Human ICT Software Engineering

**Final Report**

Simple Merge

20132682 김현빈

20132569 조성주

20134695 김명규

20134220 우정현

20131961 김민조

20162379 김성연

**목차**

1. **Project Description**
2. Summary
3. Development Environment

* Development OS
* Development Tool

1. **Analysis**
2. **Design & Implementation**
3. MVC Pattern
4. Model Design
5. View Design
6. Controller Design

* Main Controller
* Diff Algorithm

1. Class Diagram
2. **Object-Oriented Design Principles**
3. **Test**
4. Functional Unit Test Cases
5. GUI Test Case
6. **Execution Screen & Explanation**
7. **Project Management**
8. Program Description
9. Summary

이 프로젝트의 목표는 텍스트 파일 2개를 불러와서 비교하고 서로 다른 부분들을 줄 단위로 구분 해주며, 줄 단위로 복사 할 수 있으며, 불러온 텍스트 파일들을 창에서 편집할 수 있으며, 저장까지 할 수 있는 ‘Simple Merge’ 프로그램을 개발 하는 것이다.

1. Development Environment

- Development OS

프로젝트를 개발 할 때 사용 했던 운영체제는 window7, window10을 중점적으로 사용하였다

- Development Tool

JAVA 구현하였고 이에 사용된 버전은 1.8.0\_144 버전이다. GUI는 JAVA swing/awt를 사용하였고, Junit으로 Unit 테스트를 했다.

팀 프로젝트의 효율을 높이기 Git을 사용하였고 GitHUB를 이용하여 팀원들과 프로젝트 공유를 효율적으로 하였다.

본 문서는 Microsoft Word를 이용하여 작성하였다. SRS와 기타 문서들은 md를 이용하여 GitHub에서 작성하였다.

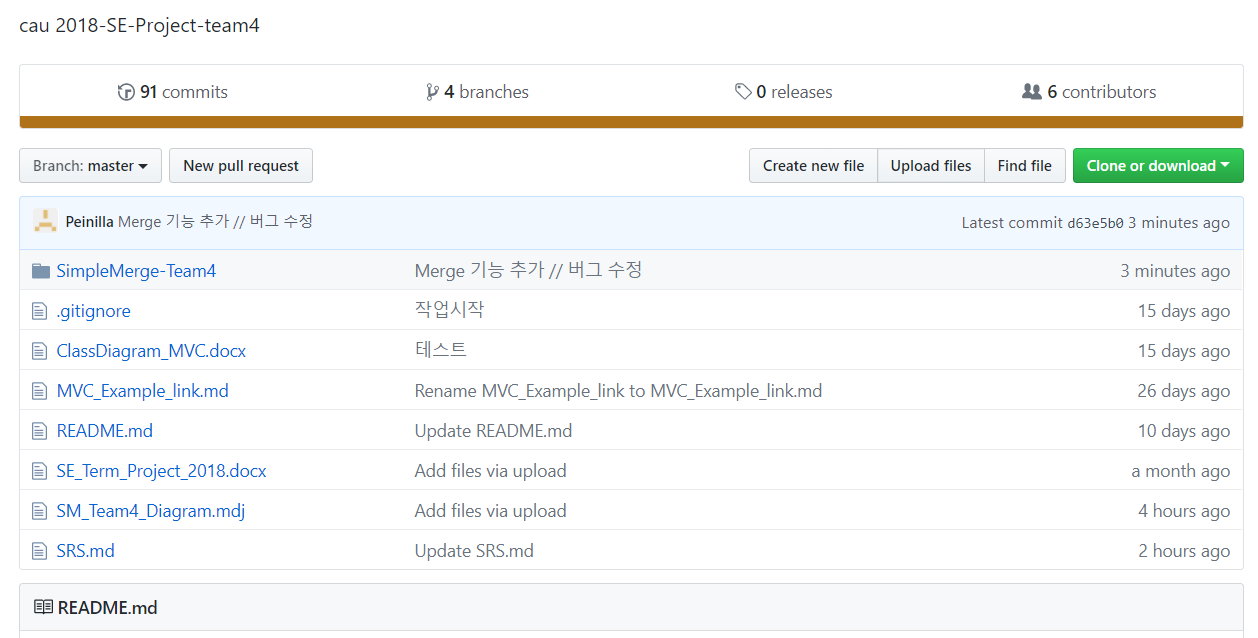


그림 1. GitHub

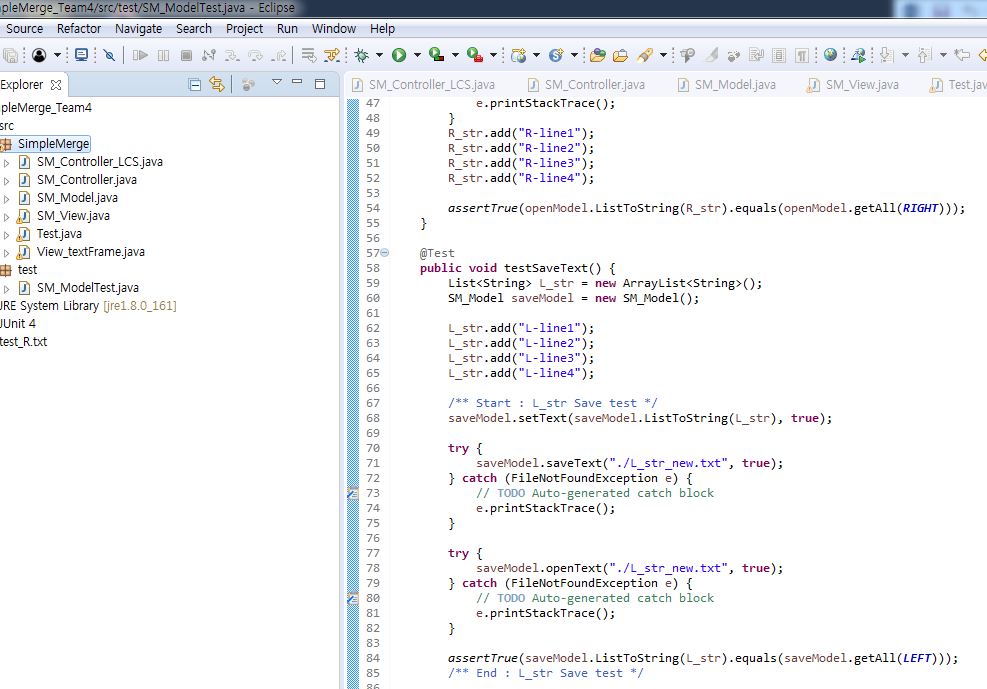


그림 2. 자바(이클립스)

1. Analysis

우리 팀은 목표로 하는 프로그램의 기능과 요구사항이 과제로서 명확하기에, 프로그램의 변화가 필요 없을 것으로 생각하여 폭포수모델(Waterfall Model)을 사용했다. 폭포수모델의 방식대로 요구사항 분석, 소프트웨어 설계, 소프트웨어 구현, 소프트웨어 시험, 소프트웨어 유지보수의 단계로 진행하였다.

먼저 요구사항 분석 단계에서는 우리가 프로그래밍해야 하는 프로그램의 주요기능들을 분석했다. 분석한 요구사항들을 알아보기 쉽게 문서화해서 SRS를 작성했다.

소프트웨어 설계 단계에서는 MVC 아키텍처 패턴에 맞춰 프로그램을 설계했다.

소프트웨어 구현 단계에서는 설계 한 것을 기초로 하여 코딩을 하였다.

소프트웨어 시험 단계에서는 다양한 방법으로 프로그램의 테스트를 해보면서 이상한 버그가 없는지 확인을 하였다.

소프트웨어 유지보수 단계는 우리 프로그램의 미숙한 부분들을 고쳐 나가는 것으로 하겠다.

**#구현하여야 하는 기능들**

**i. Load**

텍스트 파일을 불러오는 기능이다.

Edit 상태에서는 버튼이 비활성화 되어야한다.

**ii. Edit**

불러온 텍스트를 편집하거나 빈 창에서 텍스트를 작성 할 수 있는 기능이다.

**iii. Diff**

두 텍스트 파일들을 줄 단위로 서로 다른 부분들을 색칠해서 표시해주는 기능이다.

같은 줄은 색을 안 칠하고, 서로 다른 줄은 분홍색, 추가된 빈 줄들은 회색으로 칠할 것이다.

**iv. copy**

‘Diff’를 사용하여 나타낸 서로 다른 부분이 있는 줄을 위에서부터 차례대로 왼쪽 텍스트를 오른쪽에 붙여넣기 하던가, 오른쪽 텍스트를 왼쪽에 붙여넣기 하는 기능이다.

Diff가 활성화 된 상태가 아니면 비활성화 되어야 한다.

Edit 상태에서도 비활성화 되어야 한다.

**v. Save**

편집된 텍스트 파일을 저장하는 기능이다.

Edit 상태에서 비활성화 되어야 한다.

1. Design & Implementation

1) MVC Pattern

MVC 패턴은 응용 프로그램을 세 가지의 구성요소인 모델, 뷰, 컨트롤러로 나누는 디자인 패턴이다.

모델은 List<String> 을 이용해서 텍스트를 저장 하며, 컨트롤러에 의해 뷰에게 텍스트를 String으로 넘겨준다. 또한 컨트롤러를 통해 FilePath를 받아와서 텍스트를 파일로 불러오기/저장을 수행한다.

뷰는 swing을 이용하여 UI를 담당한다. 컨트롤러를 통해 변경되는 사항을 TextPane과 Button에 적용한다.

컨트롤러는 뷰의 액션을 받아와서 각 액션에 맞는 기능을 수행한다. 기능에 따라 모델과 뷰를 제어한다.

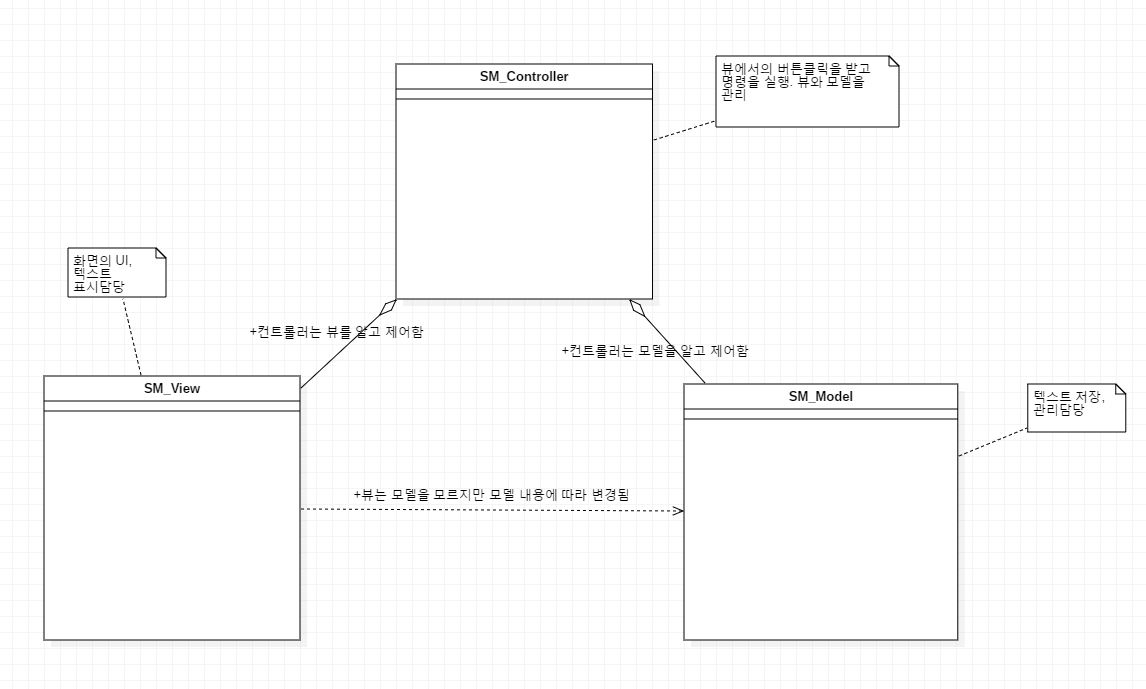


그림 3. MVC 패턴

2) Model Design

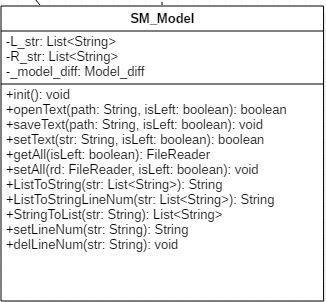


그림 4. Model design

@ SM\_Model

MVC 패턴 중 모델에 속하는 클래스로, 텍스트를 저장할 List<String> L\_str 과 R\_str를 변수로 가진다.

**생성자**를 통해 L\_str 과 R\_str을 ArrayList<String>으로 초기화한다.

**openText / saveText** 는 FilePath에 해당하는 파일을 불러오기/저장한다. 인수로 받는 FilePath가 JFileChooer를 이용한 path이기 때문에 별다른 예외처리를 하지 않는다. 좌 우 방향은 인자로 받는 boolean으로 결정한다.

처음 실행창이 켜졌을 때 완전 빈 텍스트 창이 뜨는걸 막기 위해서 스트링을 추가한 상태로 간다. 실행창에서 텍스트파일을 읽어오는 기능과 저장하는 기능을 구현하면서 리스트마다 줄 번호를 주는 기능까지 있는 클래스이다.

줄 번호를 주는 이유는 diff 상태에서 merge를 구현할 때 LCS 방법을 이용하기 편하기 위해 구현하였다.

3) View Design

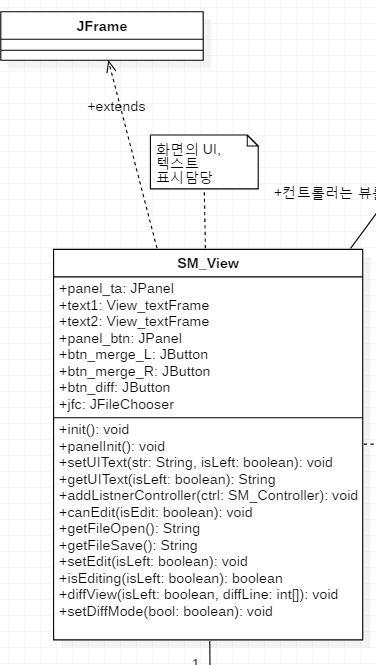


그림 5. SM\_View(Main Frame)

@ SM\_View

MVC패턴 중 뷰에 속하는 클래스로 JFrame을 상속받으며, UI구성하는 컴포넌트들을 변수로 가지며 그 중에는 view\_textFrame을 좌 우로 2개가진다.

**생성자를** Frame설정을 변경하며, 컴포넌트들을 초기화한다. 보기 좋게 만들기 위해 컴포넌트가 들어있는 패널을 BorderLayout를 이용해 Frame에 넣는다.

**addListnerController**는 버튼들에게 ActionListener인 SM\_Controller를 추가한다.

**getFileOpen / getFileSave** 는 JFileChooser를 이용하여 불러오기/저장 탐색기를 연다. JFileChooser는 txt 확장자만 열 수 있다. 파일을 선택하지 않고 탐색기 끄면 아무 일도 일어나지 않는다. 파일을 선택하면 파일의 경로를 반환한다.

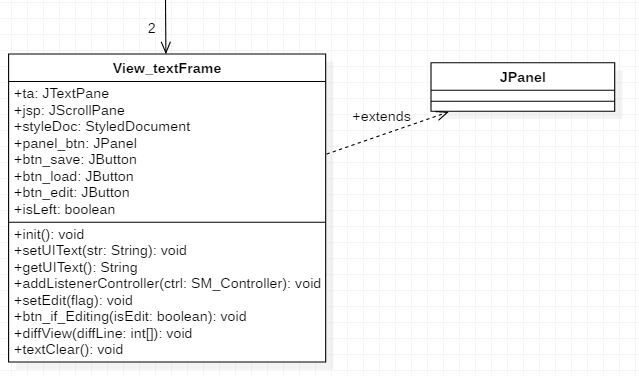


그림 6. View\_textFrame

@ View\_textFrame

MVC패턴 중 뷰에 속하는 클래스로 JPanel을 상속받으며, SM\_View에 의해 사용된다. Save/Load/Edit을 담당하는 버튼과 textPane을 변수로 가진다.

**생성자에** 인자로 받는 boolean형을 이용하여 좌우를 결정한다. 좌우 방향에 따라 버튼의 이름을 변경한다.

**diffView**는 인자로 받는 int[] diffLine을 이용한다. ( 0 = Diff Line / 1 = Same Line / 2 = Blank ) 배열의 인덱스에 해당하는 줄에 배열값에 따라 textPane에 색을 추가한다. ( 0 = 노란색 / 1 = 흰색 / 2 = 회색)

Blank는 라인넘버에 포함되지 않으며, 비교 가독성을 위해 추가한 기능이다.

4) Controller Design

-Main Controller

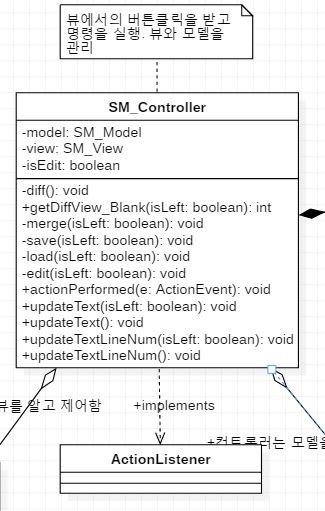


그림 7. SM\_Controller (Main Controller)

@ SM\_Controller

MVC 패턴 중 컨트롤러를 담당하는 클래스로 ActionListener 상속받는다. SM\_Controller은 뷰의 버튼들과 연동되어있어, 뷰의 액션을 통해 기능을 수행한다.

**getDiffView**는 **SM\_Controller\_LCS**를 이용해 만들어진 int 배열을 좌우에 따라 반환한다.

Merge는 다른 라인을 선택방향으로 복사한다. 한번에 한 줄씩, 위에서부터 차례대로 동작한다.

**SM\_Model**과 **SM\_Controller\_LCS**에서 구현된 것들을 ActionEvent로 받아 들여서

-Diff Algorithm

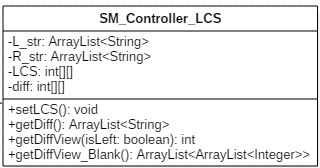


그림 8. SM\_Controller\_LCS (Diff Algorithm)

@ SM\_Controller\_LCS

MVC 패턴 중 컨트롤러를 담당하는 클래스로 Diff 기능 구현을 위해 LCS 알고리즘을 담당한다. 좌우 텍스트를 저장하는 ArrayList<String> L\_str, R\_str 2개 와 LCS 테이블을 담당하는 LCS : int[][], 테이블의 방향성을 저장하는 diff : int[][] 를 변수로 가진다.

**생성자**는 인자로 받은 ArrayList 들을 각각 L\_str과 R\_str에 복사한다.

**setLCS** 는 L\_str 과 R\_str을 한 줄씩 비교하여 만들어진 LCS 테이블을 LCS : int[][] 와 diff : int[][] 에 저장한다.

**getDiff** 를 통해 LCS : int[][] 와 diff : int[][]를 분석하여 최장 공통 부분을 ArrayList<String>으로 만든다.

**getDiffView** / **getDiffView\_Blank** 는 getDiff()를 통해 만들어진 최장공통부분 ArrayList<String>을 L\_str과 R\_str에 각각 비교하여 ( 0 = Diff Line / 1 = Same Line / 2 = Blank )을 가지는 int 배열을 만든다. int배열의 인덱스는 각 라인의 상태이다.

5) Class Diagram

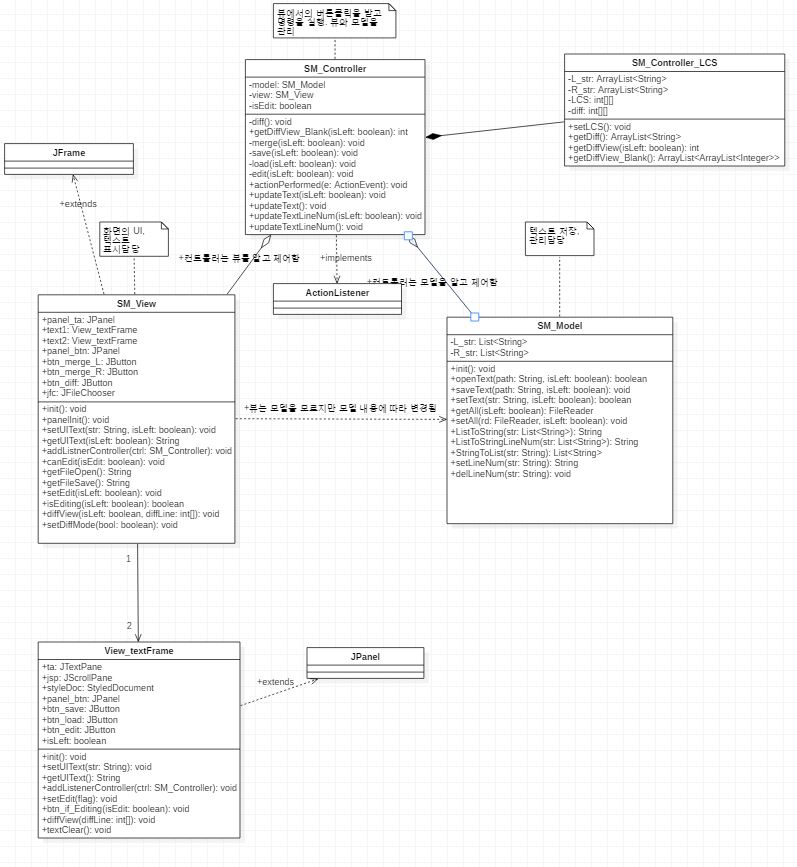


그림 9. Class Diagram

1. Object-Oriented Design Principles

**# 캡슐화 Encapsulation**

MVC패턴으로 구현되었기 때문에 View와 Model 클래스들은 모두 Public의 접근자를 가진다. 하지만 Controller는 Private 접근자를 가지는데, 이 이유는 Controller은 View와 Model 모두를 알고 있어야 하지만, View와 Model은 Controller을 몰라야 하기 때문이다.

**# 상속 Inheritance**

- SM\_Controller **implements** ActionListener

- SM\_View **extends** JFrame

- View\_textFrame **extends** JPanel

View 클래스인 SM\_View와 View\_textFrame은 각각 swing 라이브러리의 JFrame과 JPanel을 상속받는다. 이를 통해서 클래스 자체를 UI로 이용할 수 있다.

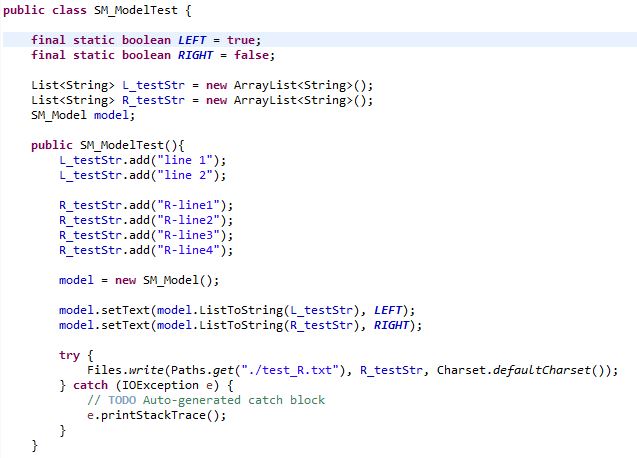
SM\_Controller은 ActionListener을 상속받는데, 이를 통해 View 클래스들의 액션을 감지(Listener)할 수 있다.

1. Test
2. Functional Unit Test Cases

우리 조는 Unit Test를 위해서 자바에 있는 Junit을 이용하여 테스트하였다.

**#SM\_ModelTest**

데이터 담당 부부인 SM\_Model을 JUnit으로 테스트 해보자.





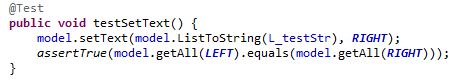
#**testOpenText 함수** : SM\_Model의 opentext 기능을 확인하는 함수이다.

주요 기능 중 하나인 ‘Load’ 기능을 담당하는 함수로 오른쪽 패널에 test\_R이라는 텍스트를 불러오기를 시도해서 잘 작동하는지 확인하다.

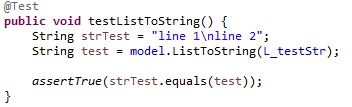


# **testSaveText 함수** : SM\_Model의 saveText 함수를 확인하는 함수이다.

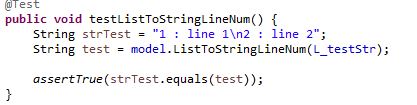
왼쪽 패널에 “L-line1”,”L-line2”, ”L-line3”, ”L-line4” 를 입력시키고 L\_str\_new로 저장을 한 뒤에, 다시 L\_str\_new를 불러와서 서로 비교해서 제대로 저장이 되었는지 확인한다.



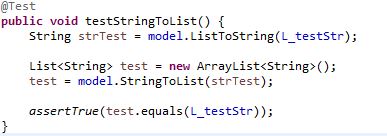
# **testSetText 함수** : SM\_Model의 setText 함수를 확인하는 함수이다.



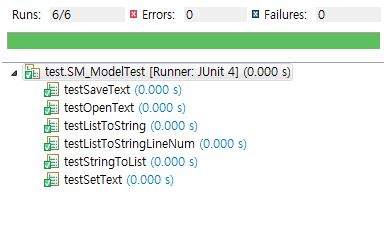
# **testListToString 함수** : SM\_Model의 ListToString 함수를 확인하는 함수이다.



# **testListToStringLineNum 함수** : SM\_Model의 ListToStringLineNum 함수를 확인하는 함수이다.

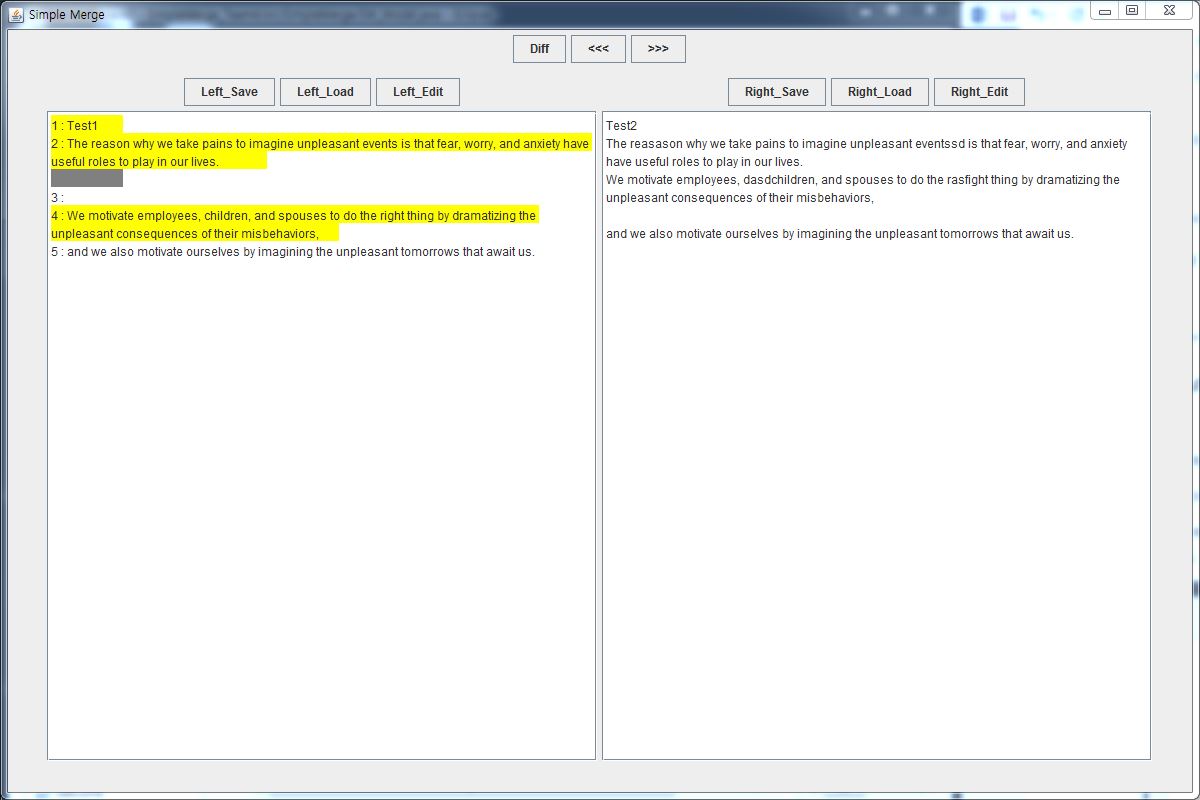


# **testStringToList 함수** : SM\_Model의 StringToList 함수를 확인하는 함수이다.



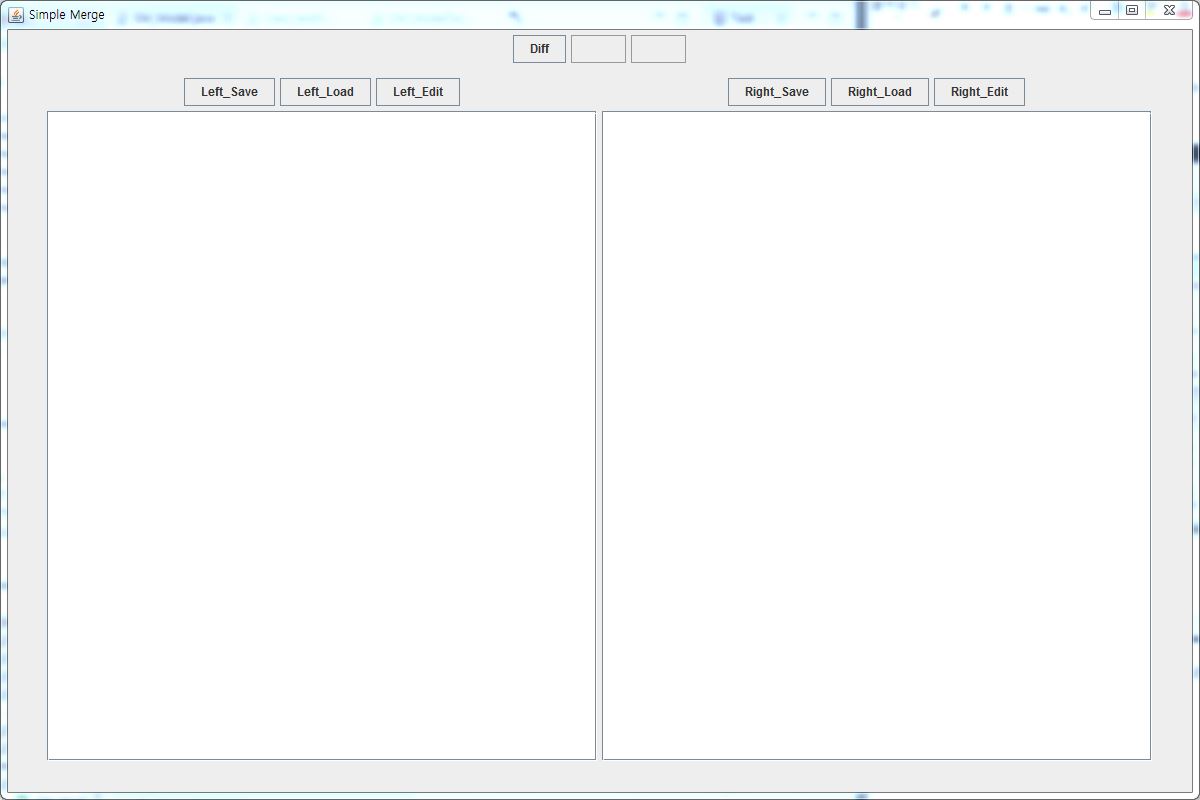
모든 테스트가 정상 작동되었다.

1. Bug Report



Diff를 누르면 규칙에 맞게 색이 칠해지는데 이 상태에서 좌우 패널 중에 하나를 저장하였을 때, 저장한쪽(여기선 오른쪽 패널을 저장하였다)의 색은 없어지는데, 저장을 안 한 쪽의 패널은 색이 그대로 칠해져 있게 된다. 똑같은 증상이 Diff 상태에서 edit을 눌렀을 때도 가끔 일어난다.

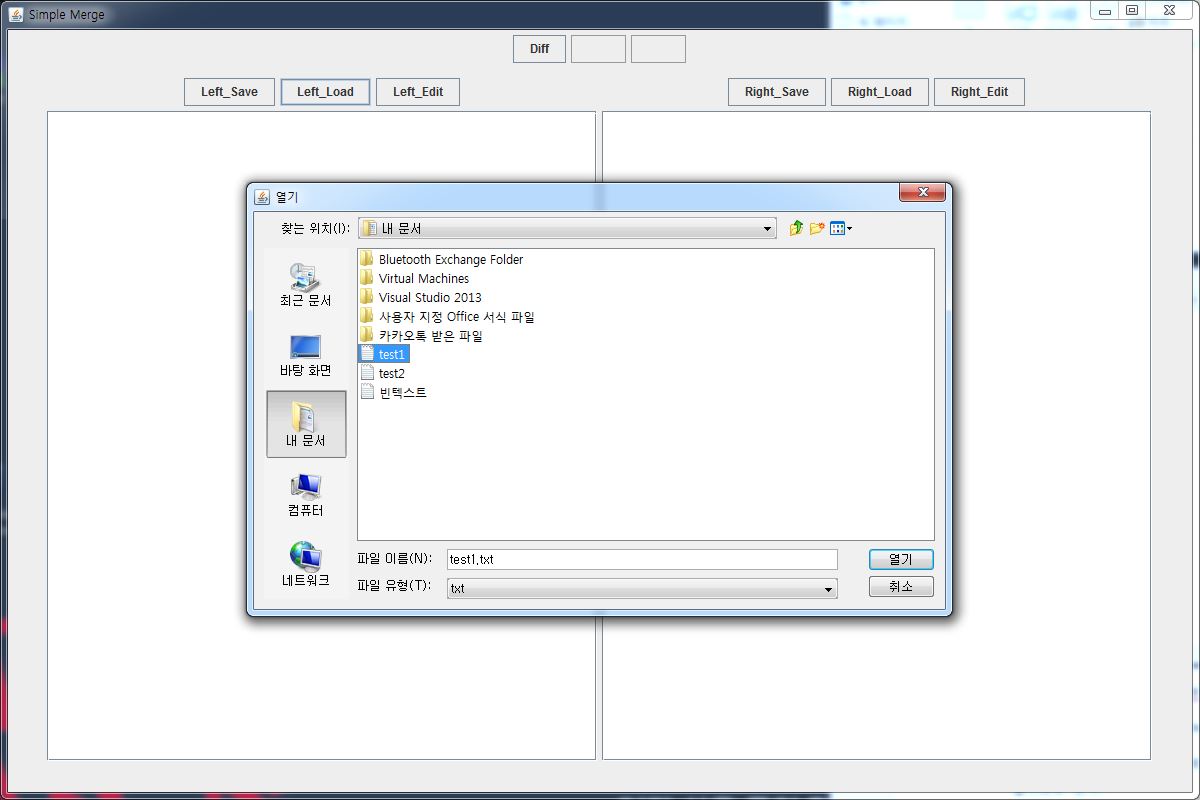
1. Execution Screen & Explanation



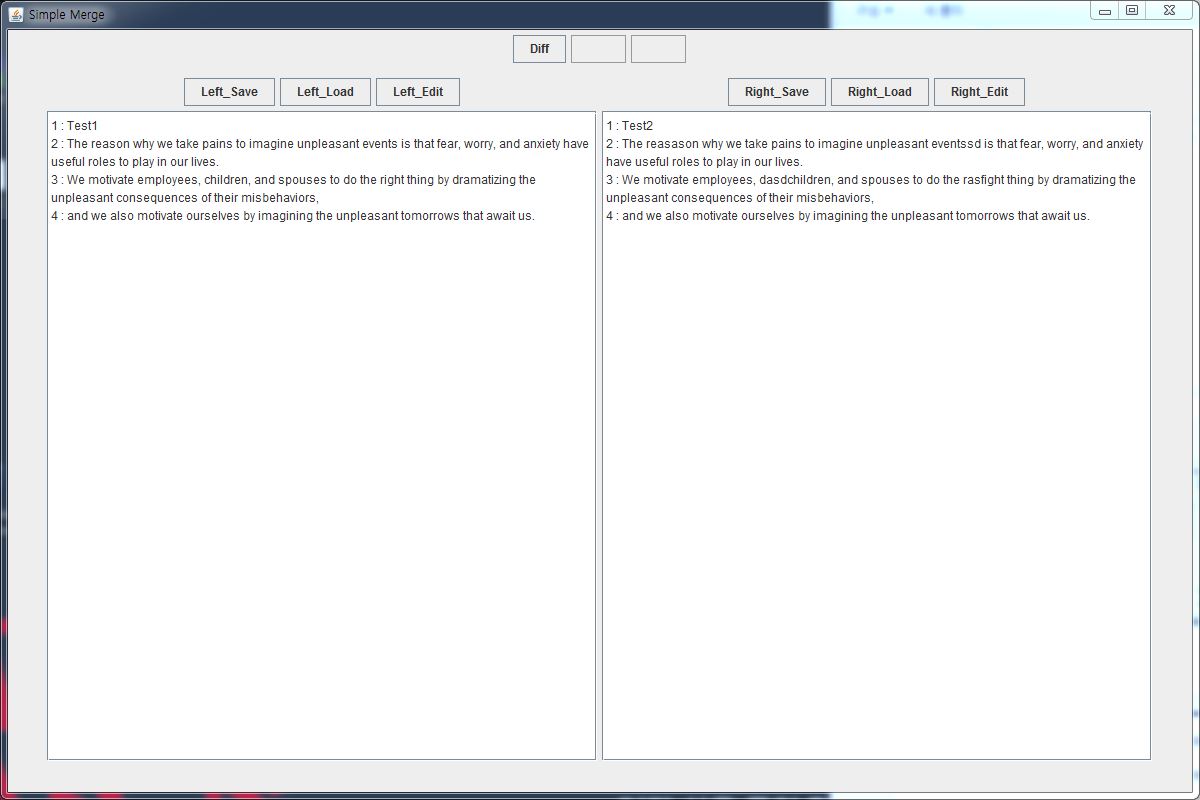
실행을 하게 되면 나오는 기본창

처음에 Diff가 안된 상태라 Merge키가 비활성화 되어있다.

이 상태에서 편집(Edit)을 누르고 패널에 직접 글을 써도 되고, Load를 이용해 텍스트 파일을 불러와도 된다.

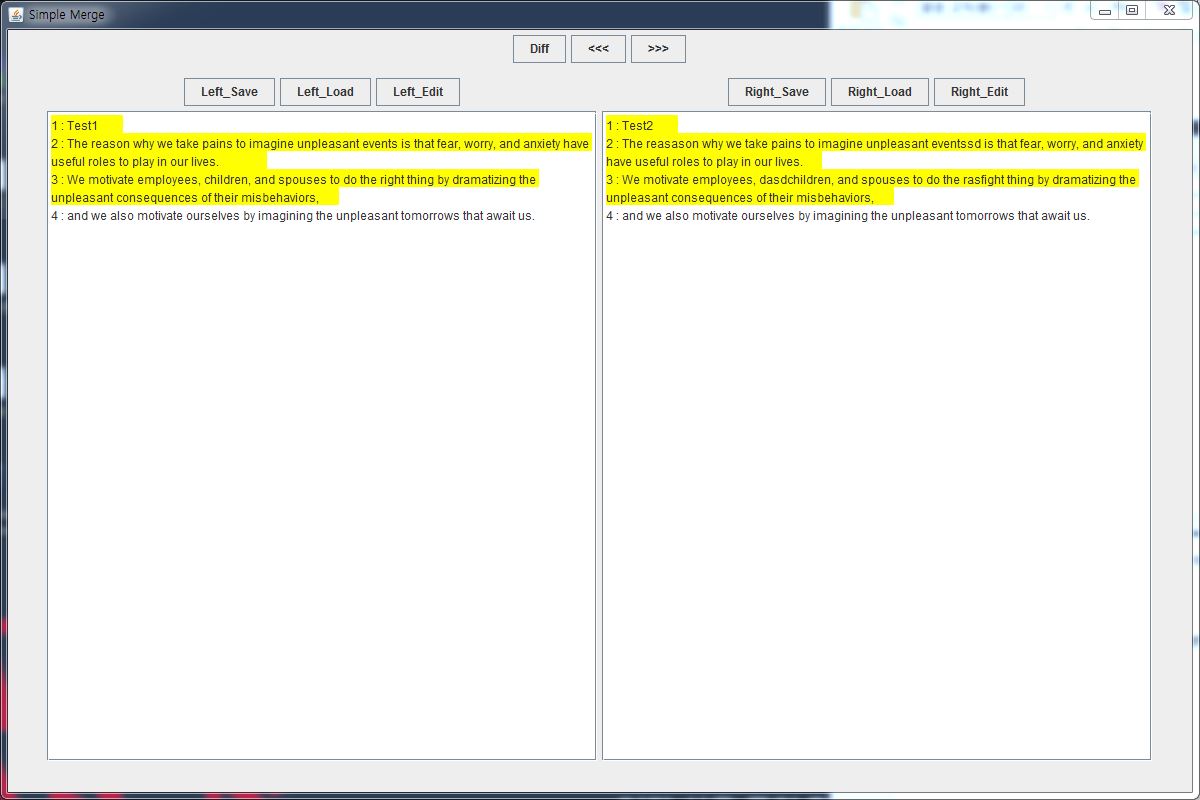


여기서는 저장된 파일을 불러와 보자.



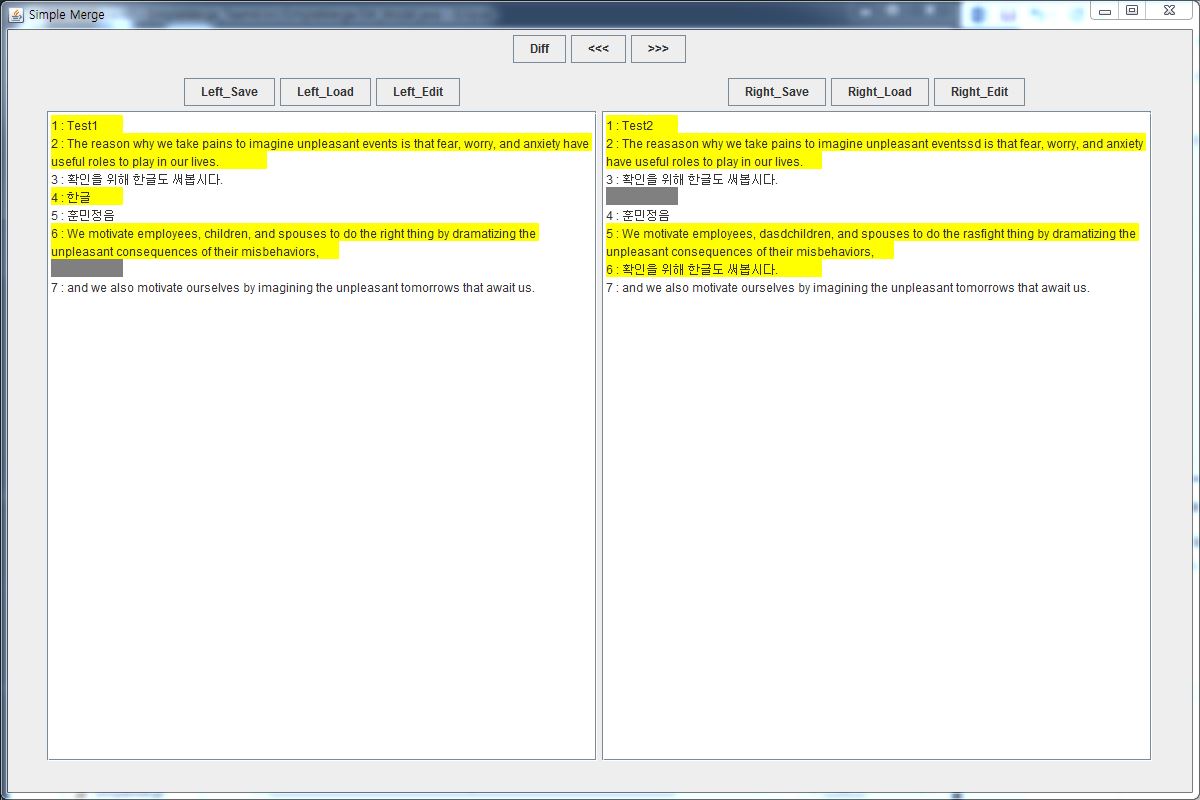
왼쪽에는 test1파일을 오른쪽에는 test2파일을 불러왔다.

편집 및 비교의 용이를 위해 각 줄마다 줄 번호를 좌측에 표시하였다.

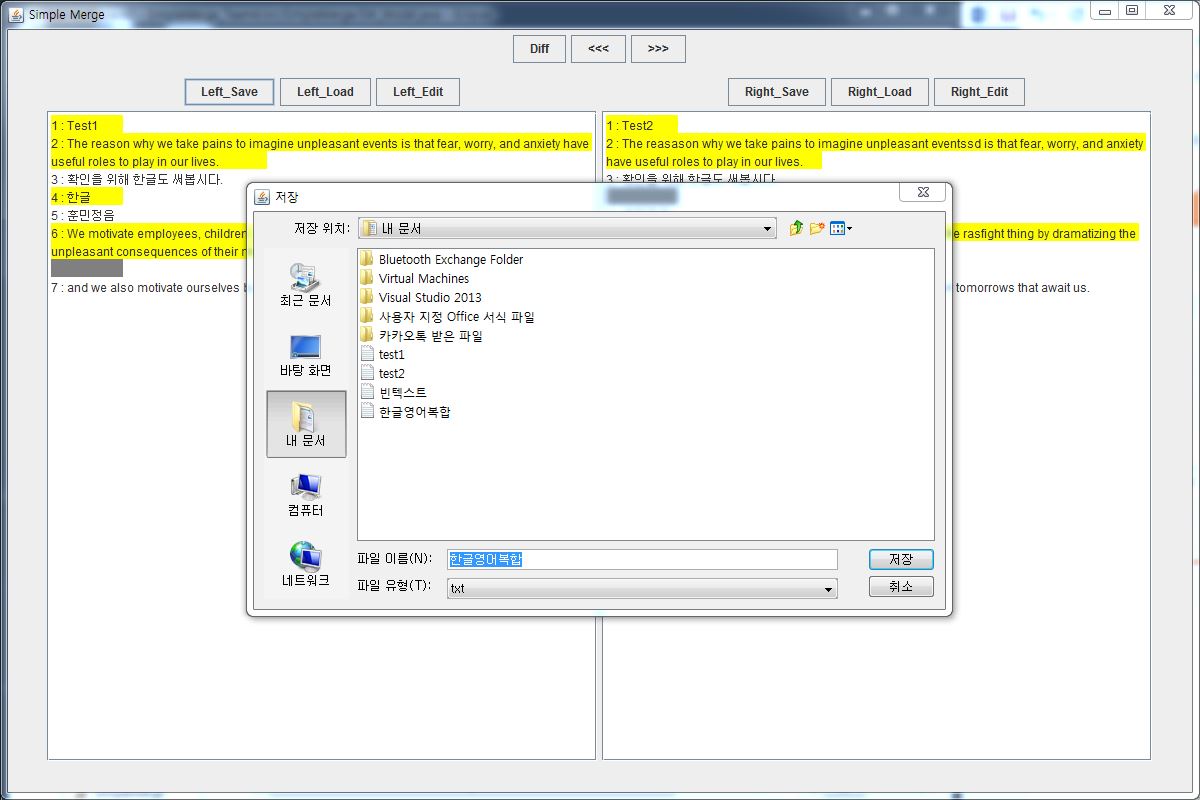


Diff를 눌러보았다.

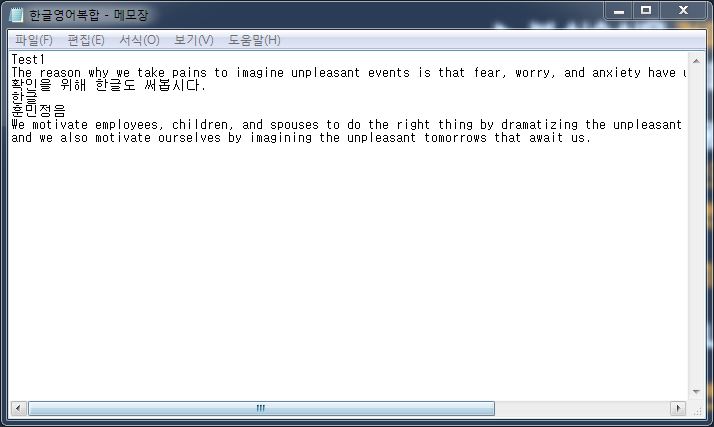
왼쪽 패널과 오른쪽 패널에서 각 라인별로 일치하는 라인을 찾고, 불일치하는 라인은 노란색으로 색칠된다.



영어 말고 한글도 가능한지 확인을 위해, Edit을 이용하여 한글도 같이 써보았다.

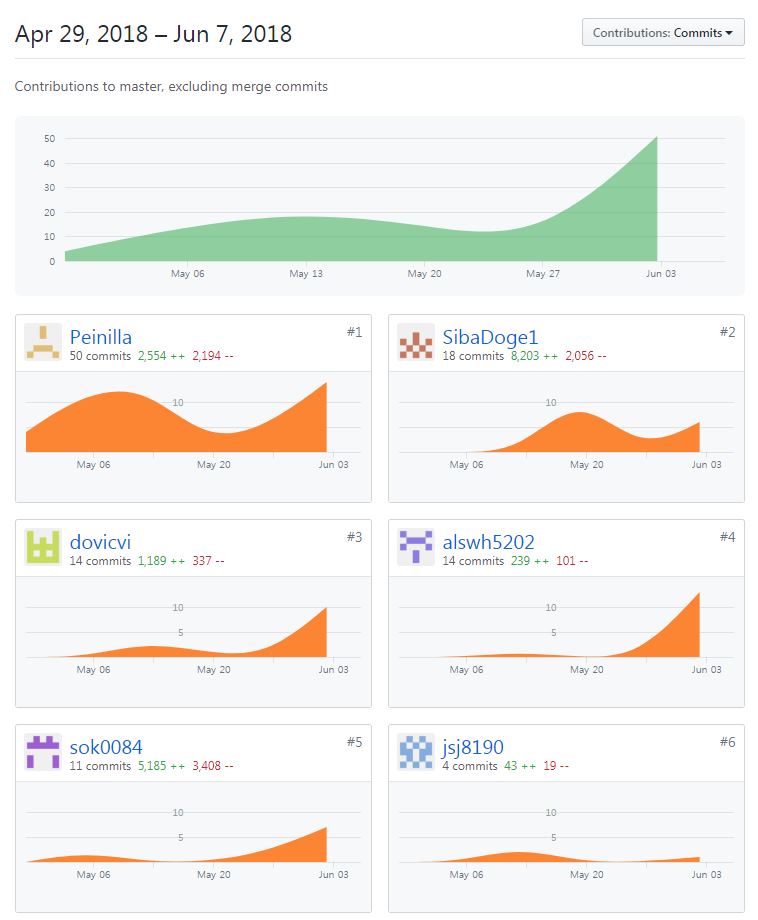


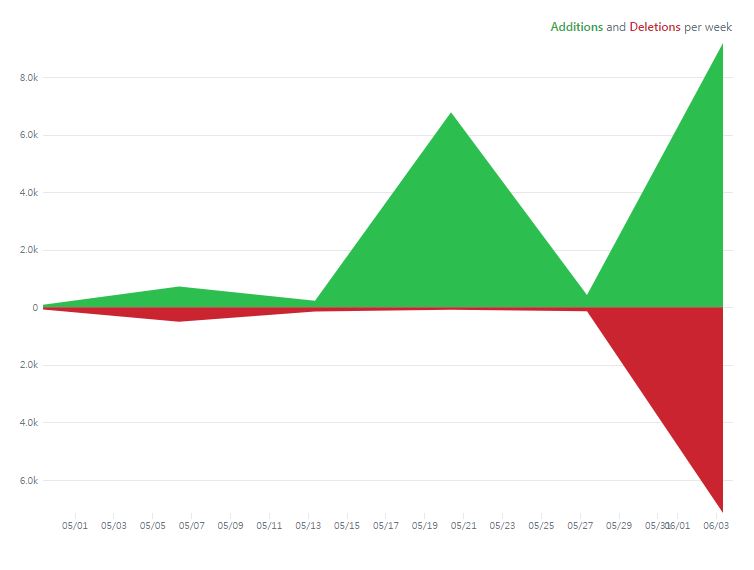
저장(Save)이 잘 되는지 확인해보자.

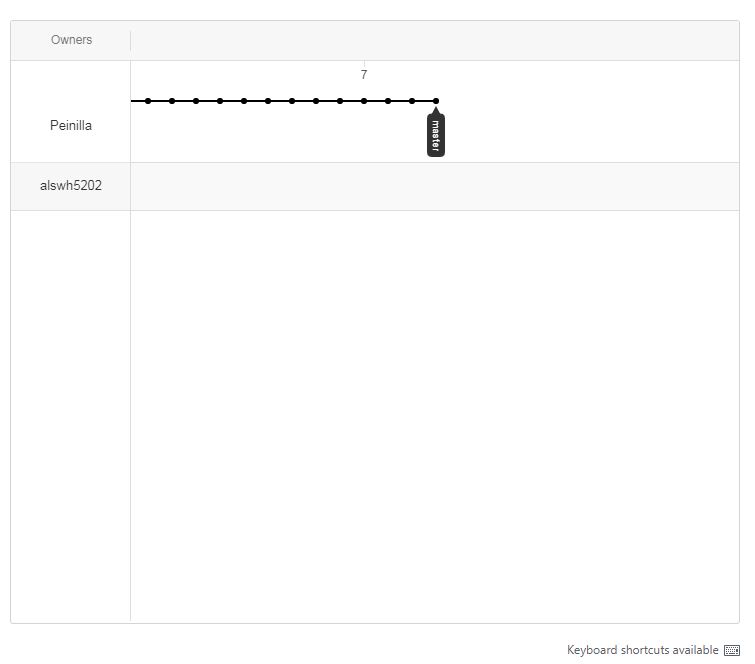


저장이 제대로 되었다.

1. Project Management
2. **GitHub**

****

****

****

**Github 경험자가 몇 명 없고, 코더가 2명뿐이라 브랜치를 따로 나누지 않고 마스터에서 계속 작업을 했다. 물론 이런 방법이 Git의 정상적인 사용방법은 아니지만, 오프라인에서의 즉각적인 커뮤니케이션과 피드백을 통해서 문제점을 해결하였다. 하지만 팀원 모두 Github에 대해서 경험을 해봤기 때문에 앞으로는 더 좋은 방법으로 사용할 수 있을 거로 생각한다.**

1. Experience of Our Team

|  |  |
| --- | --- |
| **김현빈**  **Peinilla** | 프로젝트를 통해 MVC패턴을 토대로 한 프로그램 개발을 배울 수 있었다. OOP지향 설계는 완벽하지 못하지만 팀원들과 같이 협업하는 프로그램 설계를 할 수 있는 기회를 가질 수 있어서 좋은 공부가 되었다. 팀 내에 Github 사용자가 적고, 코딩 담당이 2명뿐이라 브랜치와 PR등 Github의 주요 기능 등을 사용하지 못한 건 아쉽게 생각한다. |
| **조성주**  **jsj8190** | 이번 팀 프로젝트를 통해서 많은 것을 배울 수 있었다. 특히 요구한 사항들을 토대로 코딩해 가는 과정을 보았고, 다른 팀원들의 코딩을 보면서 부족한 프로그래밍을 향상시키는 계기가 되었다. 발표 준비를 하면서 수업시간에 배운 MVC 패턴과 Waterfall 모델을 직접 확인하고, 그 장단점을 알 수 있었다. |
| **김명규**  **dovicvi** | 이번 프로젝트를 하면서 가장 큰 경험은 계획을 갖고, SRS와 같은 요구사항 보고서를 써가면서 거기에 맞춰 과제를 해본 것이다. 과제에 명시적으로 요구하지 않았다면 막무가내로 개발을 해 나갔을 거 같다. 다른 거로는 GitHub를 써 본 것도 새로운 경험이 되었다. 처음에 어떻게 하는지 몰라서 구글에 검색해보면서 알아갔지만, 아직도 모르는 기능이 많은 것 같다. |
| **김민조**  **alswh5202** | 이번 소프트웨어 공학 팀 프로젝트를 통해서 ‘SRS’라고 불리는 소프트웨어 요구사항 명세서를 작성해보았다. ‘SRS’를 작성한 후 Github를 이용해서 팀원들과 프로젝트를 진행하니 더욱 더 작업 효율성에서 효과적인 것을 느꼈고. 다양한 개발 방식을 배워보는 기회였다. |
| **우정현**  **sok0084** | MVC패턴에 대해 알아보면서, 객체지향 프로그래밍 설계에서 쓰이는 여러 가지 설계패턴 또한 알게 되었다. JUnit 테스트를 진행하면서 실제로 프로그램 내에서 어떤 방식으로 테스트가 이루어지는지에 대해 배울 수 있었다. StarUML과 리버스 엔지니어링 툴을 사용하면서 여러 다이어그램에 대해 더 잘 알 수 있었다. |
| **김성연**  **SivaDoge** | 이번 과제에서 MVC패턴에 따른 도메인 모델 설계, 다이어그램 제작, 객체지향 디자인에 따른 클래스 설계 등을 해보며 실무에서의 디자인 방법에 대해 배울 수 있었다. 또한 Github를 통해 협업의 방식이 어떤 식으로 이루어지는지, Waterfall 모델의 실제 적용과 장단점이 무엇인지 확실히 알 수 있었다. 개인적으로 Github를 전부터 많이 써왔었기에 브랜치, PR등을 적극적으로 하려 했으나, 프로젝트가 작고 Github 경험자가 적어서 제대로 하지 못했다. 이 점에 약간 아쉬움이 느껴진다. |