**SE 2017 Spring Term Project:**

**Development of SimpleMerge Final Report**

**1. Team Members & Roles**

1) Members & Roles

2) Team Progress

**2. Program Description**

1) 개요

2) 주요 기능

**3. Development Environment**

1) Development OS

2) Development Tool

**4. SRS (Software Requirement Specification) – 수빈**

1) Introduction

2) Use Case Diagram

3) Use Case Description

4) Non-functional Requirement

5) Requirement Dependency Traceability

6) Development and Target Platforms

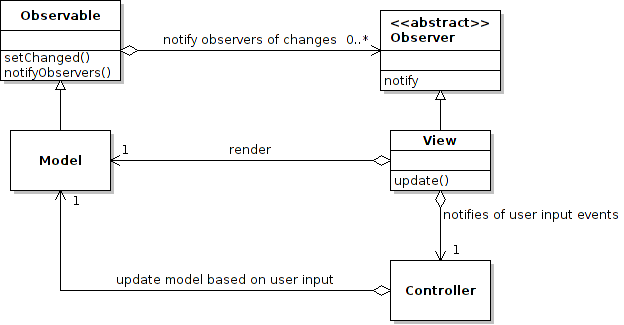
7) Project Glossary

8) Document Revision History

**5. Design & Implementation**

1) Development Process Model

2) MVC Pattern



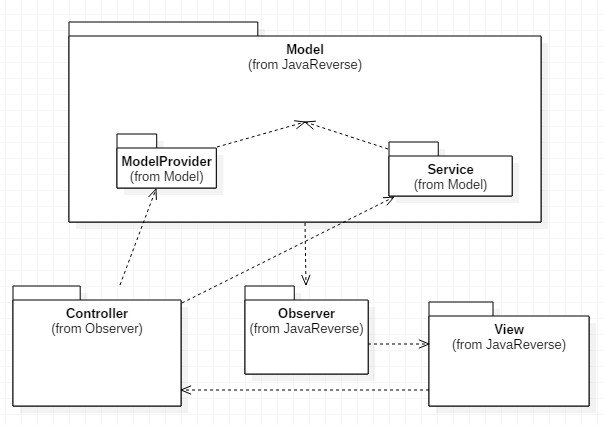
[그림 ] MVC 패턴.

우리는 프로그램을 Observer을 사용한 MVC 패턴을 이용했다. MVC 패턴이란 모듈을 역할에 따라서 모델,뷰,컨트롤러로 나누는 디자인 패턴이다.

M(Model) 모델의 역할은 자료의 구조와 알고리즘을 결정한다. 우리는 데이터 구조와 알고리즘을 좀더 세분하게 나누어 데이터 구조는 Model 패키지에, 알고리즘은 Service 패키지에 나누어 작성하였다.

V(View) 뷰의 역할은 사용자를 위한 정보를 보여준다. 프로그램의 정보가 바뀌면 뷰는 사용자가 알 수 있게 변경사항을 보여준다.

C(Controller) 컨트롤러의 역할은 유저의 입력을 관리한다. 유저가 프로그램을 이용하며 발생한 이벤트를 받아 어떤 행동을 수행 할 것인지 결정한다.



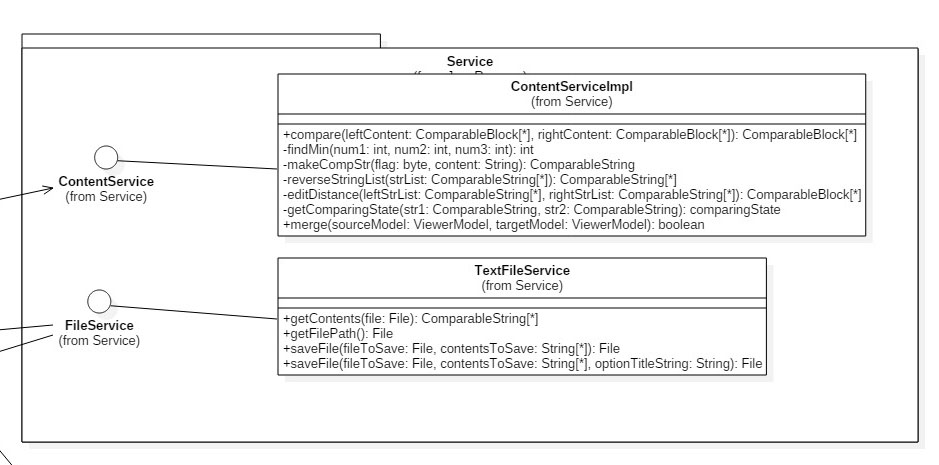
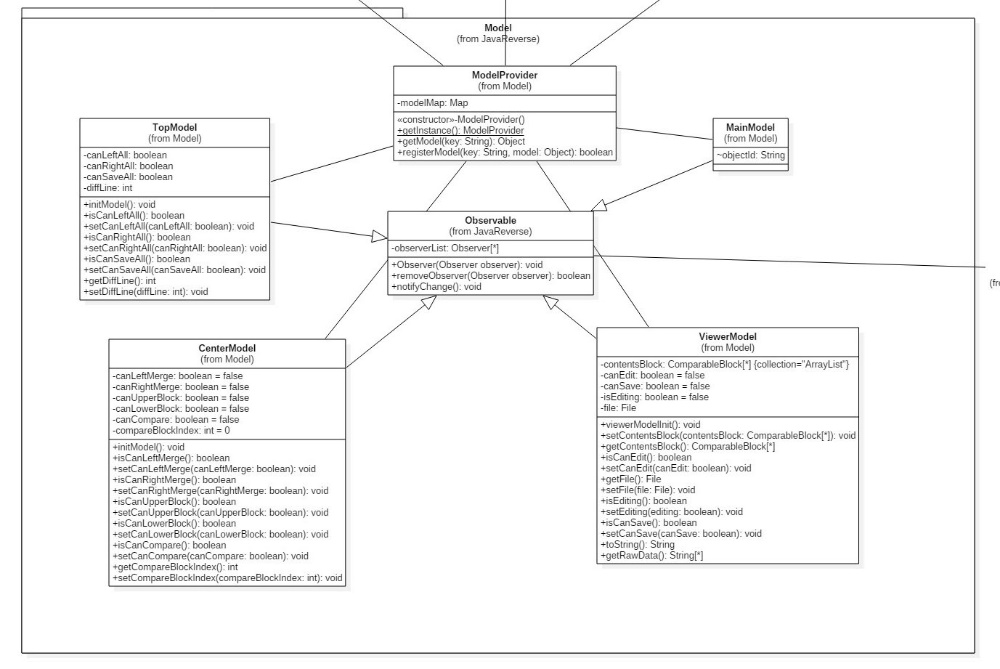
[그림 ] 우리 프로젝트의 MVC 패턴

MVC 패턴의 목적은 각 모델 뷰 컨트롤러가 서로를 몰라도 각 모듈이 작동할 수 있어 재사용성을 높이는 것에 있다. 하지만 모델이 뷰를 수정하기위해서 모델에서 뷰 객체를 가져다가 쓰면 서로 관련이 되어 버리는 결합도가 발생하게 된다.

우리는 모델과 뷰의 결합도를 낮추기 위해 옵저버 패턴을 이용하였다. 옵저버 패턴에서 모델은 직접 뷰를 수정하는 것이 아닌, 모델의 변경사항을 알리는 역할을 한다. 또한 뷰 역시 옵저버의 update 콜이 올 때 뷰를 수정하게 된다. 따라서 모델과 뷰는 서로를 직접 몰라도 되므로 결합도를 낮추는 좋은 방법이라 고 생각하여 사용했다.

모델의 데이터 object만 관리하는 ModelProvider와 데이터를 이용하여 로직을 수행하는 Service로 모델을 나누어 작성하였다.

3) Model Design



[그림 ] Model Class Diagram

데이터 구조인 Model 패키지와 데이터 관련 로직을 처리할 서비스 패키지가 있다

텍스트 관련 구조로 ComparableBlock과 ComparableString이 있다.

**ComparableString**은 상태를 나타낼 수 있는 데이터 구조로, 실제 한 줄의 String을 가지고있을 contents와 데이터의 상태를 나타내줄 state로 구성되어있다.

**ComparableBlock** 은 ComparableString을 state단위로 가지고 있는 데이터 구조이다. 같은 state의 ComparableString을 묶어 merge기능을 수행할 때 쉽게 할 수 있도록 구성하였다.

**ViewerModel**은 텍스트파일 또는 입력으로 저장된 문자들을 ComparableBlockm으로 가지고 있게 된다. 또한 Save/Load를 위한 File 객체를 가지고있고 viewerPanel의 수정,저장 가능 상태를 나타 내어줄 필드값을 가지고있다.

**CenterModel** 은 현재 비교하고있는 ComparableBlock 의 인덱스를 가지고있고, CenterPanel에서 이용 할 현재 compare,merge,up,down 기능이 이용가능한가를 나타내어 주는 필드 값을 가지고 있다.

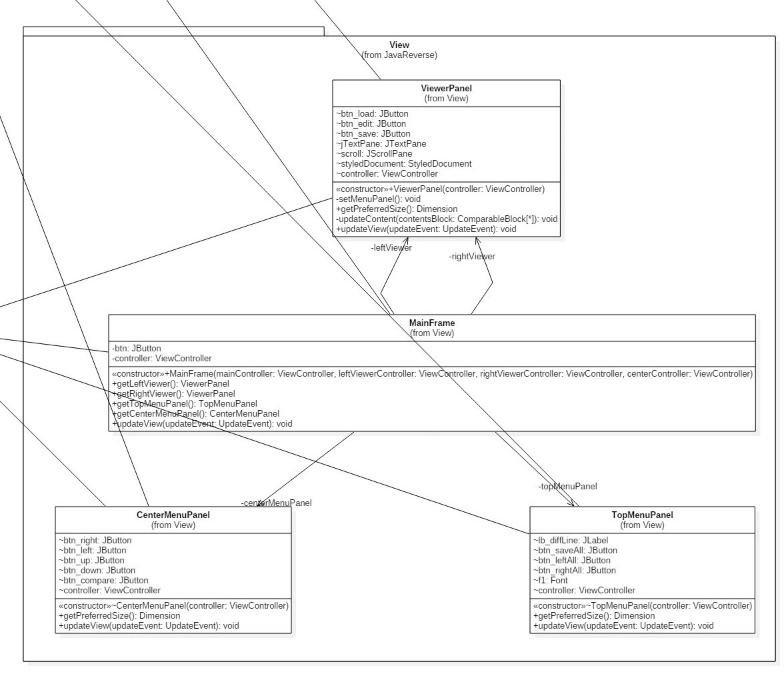
**TopModel** 은 현재 총 다른 라인수가 몇 개인지 나타내어주는 diffLine 필드값과, TopPanel에서 이용할 mergeRightAll, mergeLeftAll 기능 이용가능여부를 나타내어주는 필드 값을 가지고 있다.

**ModelProvider** 는 모델의 객체를 보관하고 저장,꺼내 쓸 수 있게 도와준다. ModelProvider는 싱글톤으로 유지되어 컨트롤러,서비스에서 모델이 필요할 경우 parameter나 setter로 넘겨주지 않고 필요한 곳에서 ModelProvider 객체를 이용하여 모델들을 꺼내 사용할 수 있다.

**ContentService – ContentServiceImpl** ComparableBlock어레이를 비교하는 Compare 함수와 viewerModel 한쪽 ComparbleBlock을 다른 한쪽 block과 같게 만들어주는 Merge 함수가있다.

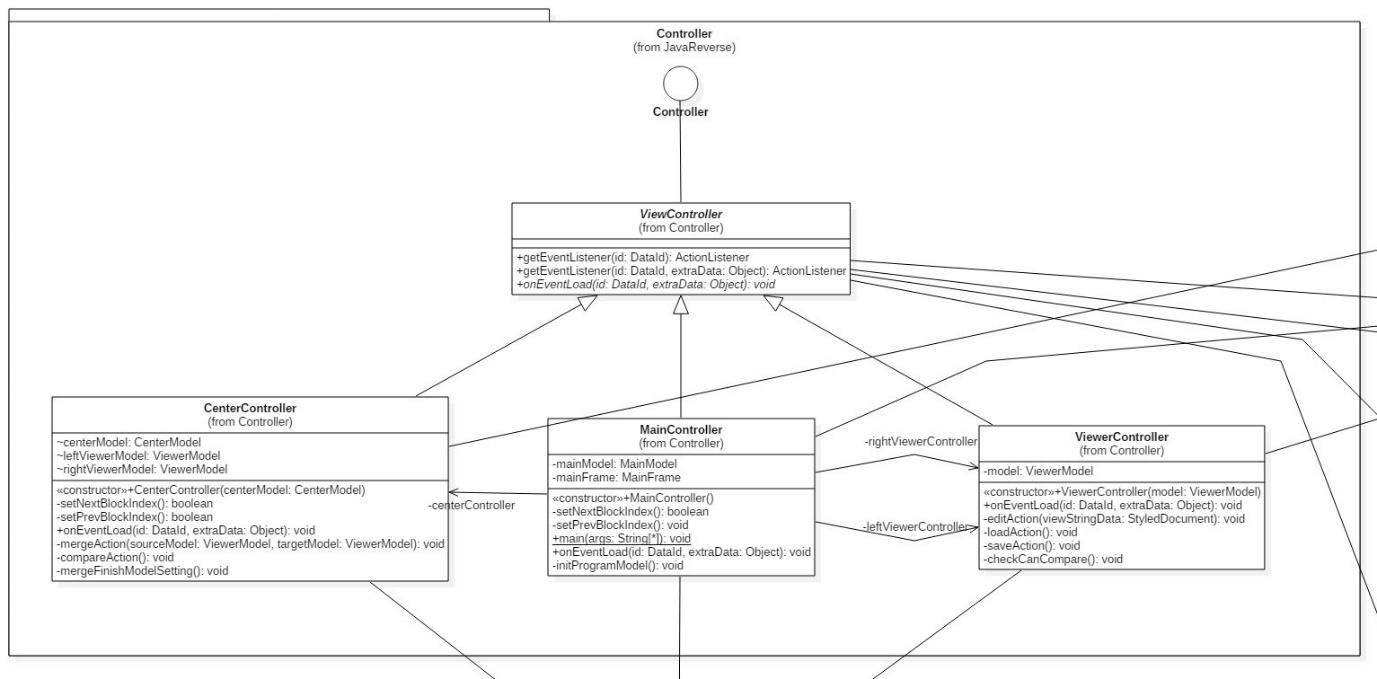
**FileService – FileServiceImpl** File객체를 얻어올수있는 getFile함수와 File객체로부터 ComparableBlock list를 얻어오는 getContents함수, File 객체를 저장하는 saveFile 함수가 있다 즉 파일 관련 로직처리는 FilerService를 통해 처리된다

4) View Design



뷰는 먼저 interface 요구사항에 맞게 swing을 이용하여 mainFrame을 작성하였다. mainFrame에 필요한 화면들을 크게 ViewerPanel 2개, CenterPanel, TopPanel로 나누어 작성하였다. 뷰가 현재 편집중인지, 불러온파일 또는 수정사항이 없는지에 따라 compare,merge,edit,file load/save를 할 수 있거나 없게 설정 하도록 하였다. 각요소에 버튼 등 이벤트가 발생하여 컨트롤러에게 전달하여야 하는 경우, 컨트롤러가 제공하는 EventListner를 받아 등록하였다.

5) Controller Design

Cont

6) Observer Design

7) Util Design

8) Class Diagram

9) State Diagram

**6. Object-Oriented Design Principles**

1) OOP Principles

2) SOLID Principles

3) UML Diagram

**7. Test**

1) Functional Unit Test Cases

2) UI Test Cases

3) System Test Cases

**8. Execution Screen & Explanation**

1) First Screen

2) Load

3) Edit

4) Compare

5) Select Block and Merge

6) Save

**9. Project Management**

1) Chosen Version Control System

2) Project Progress by Github

3) Experience of Our Team