Software Engineering Final Report

Simplemerge

Team 18

20110138 강원석

20133959 공찬형

20130895 성호준

20160030 심대범

20161493 하태윤

**Table**

1. Introduction
2. About the project
3. Git Address
4. Design
5. Package / class description
6. Class diagram
7. Frame description
8. MVC design pattern
9. Object-oriented design
10. Comparing Algorithm
11. Unit test
12. Program running screenshot
13. System test / error

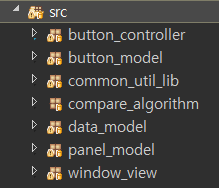
**Introduction**

이 문서는 Software Engineering term project인 Simplemerge 프로그램의 특징, 디자인 등을 설명한 문서이다. 프로젝트는 2018년 4월 17일부터 2018년 6월 8일까지 약 50일에 걸쳐 진행되었으며, Github 상에서 구현을 진행하였다.

Simplemerge는 Window 환경에서 사용가능한 Winmerge의 Java version 프로그램으로 두 문서를 비교하교, 프로그램 상에서 수정, 저장이 가능한 프로그램이다. Simplemerge는 Winmerge와 유사한 기능을 제공한다. 프로그램 구현에는 MVC design pattern과 Object-oriented design이 적용되었으며 Window Operation System 을 target으로 구현하였다.

(Github Address : <https://github.com/lenny-kong/se_termproj>)

**Design**

프로그램 소스는 총 7개의 package, 36개의 class로 구성된다.

- button\_controller : button 들의 actionlistener 를 구현

- button\_model : button 들의 형태, 위치 등을 구현

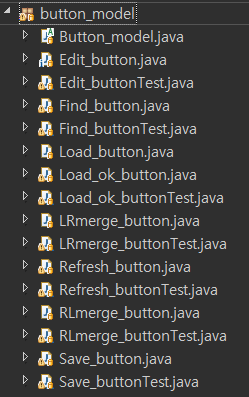
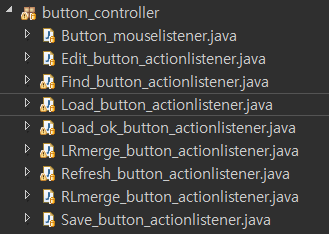
- common\_util\_lib : 프로그램 전반적으로 사용되는 method, 변수들을 구현

- compare\_algorithm : 두 파일을 비교하는 method 구현

- data\_model : 프로그램에서 사용되는 GUI\_data\_model 과 color들을 구현

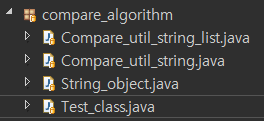
- panel\_model : 프로그램의 panel 들을 구현

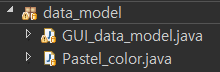
- window\_view : 프로그램의 Frame 들을 구현



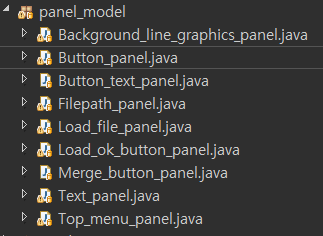
button \_controller, button\_model package 에는 프로그램의 버튼들이 구현되어 있다.

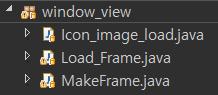
button\_model package의 button class는 button의 image, 크기, 위치 등을 구현하고, button\_controller package의 actionlistner class에 button이 눌렸을 때의 동작을 구현하였다. 또한 button\_model package 에는 unit test를 위한 test class를 같이 생성하였다.

compare\_algorithm package에는 프로그램에서 사용되는 comparing function을 구현하였다. 파일 비교는 각 line 별로 이루어지며, 알고리즘의 unit test 를 위한 Test\_class를 생성하였다.

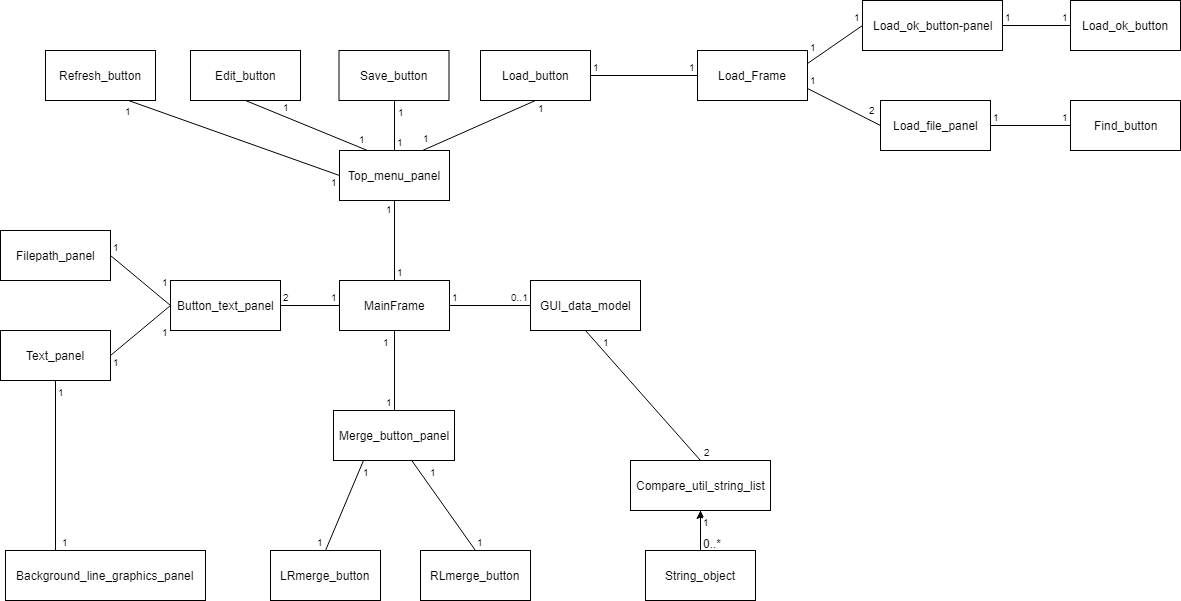


data\_model package 에는 프로그램에서 사용하는 자료형인 GUI\_data\_model을 구현하였다. GUI\_data\_model 에는 txt file의 Filepath 등이 포함된다. 또한 프로그램에서 사용하는 여러 color 들을 같이 구현하였다.

panel\_model package에는 Frame들에 들어가는 panel 들을 구현하였다. 또한 txt file이 load 했을 때, 배경 색을 칠하기 위한 method 도 구현되어 있다.

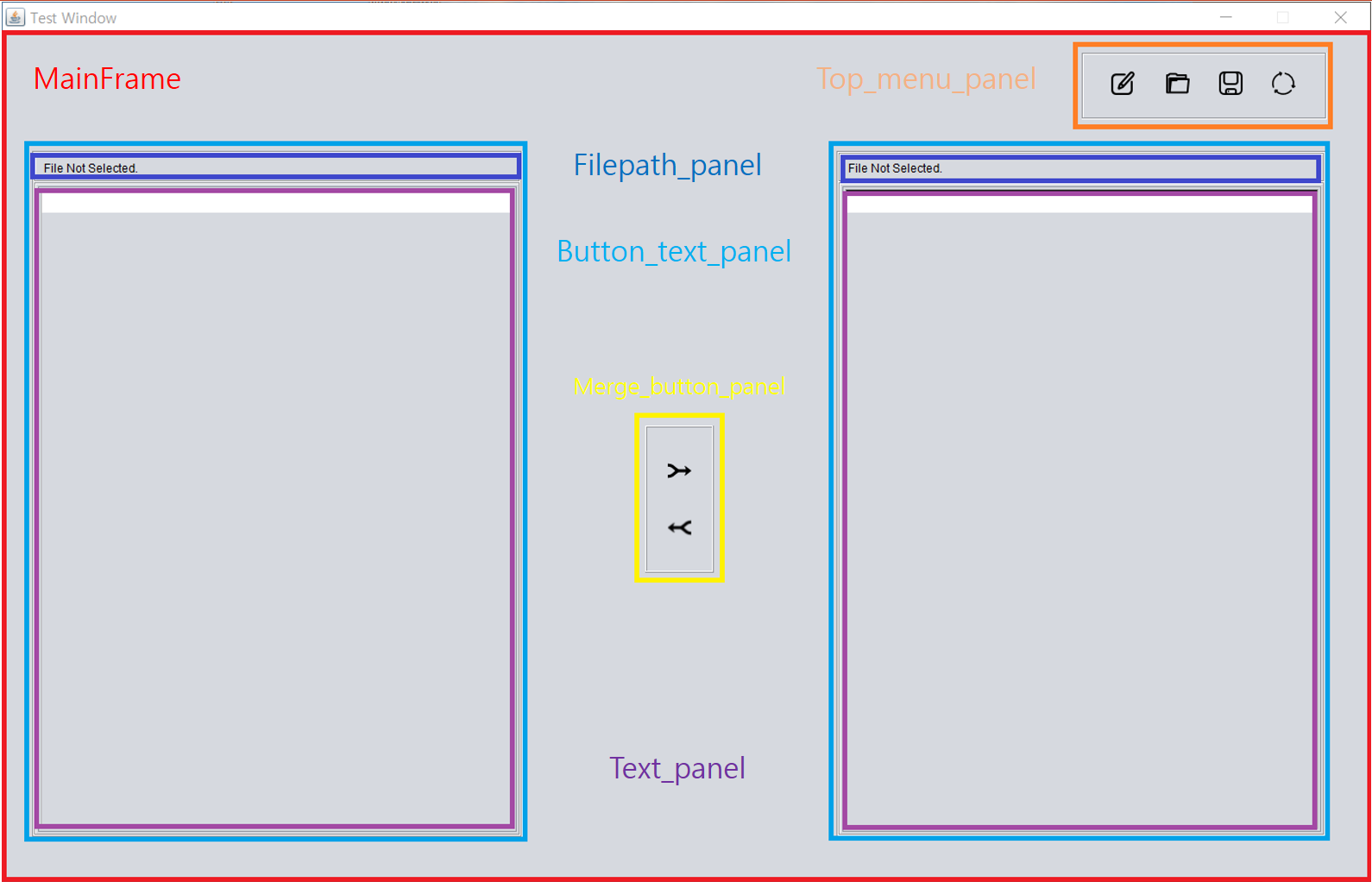
window\_view package에는 각 window 들의 Frame 들이 구현되어 있다. 또한 프로그램 icon 등이 같이 구현되어 있다.

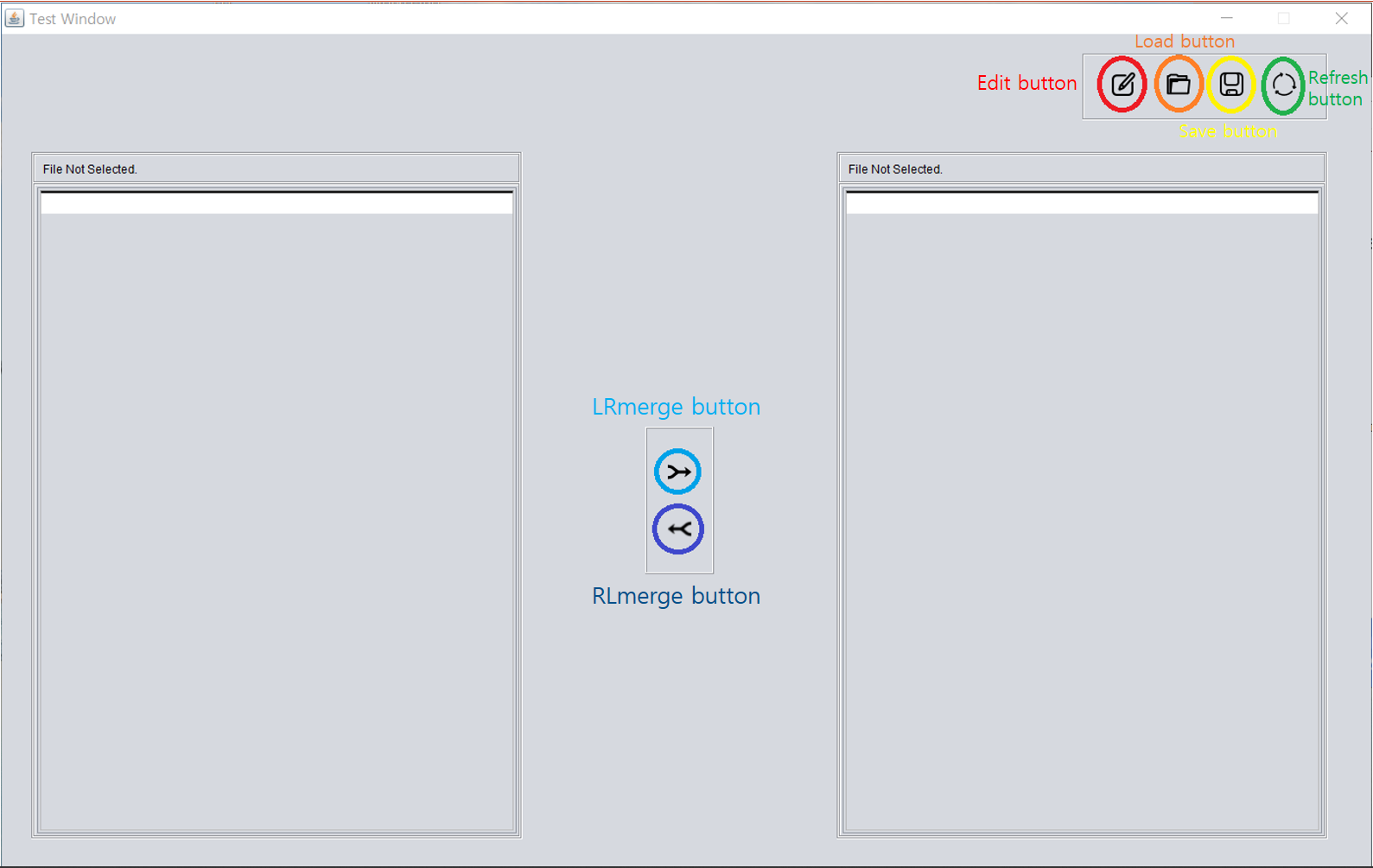
위의 class들을 diagram으로 표현하면 아래와 같다.

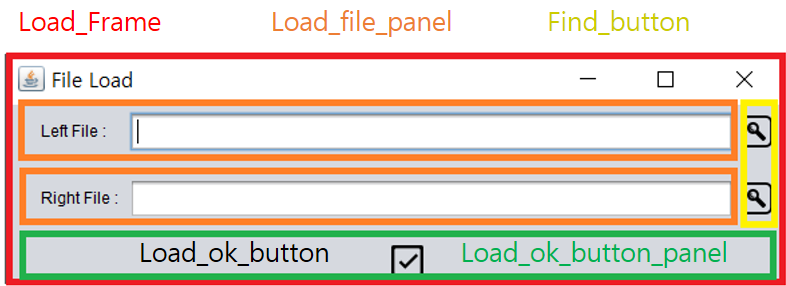


Diagram에는 생성 관계만 표현되어 있다.

실제 실행 시, 각 Frame 의 구성은 아래와 같다.







Simplemerge은 기본적으로 MVC Design Pattern을 적용하여 구현하였다. MVC Design Pattern은 프로그램을 Model, View, Controller 세 부분으로 나누어 구현하는 것을 말한다. 이는 프로그램의 유연성을 높이는 방법으로, 각각의 역할을 담당하는 부분을 나눔으로써, 유지보수를 쉽게 하고 중복 코딩을 방지한다.

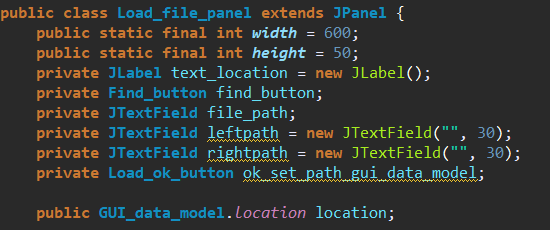
Model은 프로그램에서 사용되는 정보, 데이터를 의미하며, button\_model, data\_model, panel\_model package가 해당한다. 이 세 package는 프로그램에서 사용되는 button, panel, GUI data model 등의 정보를 포함하고 있다.

View는 텍스트 등의 사용자 인터페이스 요소를 의미한다. window\_view package가 이에 해당하며, MakeFrame class와 MainFrame class 가 Main 창을, Load\_Frame class 가 Load 창 생성을 담당한다.

Controller는 사용자의 입력에 따라 발생하는 이벤트를 처리하는 부분으로, button\_controller package가 이에 해당한다. 프로그램 사용 중, 사용자가 버튼을 클릭하면, button\_controller package 내의 class들이 처리를 담당한다.

또한 프로그램을 구현하는데 있어 Object-oriented Design을 적용하여 구현하였다. Object-oriented Design의 가장 대표적인 특징으로는 Encapsulation, Inheritance, polymorphism을 말 할 수 있는데, 이러한 특징들이 Source Code 에 나타나 있다.

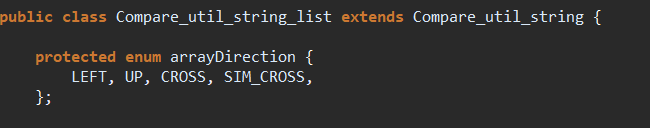
<Encapsulation>



(panel\_model package / load\_file\_panel class)

Encapsulation은 object에 대한 접근 권한을 나누어 object 내부에서의 접근과 외부에서의 접근을 구분하는 것을 뜻한다. 위의 Code 에서는 width, height, location 등이 public으로, text\_location, file\_path 등은 private 변수로 선언된 것을 볼 수 있다.

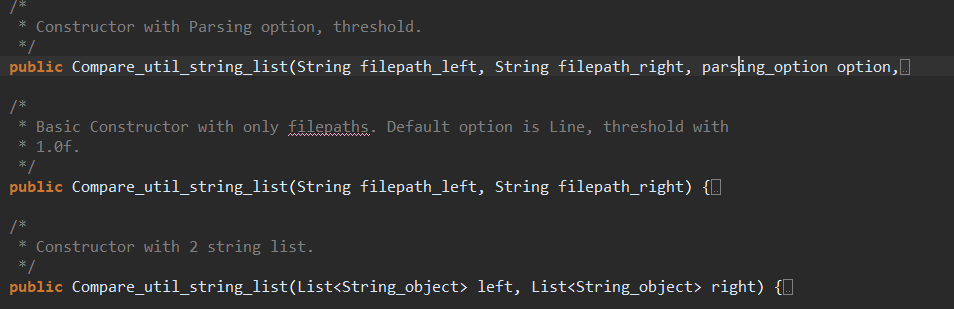
<Inheritance>



(compare\_algorithm package / Compare\_util\_string\_list class)

Inheritance는 부모 class로부터 variable이나 method를 상속받아 사용하는 것을 말한다. 위의 코드에서 subclass인 Compare\_util\_string\_list class가 부모 class 인 compare\_util\_string class 를 상속받은 것을 볼 수 있다.

<Polymorphism>

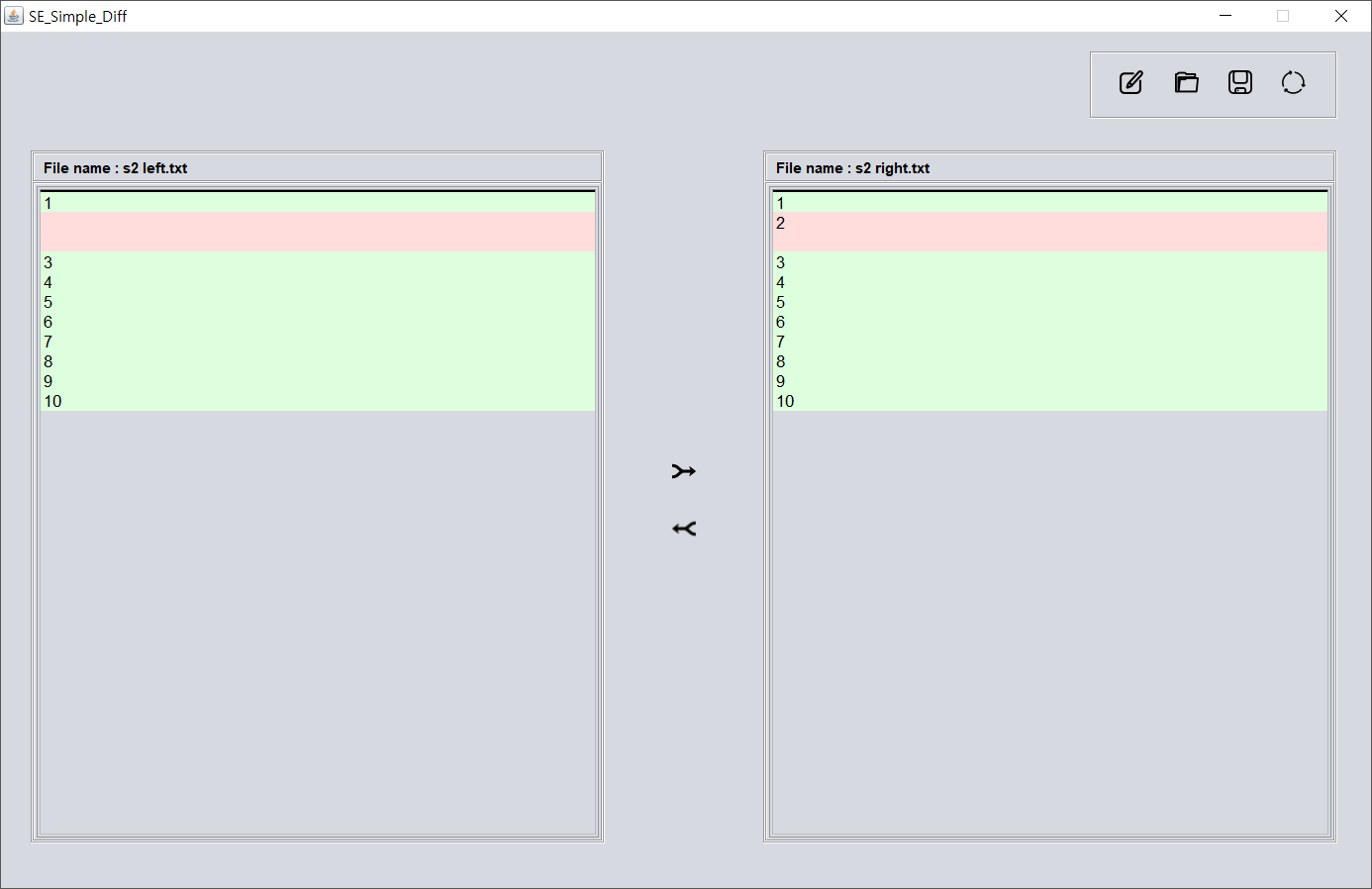
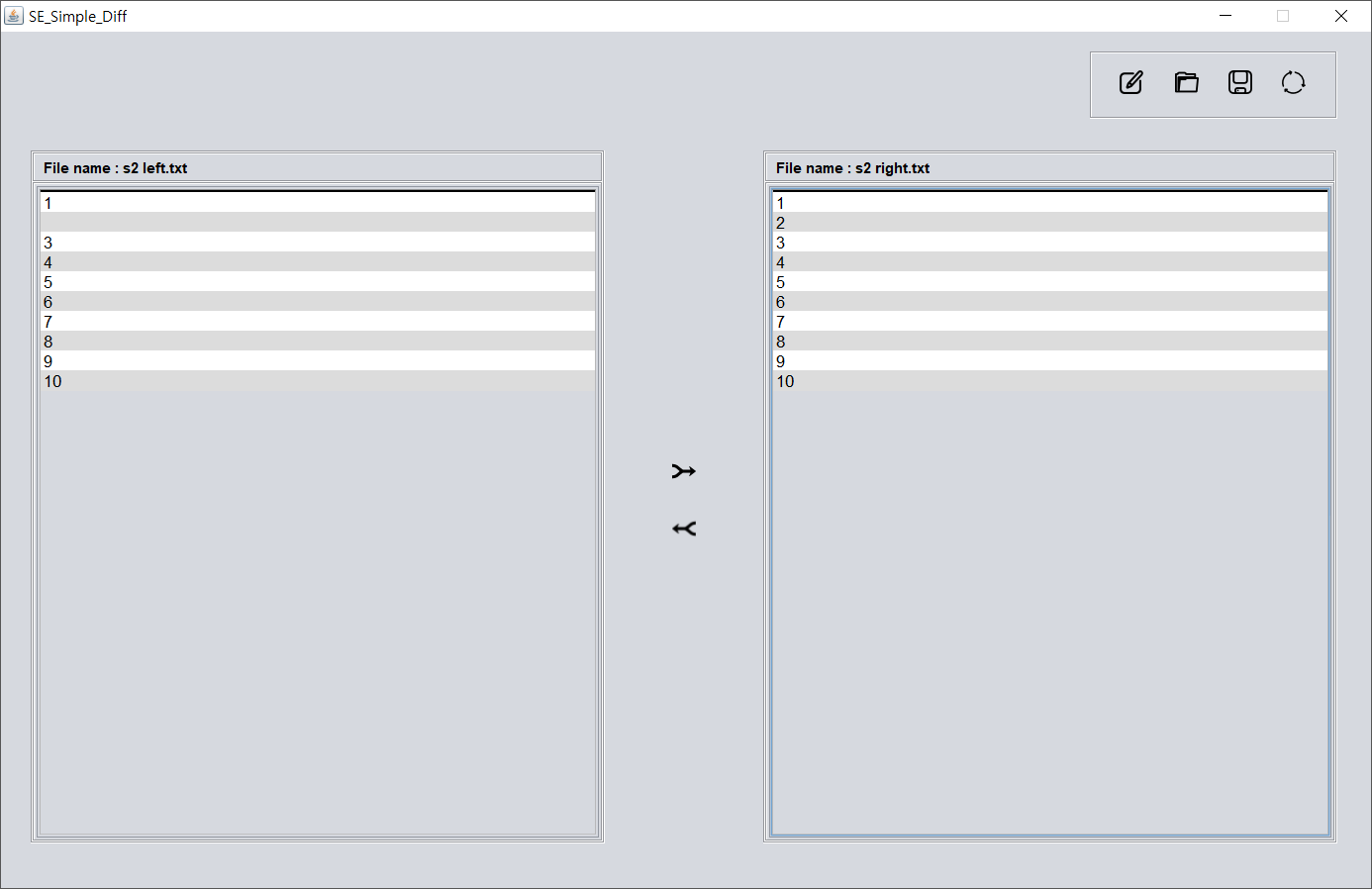
(compare\_algorithm package / Compare\_util\_string\_list class)

Polymorphism은 상속을 받거나 class에 대하여 새롭게 정의할 때 사용하는 개념이다. 크게 overriding과 overloading 두 가지로 나눌 수 있는데, 위의 Code 에 Overloading에 관련된 예시가 나와있다. 같은 생성자에서 인자로 어떤 것을 입력받느냐에 따라 다른 생성자가 실행됨을 볼 수 있다.

**<Comparing Algorithm>**

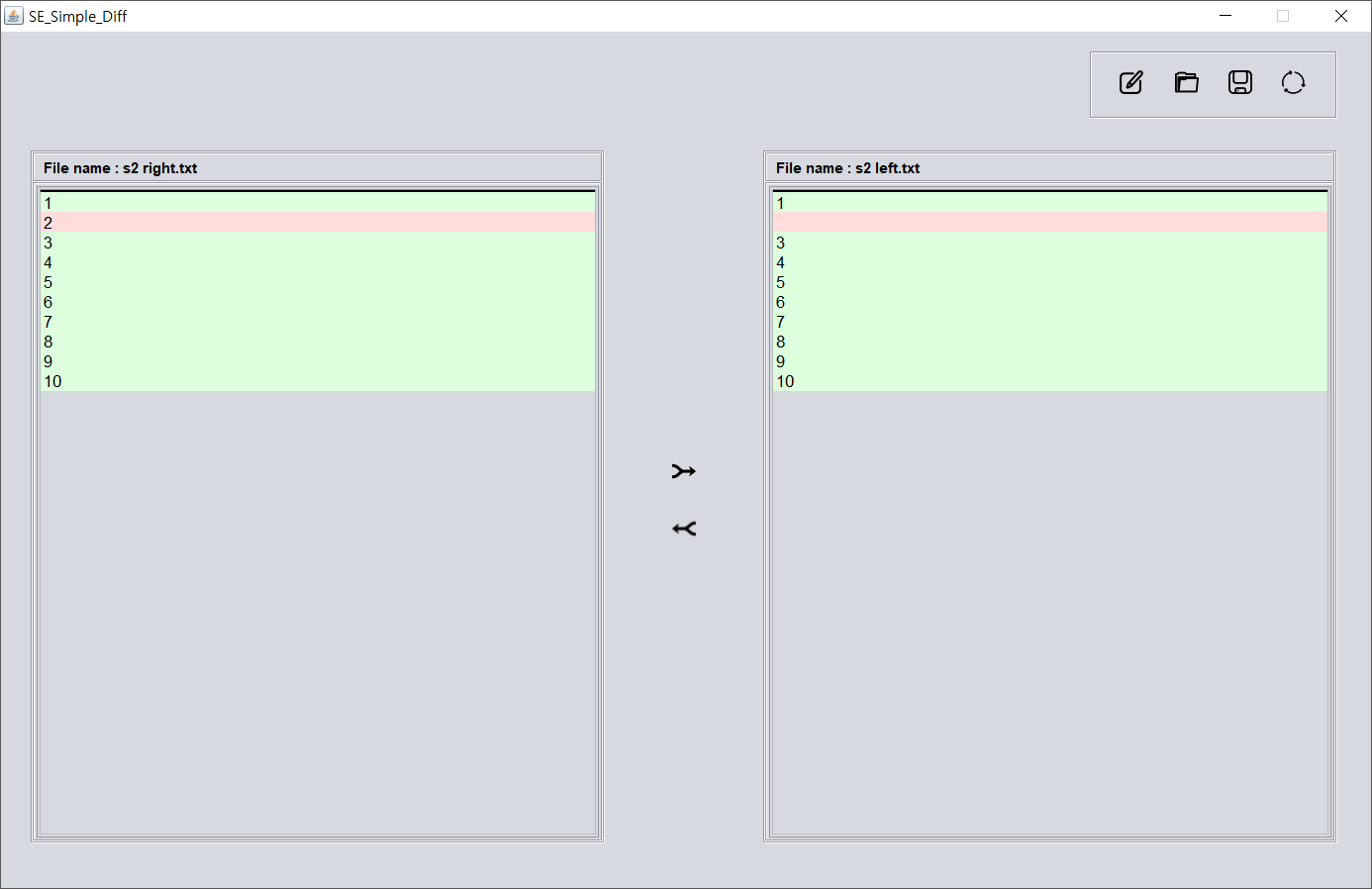
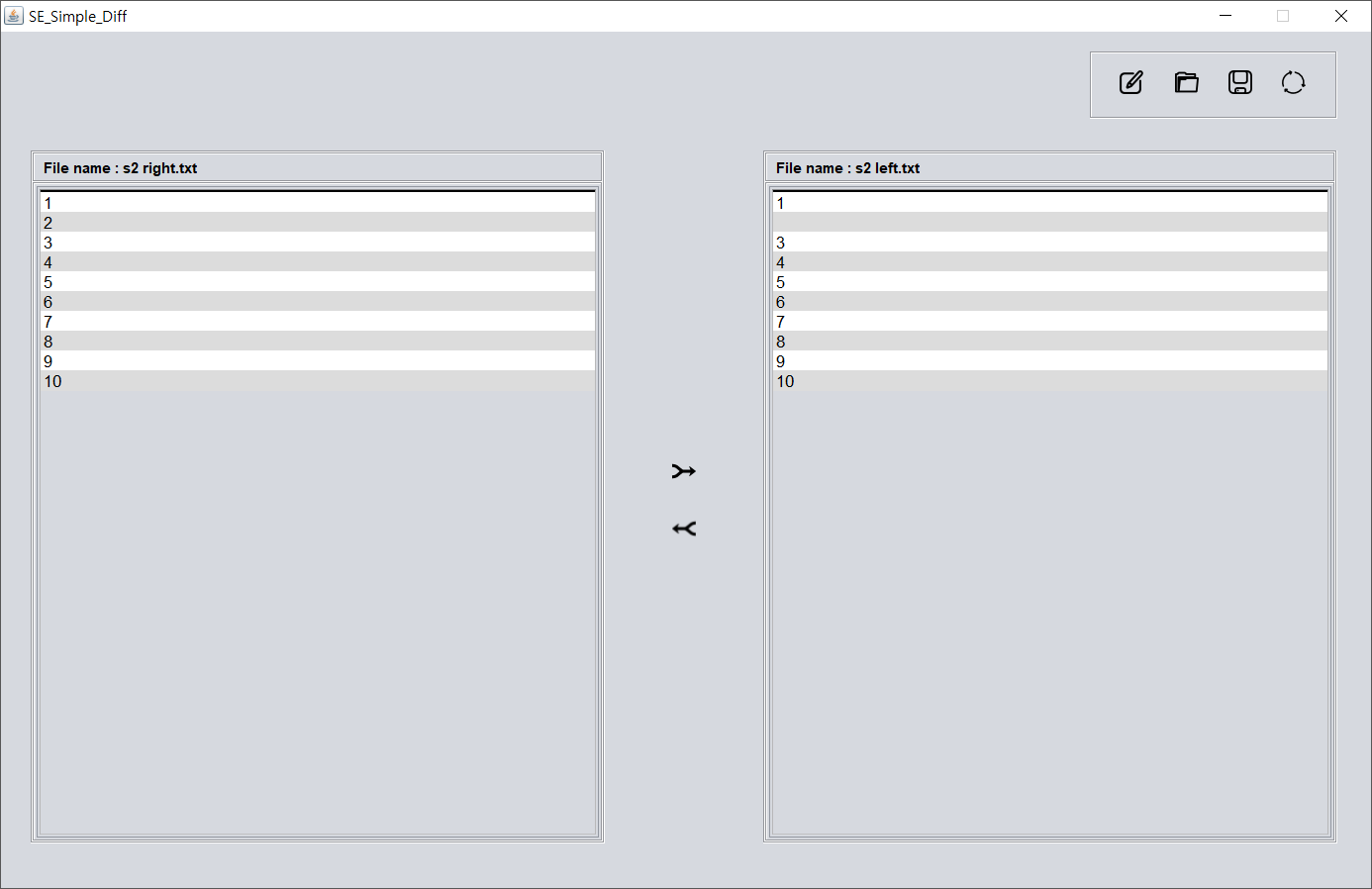
Text file 의 비교를 위한 알고리즘이 compare\_algorithm package에 구현되어 있다. 기본적으로 Longest common subsequence algorithm을 기초로 구현하였다.

Simplemerge의 비교 알고리즘은 txt file 을 line by line으로 비교하며, 왼쪽 text file 을 기준으로 비교한다. 비교 후 같은 부분은 녹색으로, 서로 다른 부분은 붉은 색으로 표시한다.



(예시1. S2left.txt & s2right.txt)

위의 경우 왼쪽 txt file 기준에서 보면 오른쪽 text file의 2가 누락된 상태로 판단한다. 이 때 프로그램은 왼쪽 파일에 2가 들어가기 위한 한 줄을 확보한다. 결과적으로 Comparing function을 실행하면 총 두 줄이 붉은 색으로 표시된다.



(예시2. S2right.txt & s2left.txt)

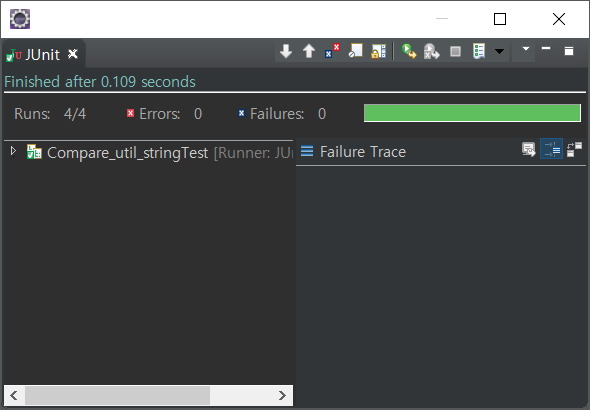
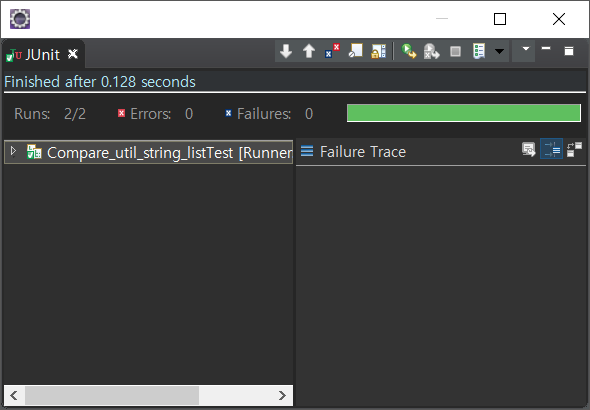
첫 번째 방법과 다르게, 왼쪽에 s2right.txt를, 오른쪽에 s2left.txt 를 load한다. 이 때 비교는 왼쪽 파일 기준으로 진행된다. 위의 경우 왼쪽 파일 기준으로 보면 오른쪽 파일에 왼쪽 파일의 2가 누락된 상태로 판단한다. 하지만 이 경우 프로그램은 왼쪽의 text를 오른쪽의 blank block에 삽입할 수 있기 때문에 추가적으로 blank block을 확보하지 않는다. 따라서 결과적으로 한 줄만 붉은 색으로 표시된다.

만약 오른쪽의 text가 blank block이 아닐 경우, 예시 1번과 같이 오른쪽 파일에서 다른 부분을 왼쪽 파일에 삽입하기 위해 blank block을 확보하는데, 이 때 확보되는 오른쪽 파일의 blank block에 왼쪽 파일의 text가 삽입된다

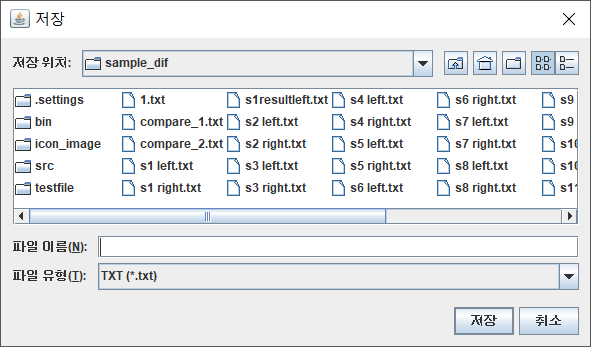
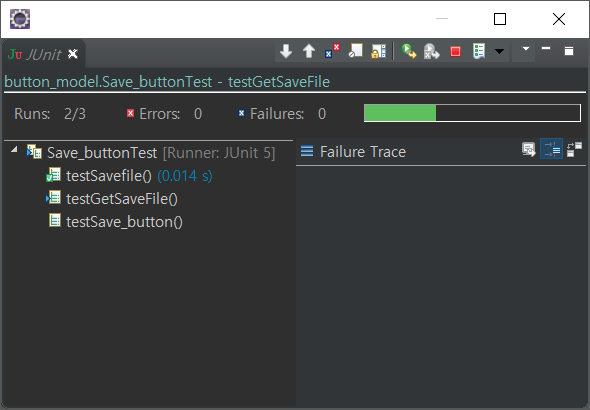
**Unit Test**

핵심 기능과 GUI를 테스트하기 위해 Junit 을 이용하여 unit test를 진행하였다.

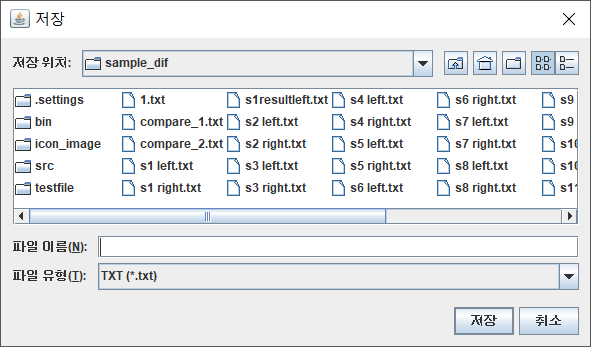
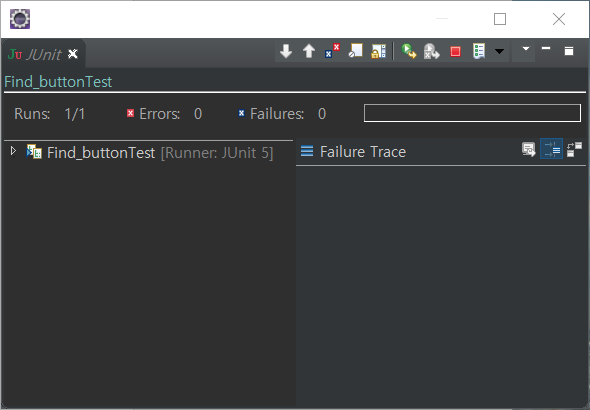
크게 text file을 비교하는 comparing algorithm과 GUI 부분을 test 하였다. 각 class 별로 test class를 만들어 unit test하기 편하게 구현하였다.



(알고리즘 부분을 담당하는 Compare\_util\_string , Compare\_util\_string\_list class unit test)



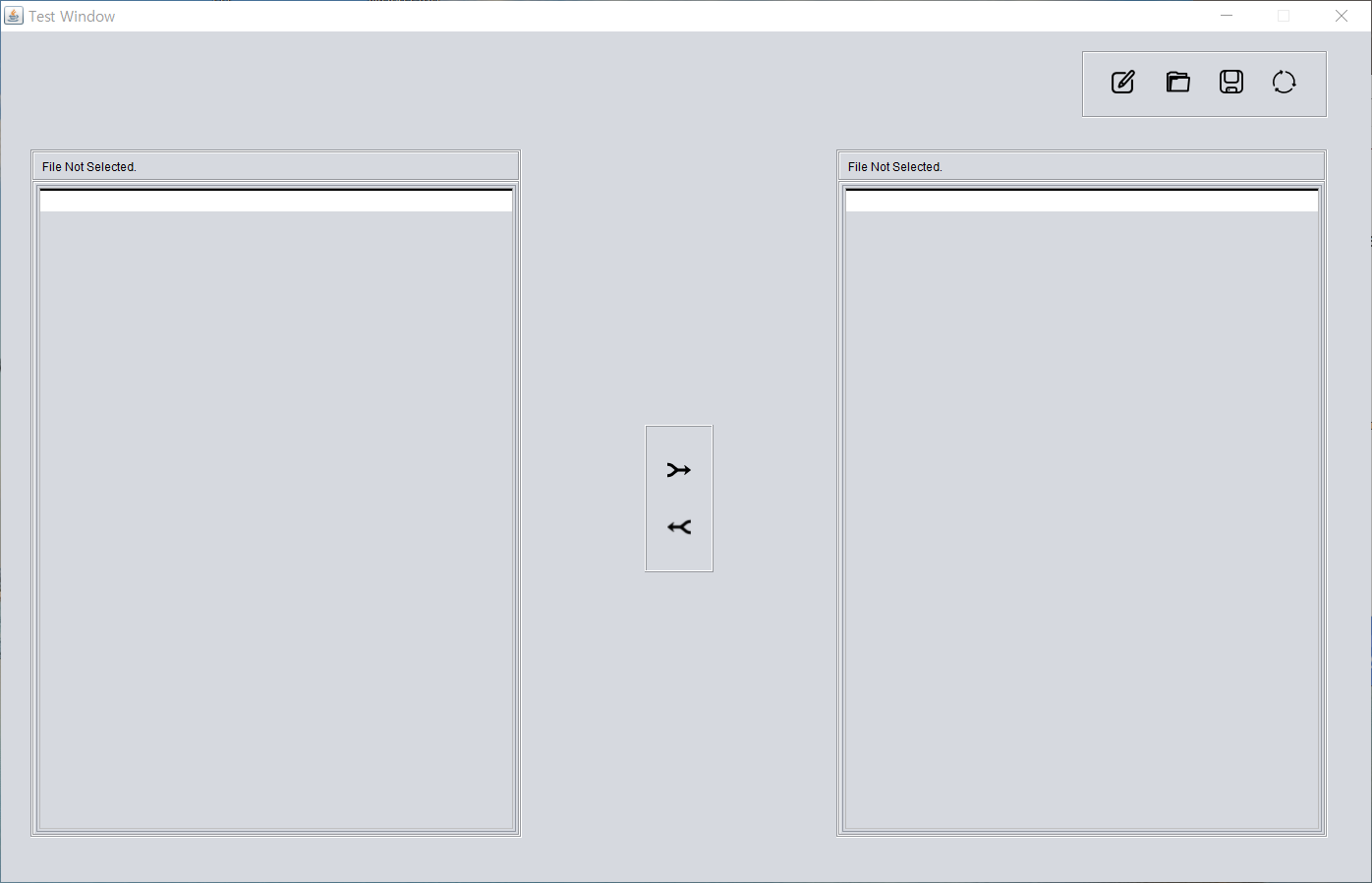
(button\_model package의 Save\_button class unit test)



(button\_model package의 Find\_button class unit test)

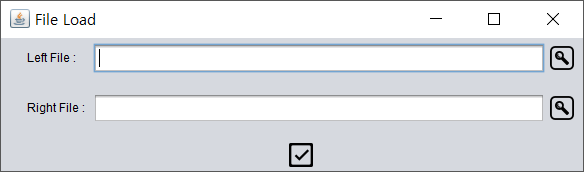
**Program Running screenshot**

<프로그램 실행 시 초기 화면>

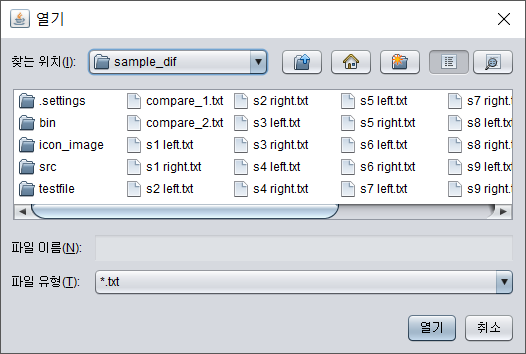


* 우측 상단 Edit, Load, Save, Refresh 버튼을 이용하여 해당 기능을 수행한다.
* 왼쪽, 오른쪽 판넬에 각각 text file 하나씩 들어간다.
* 중앙의 화살표는 Merge button이다.

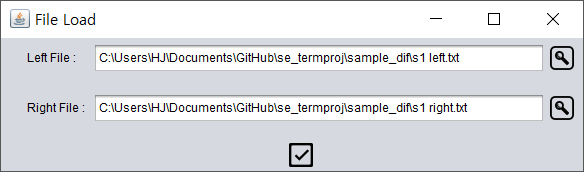
<Load 기능>



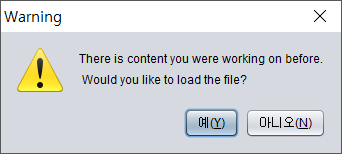
* 우측 상단의 Load 버튼을 클릭하면 text file 을 load 할 수 있는 창을 생성한다.
* 왼쪽, 오른쪽 파일을 따로 지정할 수 있다.



* 우측의 버튼을 누르면 load 할 파일을 직접 지정할 수 있다.
* Load 할 수 있는 파일유형은 txt file로 한정된다.

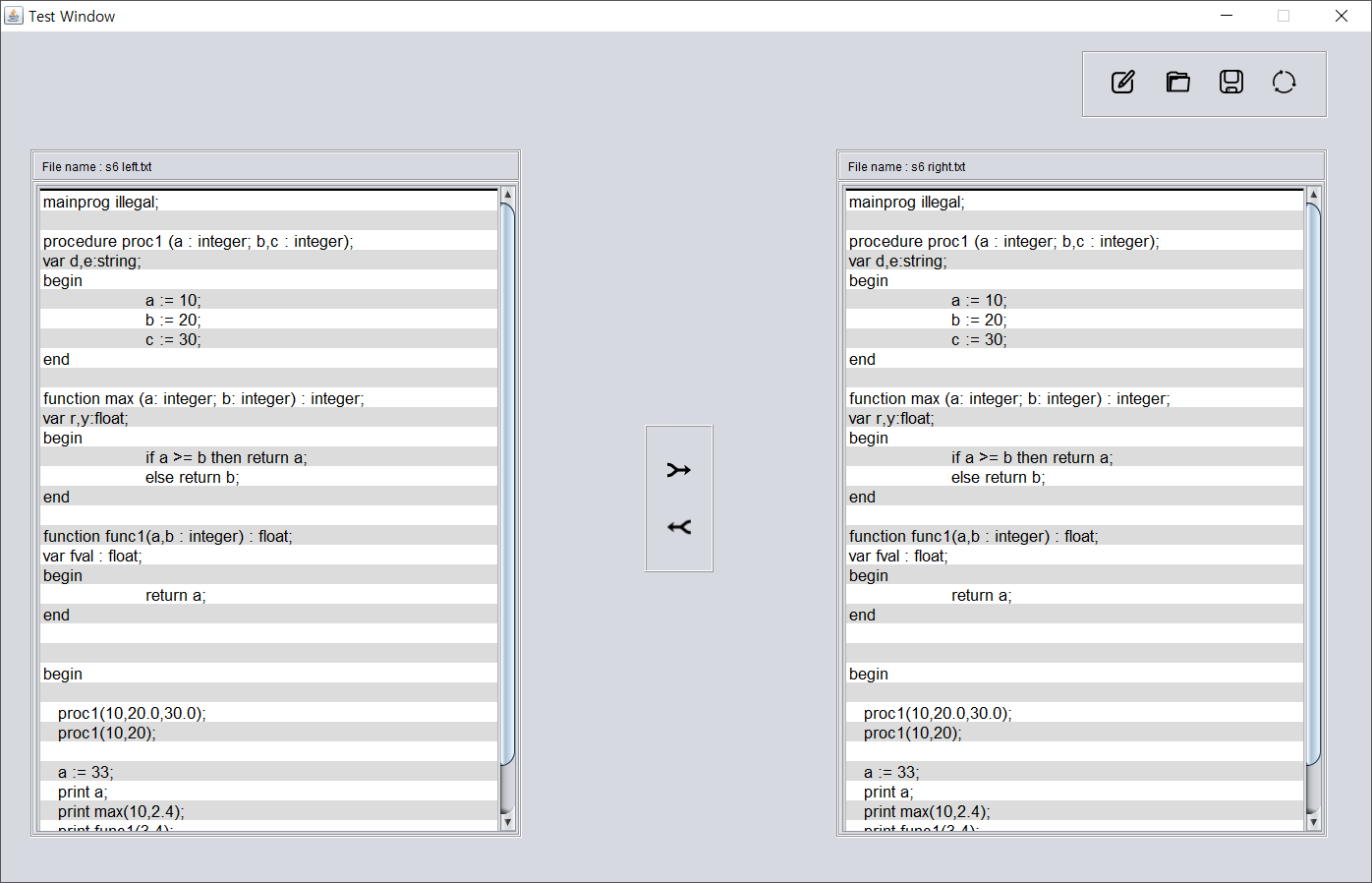


* 왼쪽, 오른쪽에 들어갈 파일을 각각 지정한 상태이다.
* 파일의 위치와 이름이 표시된다.
* 파일 지정 후, 하단의 check 버튼을 클릭한다.



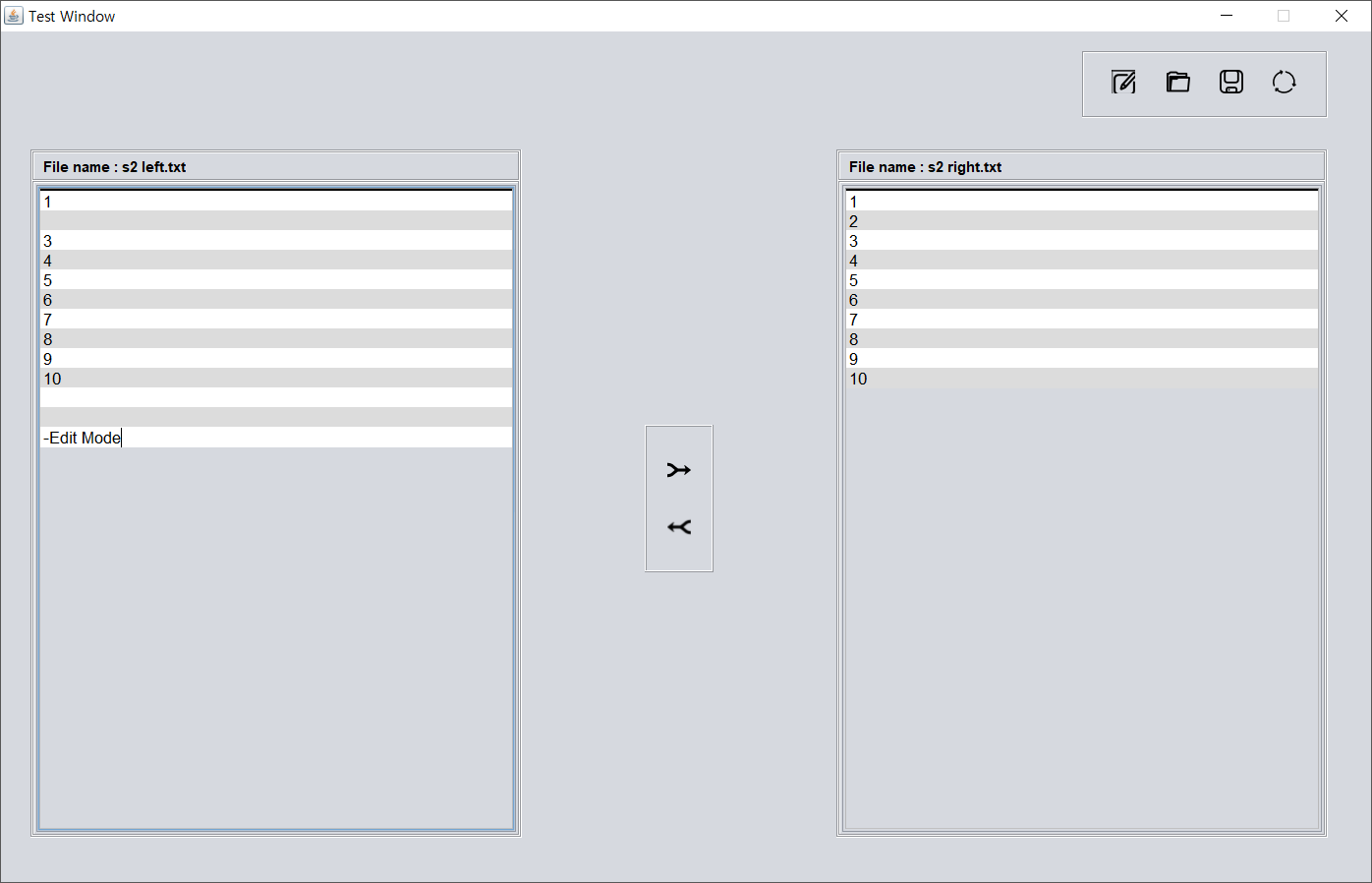
* Text file이 load 되어 있는 상태에서 새로운 text file 을 로드 할 수 있다.
* 이 때 load 창에 진입하기 전 위와 같은 경고 메시지를 출력한다.

<Loaded>



* Txt 파일이 load 된 화면이다.
* 왼쪽, 오른쪽 판넬에 각 txt 파일이 표시된다.
* 각 판넬의 위쪽에는 현재 Load 되어 있는 파일의 이름이 표시된다.
* 각 라인마다 배경 색을 다르게 하여, 가독성을 높임
* Txt 파일의 내용이 길면, 각 판넬의 우측에 scroll bar 가 생성된다.

<Edit>

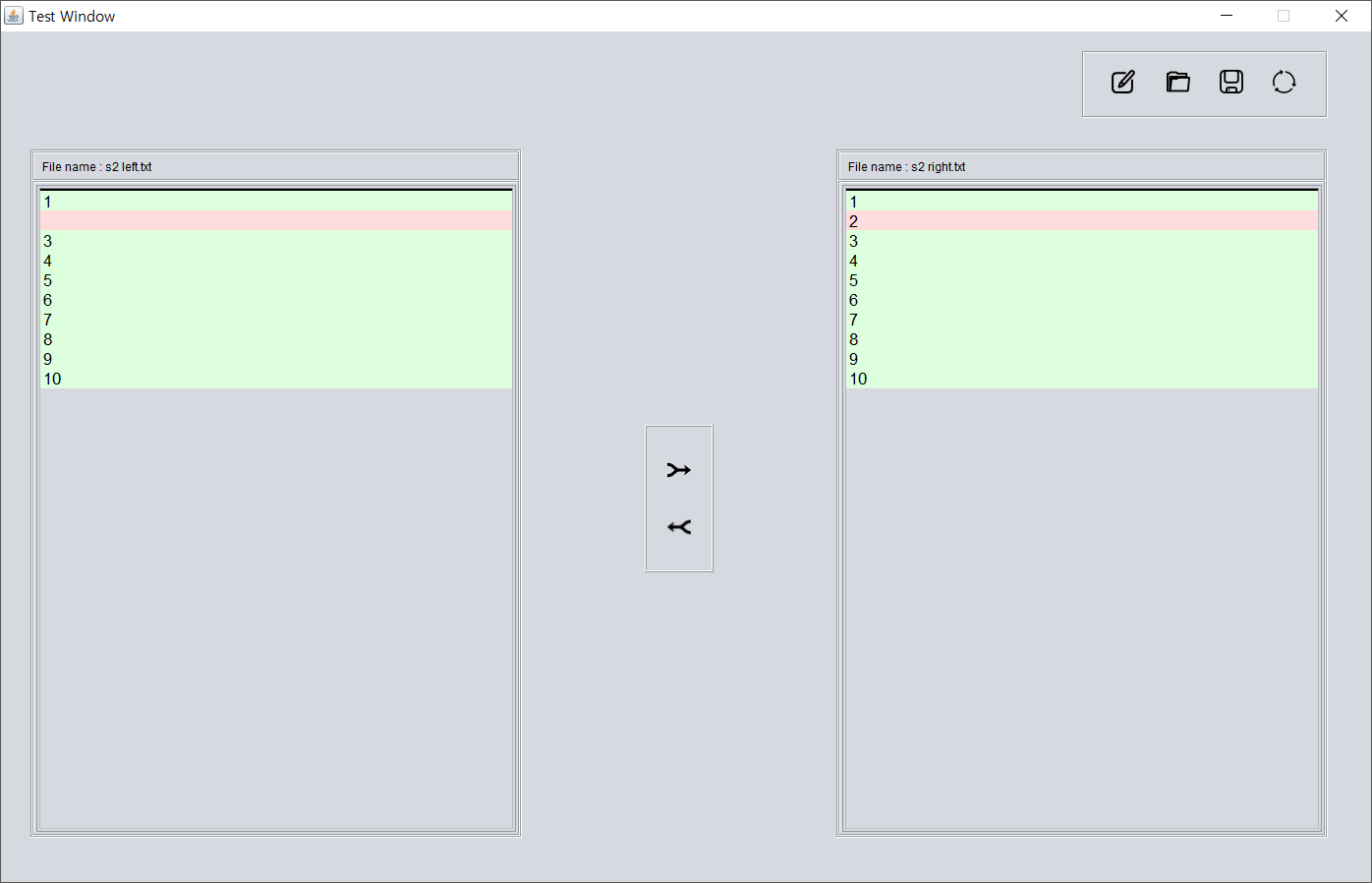


* 우측 상단의 Edit 버튼을 클릭해서 Edit Mode 로 진입할 수 있다.
* Edit 모드에서는, Txt file에 새로운 내용을 추가하거나 지울 수 있다.

(왼쪽 text 파일의 가장 아래에 새로운 내용이 추가된 것을 볼 수 있다.)

* Edit 모드를 종료 후 별도의 Save 과정이 필요하다.

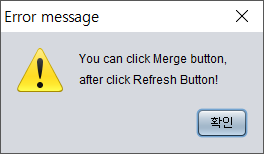
<Comparing(Refresh)>



* 파일이 load 된 상태에서 우측 상단의 refresh 버튼을 누르면 두 개의 txt 파일을 비교한다.
* 파일의 비교는 각 Line 별로 이루어 진다.
* 비교 후 내용이 같으면 녹색, 다르면 붉은 색으로 표시된다.
* 좌측 예시의 경우 두 개의 text file이 identical 해서 모든 라인이 녹색으로 표시된다.

우측 예시의 경우 중간의 한 줄이 누락되어서 붉은 색으로 표시되었다.

* Refresh 버튼을 누른 후 Merge 기능이 활성화된다.

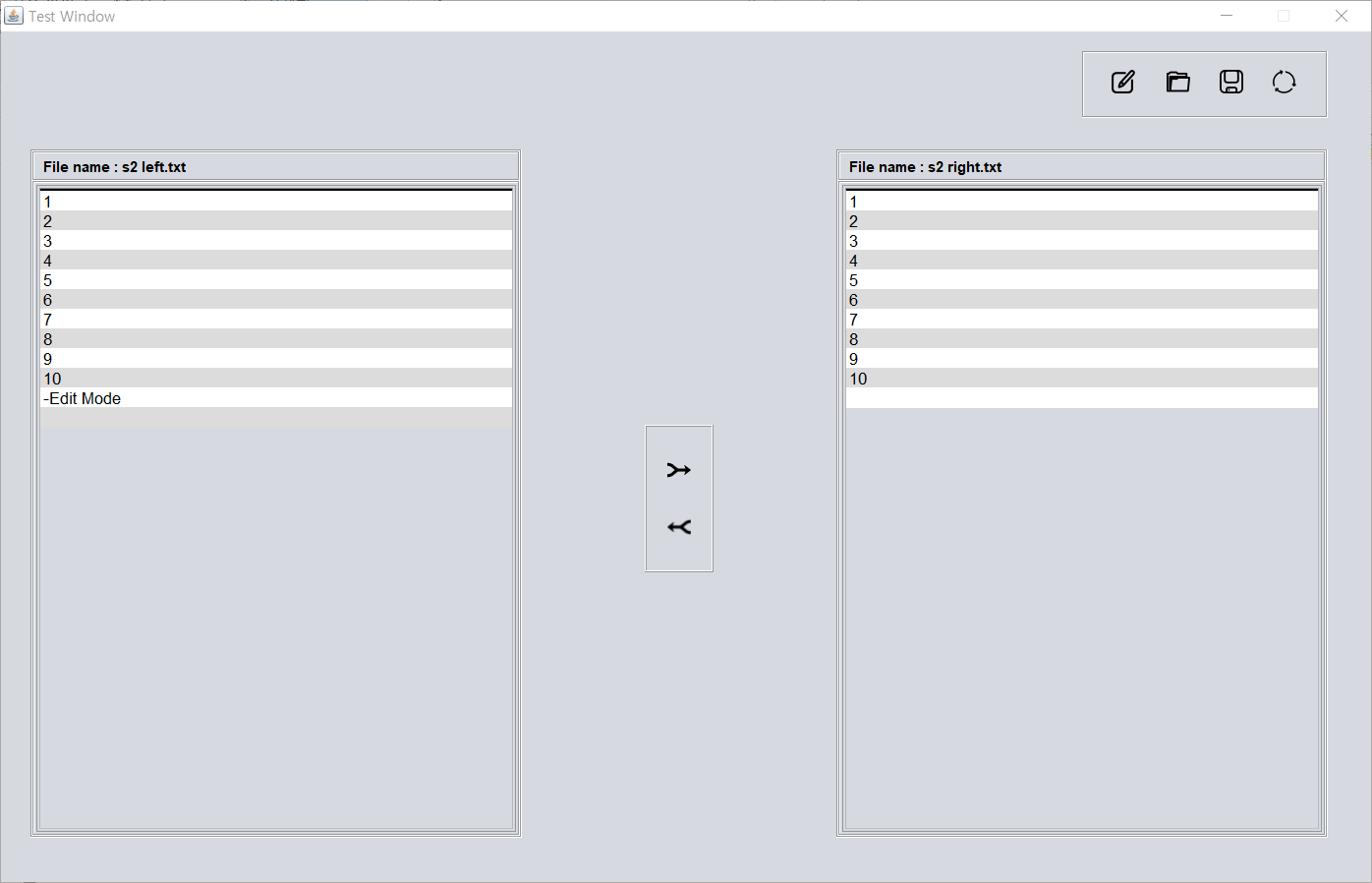


* Refresh 버튼을 누르기 전 Merge 기능을 사용하려 하면 다음과 같은 error message가

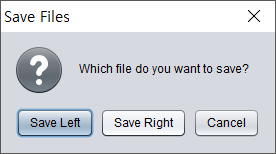
출력된다.

<Save>

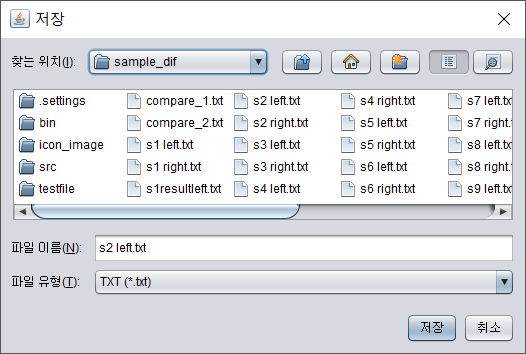
* Merge 나 Edit 등 text file 을 변경한 후 Save 버튼을 눌러 변경된 text file 을 저장할 수 있다.



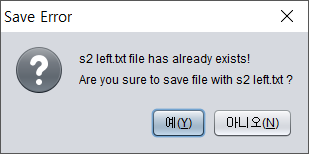
* 위의 예시는 왼쪽의 text file의 아래쪽에 새로운 내용을 추가하고 merge를 수행하였다.



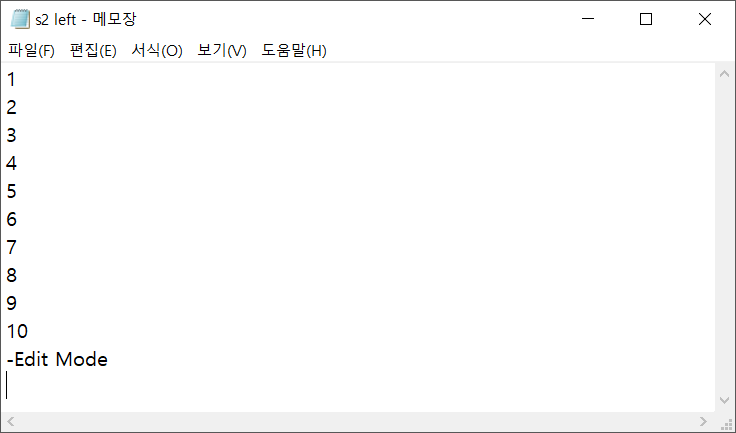
* Save 버튼을 누르면 저장할 파일을 지정하는 창이 출력된다.



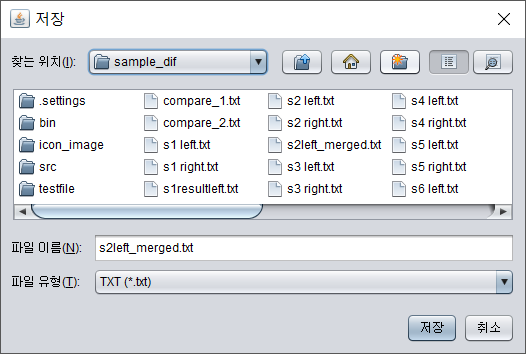
* 위의 예시의 왼쪽 파일을 저장하기를 선택한다.
* 기존 text file에 덮어쓸 수 도 있고, 새로운 text file 을 생성할 수도 있다.



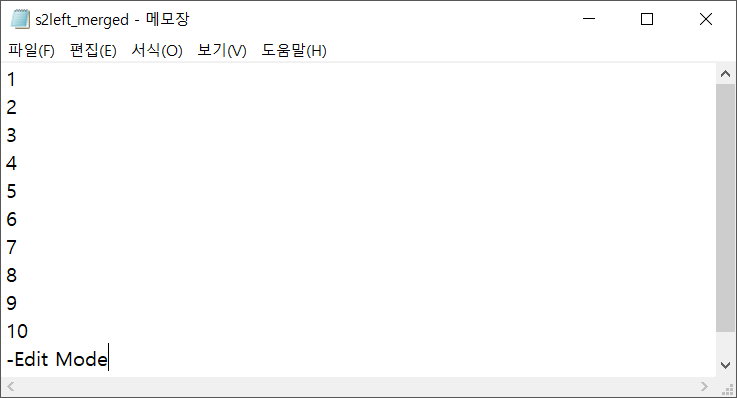
* 기존 파일을 덮어쓰기를 선택하면 위의 경고창이 출력된다.



* 덮어 쓴 후, text file 이 변경되었다.



* 저장 시 새로운 이름을 가진 text file 을 생성할 수도 있다.
* S2left\_merged.txt 라는 이름을 가진 새로운 파일을 생성하였다.

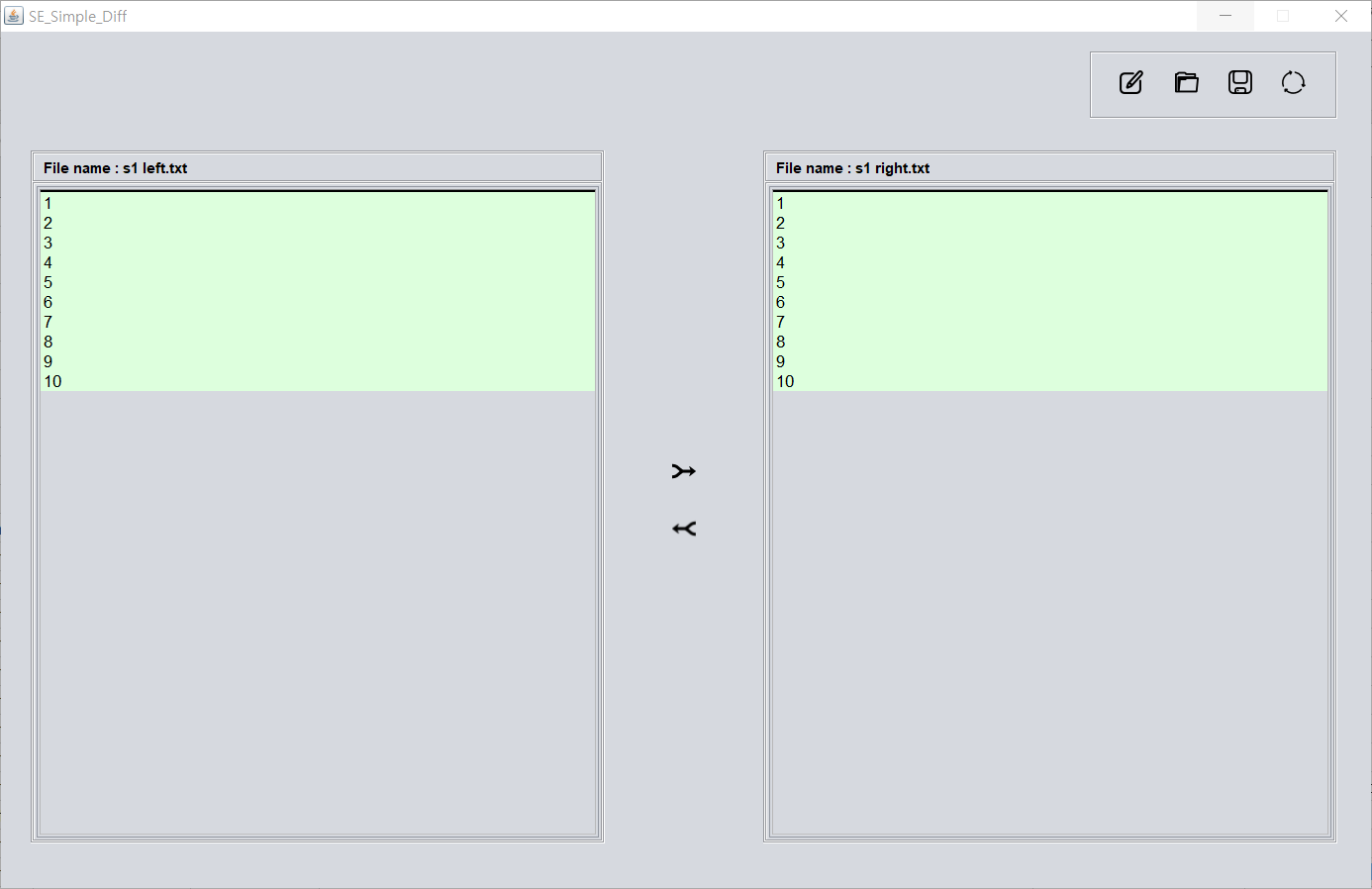


* 새로운 이름으로 저장 시, 해당 이름의 txt file 이 생성된다.

**<System test>**

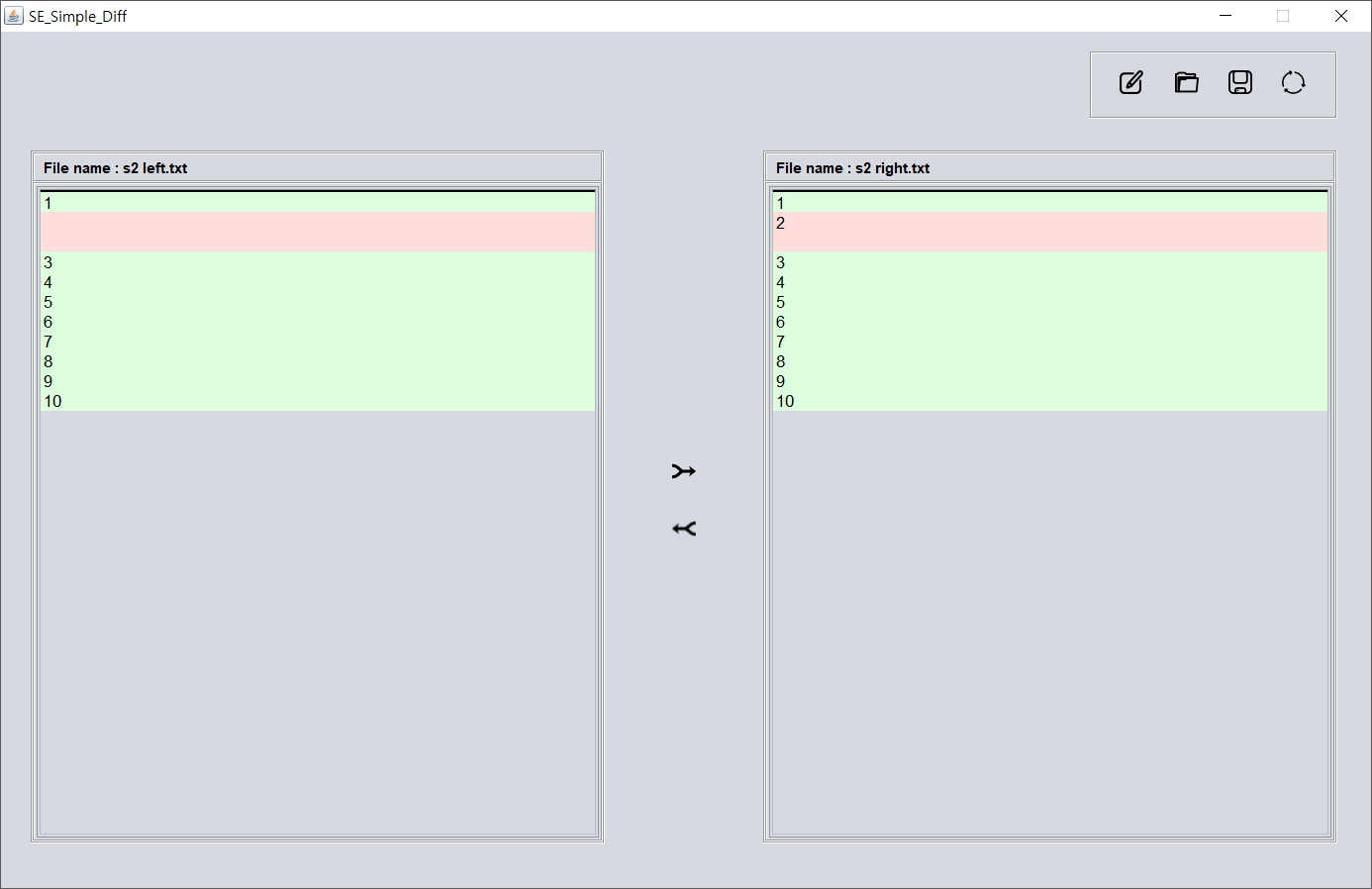
Test 는 Sample Data and check list 의 text file들을 이용하여 진행하였다.

* S1left.txt & s1right.txt



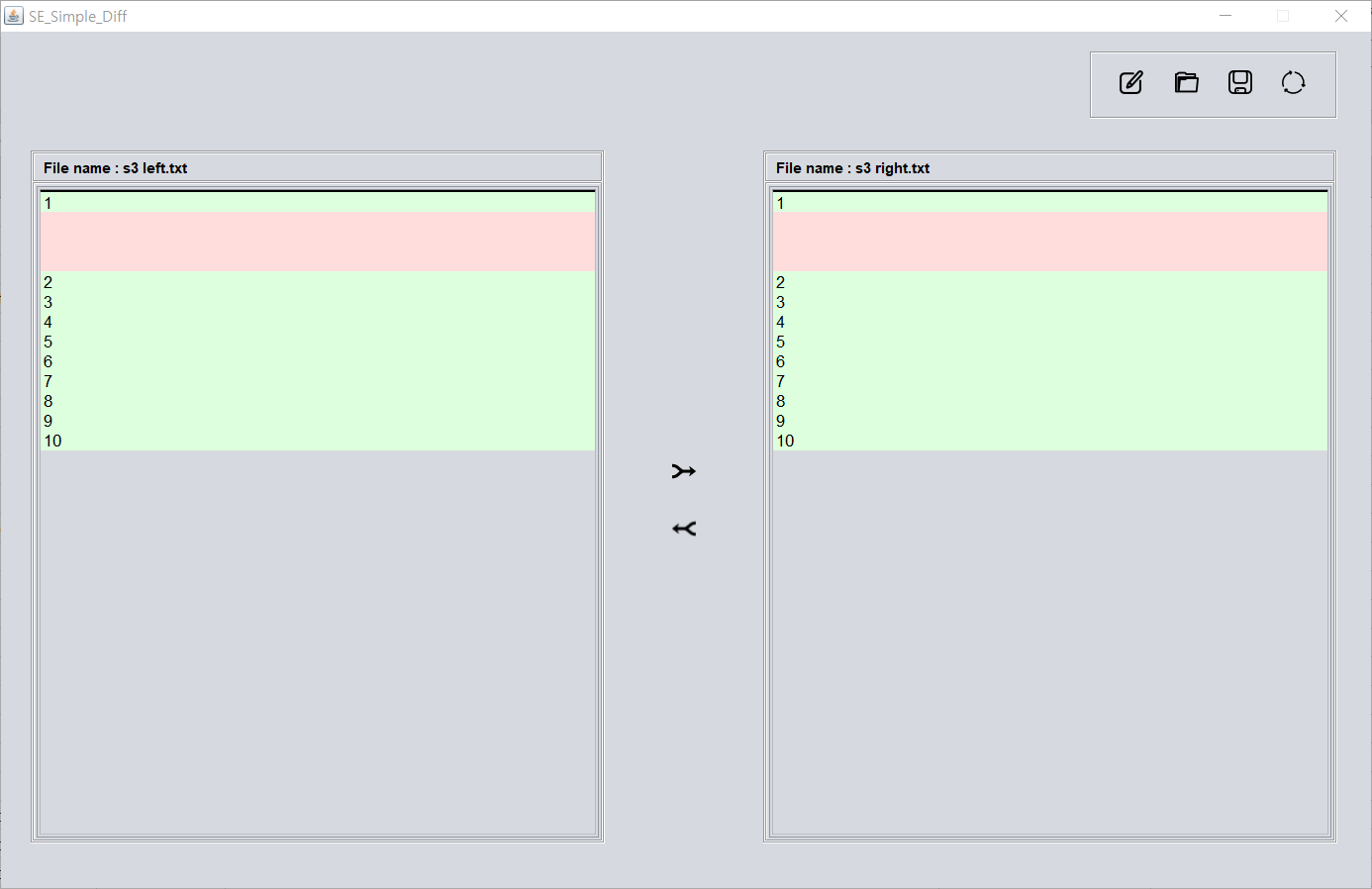
두 text 파일이 일치하므로 모든 line 이 identical 하다.

* S2left.txt & s2right.txt



S2left.txt에 두 번째 줄에 2가 누락되어 있다. 따라서 두 파일을 비교했을 때 “(blank block)”과 “2”가 다르기 때문에 두 번째 줄을 서로 다르다고 판단한다. 이 때 Comparing Algorithm이 왼쪽 파일을 기준으로 동작하므로 왼쪽 파일에 2를 삽입하기 위한 한 줄을 추가적으로 확보하여 총 두 줄이 붉은 색으로 표시된다.

* S3left.txt & s3right



Text의 내용은 모두 같지만 s3left.txt 파일의 1과 2 사이에 몇 개의 blank block이 삽입되어 있다. 따라서 s3right.txt 파일에 blank block 만큼의 line 을 확보하고 붉은색으로 표시한다.

* S4left.txt & s4right.txt



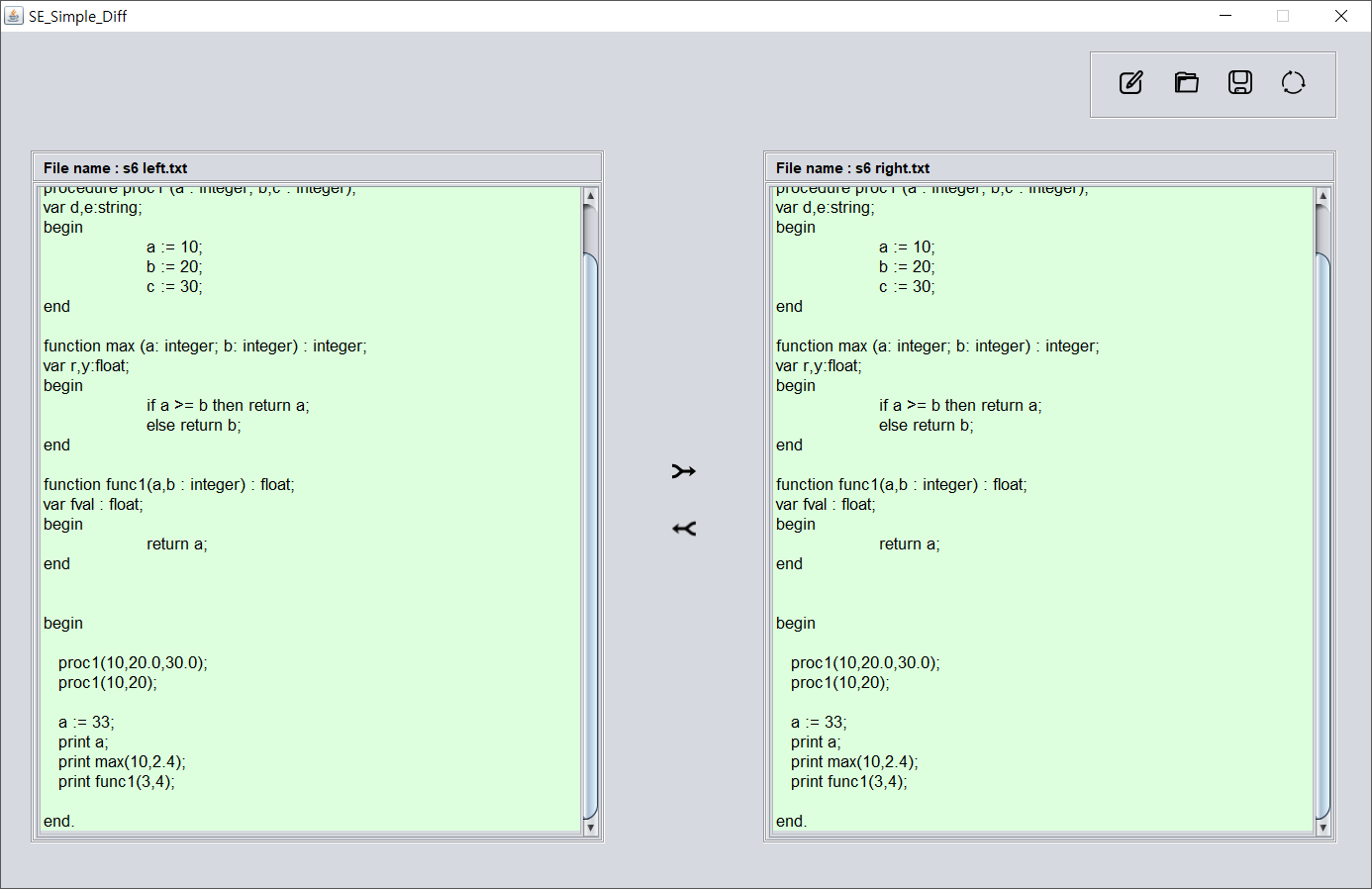
S3left.txt 파일과 같이 s4left.txt 파일에 blank block들이 삽입되어 있다. 따라서 s4right.txt 파일에 blank block을 삽입하기 위한 line 을 확보하고 이를 붉은색으로 표시한다.

* S5left.txt & s5right,txt



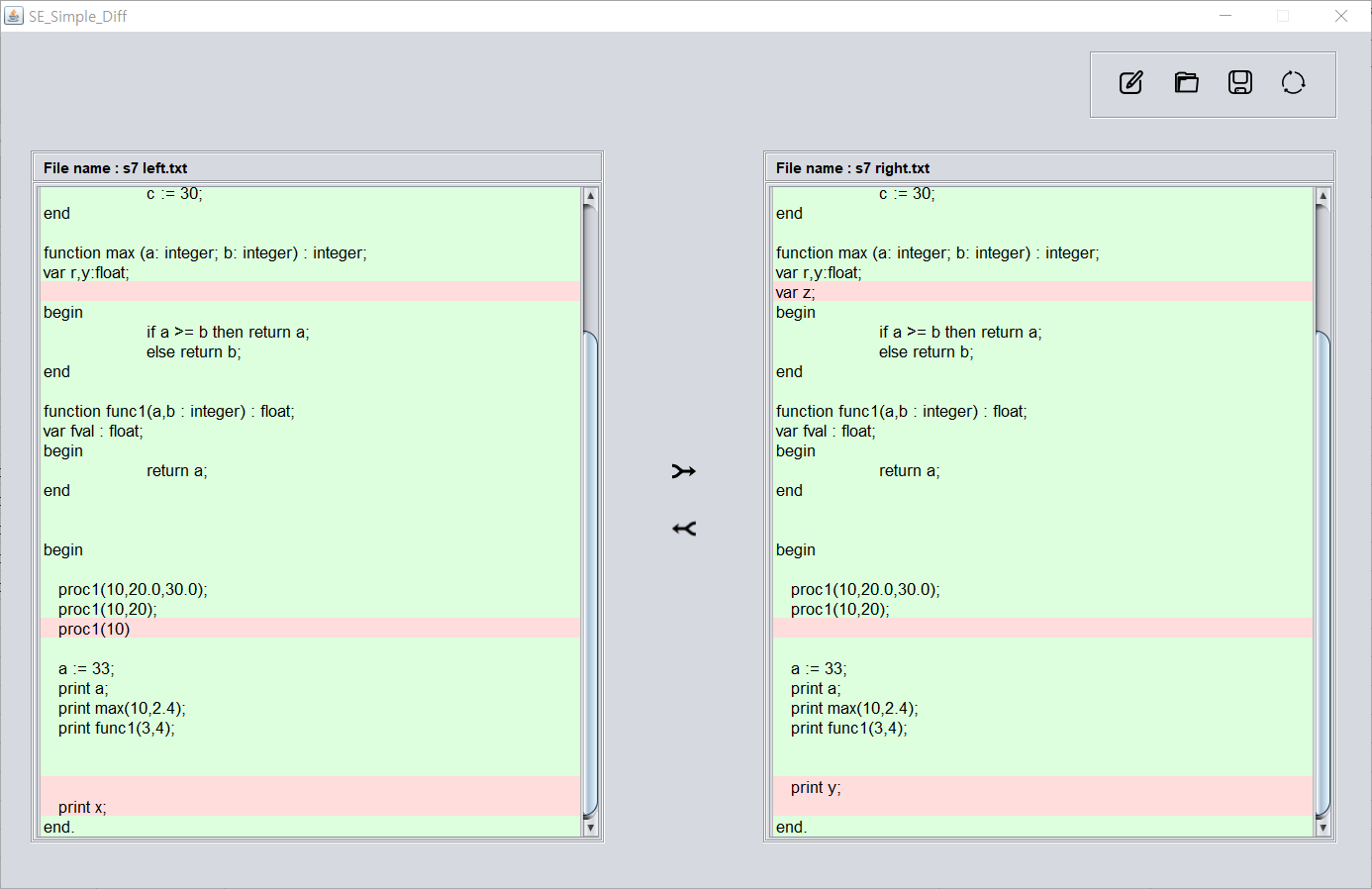
S5left.txt. 파일에 blank block과 여러 개의 숫자가 추가로 삽입되어 있다. 따라서 s5right.txt 파일에 숫자들과 blank block을 삽입하기 위한 line을 확보하고 붉은 색으로 표시한다.

* S6left.txt & s6right.txt



두 text 파일이 일치하므로 모든 line 이 identical 하다.

* S7left.txt & s7right.txt



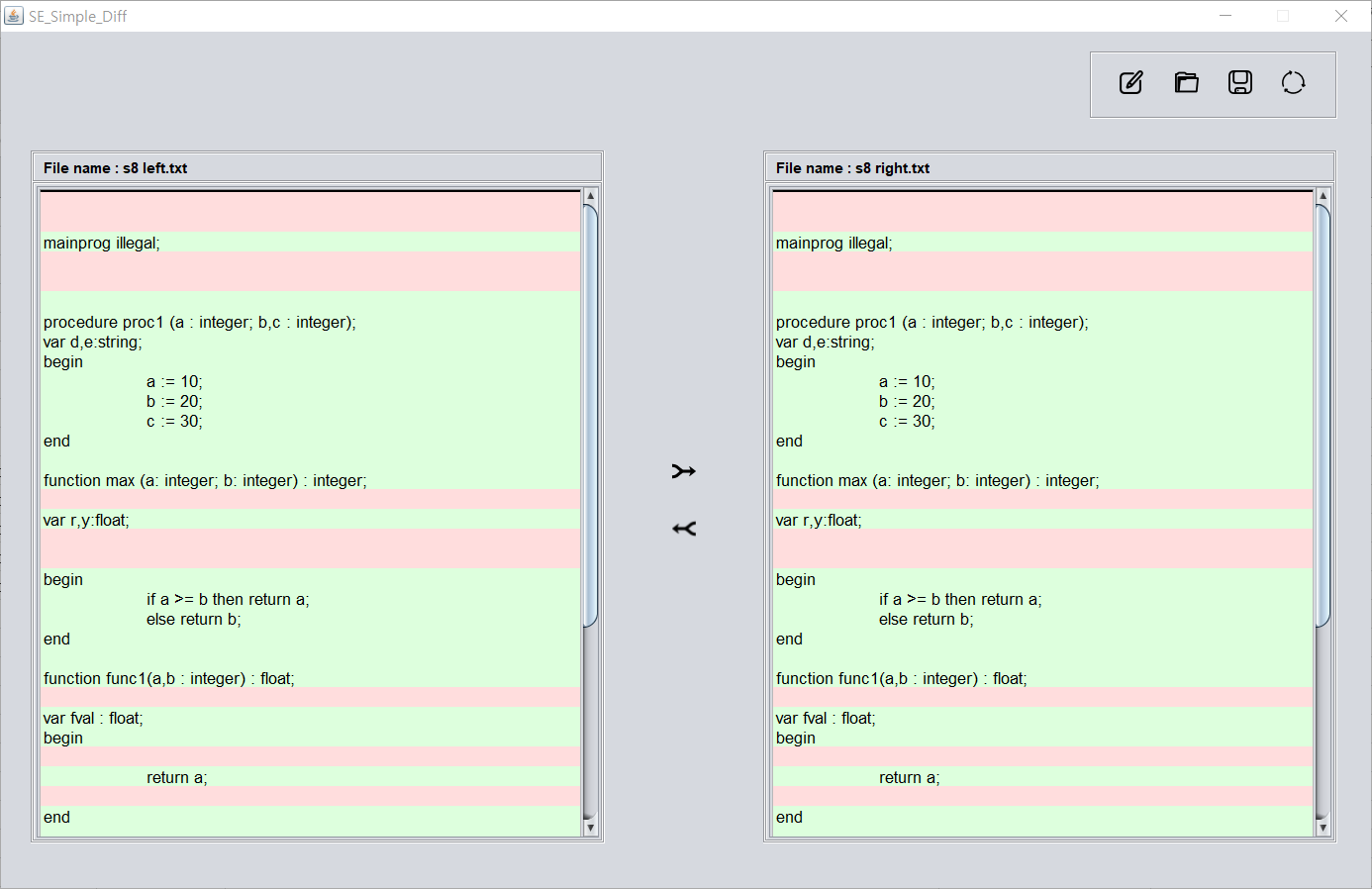
두 파일의 세 번째 line 이 서로 다르다. 이 때 오른쪽 파일의 text를 왼쪽 파일에 삽입하기 위해 한 줄을 추가적으로 확보한다. 따라서 총 두 줄이 붉은 색으로 표시된다.

왼쪽 파일의 13번째 line에 한 line이 누락되어 있다. 이 때 왼쪽 파일의 blank에 오른쪽 파일의 text를 삽입할 수 있으므로, 추가적으로 line을 확보하지 않는다. 따라서 한 줄이 붉은 색으로 표시된다.

오른쪽 파일의 30 번째 line에 한 line이 누락되어 있다. 따라서 한 줄을 붉은색으로 표시된다.

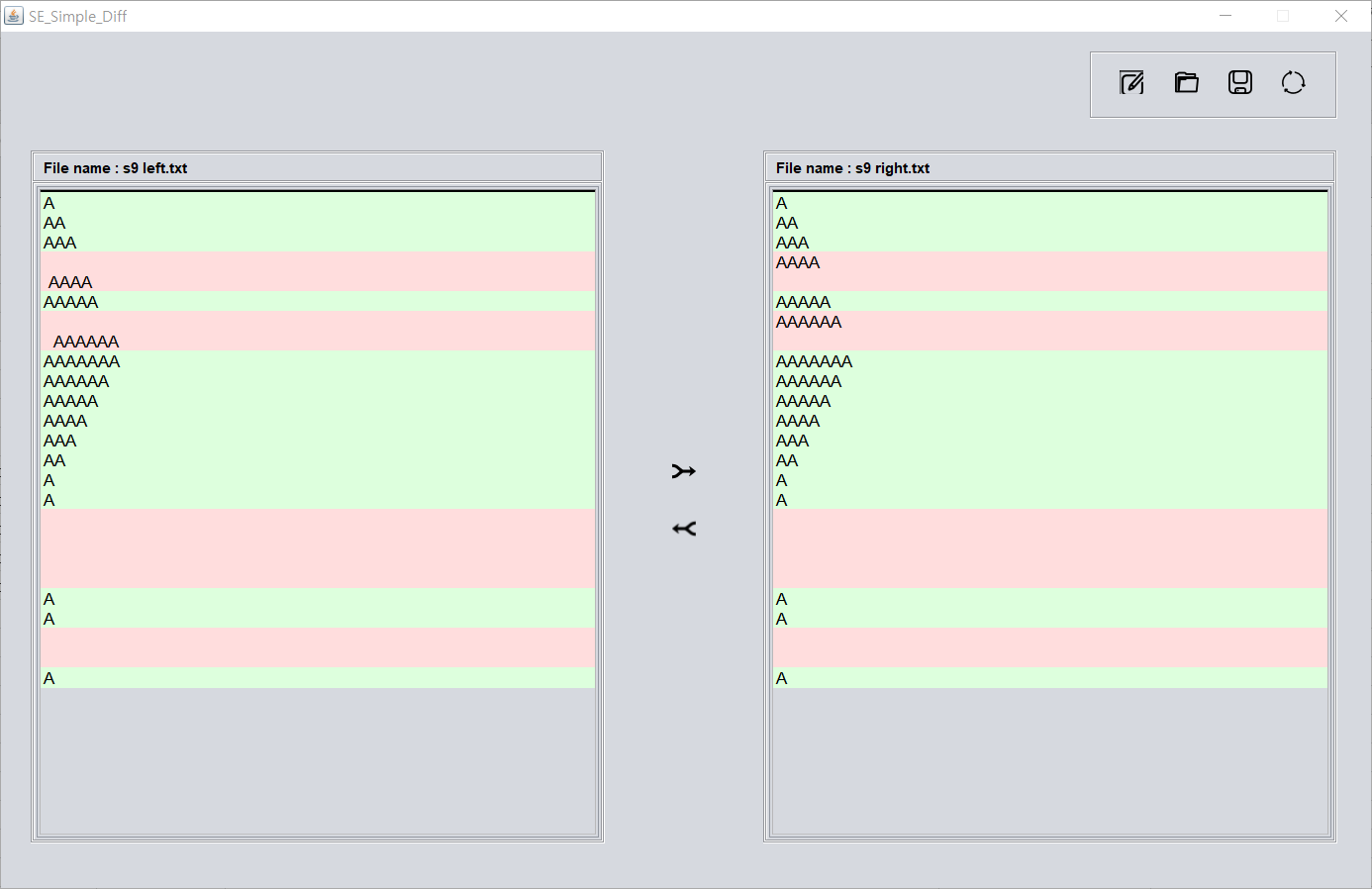
두 파일의 38번째 line 이 서로 다르다. 이 때 오른쪽 파일의 text를 왼쪽 파일에 삽입하기 위해 한 줄을 추가적으로 확보한다. 따라서 총 두줄이 붉은 색으로 표시된다.

* S8left.txt & s8right.txt



두 파일의 text는 일치하지만 왼쪽 파일에 blank block이 삽입되어 있다. 따라서 오른쪽 파일에 blank block 만큼의 line을 확보하고 붉은색으로 표시한다.

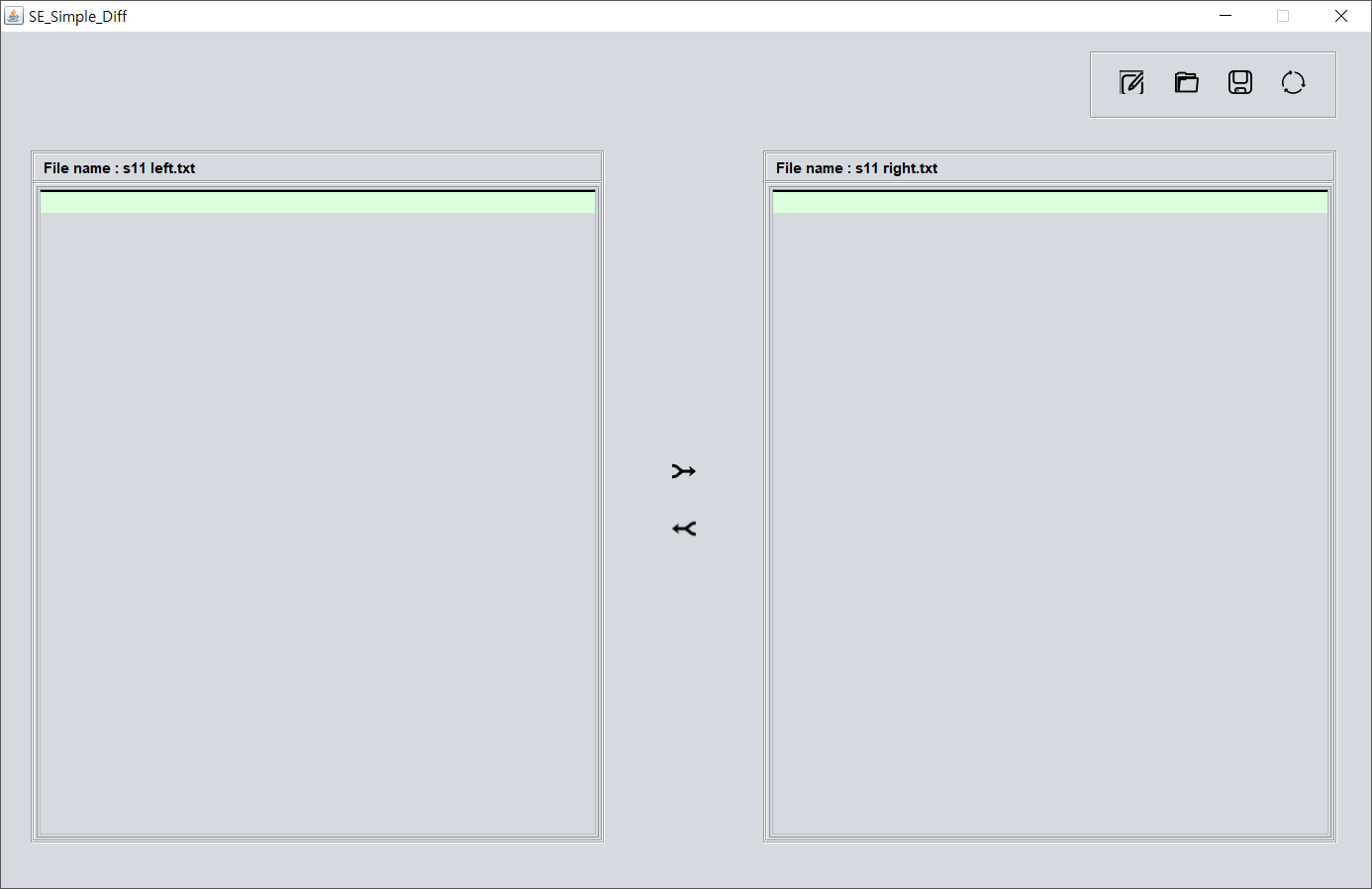
* S9left.txt & s9right.txt



두 파일의 3,5번째 줄이 서로 다르다. 이 때 오른쪽 파일의 text를 왼쪽 파일에 삽입하기 위해 각 한 줄씩 추가적으로 확보한다. 따라서 총 네 줄이 붉은 색으로 표시된다.

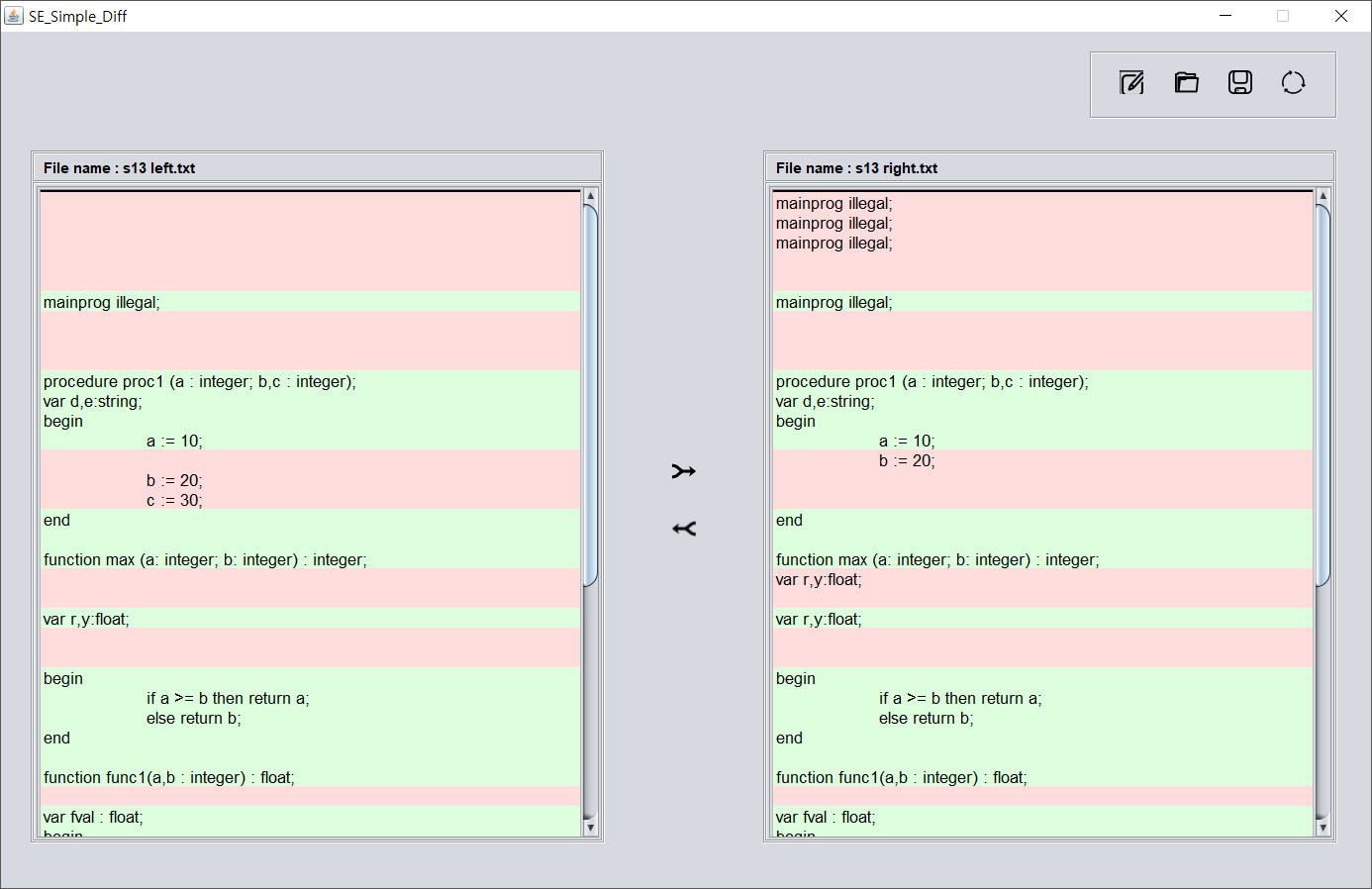
왼쪽 파일에 blank block들이 삽입되어 있다. 따라서 오른쪽 파일에 blank block을 삽입하기 위한 line을 확보하고 이를 붉은 색으로 표시한다.

* S11left.txt & s11right.txt



두 text 파일이 blank 파일로 서로 일치하므로 identical 하다.

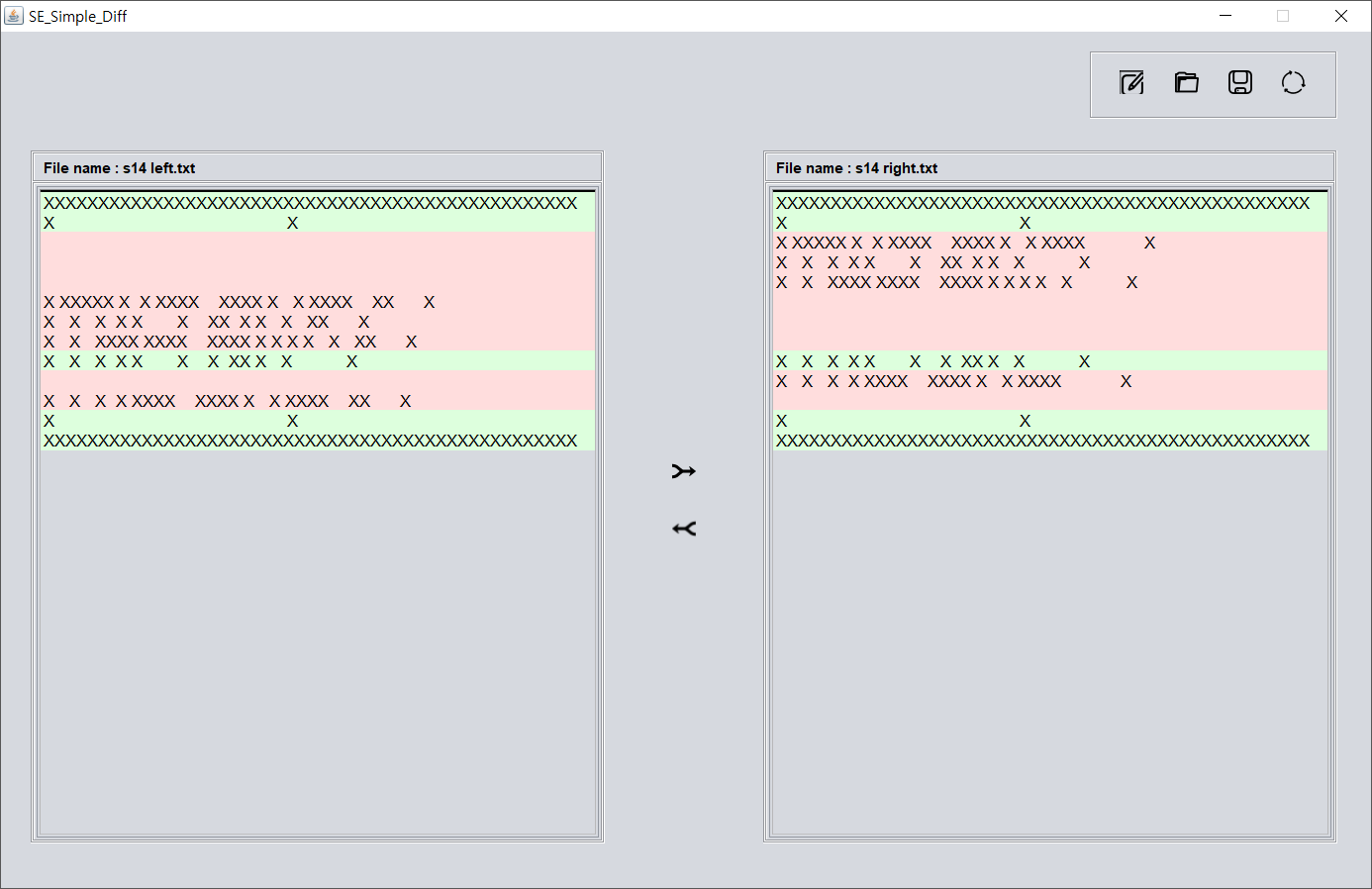
* S13left.txt & s13right.txt



두 파일을 비교했을 때, 양쪽 파일에 서로 누락된 내용이 존재한다. 이러한 내용을 삽입하기 위한 line을 확보하고 붉은 색으로 표시한다.

또한 blank block이 존재할 경우 blank block을 삽입할 line 을 확보하고 이를 붉은색으로 표시한다.

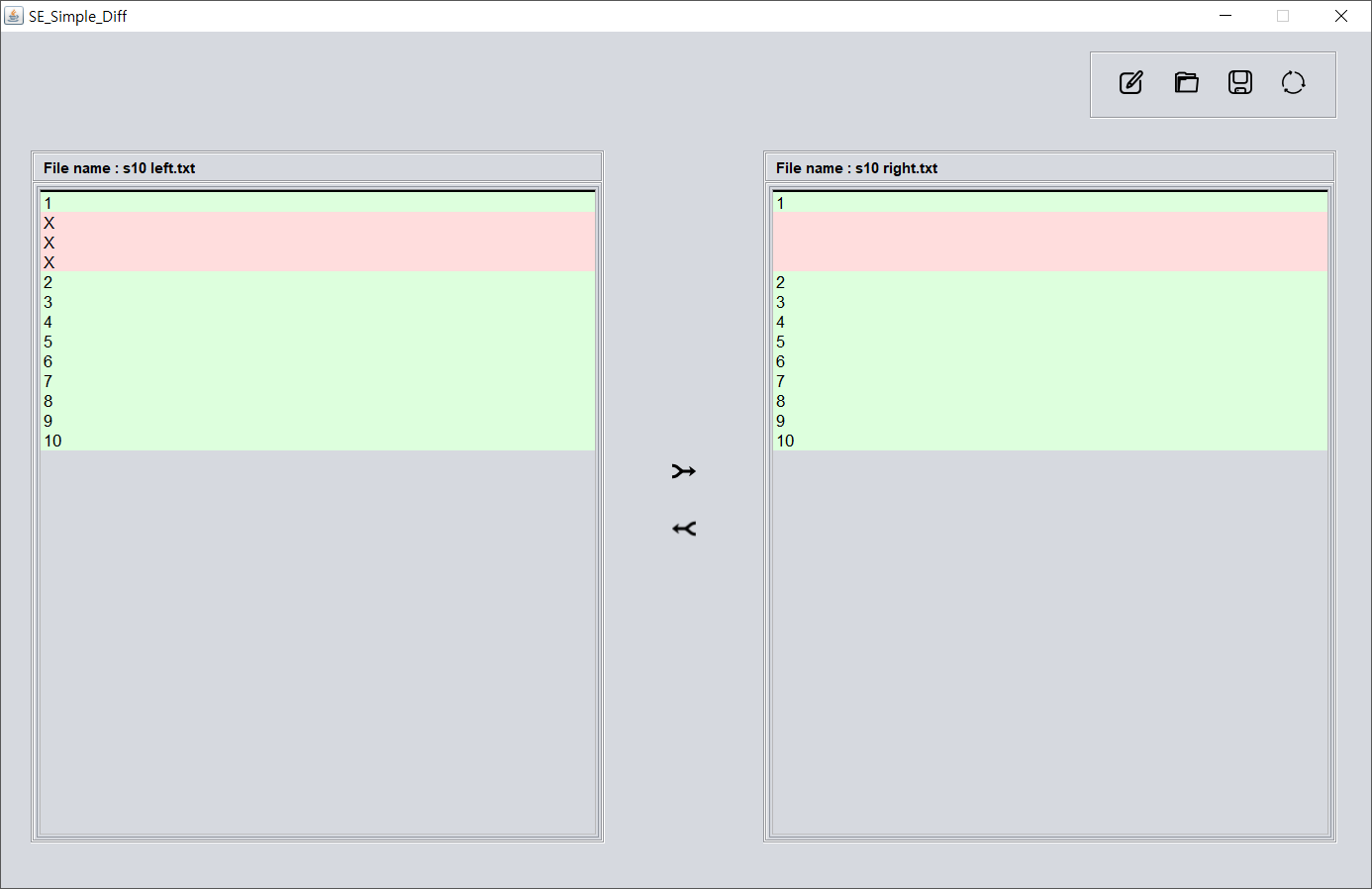
* S14left.txt & s14right.txt



두 파일을 비교했을 때, 3~5번 line과 7번 line 이 서로 다르다. 이 때 다른 부분을 왼쪽 파일에 삽입하기 위해 각각 3줄, 1줄을 추가로 확보하고 이를 붉은색으로 표시한다.

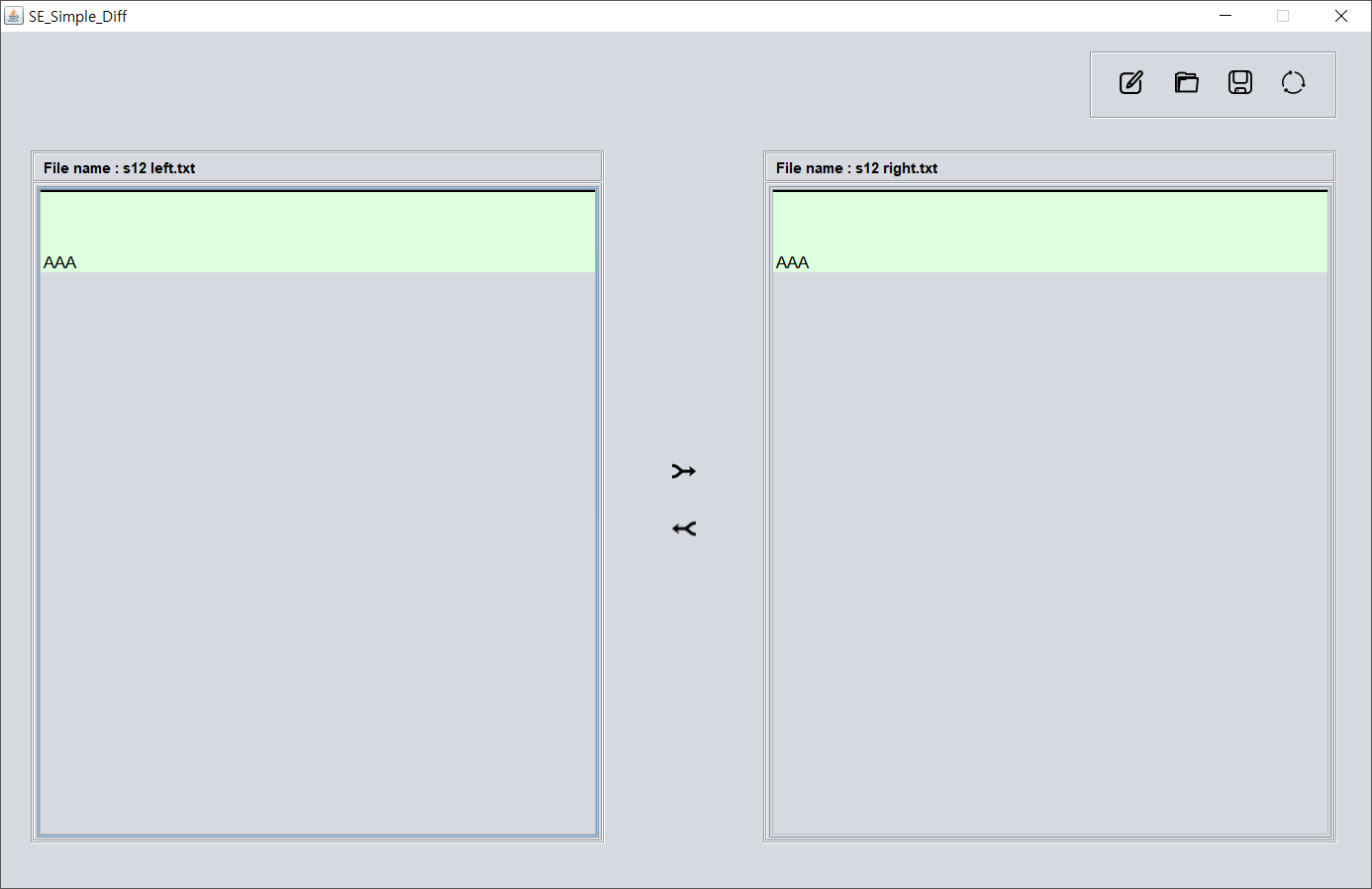
***<ERROR>***

* S10left.txt & s10right.txt



S10left.txt, s10right.txt 의 경우 Compare는 정상동작한다. Merge function의 경우 XXX text 가 다른 쪽의 blank block으로 삽입 가능하지만 blank block이 맞은편으로 삽입되지는 않는다.

* S12left.txt & s12right.txt



S12left.txt & s12right.txt 의 경우 text “AAA” 아래의 blank block 이 인식되지 않는다.