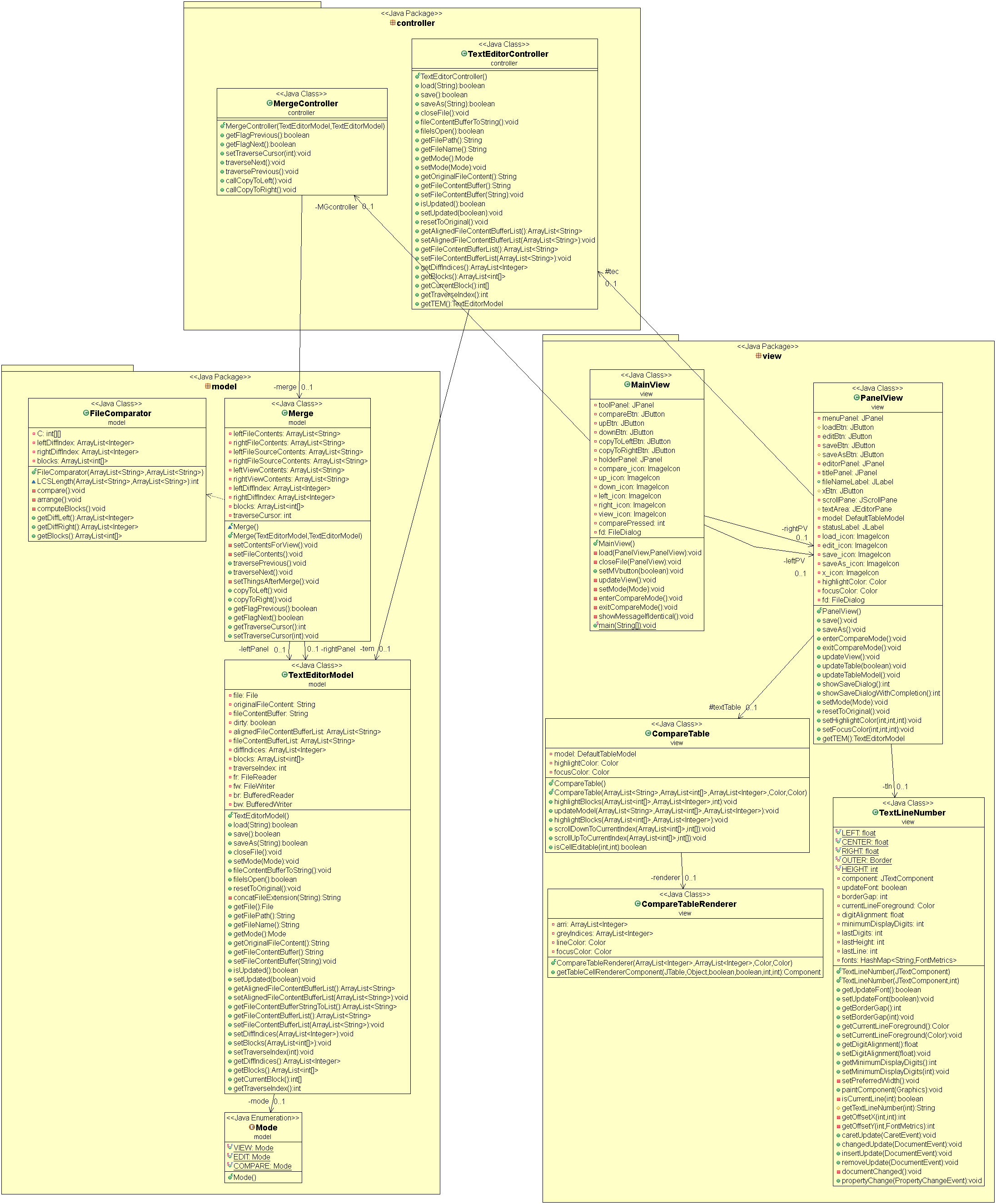
**손승표 (20162581) 윤신영 (20163657)**

1. **Domain model (class diagram)**

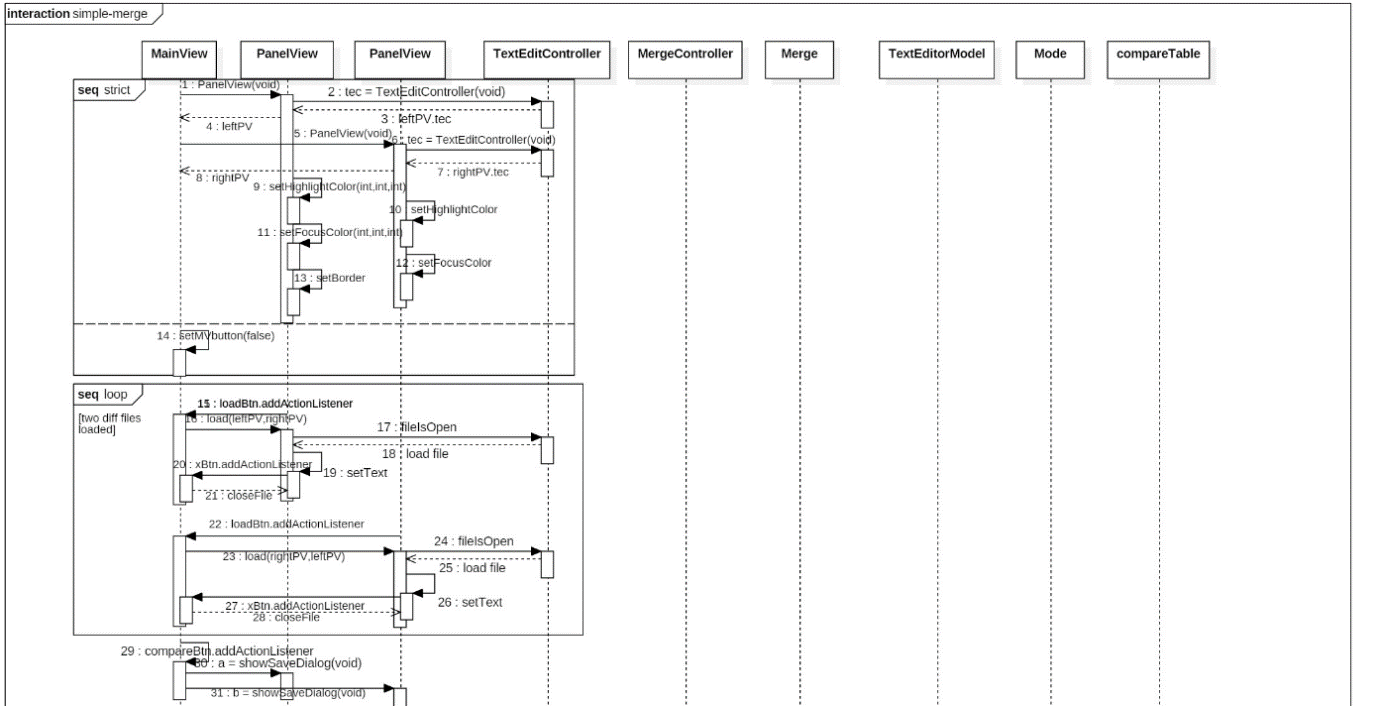


(you can see more clear diagram images at Github ‘Diagram’ folder!)

1. **Software architecture & Design model**



**[System sequence diagram]**

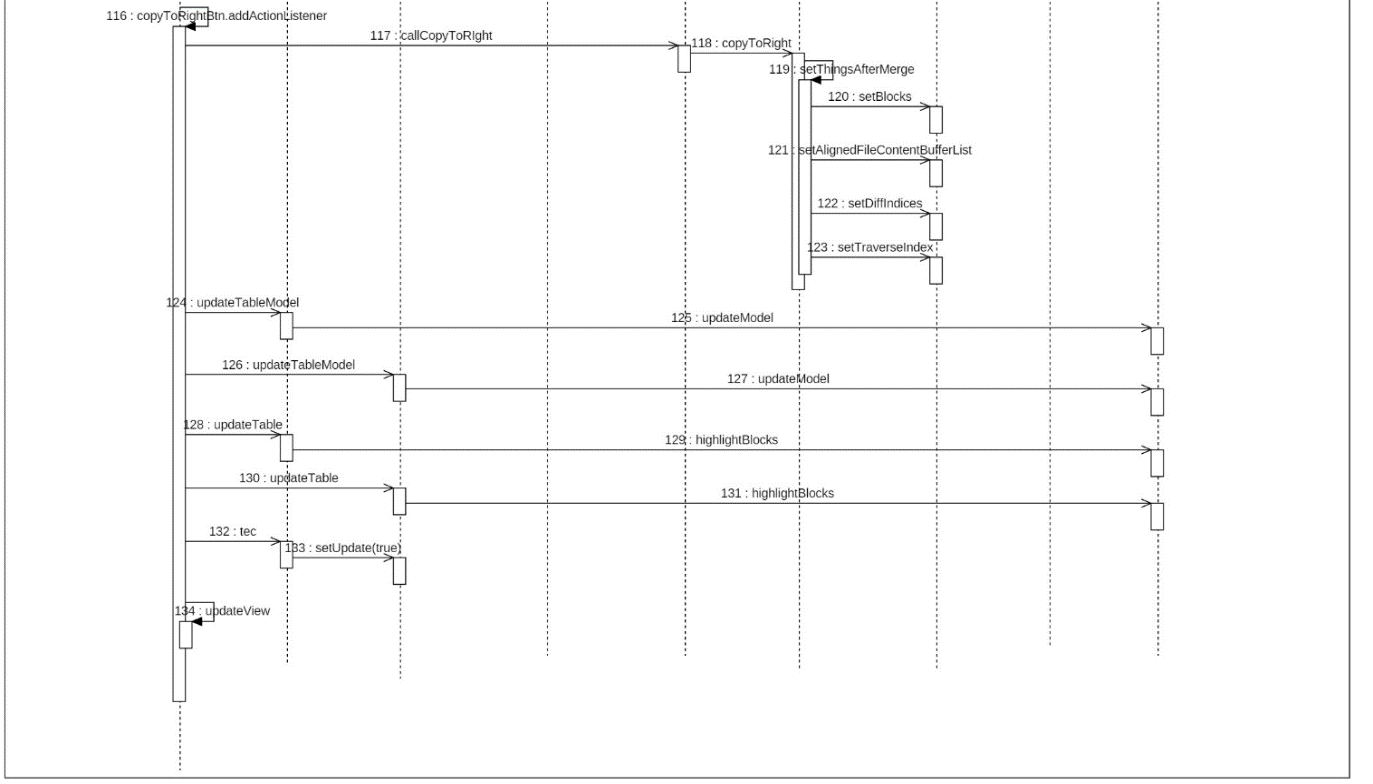


스크린샷이(가) 표시된 사진

높은 신뢰도로 생성된 설명텍스트이(가) 표시된 사진

매우 높은 신뢰도로 생성된 설명지도, 텍스트이(가) 표시된 사진

매우 높은 신뢰도로 생성된 설명지도이(가) 표시된 사진

높은 신뢰도로 생성된 설명

1. **Major design decision**

[Why the ActionListener of Load Button and X Button is in the MainView?]

Load and X Button is in PanelView but their ActionListener is in MainView. It’s because we could not load same file in both panel, so to prevent this action, ActionListener of Load Button is moved to MainView. The pre-condition of Compare Mode is that both Panel have file and it is in View Mode. X button is in MainView to prevent pressing Xbutton during Compare Mode.

[Why we modified return types in TextEditorModel?]

At first, load(), save(), saveAs() functions in TextEditorModel had void return type. But, since we had to check if that function is successful to show in the PanelView. PanelView should not change Modes or update view if expected function fails. So, we decided to change return type as boolean and checked its success.

[How we sent Compare results to TextEditorModel?]

We had a long discussion about how to send the compare results to the TextEditorModel(TEM) instance. We decided to give the right/left TEMs as parameters when creating the Merge instance. So after comparing two files, the Merge instace will set the fileContentBufferList, diffIndices, blocks, and traverseIndex in the TextEditorModel.

[How to compare the texts?]

By using the algorithms for longest common subsequence(LCS) problem and sequence alignment, we compute the matrix C which stores the LCS length and compare two panels based on the matrix.

[How to distinguish two cases? 1. Different strings are in same line index in each panel 2. The string is in only one panel]

For case 1, we multiply (-1) to the index of string. For case 2, the string is stored in the panel which the string is in and the blank value(0) is stored in another panel. This is for computing the start and end indices of blocks. When the getter function is called, returned data is the arraylist that all elements of left panel are changed to negative integer and all elements of right panel are changed to positive integer to distinguish two panel’s indices.

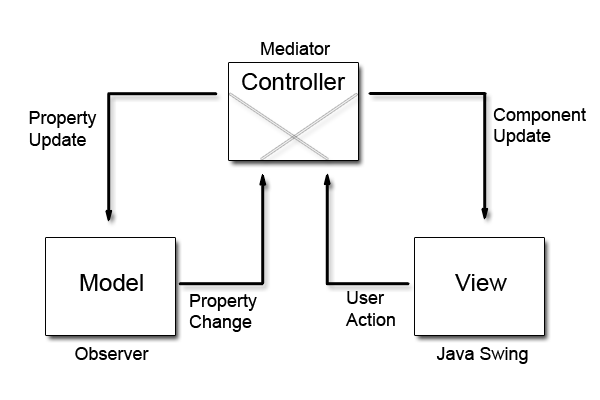
[What data type would be returned to Merge class after comparing in FileComparator class?]

In FileComparator class, the difference of two panel is stored in each integer type arraylist and the block, which store the block’s start and end index, is stored in integer array type arraylist. When merge call getter functions by FileComparator object, all elements in left panel’s arraylists are changed to negative integer, and all in right panel’s are changed to positive integer.

[What data type would be returned after calling traverse function?]

There is a cursor for traverse function which is an integer type index for the block array and it represents current location of cursor. If traverse function is called, the block array and the cursor are returned.

1. **Explanation about how MVC concept was applied**.



We define ‘view’ classes as {MainView, PanelView, TextLineNumber, CompareTable, CompareTableRenderer}, ‘model’ classes as {Merge, TextEditorModel, FileComparator, Mode}, and ‘controller’ as {MergeController, TextEditorController}.

As you can see these two class diagrams (domain model and design model), all the classes in view are not directly connected to any class in model. In other words, controller is the only mediator between model and view. Also, when state of model is changed, controller receives the signal and sends it to the view so that the view can apply the change and show it to the user.

So, concept of MVC model is applied to our program.

1. **Explanation about how OO design principles were applied (with code examples)**

[Encapsulation]

We set member variables as private or protected (there are only few exceptions), also we made public getter & setter methods to access those private or protected member variables. For some methods which are used only within in one class, we set them as private, otherwise, we set methods as private.

[Inheritance]

We use inheritance when we design ‘view’ classes. Here are the pictures of code.

스크린샷이(가) 표시된 사진

매우 높은 신뢰도로 생성된 설명

[Object-Oriented Programming]

We viewed the text files based on the concept of “objects” which contain data and actions, called “methods”. We modified the data such as file contents and mode, using methods. We designed the program by making the objects interact with each other. We could understand the interaction between program components than using procedural programming.

1. **Explanation about how our program was designed to be testable by Unit-test tools**

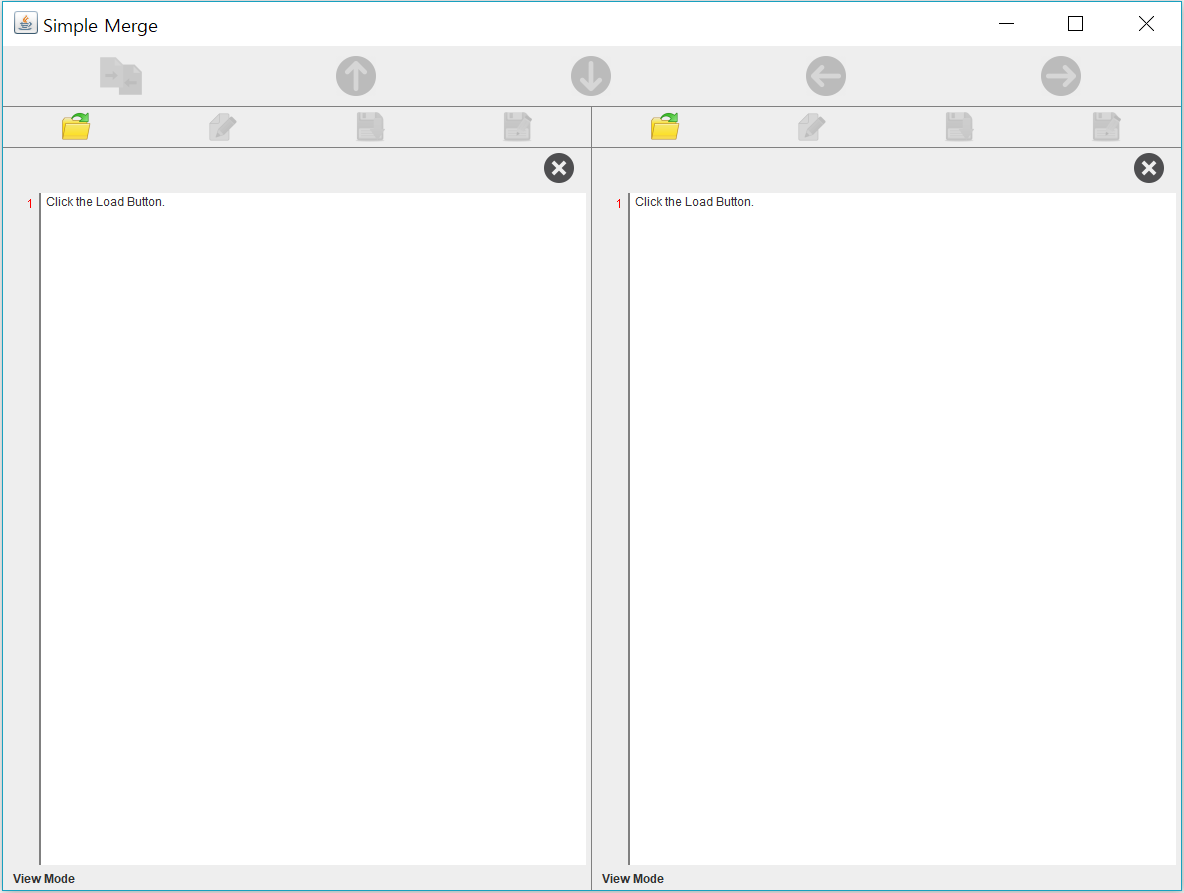
Unit-test tool을 사용하기 위해 test라는 package를 따로 만들고, 그 안에TextEditorModelTests.class, FileComparatorTests.class, MergeTests.class를 생성했다. Junit을 통해 각각의 메소드들을 테스트 할 수 있었다. Setup()을 통해 먼저 전제조건을 정해주고, assertEquals()를 이용해 기대한 값과 계산된 값을 비교해 test했다.

TextEditorModelTests.class 에서 load, save, save as, getFilePath, getFileName 등 파일을 제대로 불러오고 스트링으로 변환, 변경 후 save까지도 가능한지 test했다.

FileComparatorTests.class 에서는 두 개의 ArrayList<String>(이하 AL<S>)을 읽고 각각의 공통된 부분과 다른 부분을 고려해 ArrayList<Integer>를 만드는데 임의로 추가하는 공백의 값은 0으로 나머지는 AL<S> index로 저장했을 때의 값을 test했다.

MergeTests.class를 이용해, 각각의 다른 부분의 내용을 가리키는 block을 잘 traverse하는지 merge 또한 잘 해내는지를 test 했다.

1. **Usage of program & screen shots of examples**

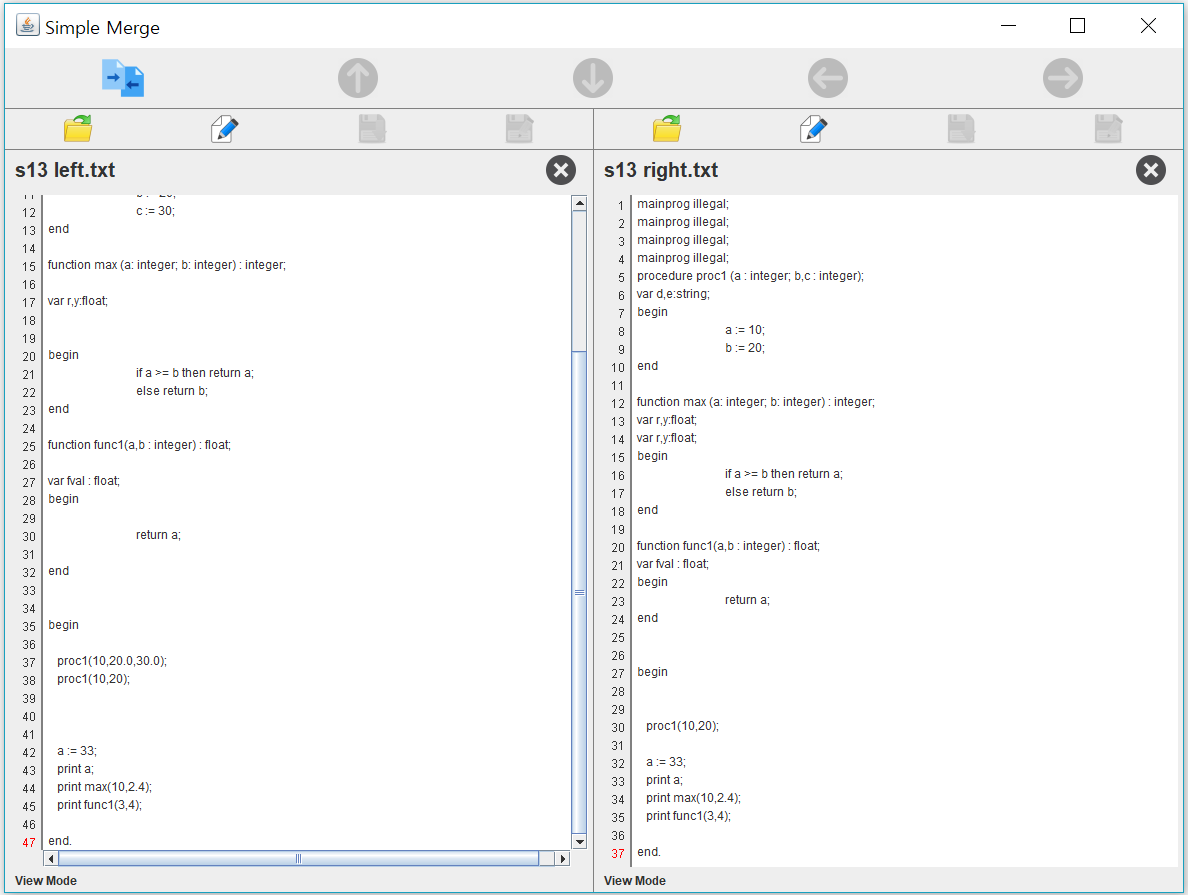


1. 프로그램의 시작화면. 활성화되어 있는 Load 버튼을 클릭하여 파일 load

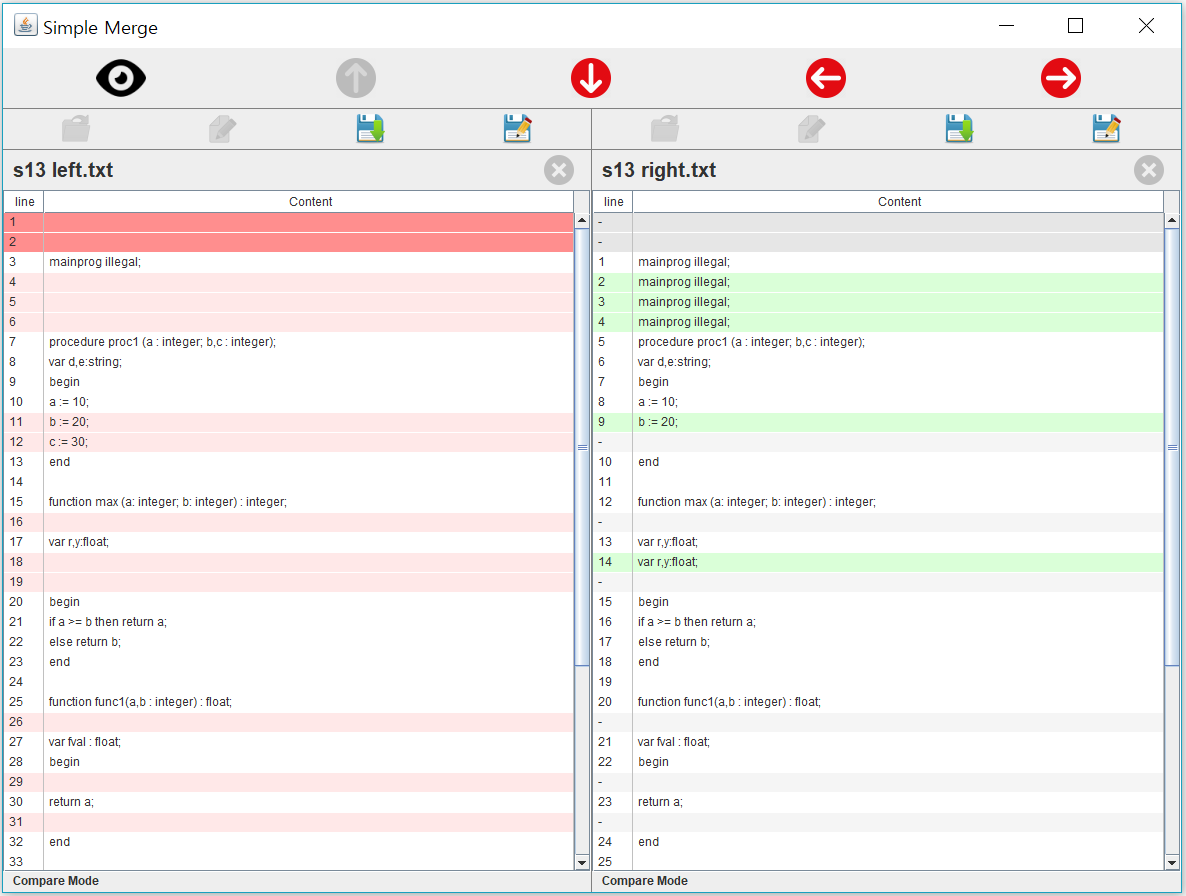
스크린샷, 컴퓨터이(가) 표시된 사진

매우 높은 신뢰도로 생성된 설명

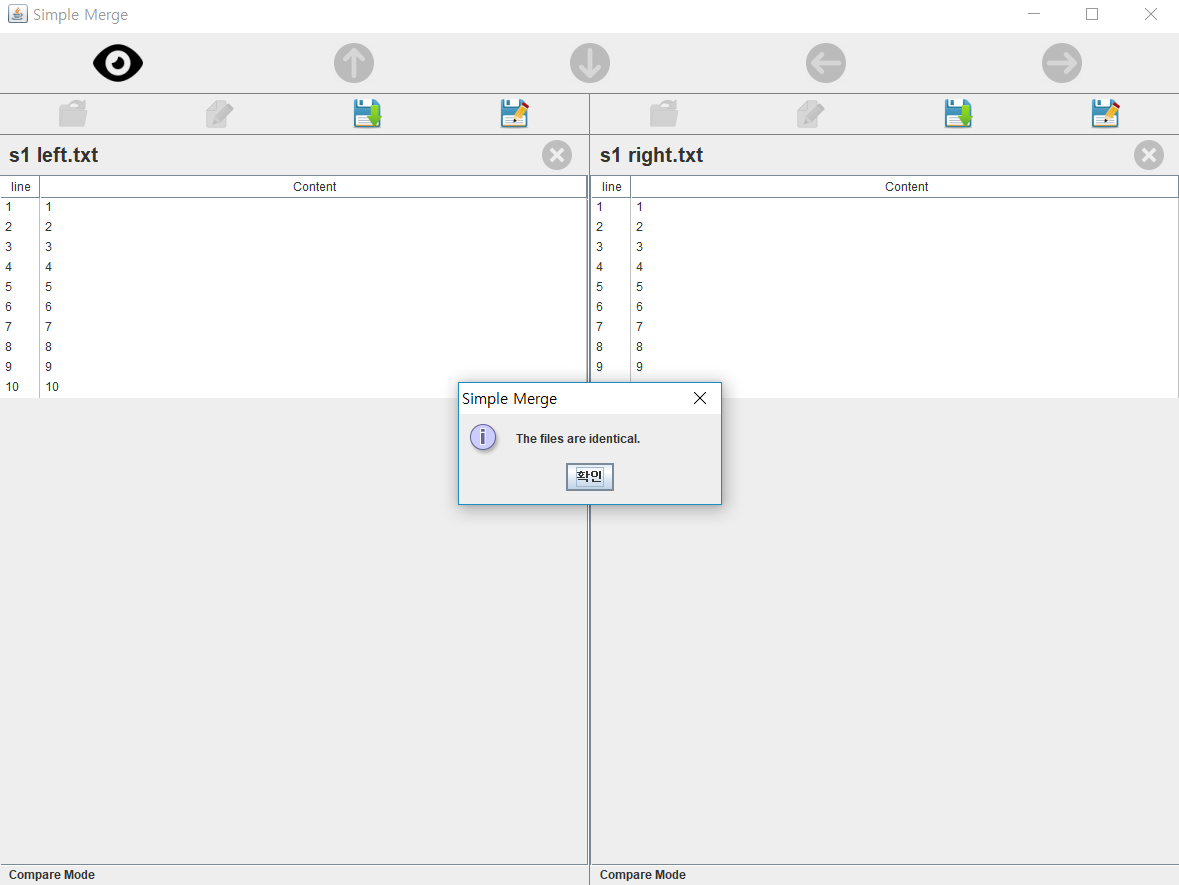
1-1. load한 파일이 중복인 경우 중복임을 알리는 message dialog 출력



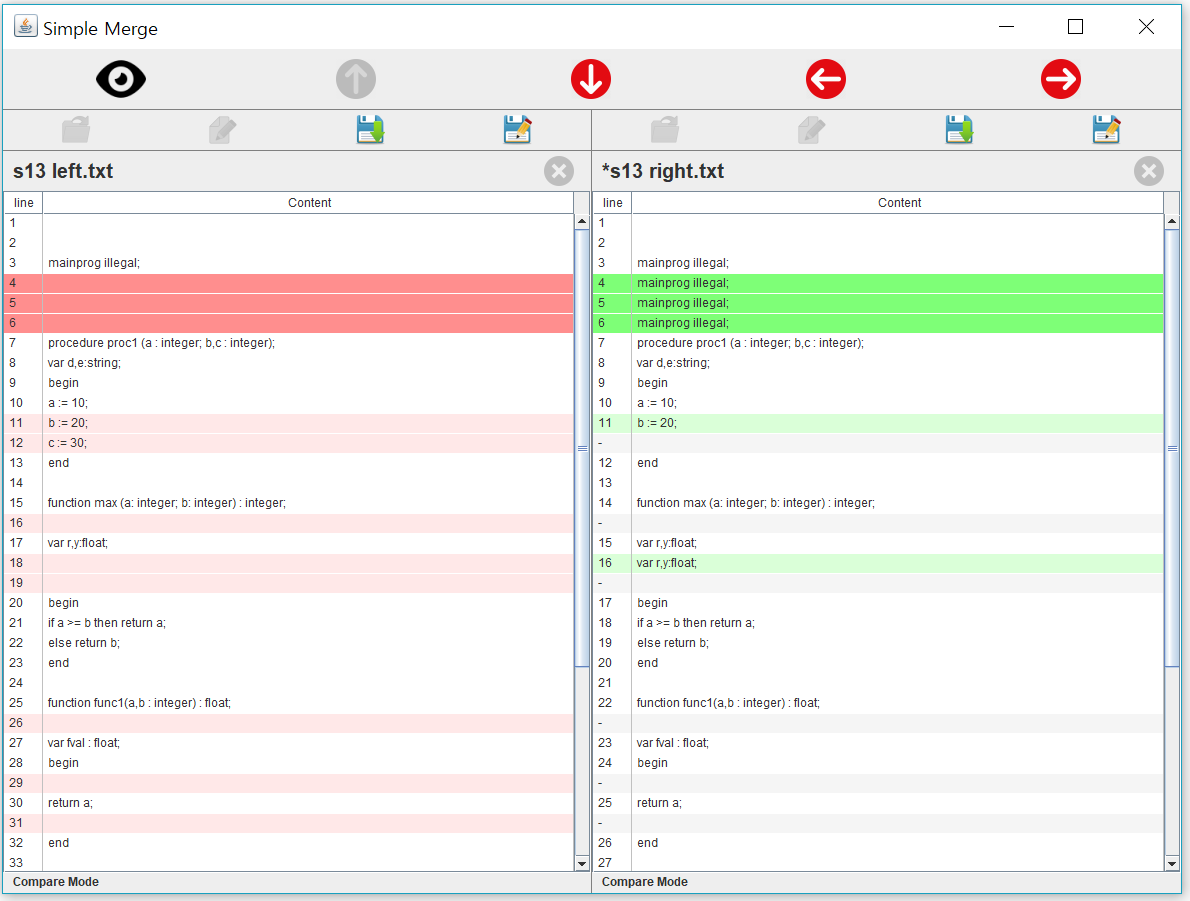
1. 파일이 Load된 상태. 패널의 밑부분에는 현재 패널의 모드가 표시되고(스크린 샷은 View Mode) 현재 커서가 있는 줄의 index는 빨간색으로 표시된다. Compare 버튼과 edit 버튼이 활성화되었다.



3. compare 버튼을 클릭한 후, Traverse와 Merge 버튼이 활성화되었다. 줄 정렬을 위해 추가된 공백줄은 회색으로 표시되고, index는 ‘-‘로 표시됨. 더 이상 traverse할 수 있는 블록이 없으면 해당 버튼 비활성화.

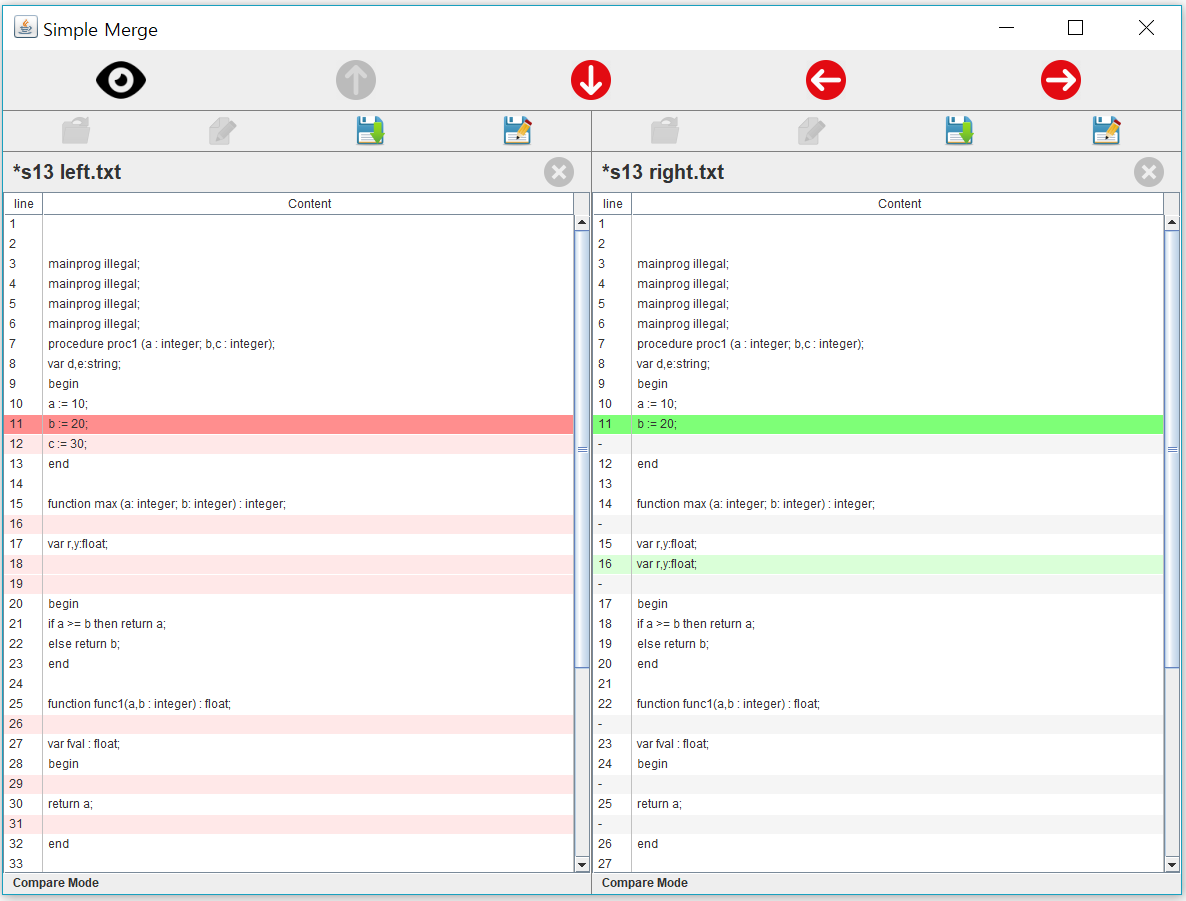


3-1. compare 연산 결과 두 텍스트파일이 동일한 경우 이를 알리는 message dialog 출력

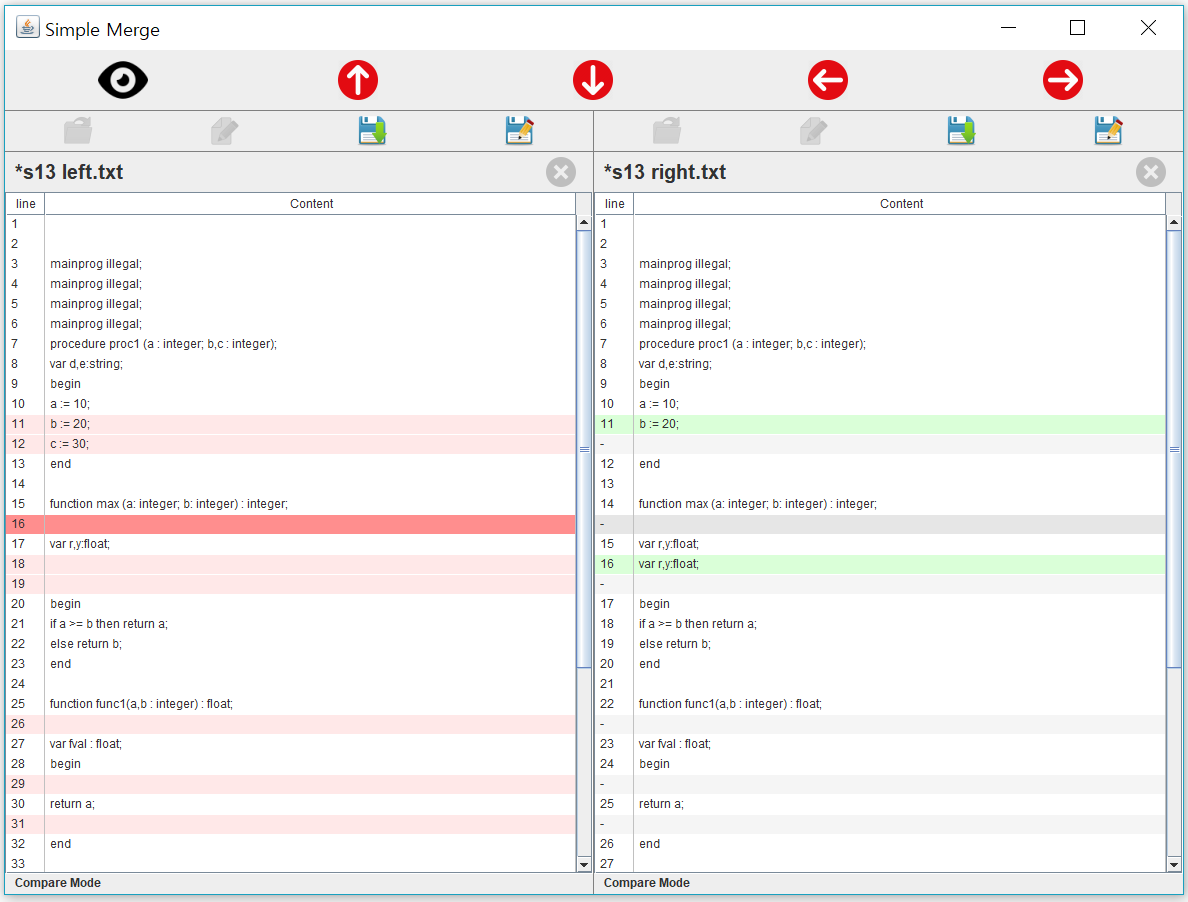


4. Copy to right 버튼을 한 번 클릭한 결과

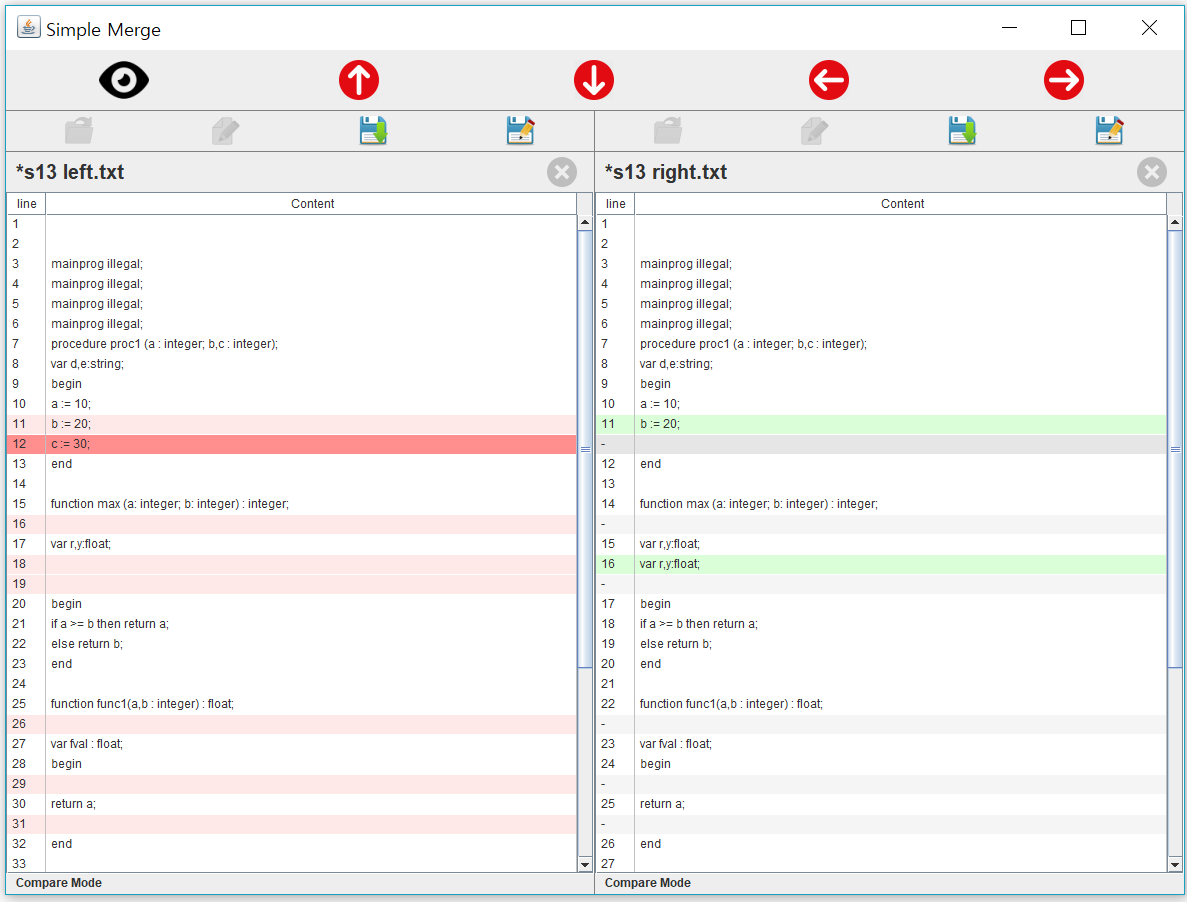
오른쪽 파일이 변경되어 파일 이름 앞에 \*표시가 붙는다.



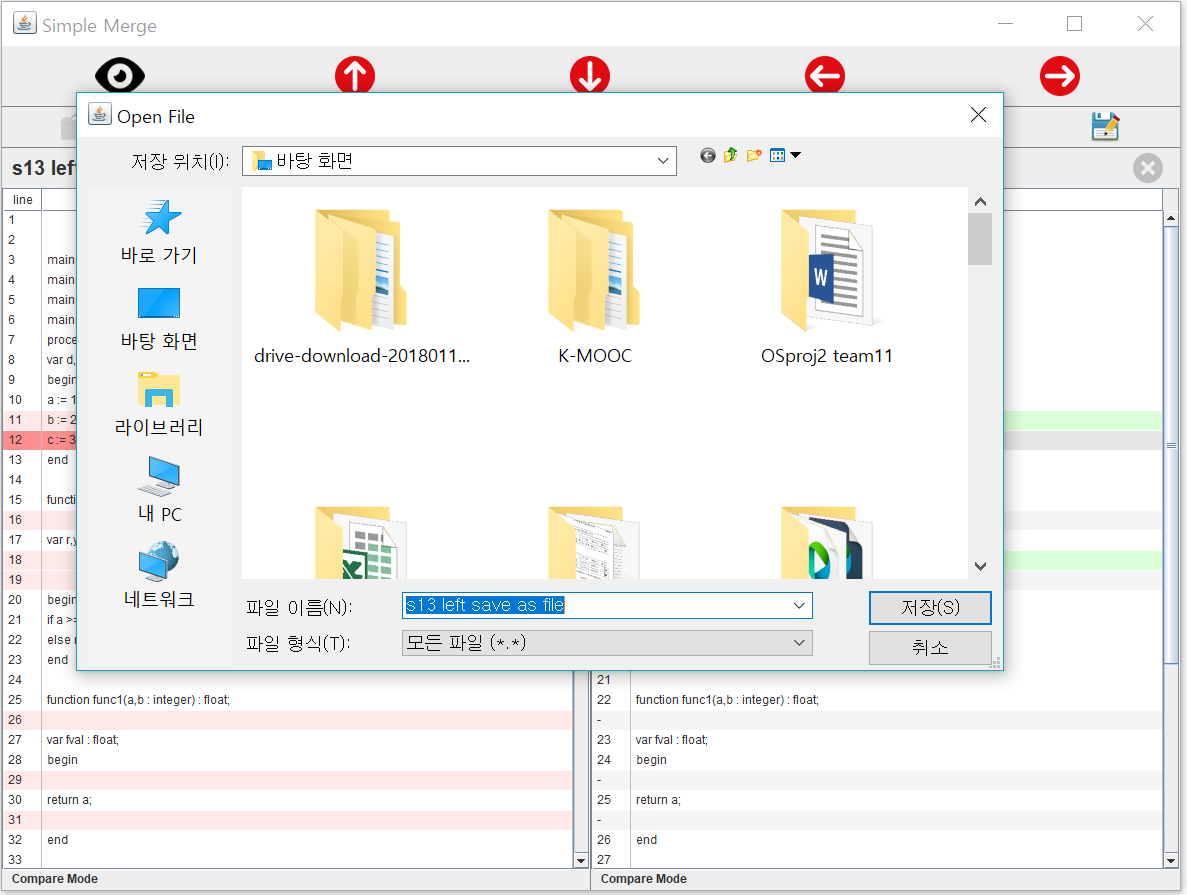
5. Copy to left 버튼을 한 번 클릭한 결과. 왼쪽 파일이 변경되어 파일 이름 앞에 \*표시가 붙는다. (현재 커서가 있는 곳은 b:=20;으로 둘 다 동일해 보이지만 한 쪽에 tab이 있음)

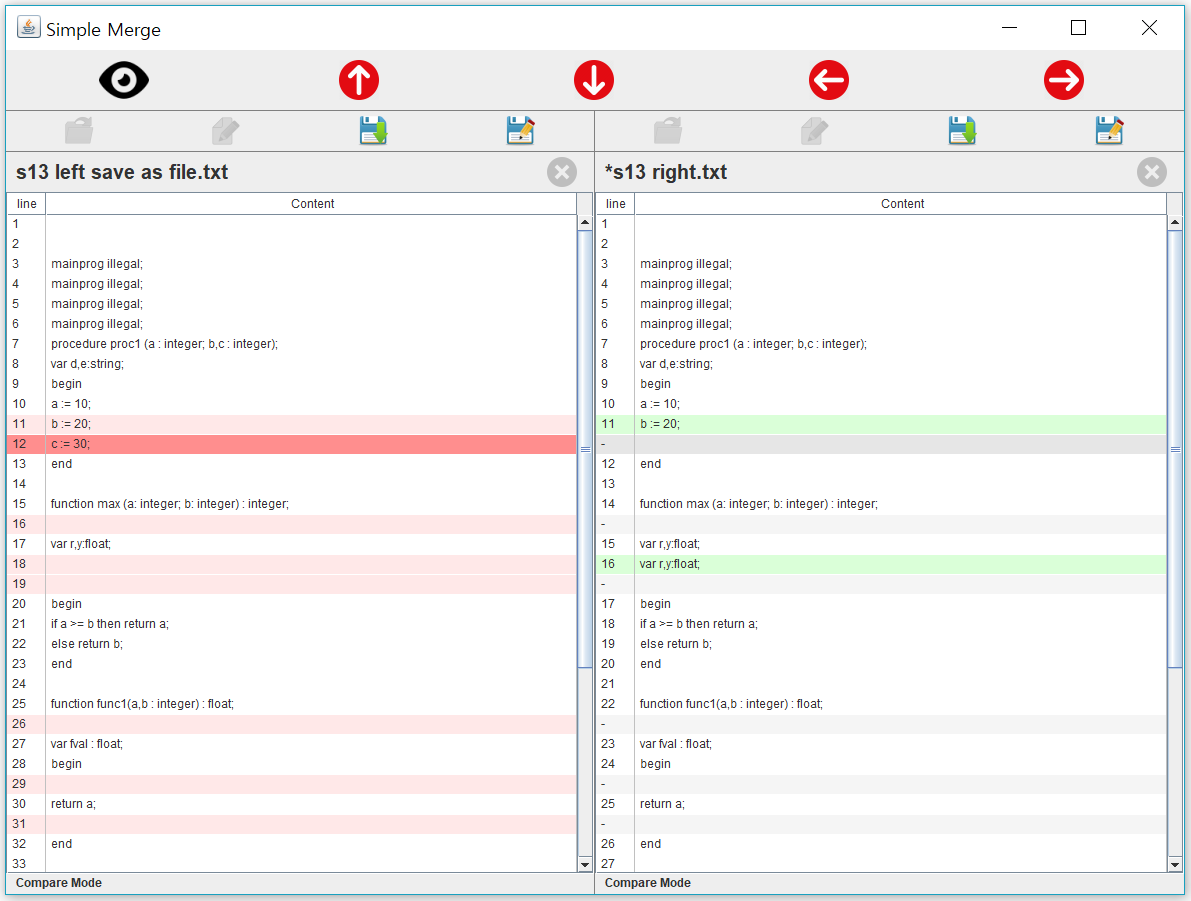


6. traverse next 버튼을 두 번 클릭한 결과

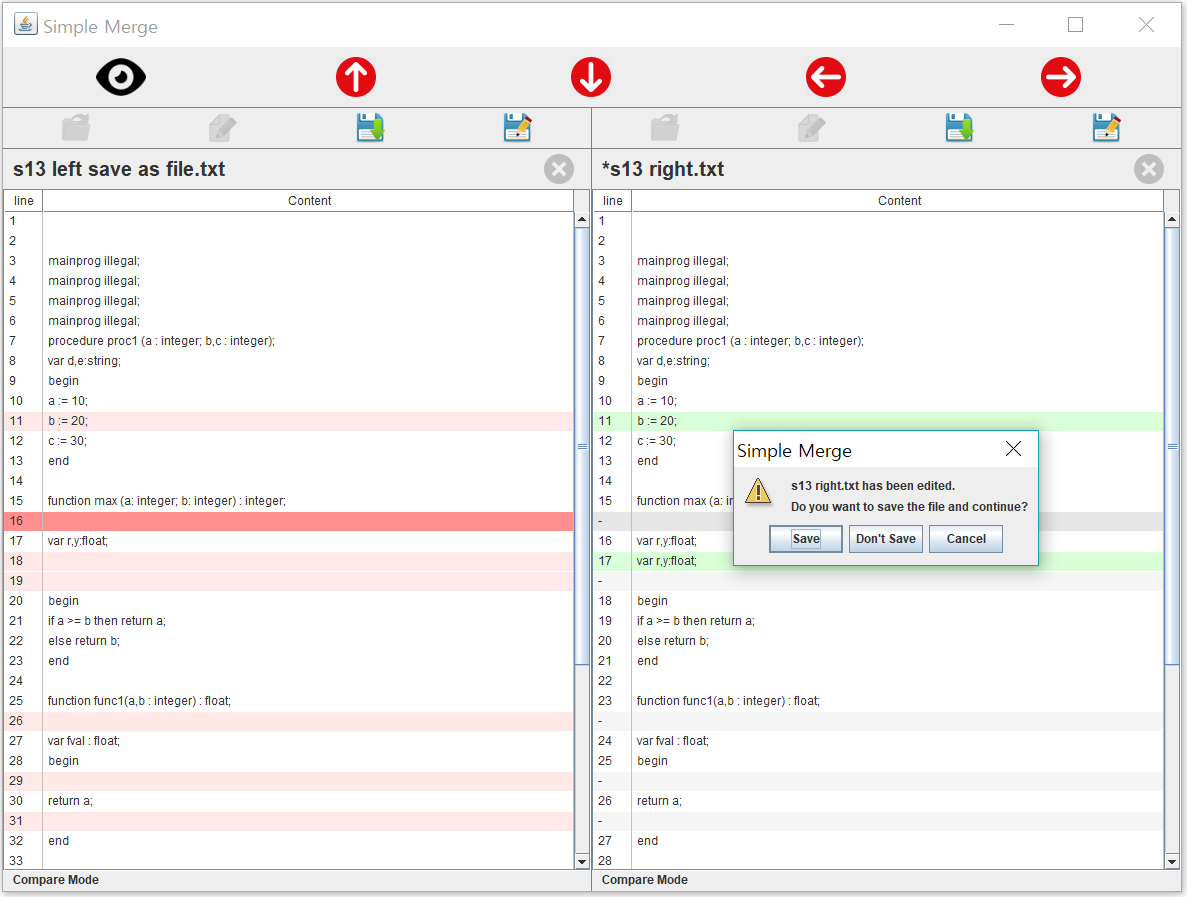


7. traverse previous 버튼을 한 번 클릭한 결과. 여기서 save 버튼을 클릭하면 해당 파일에 변경된 내용이 save된다.

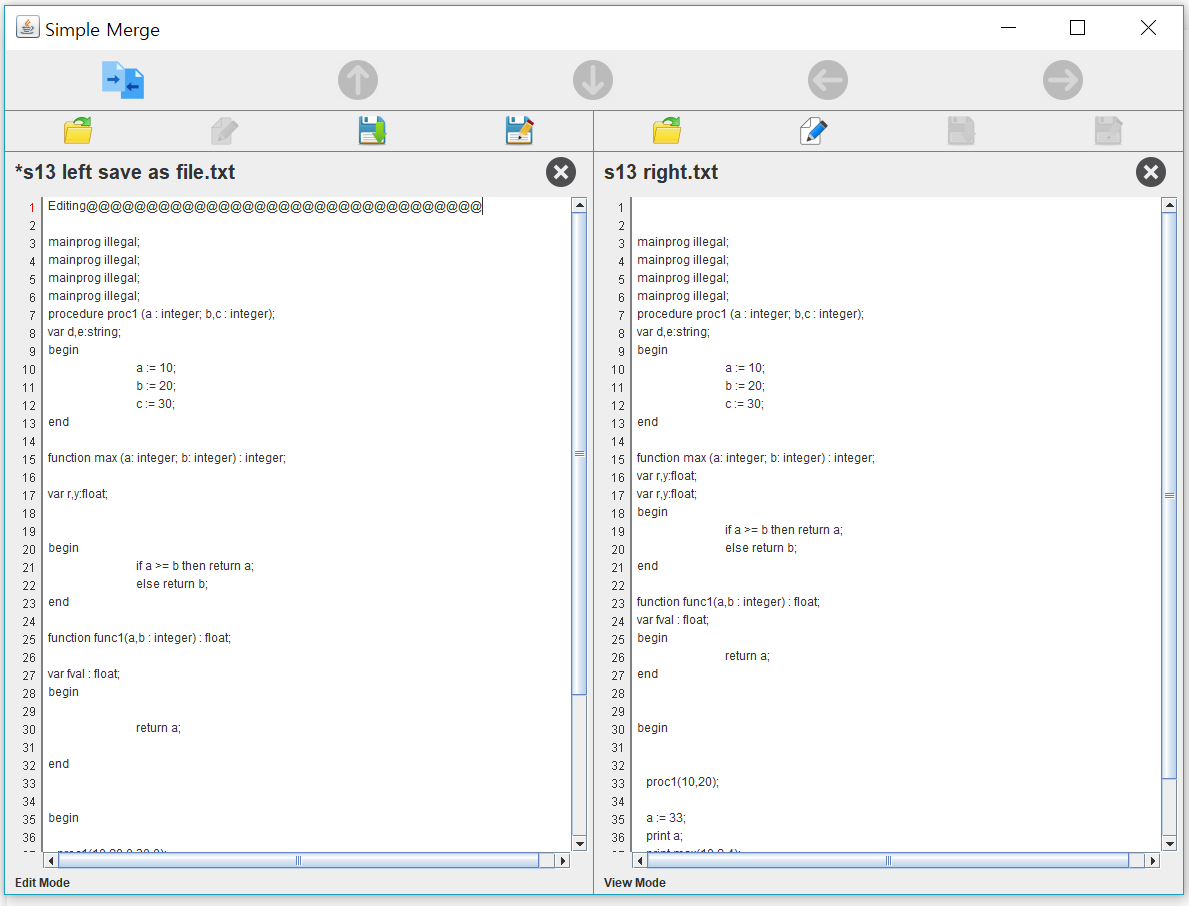




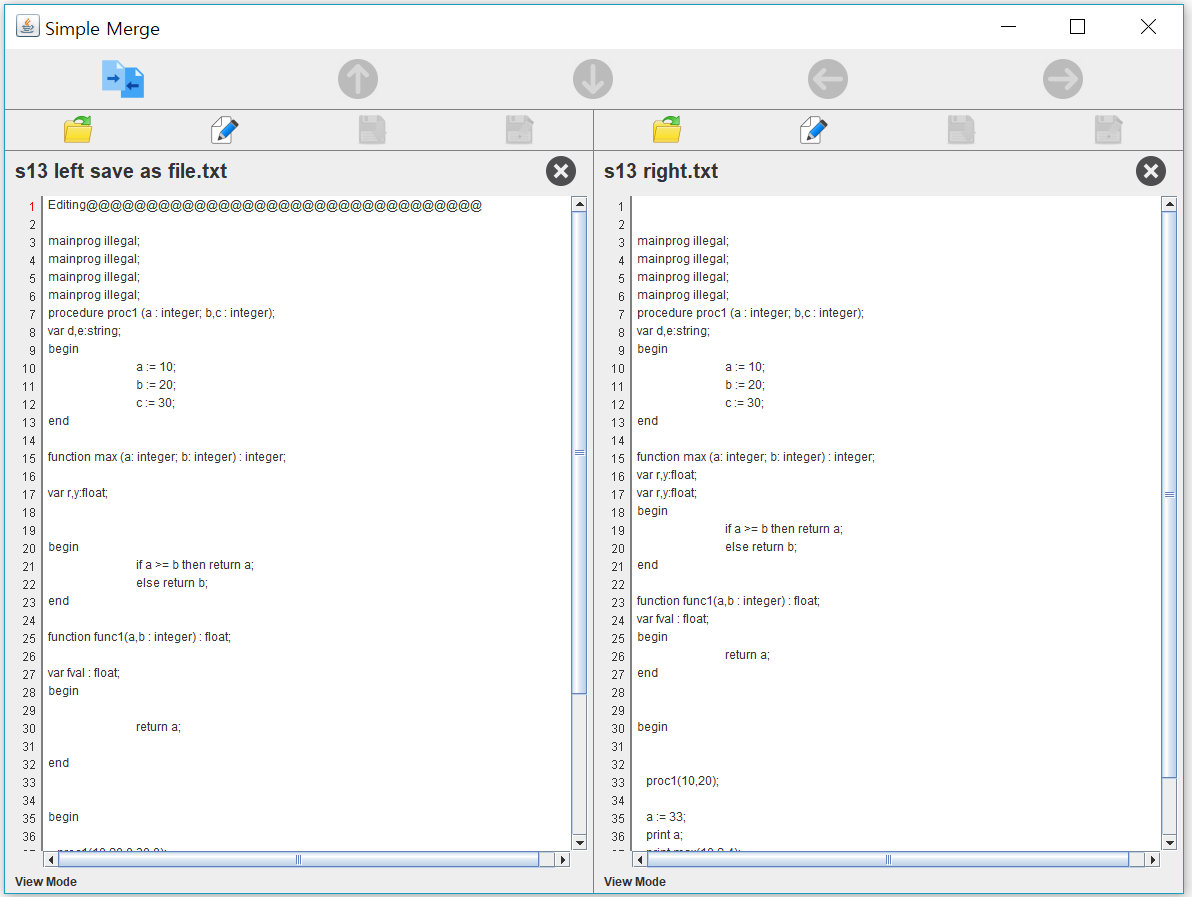
8. 왼쪽 파일을 save as로 저장한 결과. Save as로 파일을 저장한 이후에는 해당 패널의 파일이 save as로 저장한 파일로 대체된다. 이 때 파일의 확장자를 지정해주지 않으면 기존 파일의 확장자로 저장된다.



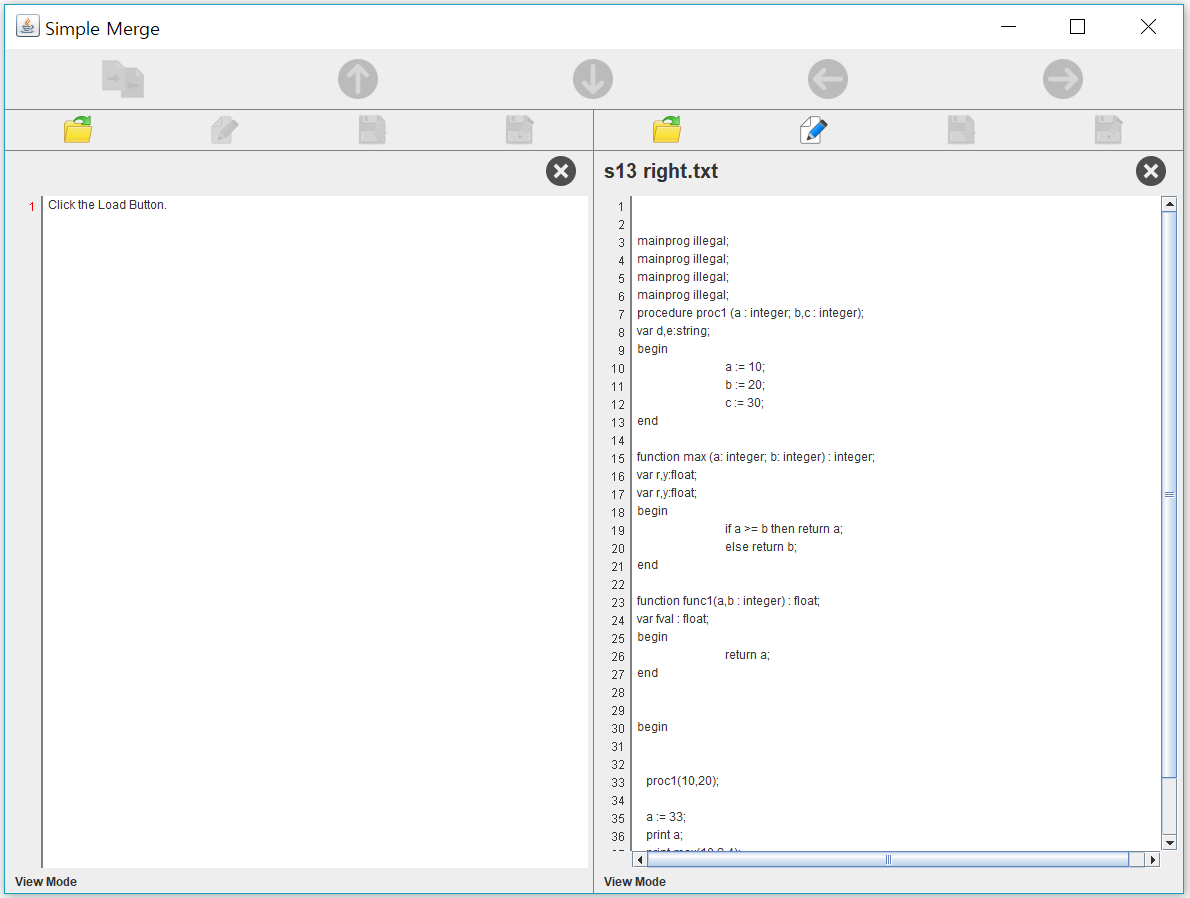
9. 저장 이후 copy to right를 한 번 더 실행하여 오른쪽 파일을 변경시켰다. 이후 왼쪽 위의 escape compare mode 버튼을 누르면 변경사항을 저장할 것인지 묻은 message dialog가 출력된다.



10. compare모드에서 view모드로 탈출하고, 왼쪽 패널의 edit 버튼을 누른 상태. 왼쪽 밑의 mode 상태를 보면 왼쪽 패널이 edit mode가 되어있음을 알 수 있다.



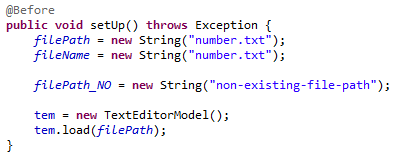
11. save 버튼을 누르면 자동으로 edit 모드를 탈출하여 view mode로 전환된다.



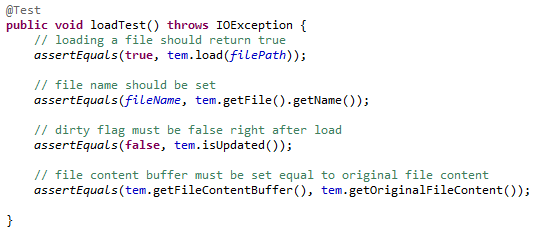
12. ‘X’ 버튼을 누르면 파일이 닫힌다.

1. **Functional unit test cases and results**

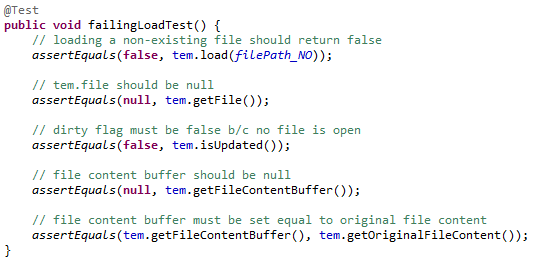
* setUp – test case로 사용할 파일의 경로와 이름을 설정하고, 테스트 하고자 하는 클래스의 객체를 만든다.



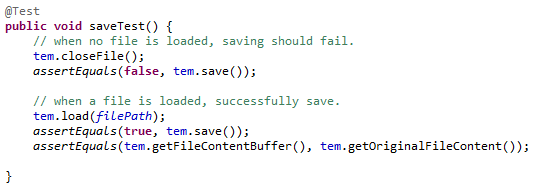
* load() 함수를 테스트한다. 파일을 성공적으로 불러오면 true를 반환하며, 파일 이름, dirty flag, 그리고 파일의 내용이 알맞게 설정됐는지 확인한다.



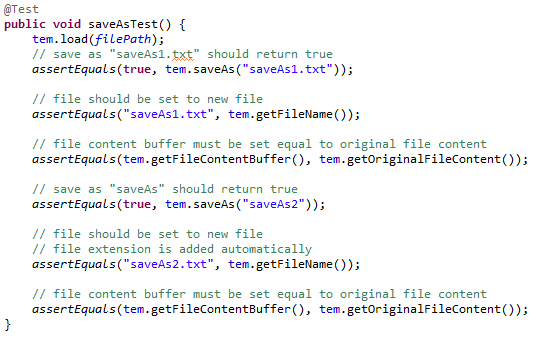
* load() 함수가 실패하는 경우를 테스트한다. 존재하지 않는 파일을 불러오려고 하였을 때 false를 반환하고, 파일의 이름과 내용이 적절하게 null로 설정됐는지 확인한다.



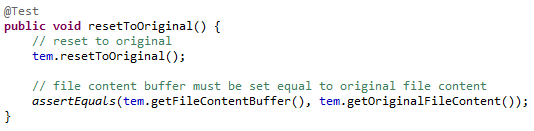
* save() 함수를 테스트한다. 파일이 열려 있지 않은 경우에는 false를 리턴하고, 그렇지 않으면 저장하고 true를 리턴한다. fileContentBuffer가 잘 설정되었는지 확인한다.



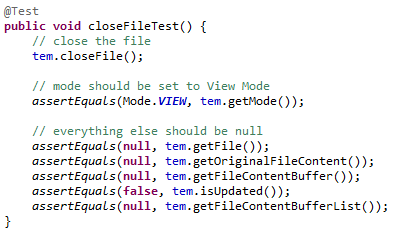
* saveAs() 함수를 테스트한다. Save()함수와 같은 방식으로 테스트하며, 저장 시 파일 확장자를 입력하지 않은 경우 열려 있던 파일의 확장자를 자동으로 붙여 주는지 확인한다.



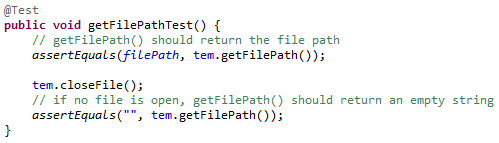
* resetToOriginal() 함수를 테스트한다. 함수가 호출되었을 때 fileContentBuffer가 originalFileConten로 set 되었는지 확인한다.



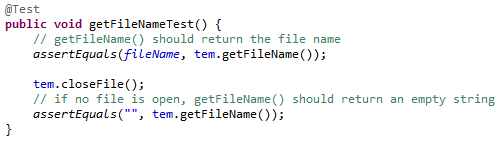
* closeFile() 함수를 테스트한다. 파일이 닫혔을 때 모드가 제대로 설정 되었는 지 확인하고, 나머지 필드들이 null로 지정되었는지 확인한다.



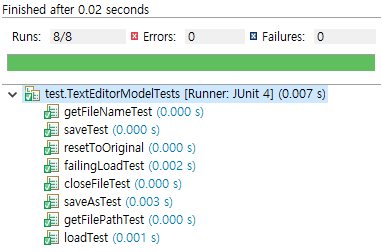
* getFilePath() 함수를 확인한다. 파일이 없을 때 null을 반환하지 않는지 테스트한다..

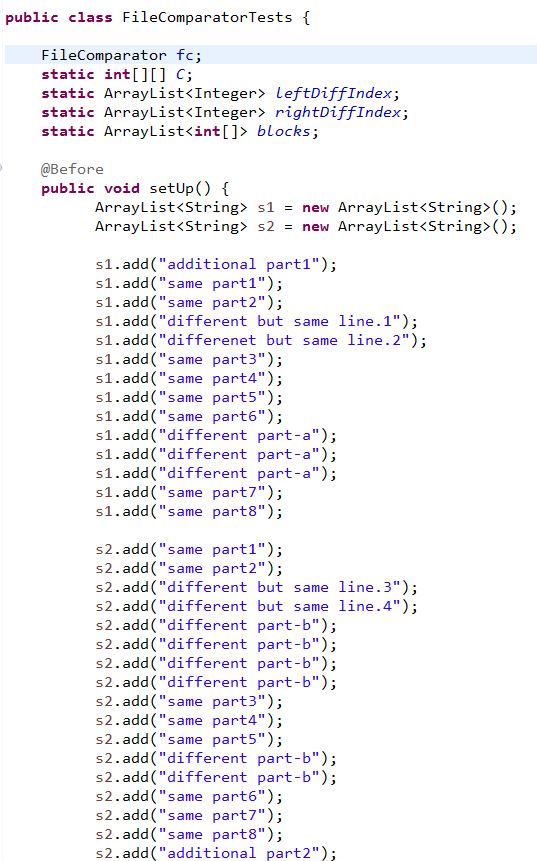


* getFileName() 함수를 확인한다. 파일이 없을 때 null을 반환하지 않는지 테스트한다.



* 테스트 결과 모든 테스트가 성공적으로 종료되었다.

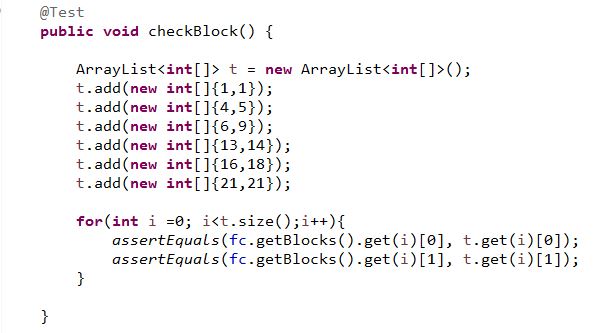




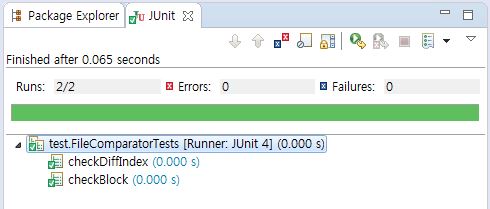
* FileComparatorTests 에서는 앞서 2-(4)에서 말했듯이 두 개의 AL<S>를 받아 AL<Integer>로 리턴 시키고 기대 값과 비교하는 test를 진행했다.



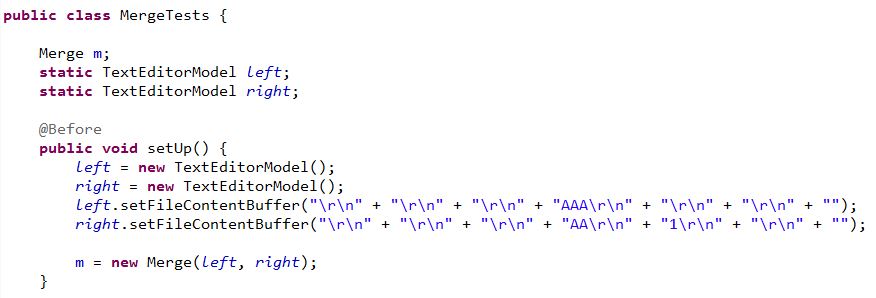
* 두 개의 내용에서 다른 부분과 공통된 부분을 미리 계산하여 그 값을 비교했다.



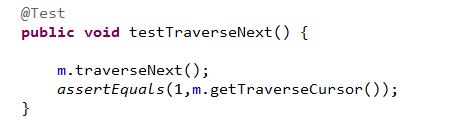
* 저 부분에서 서로 다른 내용을 가리키는 block의 변수 또한 값을 비교했다.



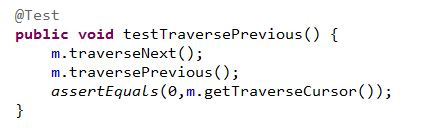
* 이상 없었을 때까지 코드를 수정했다. (본 예제에서는 이상 없음.)

)

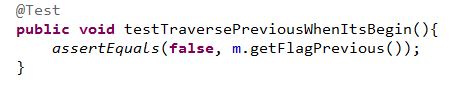
* MergeTests 에서는 traverse 와 merge가 잘 작동하는지 테스트했다. 이 또한 두 개의 텍스트를 setup으로 setting 해주었다. 위 상태에서의 block은 ( {4,4},{5,5},{7,7} )가 된다.
* 초반 cursor index는 0으로 block이라는 이름을 가진 ArrayList<int[]>의 index이다.



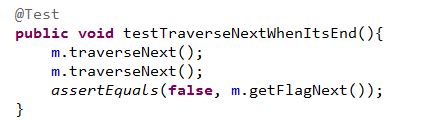
* traverseNext를 하게되면 0에서 +1 증가한 1을 가리키게 된다.



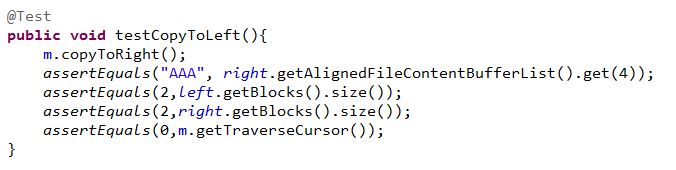
* traverseNext를 하고 다시 traversePrevious를 하면 +1되었다 다시 -1 되었기에 처음 커서값 0으로 그대로이다.



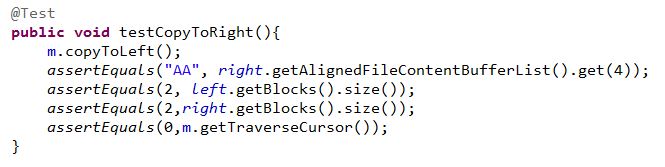
* 현 Cursor값이 0이므로 previous로 둘러볼 block이 없어서 FlagPrevious의 값은 false이어야 한다.



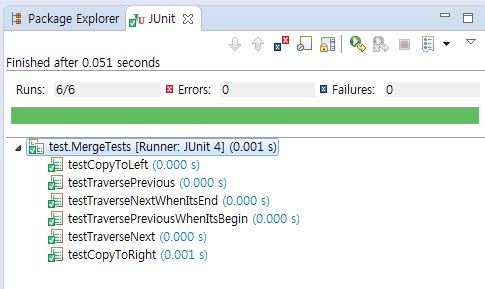
* Block의 최대 인덱스가 2이고, cursor을 2까지 옮겨주면 다음으로 둘러볼 block이 없기에 FlagNext는 false가 되어야 한다.



* 위 string에서 4번째 문장을 copytoRight을 하면 right의 4번째 문장이 AA에서 AAA로 바뀌어있어야 한다. 또한 left와 right의 Block(둘이 항상 같은 값을 가지고 있다.) 사이즈는 합쳐졌기에 -1되어야하고, 커서는 0번 index에서 merge했기에 그대로 0번에 머물러야 한다.



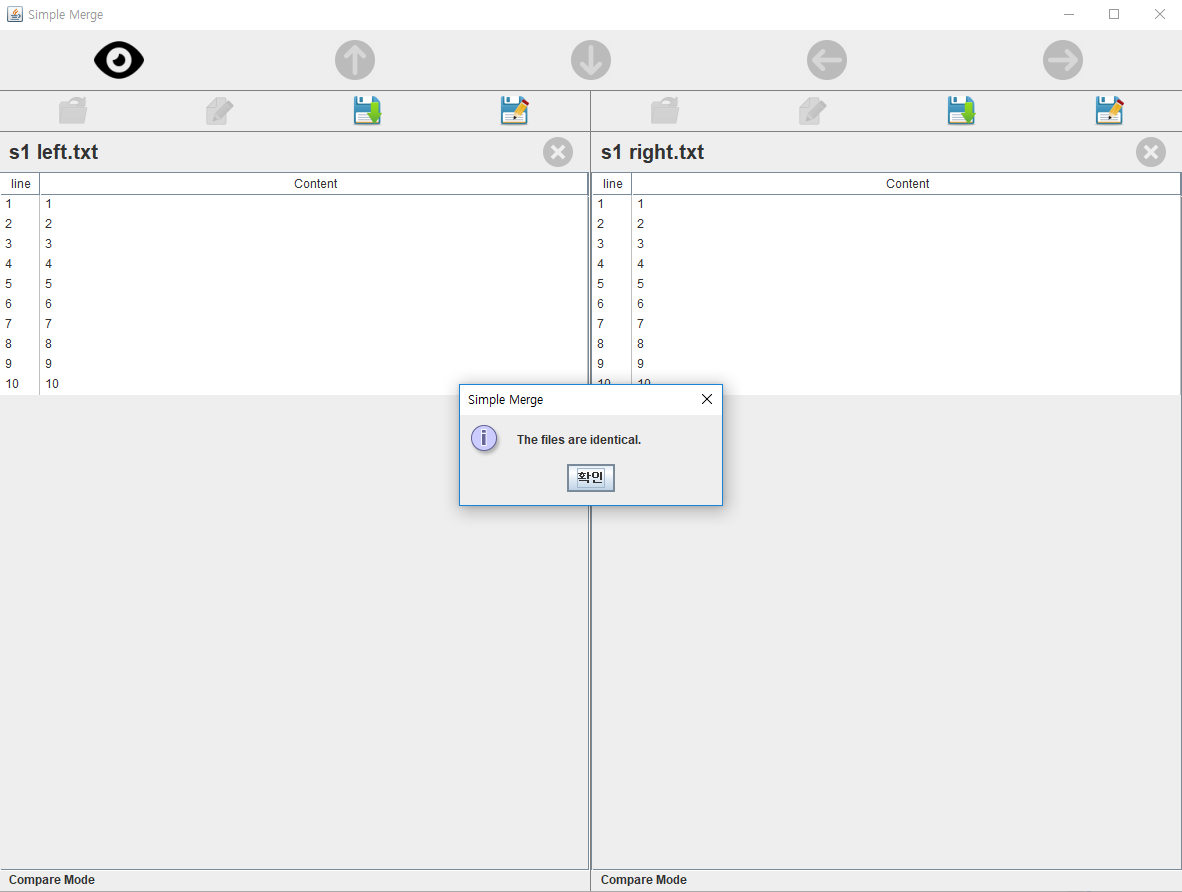
* 위 string에서 4번째 문장을 copytoLeft을 하면 leftt의 4번째 문장이 AAA에서 AA로 바뀌어 있어야 한다. 또한 left와 right의 Block(둘이 항상 같은 값을 가지고 있다.) 사이즈는 합쳐졌기에 -1되어야하고, 커서는 0번 index에서 merge했기에 그대로 0번에 머물러야 한다.



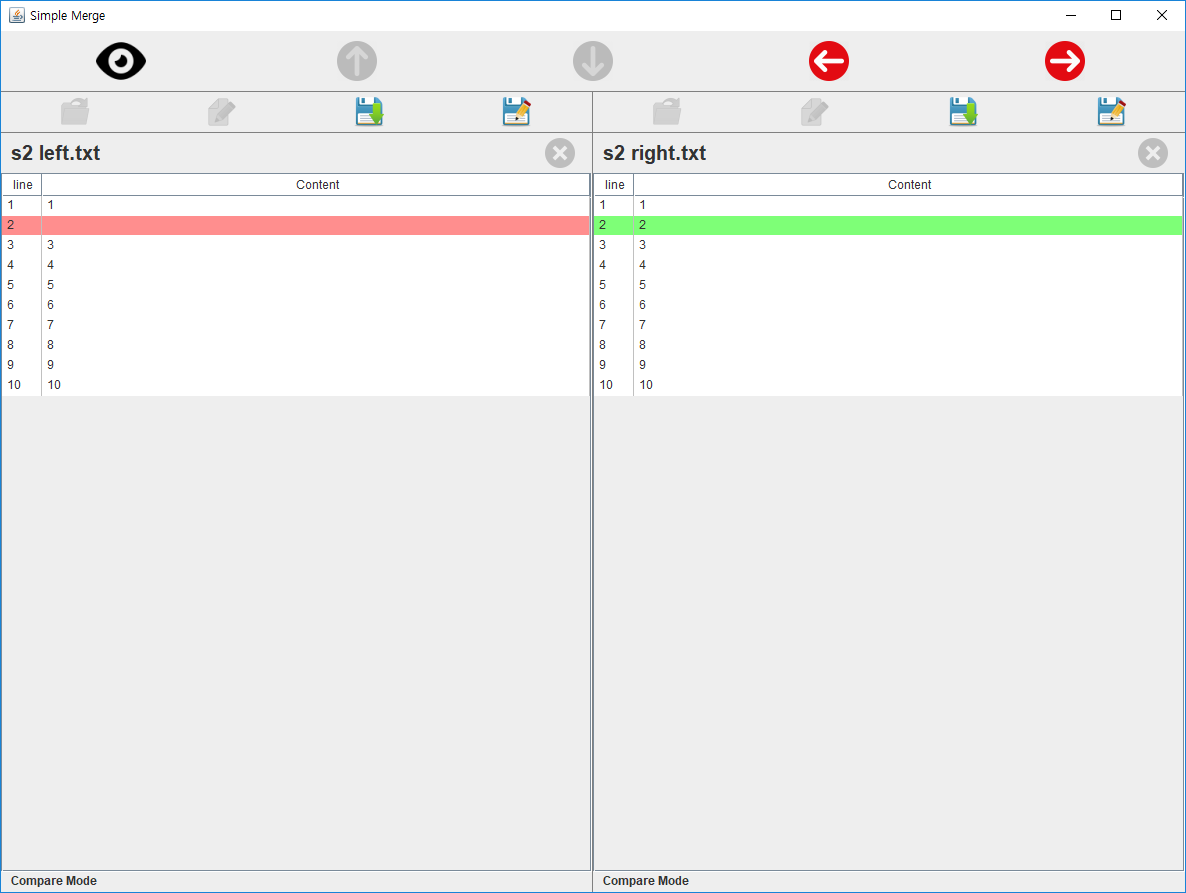
우리가 생각했던 기대값과 계산된 값이 일치할 때까지 test했다. 결국 test를 모두 통과했다.

1. **System test cases and results**

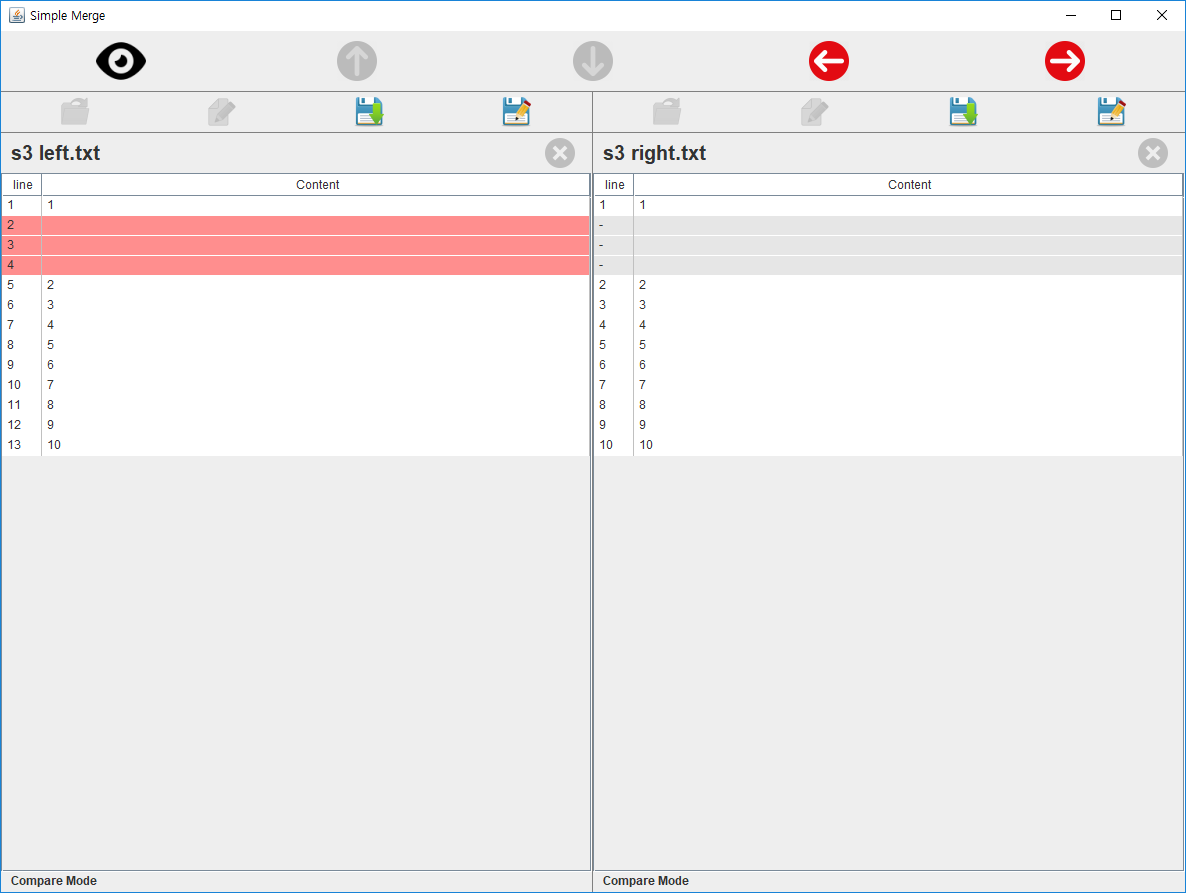
**S1. Result :** 두 파일이 동일함을 보여준다.



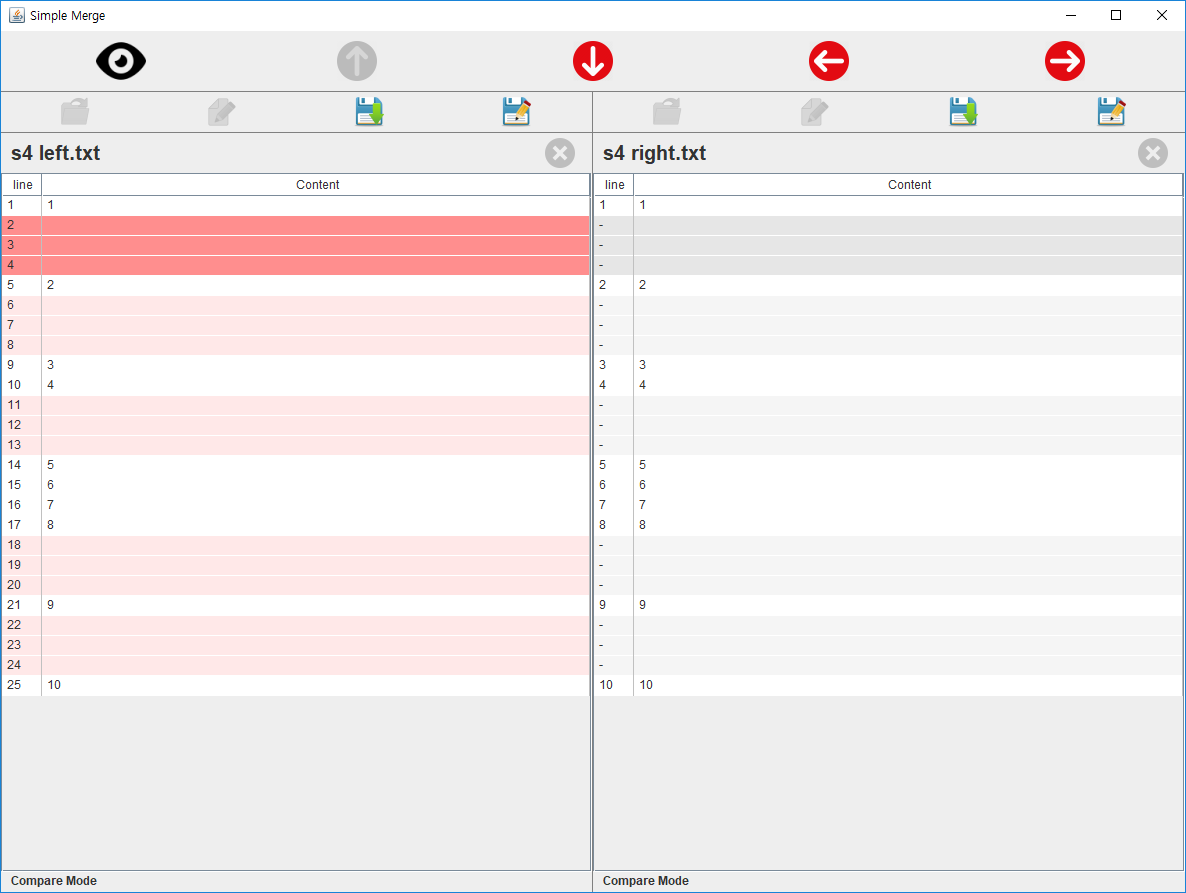
**S2. Result :** 왼쪽 파일에 2가 없음이 나타난다.



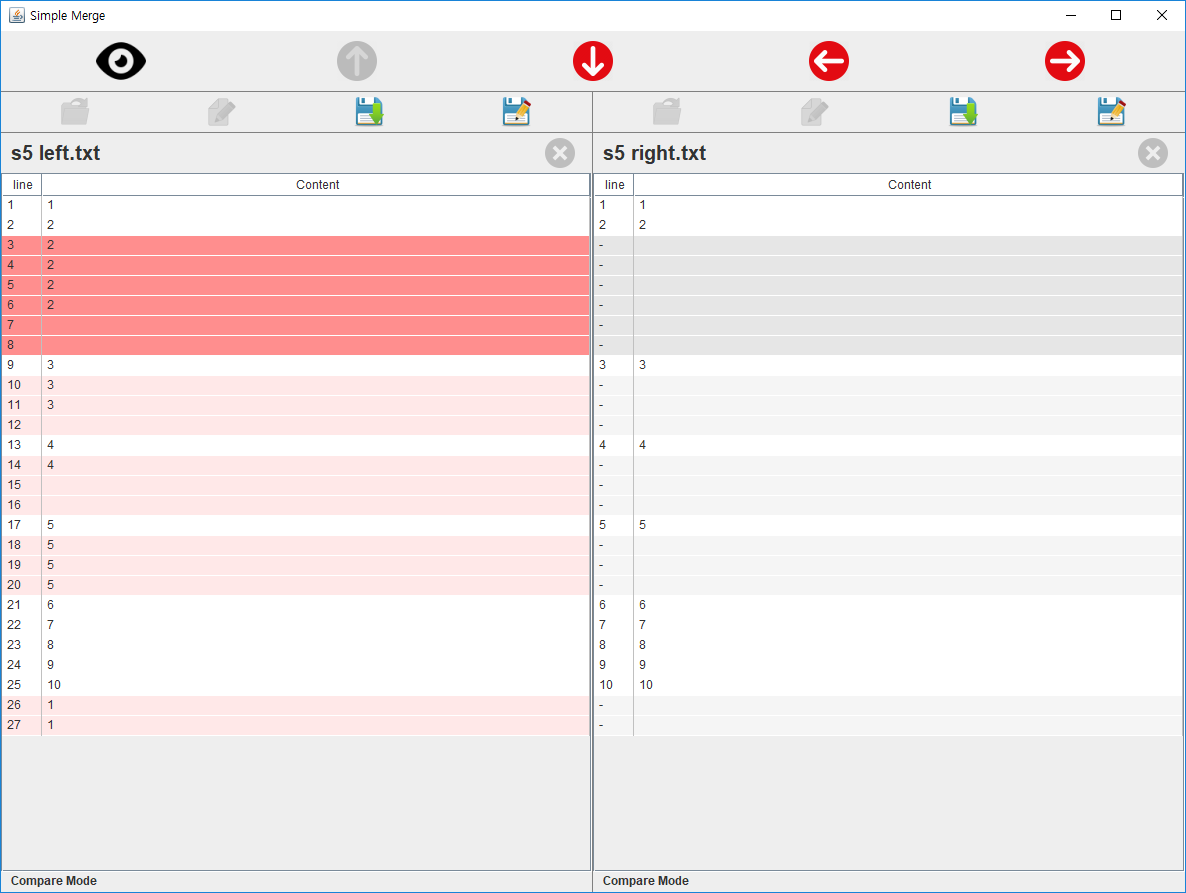
**S3. Result :** 왼쪽 파일에 빈줄이 추가되어 있음이 나타난다.



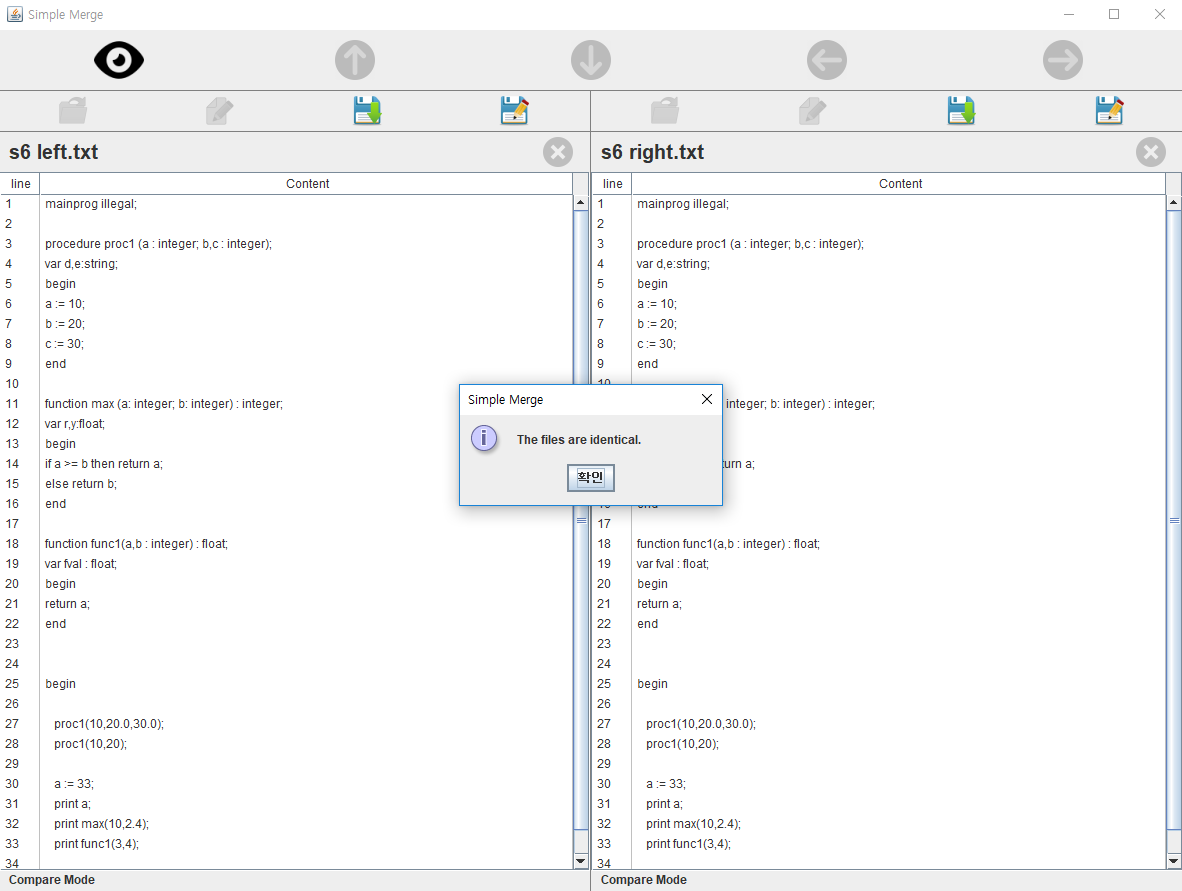
**S4. Result :** 여러 개의 빈 줄 블록이 있음이 나타난다.



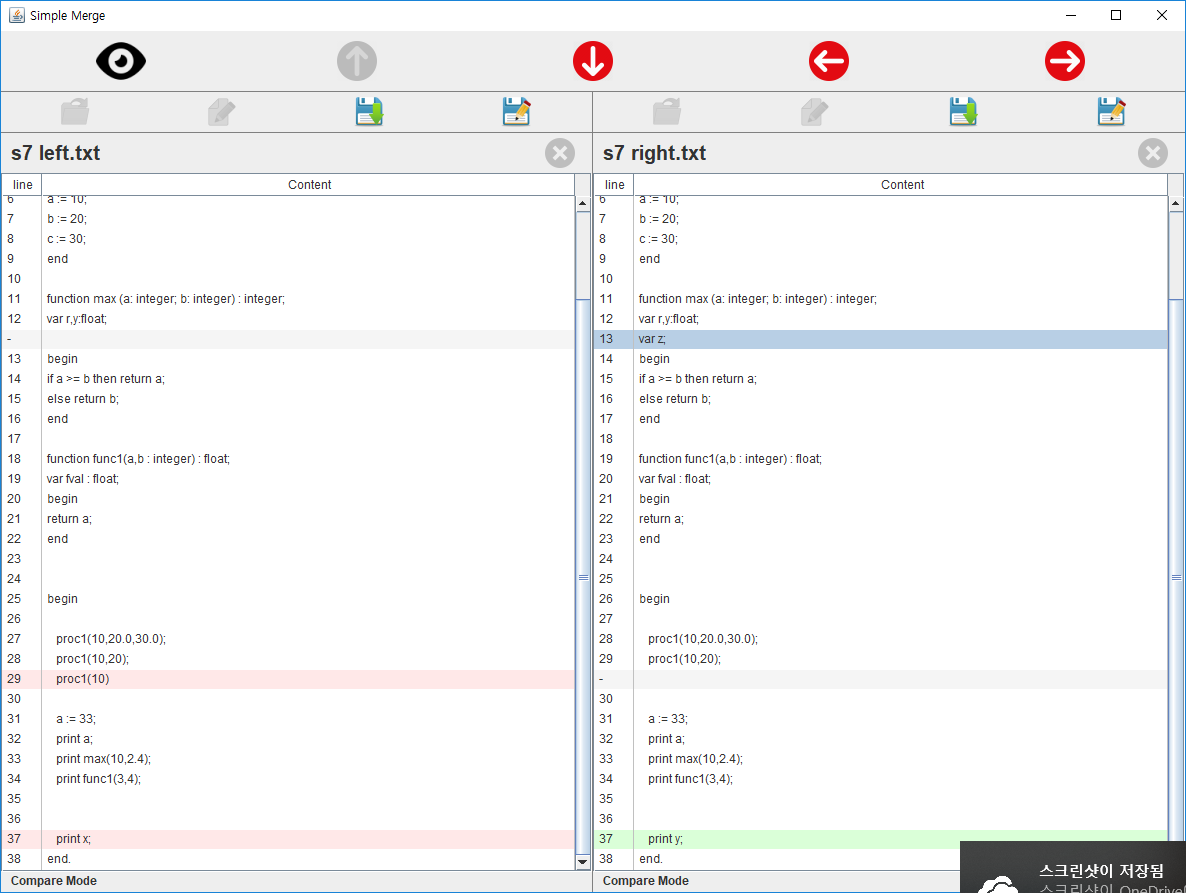
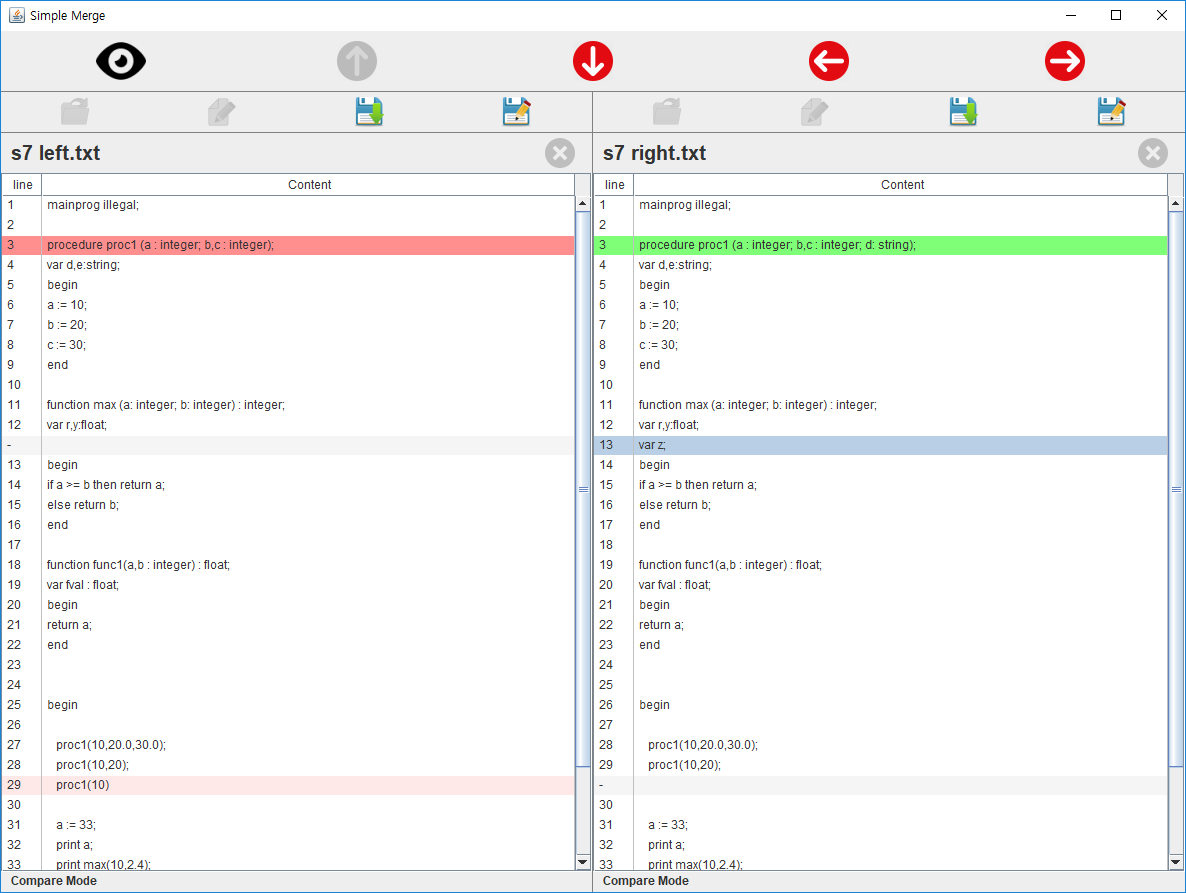
**S5. Result :** 여러 개의 빈 줄 블록이 있음이 나타난다. 또한 왼쪽 파일과 오른쪽 파일에서 다르게 나타나는 부분도 같이 블록에 포함되어 있다.



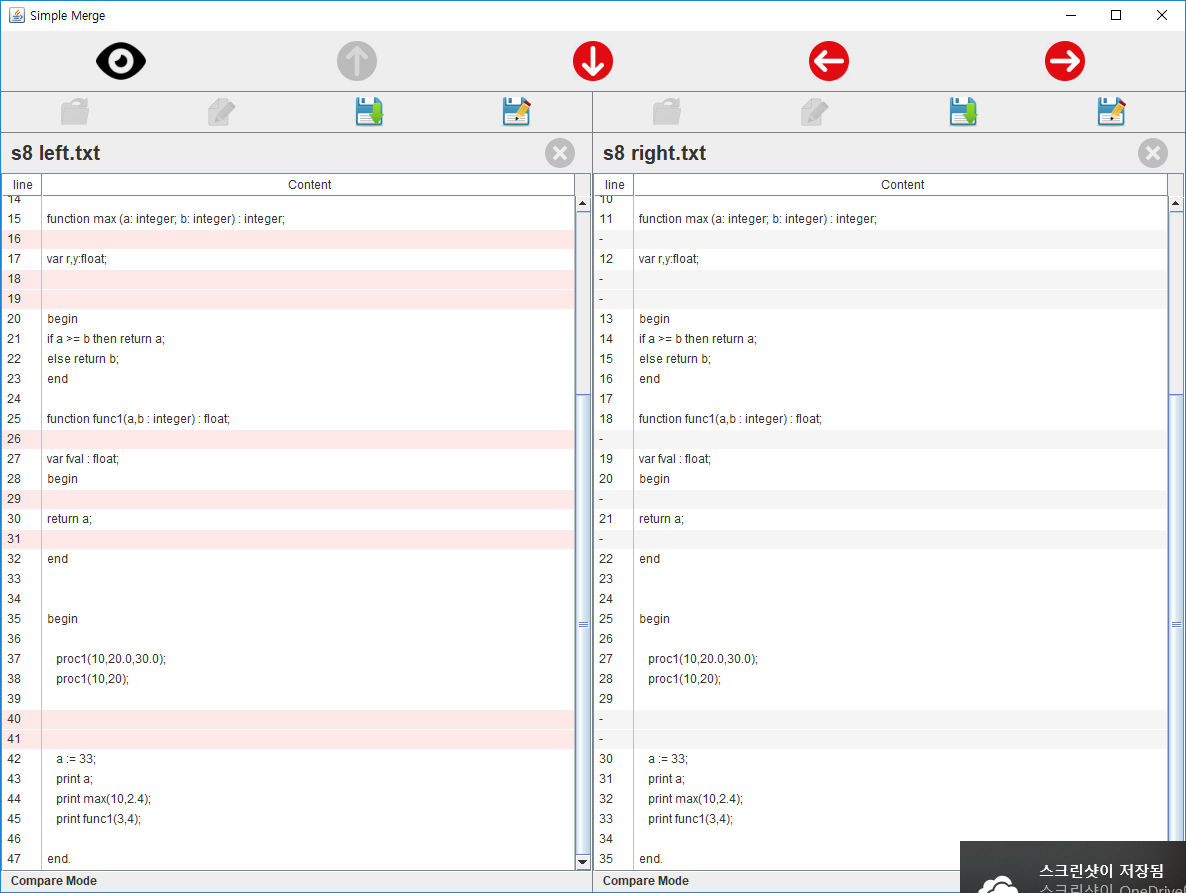
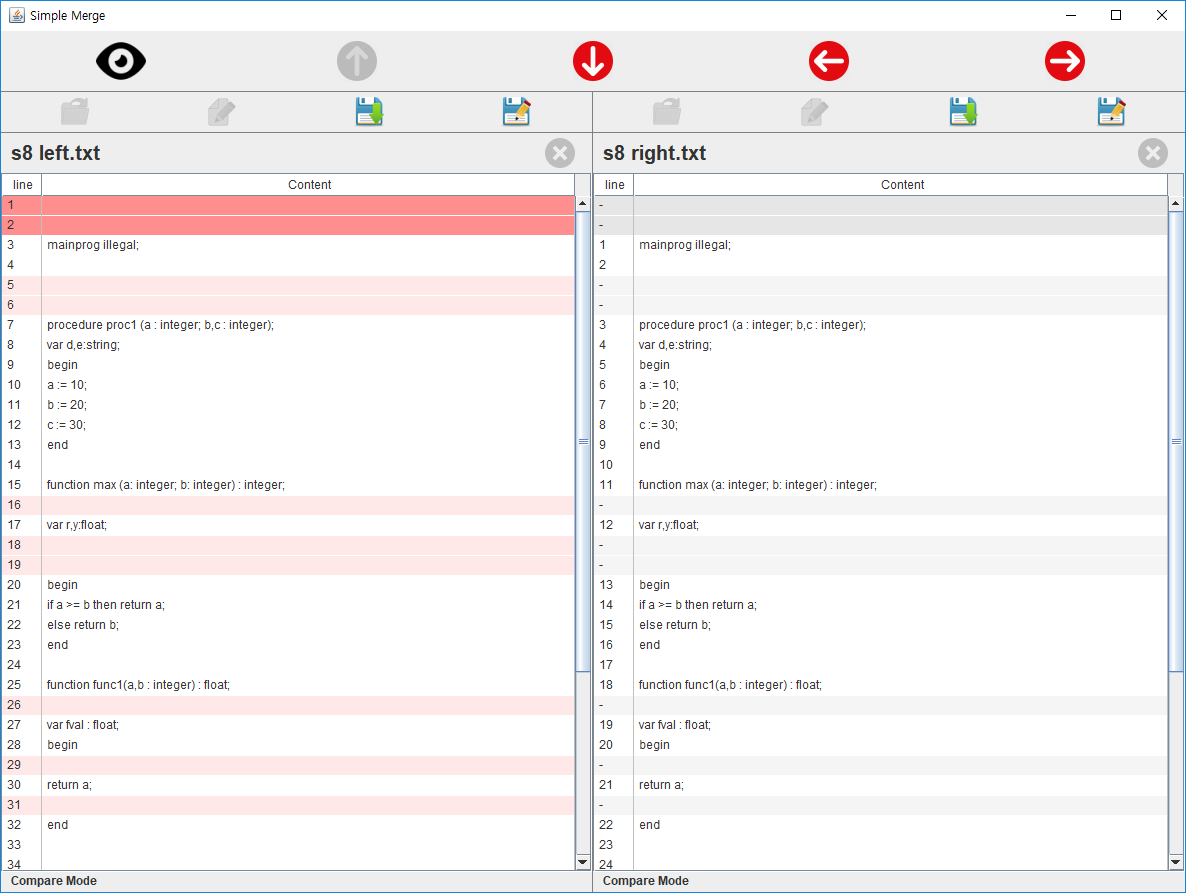
**S6. Result :** 두 파일이 동일함이 나타난다.



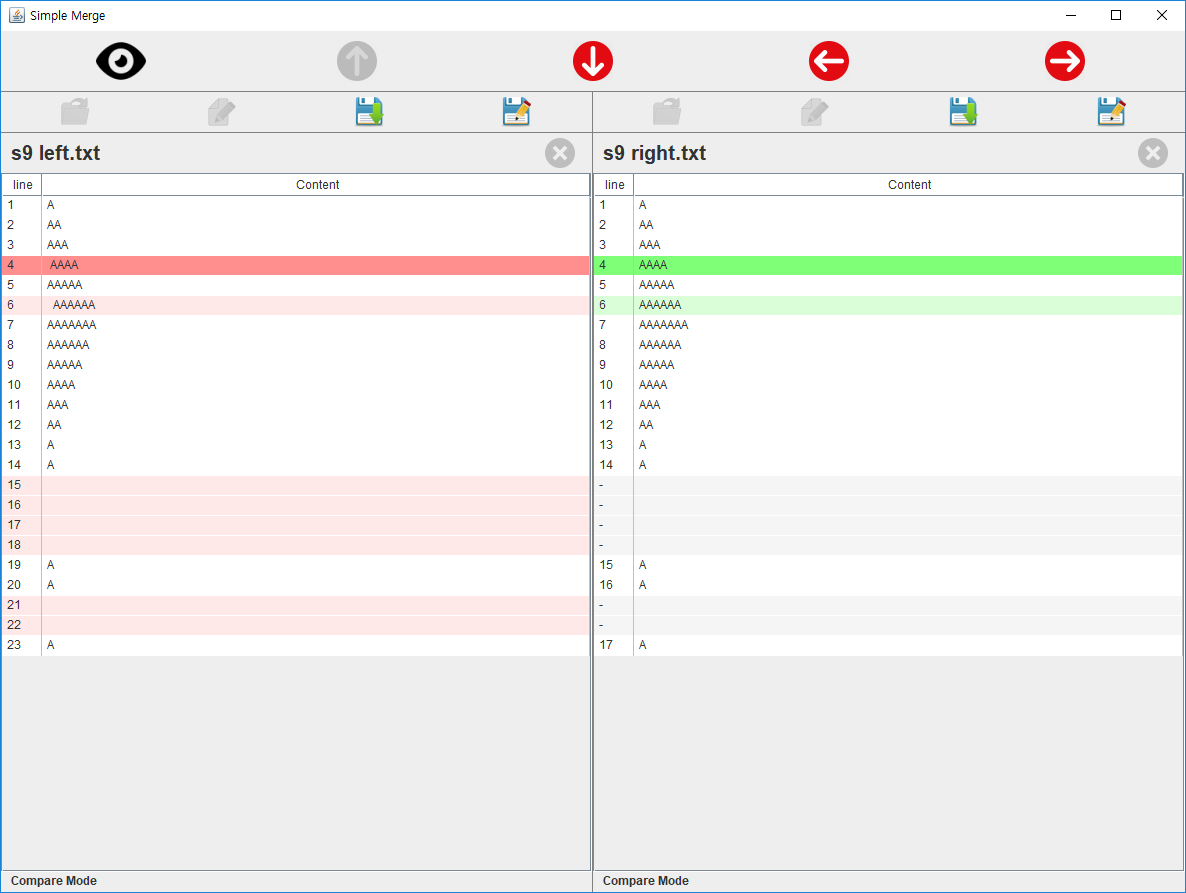
**S7. Result :** 3번째 줄이 다름, 왼쪽 파일에만 빈 줄이 있음, 오른쪽 파일에만 빈 줄이 있음, 37번째 줄이 다름이 나타난다.



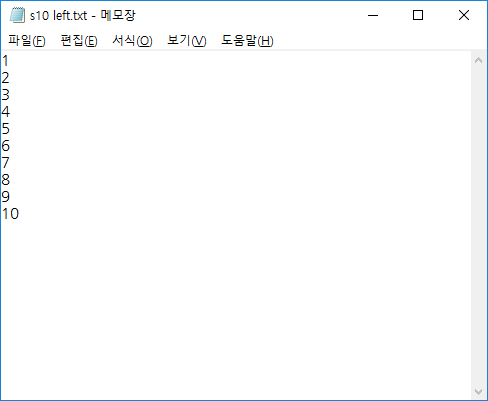
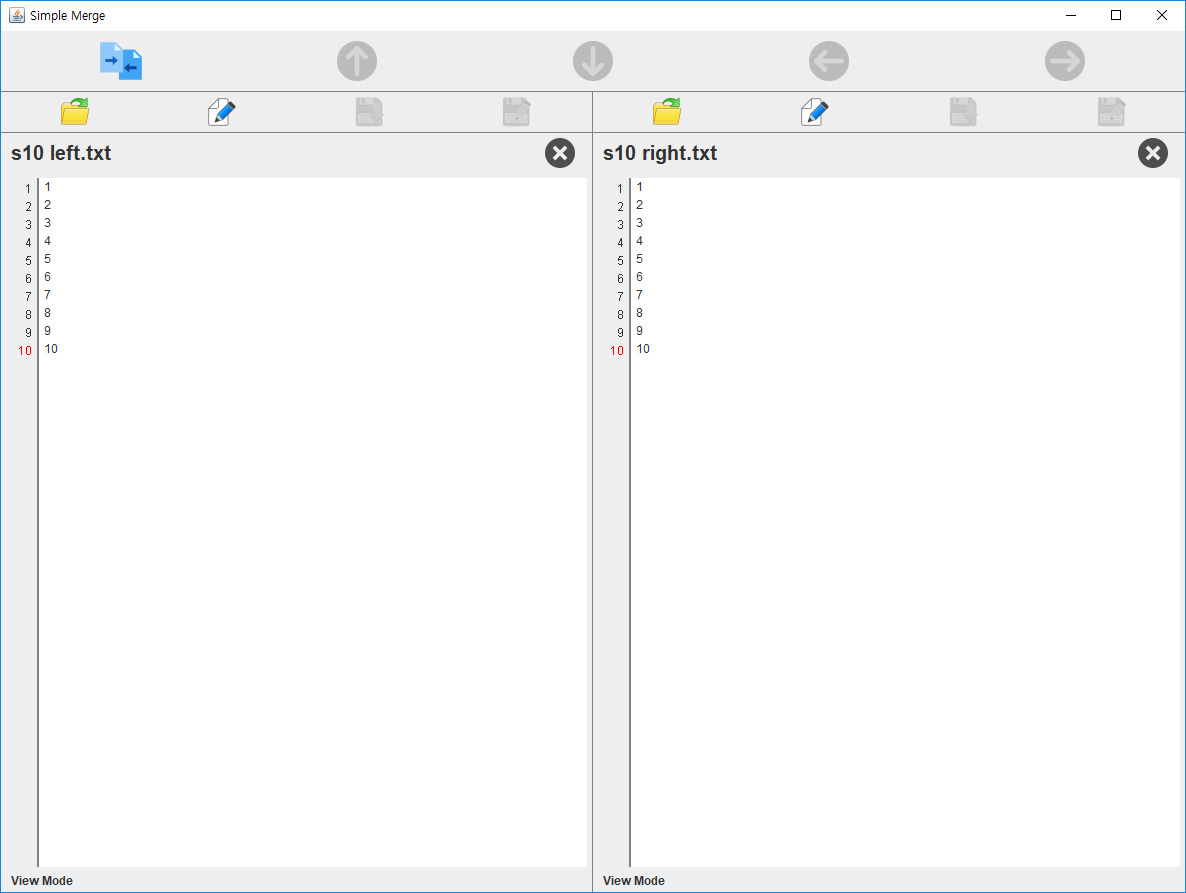
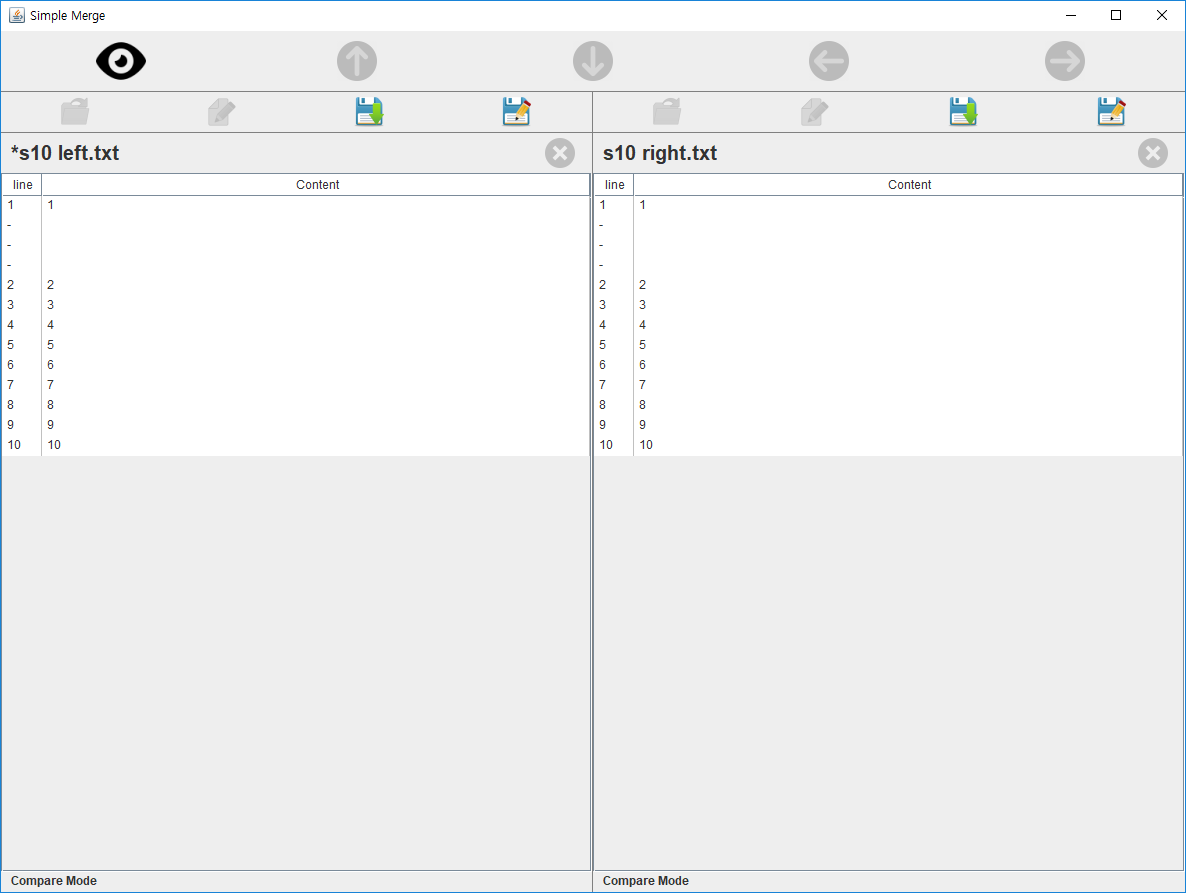
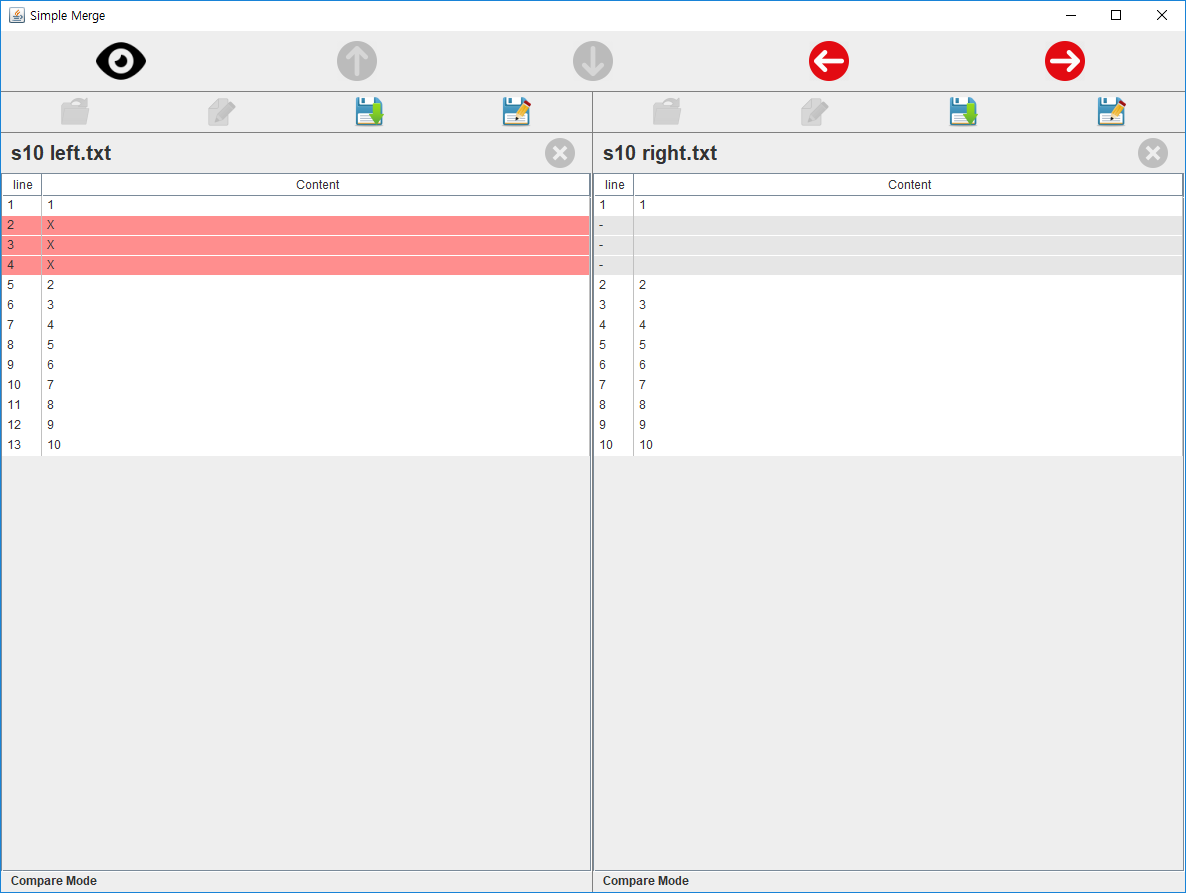
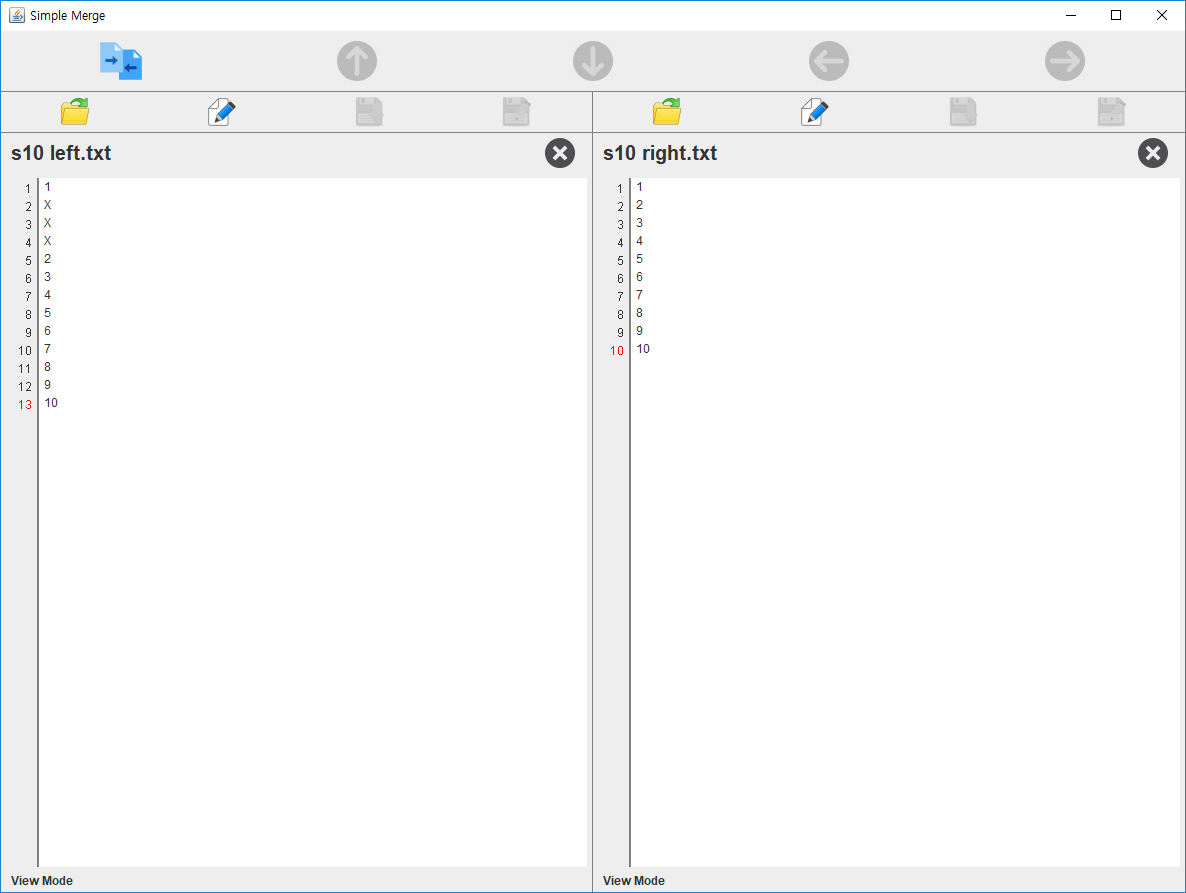
**S8. Result :** 여러 개의 빈 줄 블록이 있음이 나타난다.



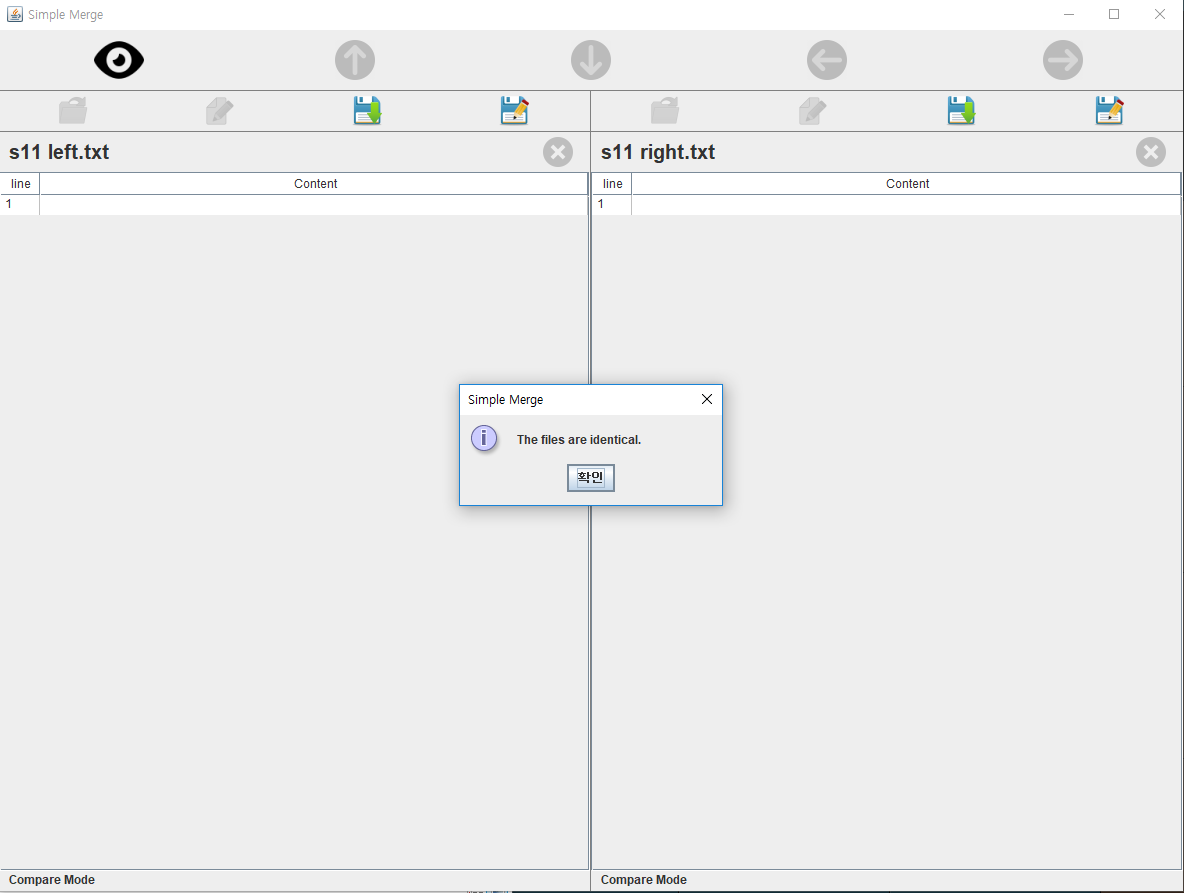
**S9. Result :** 여러 개의 빈 줄 블록이 있음, 4, 6번째 줄이 다름이 나타난다.



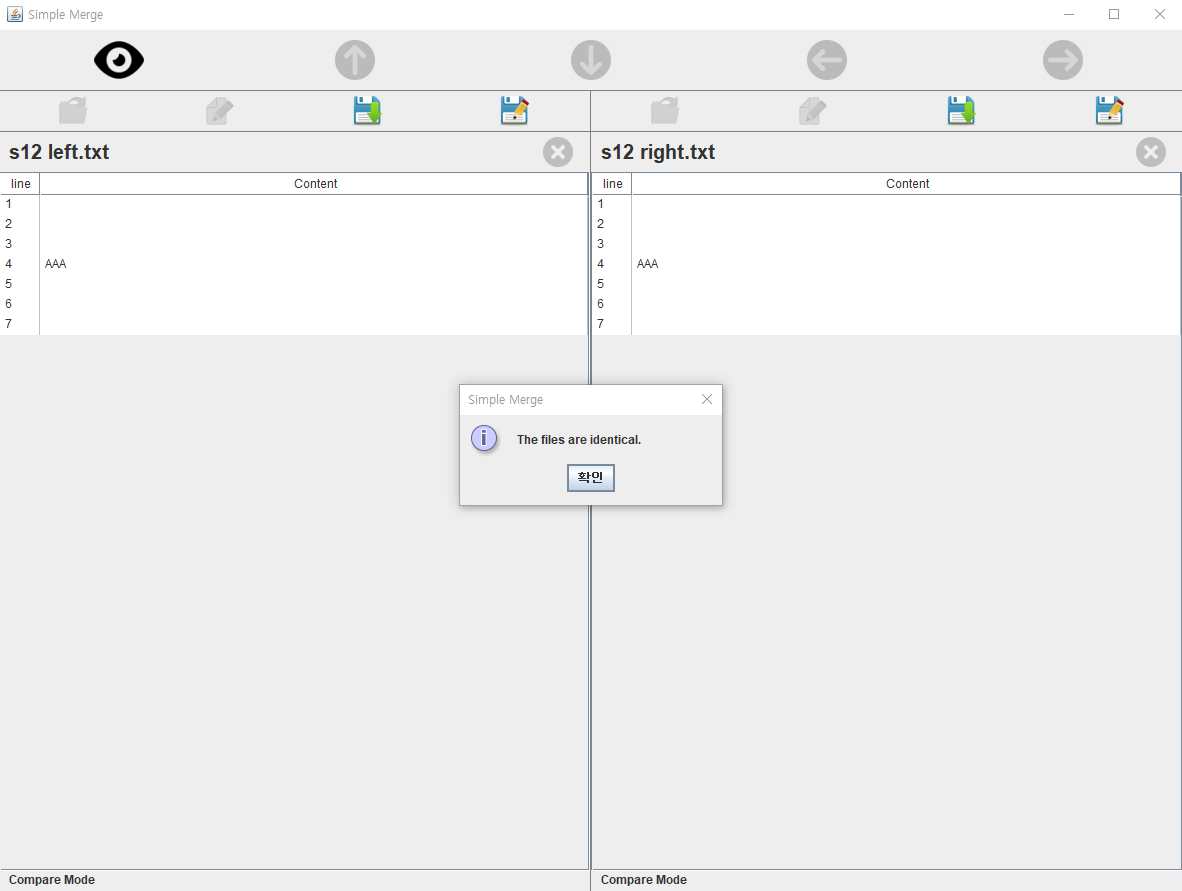
**S10. Result :** Compare 버튼을 누르면 양쪽 파일에서 다른 부분이 색칠되고, Copy to Left 버튼 ( <- ) 을 누르면 3번째 사진과 같이 파일 내용이 동일해지는 것을 볼 수 있다. Save 버튼을 누르고 Compare 모드에서 벗어나기 위해 눈동자 모양의 버튼을 클릭하면 View 모드로 전환되며, 두 파일의 내용이 완전히 동일함을 볼 수 있다. 이후 왼쪽 파일을 메모장으로 켜면 파일의 변경 내용이 제대로 Save 되었는지를 볼 수 있다.



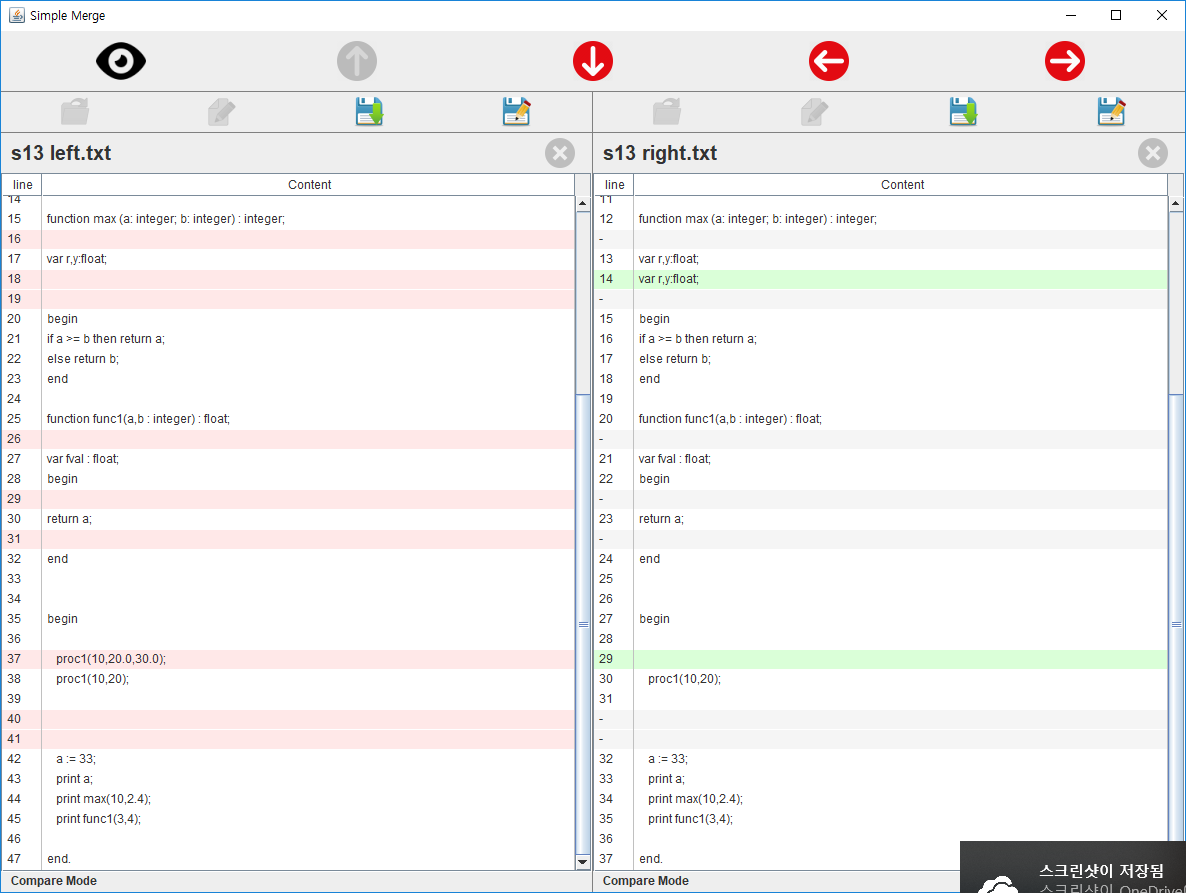
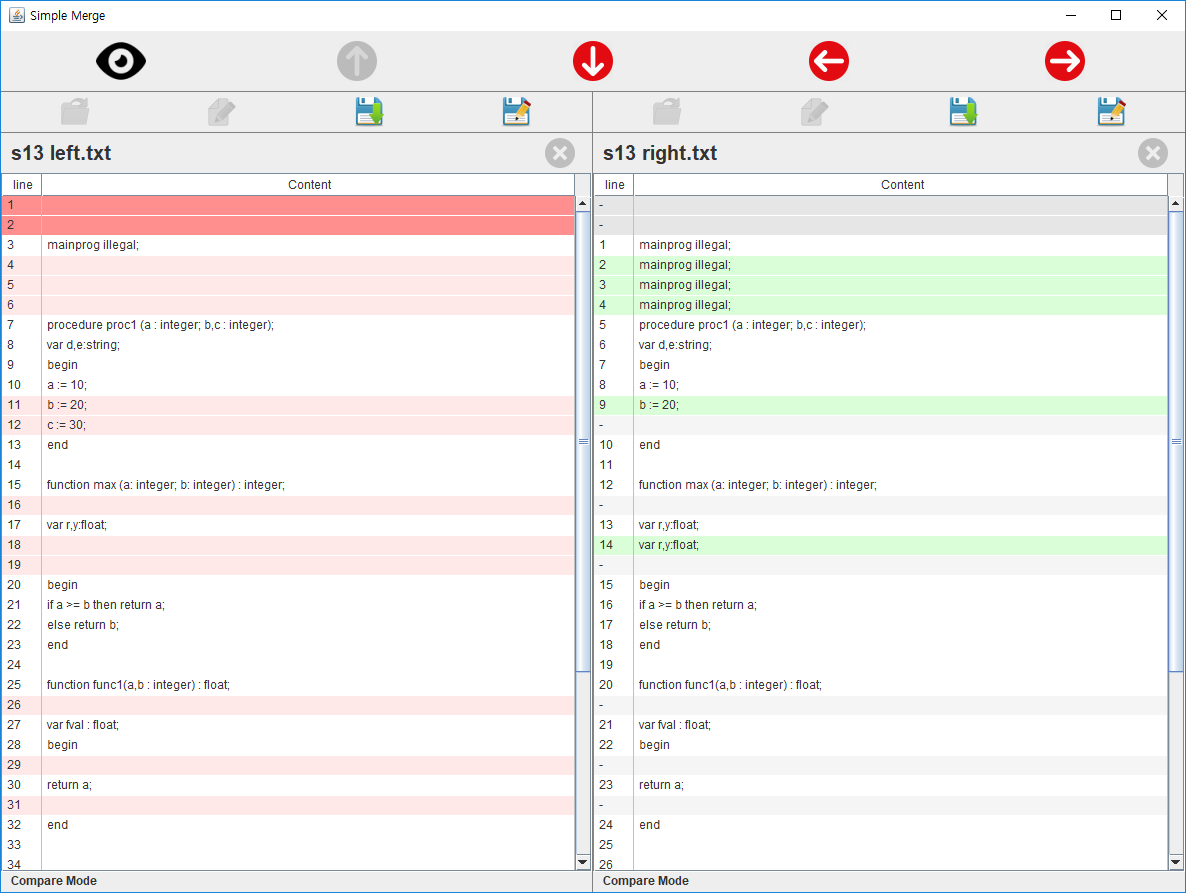
**S11. Result :** 비어 있는 두 파일의 내용이 동일함을 볼 수 있다.



**S12. Result :** 3줄의 빈 줄, 4번째 문장, 다시 3줄의 빈 줄이 나타남을 알 수 있으며, 두 파일의 내용이 동일함을 볼 수 있다.



**S13. Result :** 여러 개의 빈 줄 블록, 다른 내용 블록들이 모두 색칠되어 있는 것을 볼 수 있다.



**S14. Result :** 세 줄 블록의 내용이 다르고, 한 줄 블록의 내용이 달라서 색칠이 되어있음을 알 수 있다.

