

**Software Engineering Class01 term project**

**SimpleMerge Team7**

**Analysis & Design, Implement and Test Documents**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Team Leader 김정민 |  | Team Member  김상헌, 양종만, 이동현, 임기찬, 황선준 |  | Writer  양종만, 임기찬, 황선준 |

COTENTS

1. [Analysis & Design Doucuments 1](#_Toc349734314)

Analysis

[Design](#_Toc349734318)

1. [Implementation 2](#_Toc349734319)

[Usecase](#_Toc349734320) Diagram

[Usecase Descriptions](#_Toc349734324)

1. Test [3](#_Toc349734331)

[Quality Requirements](#_Toc349734337)

[Constraints](#_Toc349734338)

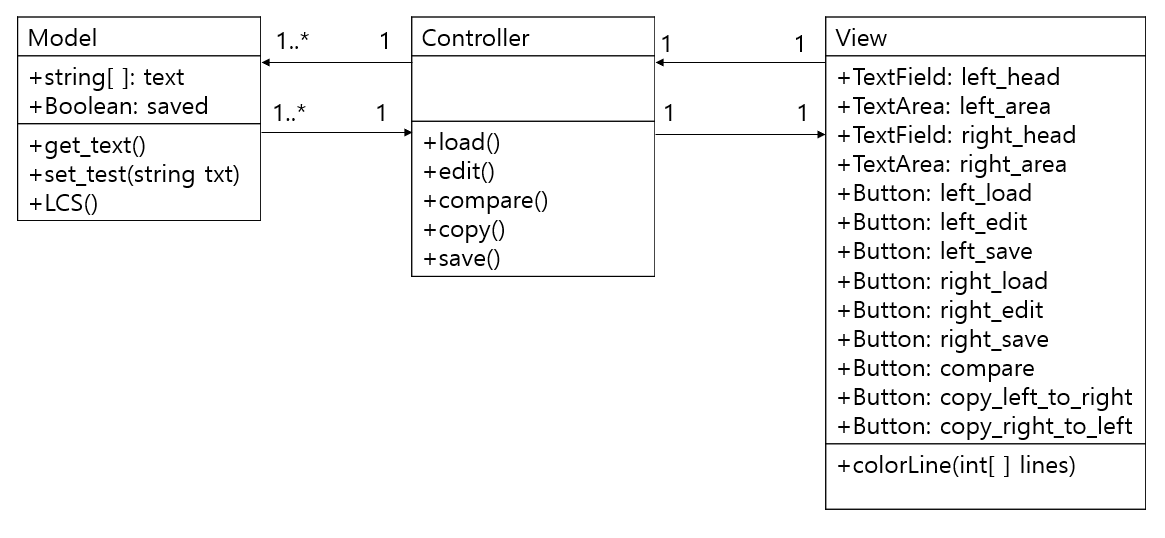
# Analysis & Design

## Analysis

SimpleMerge 프로그램은 text파일을 불러오거나 직접 입력하여 그 내용을 비교, 비교 결과를 토대로 내용의 복사가 가능하며, 복사를 포함한 편집 결과를 또다른 텍스트 파일로 저장할 수 있게 해준다. 그리하여 필수적으로 구현되어야 할 기능은 다음과 같이 선정하였다.

* 텍스트 불러오기 (load)
* 텍스트 편집하기 (edit)
* 텍스트 비교하기 (compare)
* 텍스트 저장하기 (save)

## Design



MVC 모델을 기반하여 SimpleMerge 프로그램을 디자인 하였다. 사용자가 Controller를 조작하면 Controller는 Model을 호출하여 데이터를 가져오고, 이 정보를 View에 보내 View에서 시각적인 표현을

제어하여 사용자에게 보여준다.

1. **Controller**

View와 Model의 상호 작용을 관리한다. 사용자가 View를 통해 Controller를 조작하면 Controller는 대응하는 정보, 또는 메소드의 결과를 Model에서 가져와 View에 보낸다.

Load(): Model에서 해당하는 파일의 정보를 가져와 View에 보낸다.

Edit(): 사용자가 View에서 Edit 버튼을 누르면 Model의 text와 연동되어 View에서 파일이 변할 때 같이 변한다.

Compare(): Model의 LSC() 메소드를 실행시키고 그 결과를 View에 보낸다. 메소드 실행결과, 변동사항이 있다면 isCompared를 true로 set한다.

Copy(): isCompared가 true라면 선택된 영역의 정보를 맞은편의 대응하는 영역에 복사한다.

Save(): Model에서 저장할 정보를 해당하는 파일 위치에 저장한다.

1. **Model**

Model엔 파일을 저장한다. Compare의 결과 변동이 있으면 isCompared를 true로 set하며, copy의 결과 양쪽이 같아지면 false로 set한다. LCS는 Model에 포함되어, Controller에서 compare를 진행하고자하면 Model에서 자체적으로 LCS를 수행하여 결과를 저장한다.

Get\_text(): Model 안에 있는 text를 가져오는 메소드. 파일을 저장할 때 사용한다.

Set\_text(String txt): Model에 text를 set하는 메소드이다.

LCS(): LCS를 수행하는 메소드

1. **View**

유저에게 보여지는 부분들이다. TextField와 TextArea는 Text를 위한 영역이며, Button들은 load, edit, save, compare, copy에 쓰일 버튼들이다.

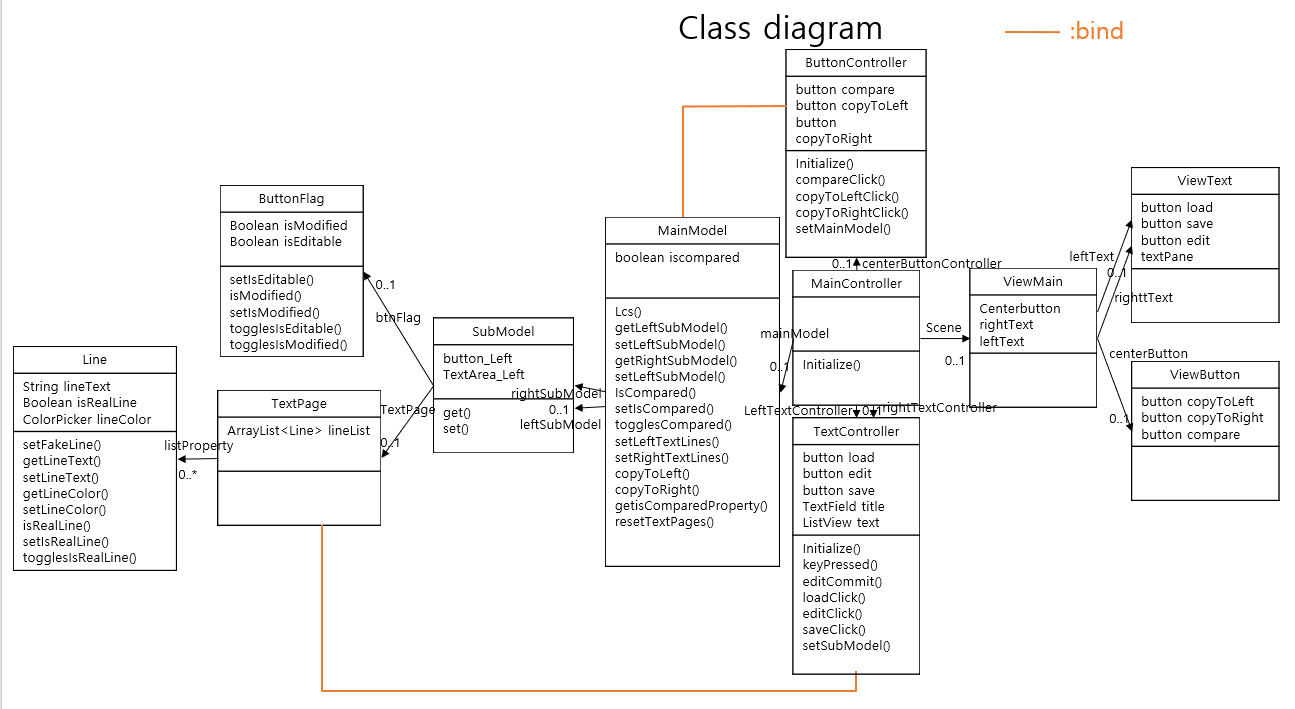
ColorLine(int[] lines): line을 색칠하는 method이다.

# Implementation

## Implementation log

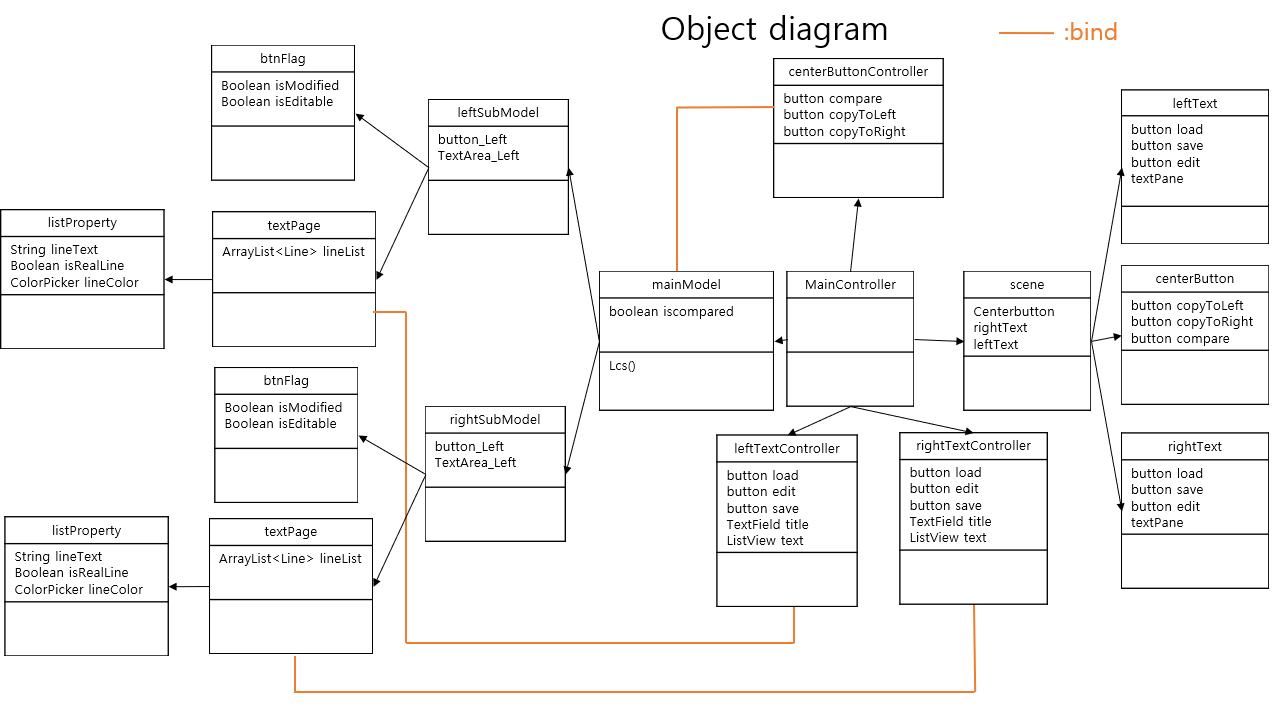
|  |  |  |
| --- | --- | --- |
| **순서** | **구현 내용** | **날짜** |
| 1 | View 및 View Controller 프로토타입 구현 | 05.16 |
| 2 | 파일 입출력 프로토타입 구현 | 05.17 |
| 3 | MODEL 프로토타입 구현 | 05.19 |
| 4 | MODEL의 클래스인 Line, ButtonFlag, MainModel 구현 | 05.19 |
| 5 | MODEL의 클래스였던 Text.java 삭제 | 05.19 |
| 6 | Css 파일, fxml css 적용 | 05.23 |
| 7 | LCS 프로토타입 구현 | 05.23 |
| 8 | View 분할 | 05.24 |
| 9 | textArea를 ListView로 변경, TextController에서 TextPage를 사용. Bind를 위해 TextPage 수정. | 05.27 |
| 10 | LineTest, SubModelTest 추가. Line, Button, SubModel 클래스 변경 | 05.28 |
| 11 | Line : LineColor 추가. Toggle 기능 추가  ButtonFlag : saveFlag, editFlag이 isSaved, isEdited로 수정됨. Toggle 기능 추가  MainModel : CF가 isCompared로 변경. Toggle 기능 추가 | 05.29 |
| 12 | LineColor가 JAVAFX를 사용하도록 변경.  Line 클래스에서 fakeline을 위한 생성자와 메소드가 추가됨.  SubModel과 TextPage에 메소드 추가 | 05.29 |
| 13 | Enter키 입력하면 다음줄로 넘어가능 기눙 추가. 빈 줄에 backspace를 입력하면 줄이 지워지는 기능 추가 | 05.29 |
| 14 | isModified Boolean 추가 | 06.02 |
| 15 | isSaved 삭제 | 06.02 |
| 16 | load를 실행할 때 변경사항 있을 시 안내문 뜨는 기능 추가 | 06.02 |
| 17 | ButtonFlagTest, TextPageTest 구현. | 06.02 |
| 18 | LCS 및 LCStest 추가. Model에 변경사항들 있음 (setTextLines, getTextLines등…) | 06.02 |
| 19 | TextPage에서 Save를 위한 getTextField 기능 추가 | 06.03 |
| 20 | Line에서 Color 세팅 | 06.04 |
| 21 | Line의 getColor 메소드 수정 | 06.04 |
| 22 | LCS에서 불필요한 코드 삭제 및 LCStest 최신화 | 06.04 |
| 23 | TextPage에서 addLineText(String str), addFakeLine() 추가 | 06.04 |
| 24 | LCS 버그 수정 | 06.04 |
| 25 | Model에서 Copy하도록 추가 | 06.04 |
| 26 | LCS 리팩토링 | 06.04 |
| 27 | LCS 심각한 버그 수정 | 06.04 |
| 28 | 입출력 받는 파일의 인코딩을 UTF-8으로 변경 | 06.05 |
| 29 | LineTest 수정, TextPageTest 구현 | 06.05 |
| 30 | TextPageTest에 testIsRealLineModification, testLoadAndSaveFuncInTextPage() 메소드 추가 | 06.05 |
| 31 | MainModelTest 구현 | 06.06 |
| 32 | LCS 버그 수정 및 LCStest 최신화 | 06.06 |
| 33 | MainModel의 에러 처리 추가 | 06.06 |
| 34 | SubModelTest 최신화 | 06.06 |
| 35 | MainModel 리팩토링 | 06.06 |
| 36 | Line 클래스에서 발견된 에러 수정 | 06.06 |
| 37 | MainModel copy 메소드 변경 | 06.07 |
| 38 | MainModelTest 추가 | 06.07 |
| 39 | ControllerTest 추가 | 06.07 |
| 40 | MainModel에 reset 메소드 추가 | 06.07 |
| 41 | TextController 최신화 | 06.07 |

## Class Diagram

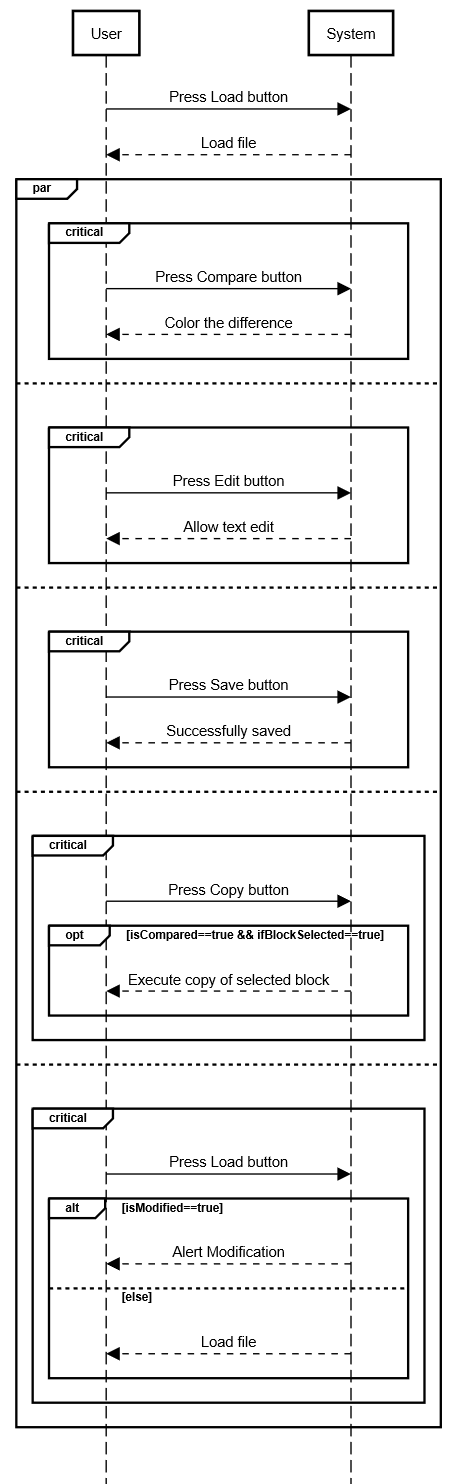


Implementation이 끝난 후 최종적인 Class Diagram의 모습이다.

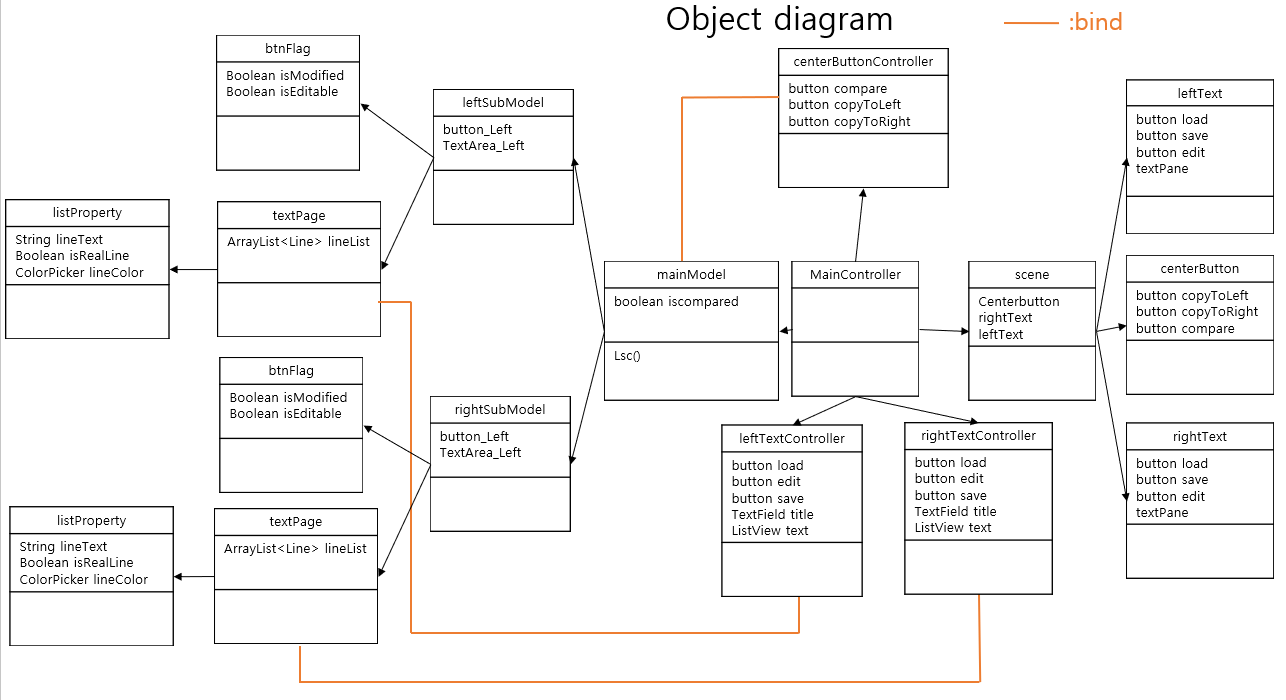
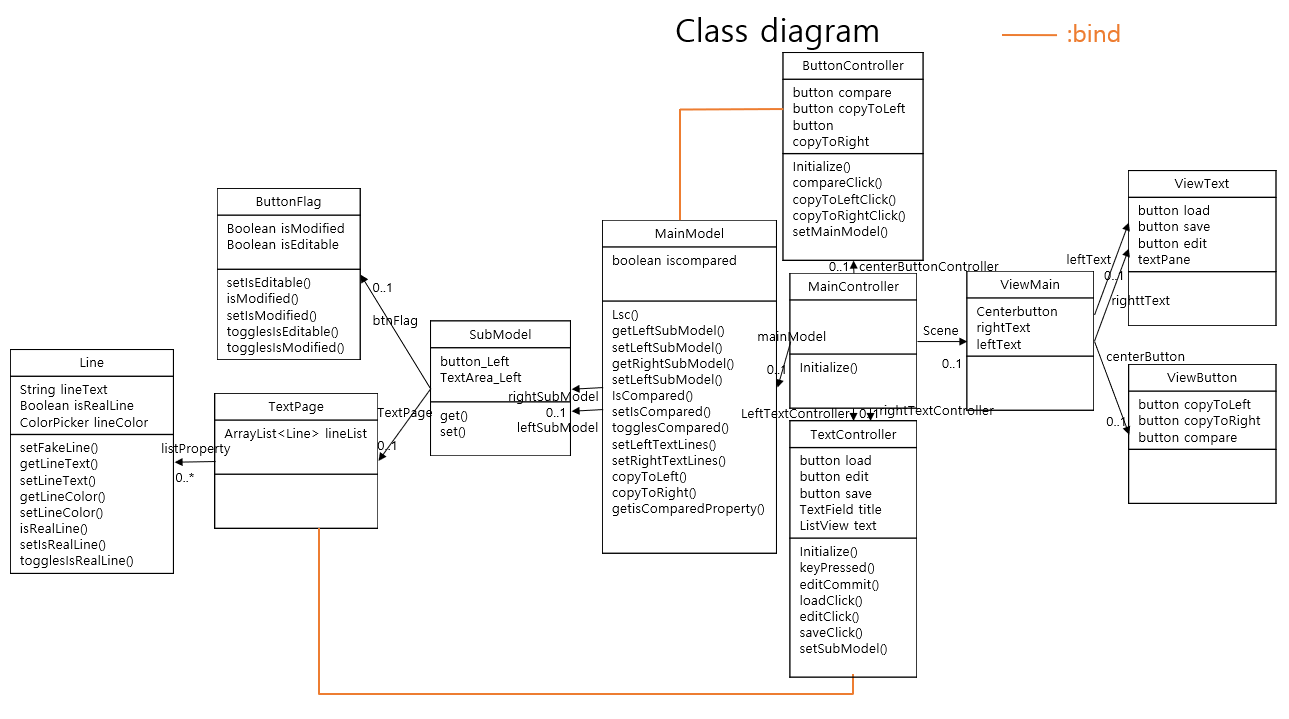
## Object Diagram



## Sequence Diagram

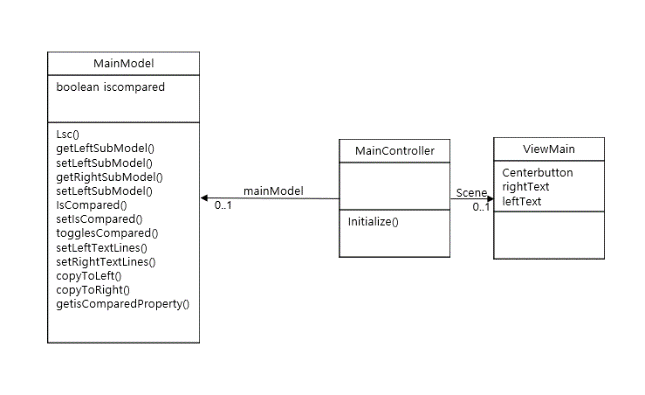


## Object Oriented Design Principle



1. Single Responsibility Principle

SimpleMerge 기능은 크게 3개로 나뉜다. 사용자에게 보여지는 View, 데이터를 저장하는 Model, 이 두 사이에서 데이터의 흐름을 담당하는 Controller로 나눌 수 있다. 따라서 SimpleMerge는 MVC에 따라 각각의 기능이 분리된다.



1. Open-Closed Principle

Model 내의 SubModel 클래스는 Simple Merge의 load, edit, save 와 텍스트 창을 담당하는 클래스이다. 이 클래스는 파일을 받아와 텍스트 창에 띄우고 사용자는 버튼을 사용하여 파일을 변경할 수 있다. SubModel의 역할은 중앙의 compare, copy버튼과 edit, save, load 버튼이 포함된 두개의 텍스트 창으로 나뉠 수 있다. 이 두 역할을 Button Flag와 Text Page 클래스 두개로 나누어 SubModel에서 불러온다. 또한, Text Page 내의 텍스트를 저장하기 위해 Line이란 class를 만들어 불러온다.

public class SubModel {

private ButtonFlag btnFlag;

private TextPage text;

public TextPage getTextPage() {

|  |
| --- |
|  |
|  | return this.text; |
|  | } |
|  | public ButtonFlag getButtonFlag() { |
|  | ButtonFlag ret = this.btnFlag; |
|  | return ret; |
|  | } |

1. Liskov Substitution Principle

Simple Merge에서 텍스트 창은 왼쪽과 오른쪽에 각각 나타나고 동일한 작업을 수행한다. 따라서 edit, save, load 버튼과 텍스트 창이 포함된 SubModel class를 선언하여 Left SubModel과 Right SubModel을 객체화 시킨다. 또한, View에서도 ViewText class를 선언하여 Left Text와 Right Text로 객체화 시켜 보여진다.

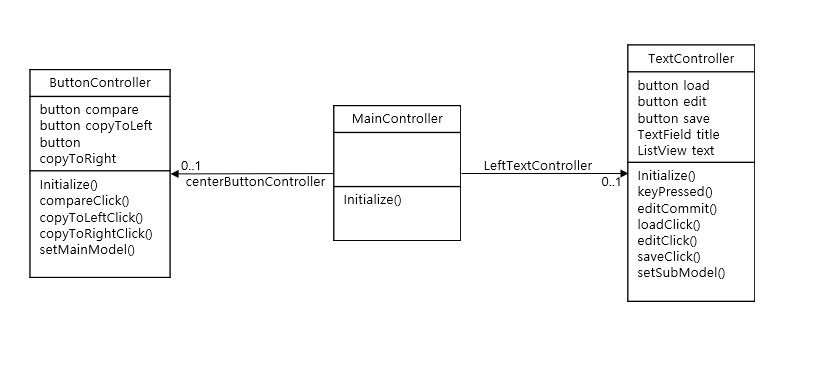
public class MainModel {

|  |
| --- |
|  |
|  | private SubModel leftSubModel; |
|  | private SubModel rightSubModel; |
|  |  |
|  | public MainModel() { |
|  | this.isCompared = new SimpleBooleanProperty(false); |
|  | this.leftSubModel = new SubModel(); |
|  | this.rightSubModel = new SubModel(); |
|  | } |

}

1. Interface Segregation Principle

Main Controller는 View와 Model 사이의 데이터 흐름을 관리한다. 이 데이터는 두 가지로 나뉜다. Button flag를 관리와 텍스트 관리이다. 따라서 button flag를 담당하는 Button Controller와 텍스트를 관리하는 Text Controller class를 선언해 두가지의 데이터가 서로에게 영향을 받지 않는다.



1. Dependency Inversion Principle

SimpleMerge는 크게 세가지 기능(Model, View, Controller)으로 나뉘어서 분류되어 있다. 그러나 각각의 기능들은 서로에게 직접적으로 영향을 주는게 아닌 MainModel, MainView 그리고 MainController를 통해 작업이 수행된다. 각각의 Main class들은 들어오는 요청을 처리하면 Sub Model들이 작업을 진행한다. 이로써 SubModel들은 요청 처리 책임에서 벗어난다.

# Test

## Unit test results

|  |  |  |  |
| --- | --- | --- | --- |
| **테스트 파일 이름** | | **ButtonFlagTest.java** | |
| **No.** | **테스트 케이스 이름** | **테스트 케이스 설명** | **테스트 결과** |
| 1 | Test ButtonFlag | ButtonFlag의 생성자를 테스트한다.  isEditable과 isModified는 False이다. | 정상 |
| 2 | Test Set isEditable | isEditable을 True로 설정하는 메소드를 테스트한다. | 정상 |
| 3 | Test Set is Modified | isModified를 True로 설정하는 메소드를 테스트한다. | 정상 |
| 4 | Test Set Toggle isEditable | isEditable을 Toggle하는 메소드를 테스트한다.  이 테스트케이스에서는 True인 상태에서 Toggle하여 False로 바뀌는지 테스트. | 정상 |
| 5 | Test Toggle isModified | isModified를 Toggle하는 메소드를 테스트한다.  이 테스트케이스에서는 True인 상태에서 Toggle하여 False로 바뀌는지 테스트. | 정상 |

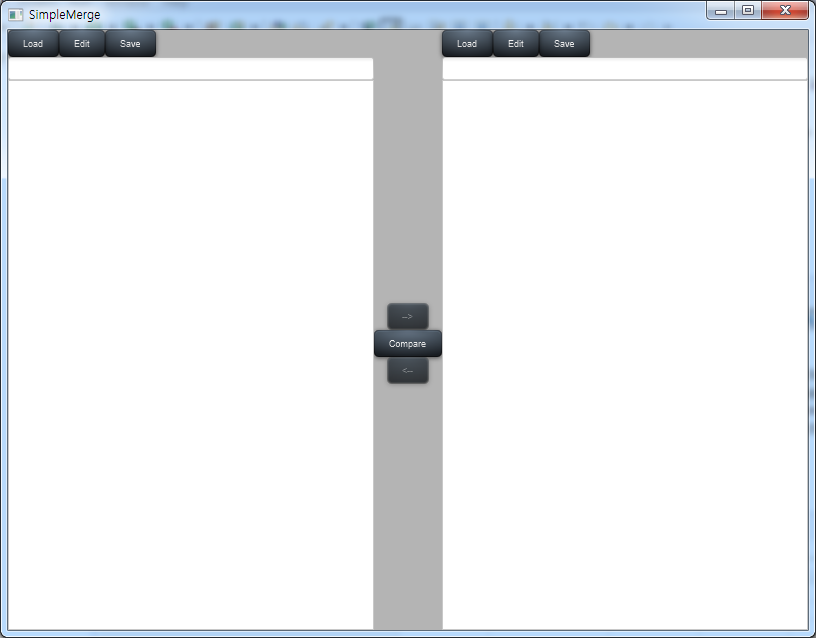
|  |  |  |  |
| --- | --- | --- | --- |
| **테스트 파일 이름** | | **SubModelTest.java** | |
| **No.** | **테스트 케이스 이름** | **테스트 케이스 설명** | **테스트 결과** |
| 1 | Test TextPage | SubModel의 TextPage를 set하는 메소드를 테스트한다. | 정상 |
| 2 | Test ButtonFlag True | SubModel의 ButtonFlag를 True로 Set하고  정상적으로 True로 Set 되었는지 테스트한다. | 정상 |
| 3 | Test ButtonFlag False | SubModel의 ButtonFlag를 True로 Set하고 False와  비교하였을 때 다르다는 결과가 나오는지 테스트한다. | 정상 |
| 4 | Test isModified | SubModel의 isModified가 False인지 확인하고,  그것을 Toggle하여 True가 되는지 확인한다. | 정상 |
| **테스트 파일 이름** | | **ButtonControllerTest.java** | |
| **No.** | **테스트 케이스 이름** | **테스트 케이스 설명** | **테스트 결과** |
| 1 | Test CopyToRight | copyToRight 버튼이 눌리는 것을 테스트 | 정상 |
| 2 | Test CopyToLeft | copyToLeft 버튼이 눌리는 것을 테스트 | 정상 |
| 3 | Test Compare Equal | LeftSubModel과 RightSubModel에 같은 내용을 넣고  Compare 했을 때 변경사항이 없는지 테스트 | 정상 |
| 4 | Test Compare Non Equal | LeftSubModel과 RightSubModel에 다른 내용을 넣고  Compare 했을 때 변경사항이 생기는 것을 테스트 | 정상 |

|  |  |  |  |
| --- | --- | --- | --- |
| **테스트 파일 이름** | | **LCStest.java** | |
| **No.** | **테스트 케이스 이름** | **테스트 케이스 설명** | **테스트 결과** |
| 1 | test DoLCS | 전체적인 LCS 동작을 테스트 | 정상 |
| 2 | test Return Largest Index | 3개의 int를 인자로 받아서 그 중 가장 큰 것의 Index를 정상적으로 리턴하는지 테스트 | 정상 |
| 3 | test Reset List Array Before LCS | LCS 연산 수행 전, 양쪽의 List의 초기화를 테스트 | 정상 |
| 4 | test Initialize Array | lcsCount배열과 isSame 배열의 초기화를 테스트 | 정상 |
| 5 | test Make LCS Matrix | lcsCount배열을 계산하는 것을 테스트 | 정상 |
| 6 | test Make Result Stack | lcsCount배열을 토대로 Stack을 만드는 것을 테스트 | 정상 |
| 7 | test Make List From Stack | Stack을 토대로 List를 만드는 것을 테스트 | 정상 |
| 8 | test Modify MainModel | List를 토대로 MainModel을 갱신한다. | 정상 |

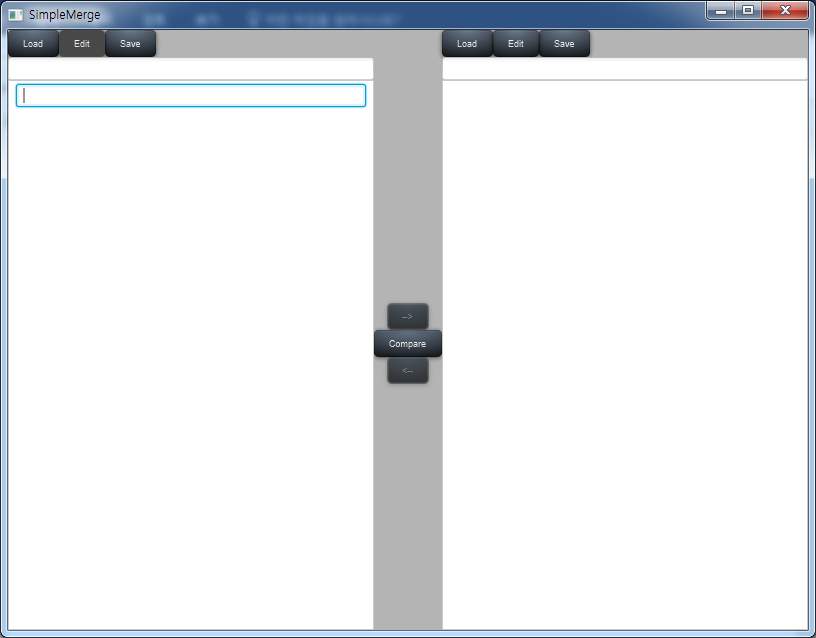
|  |  |  |  |
| --- | --- | --- | --- |
| **테스트 파일 이름** | | **LineTest.java** | |
| **No.** | **테스트 케이스 이름** | **테스트 케이스 설명** | **테스트 결과** |
| 1 | test Line Constructor | Line 생성자의 동작이 정상적인지 테스트 | 정상 |
| 2 | test Line Modification With FakeLine | Line을 FakeLine으로 설정하는 동작이  정상적인지 테스트 | 정상 |
| 3 | test Line Modification with RealLine | Line을 RealLine으로 설정하는 동작이  정상적인지 테스트 | 정상 |
| 4 | test Line Color | Line의 Color를 받아오는 동작이 정상적인지 테스트 | 정상 |
| 5 | test RealLine | Line이 RealLine인지 검사하는 동작이 정상적인지 테스트 | 정상 |

|  |  |  |  |
| --- | --- | --- | --- |
| **테스트 파일 이름** | | **TextPageTest.java** | |
| **No.** | **테스트 케이스 이름** | **테스트 케이스 설명** | **테스트 결과** |
| 1 | test TextPage Default Constructor | TextPage의 기본생성자의 동작이 정상적인지 테스트 | 정상 |
| 2 | test TextPage Constructor with File | File을 인자로 받는 TextPage의 생성자의 동작이 정상적인지 테스트 | 정상 |
| 3 | test Load And Save Func In TextPage | TextPage안의 Load와 Save 기능의 동작이 정상적인지 테스트 | 정상 |
| 4 | test IsRealLine Modification | TextPage를 RealLine을 FakeLine으로, FakeLine을 RealLine으로 바꾸는 동작이 정상적인지 테스트 | 정상 |
| 5 | test Set Line Color | TextPage의 특정 라인의 색깔을 설정하는 동작이 정상적인지 테스트 | 정상 |
| **테스트 파일 이름** | | **MainModelTest.java** | |
| **No.** | **테스트 케이스 이름** | **테스트 케이스 설명** | **테스트 결과** |
| 1 | Test MainModel Constructor | MainModel의 기본 생성자를 테스트한다. | 정상 |
| 2 | Test Set Left SubModel | LeftSubModel을 Set하는 것을 테스트한다. | 정상 |
| 3 | Test Set Right SubModel | RightSubModel을 Set하는 것을 테스트한다. | 정상 |
| 4 | Test Set isCompared | isCompared를 Set하는 것을 테스트한다 | 정상 |
| 5 | Test Toggle isCompared | isCompared를 toggle하는 것을 테스트한다 | 정상 |
| 6 | Test CopyToRight | LeftSubModel의 Text를 RightSubModel으로 Copy 하는 것을 테스트한다. | 정상 |

## Usage of Program



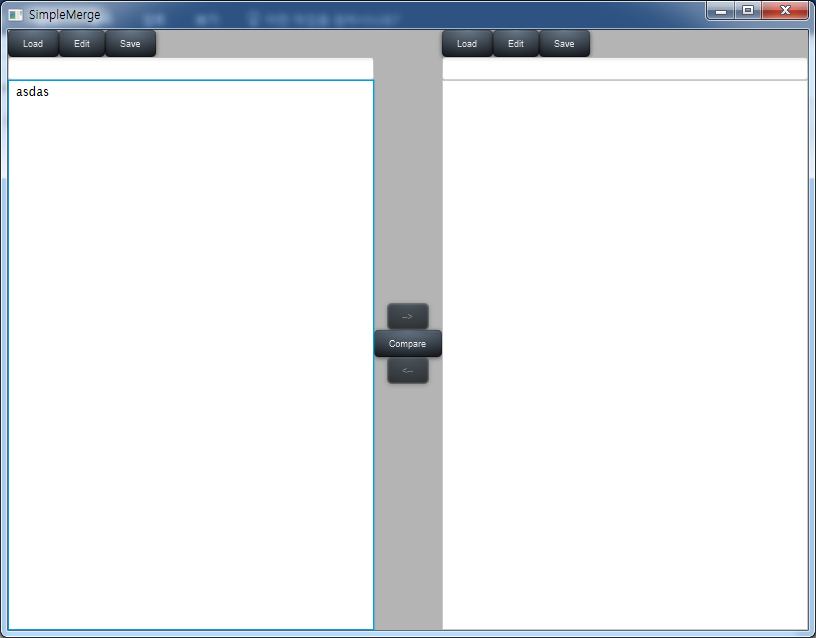
맨 처음 프로그램을 실행했을 때의 모습이다. 양쪽이 비어있다. 이 상태에서는 양쪽에 text를 쓸 수도 없다.

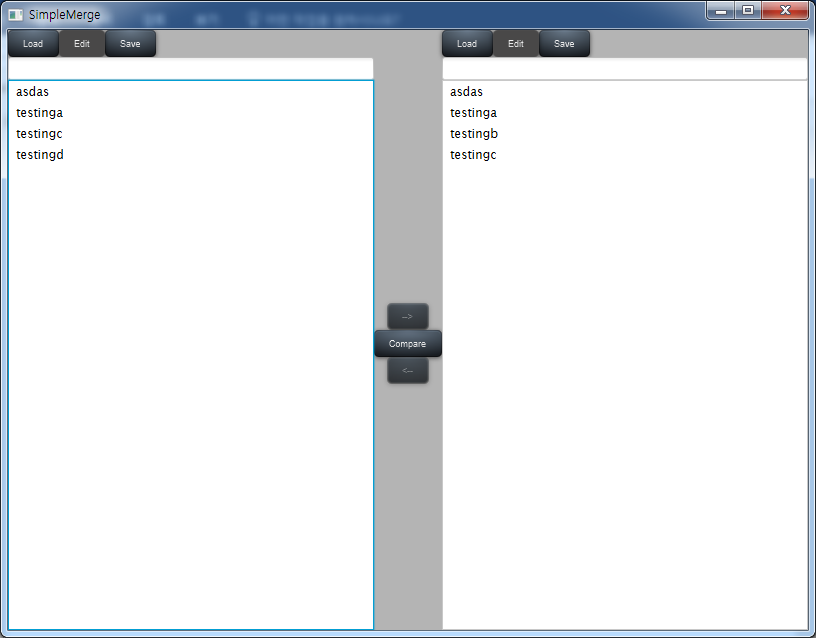


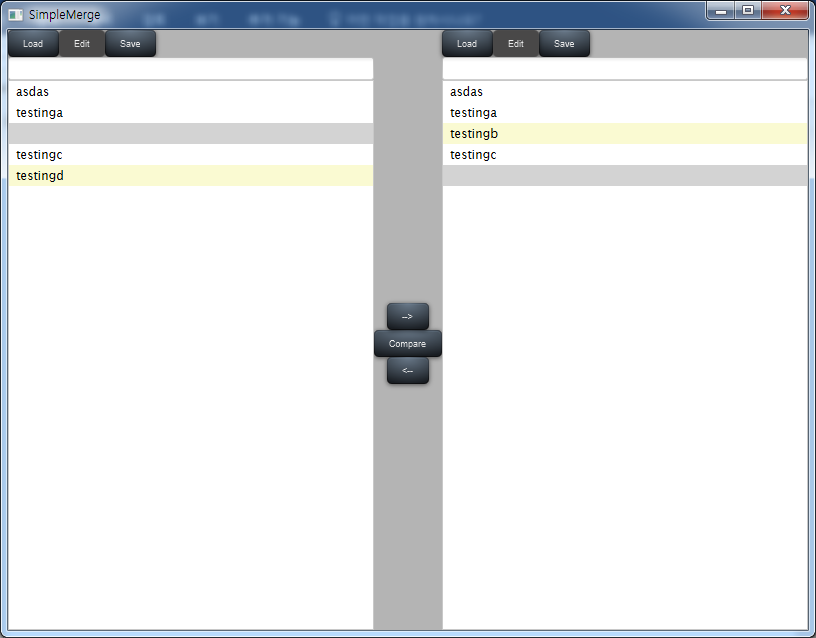
위쪽의 edit 버튼을 누르면 왼쪽 text area를 편집할 수 있게 된다.

첫 번째 줄 영역에 더블클릭을 하면 그 줄을 edit 할 수 있게 된다. 내용을 입력 후 Enter키를 입력하면 내용이 입력되며 다음 줄이 자동으로 추가된다.

edit버튼을 한번 더 누르면 편집을 비활성화 할 수 있다.



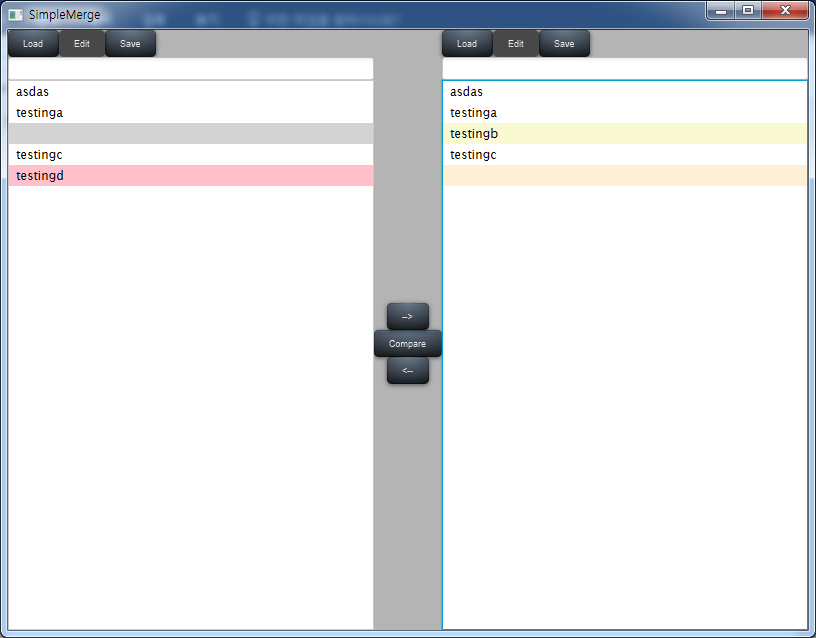
내용이 입력된 모습이다. 

내용을 더 입력하였다. 

그 후 Compare 버튼을 누른 모습이다.

노랑색 영역은 비교하여 다른 것으로 판명된 라인이며, 회색 영역은 실제로 있는 라인이 아닌 fakeline으로, 비교 대상과 위치를 맞추기 위해 생성된 라인이다. 이 상태에서 선택은 노랑색 혹은 회색 영역 중에서 할 수 있으며 양쪽의 선택영역은 따로 취급한다.

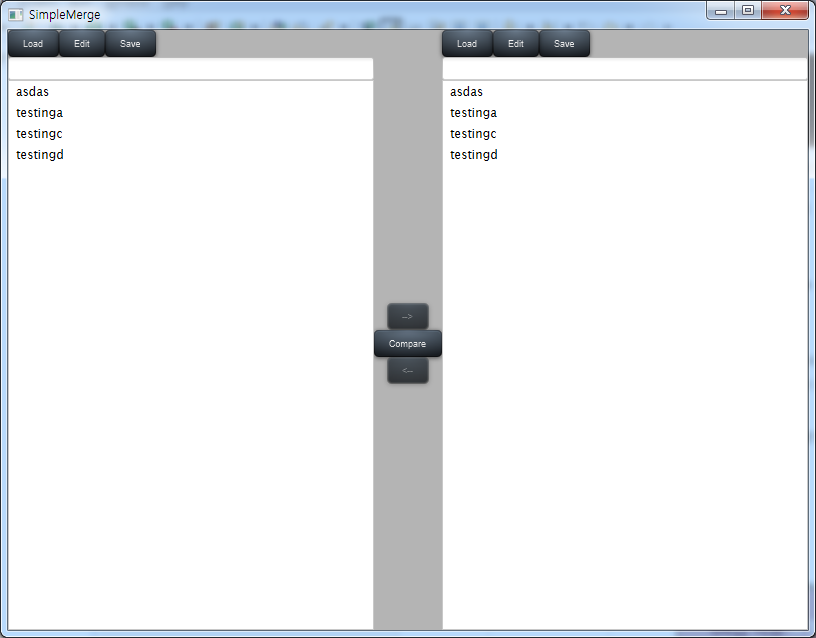
가운데에 화살표 버튼들이 활성화 된 것을 볼 수 있다.

노랑색 영역을 그 영역이 분홍색으로, 회색 영역은 옅은 분홍색으로 바뀌게 된다.

이대로 화살표를 누르면 선택된 영역이 반대쪽의 대응하는 영역에 복사된다.

분홍색 영역을 복사하면 대응 영역에 그대로 입력되며, 옅은 분홍색 영역은 대응 하는 영역을 삭제한다.

화살표를 누를 때, edit이 활성화 되어있으면 동작이 수행되지 않는다.



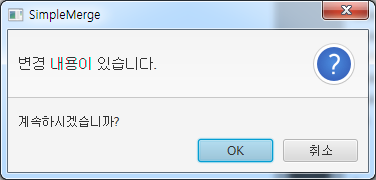
왼쪽에 있는 영역들을 다 오른쪽으로 옮긴 모습이다.

옅은 분홍색 영역에 대응하던 testingb 라인은 삭제되었고, 분홍색 영역에 대응하던 영역에는 testingd가 복사되었다.

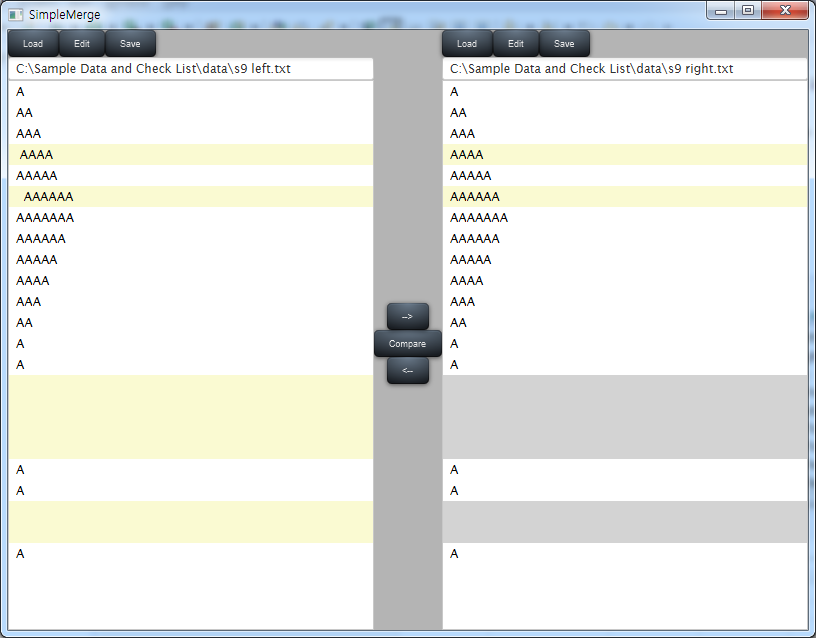
양쪽의 내용이 같아져서 화살표 버튼이 비활성화 되었으며, Compare 버튼을 눌러도 내용이 같아서 화살표 버튼은 활성화되지 않는다.



양쪽에 파일을 load한 모습이다. 텍스트 칸 위쪽으로 파일 경로가 나오는 것을 볼 수 있다.

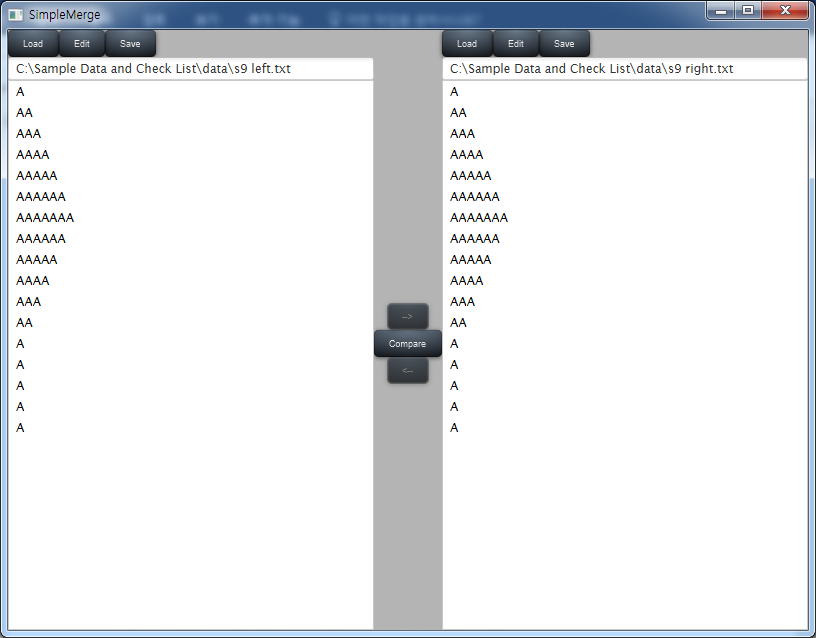


Load를 할 때 텍스트에 변경사항(edit, copy 등)이 있다면 다음과 같은 안내창이 뜨게 된다.



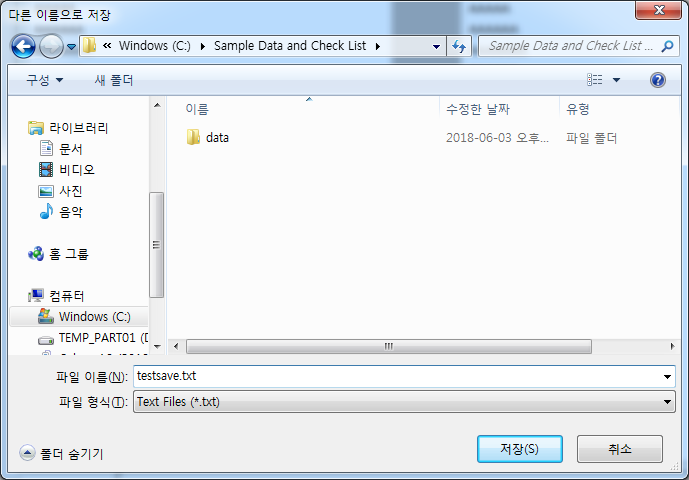
Compare 버튼을 누르면 정상적으로 동작하는 것을 확인할 수 있다.

이번엔 전부 왼쪽 화살표로 동작해보겠다.

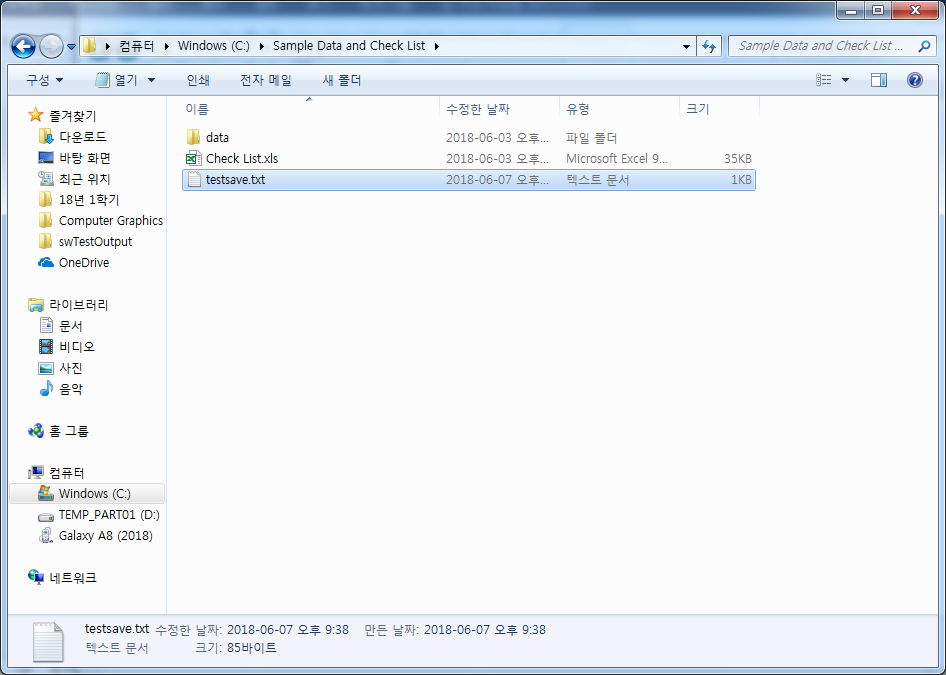


동작이 완료되었고 화살표 버튼이 비활성화 되었다.

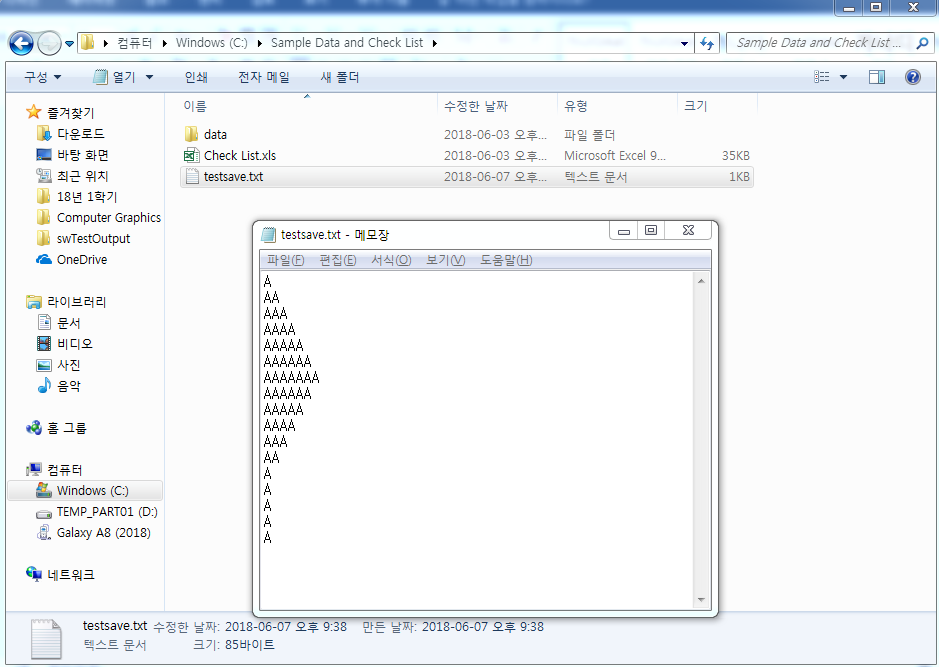
이 상태로 저장을 수행해보겠다.



다음 경로에 testsave라는 이름으로 텍스트 파일을 저장한다.



Testsave 텍스트파일이 정상적으로 생성되어 있는 것을 볼 수 있다.



텍스트파일의 내용 또한 정상적으로 저장되어있다.