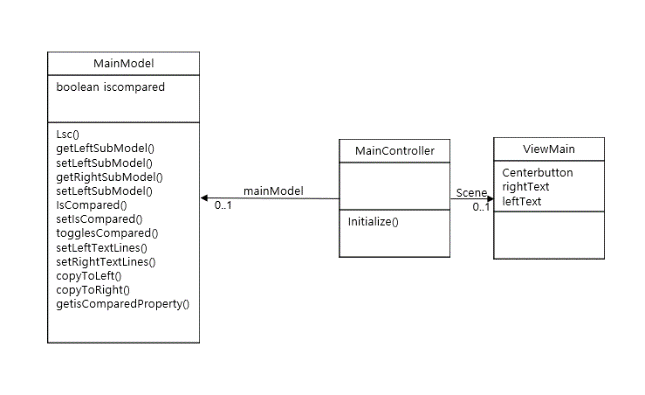


1. Single Responsibility Principle

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.

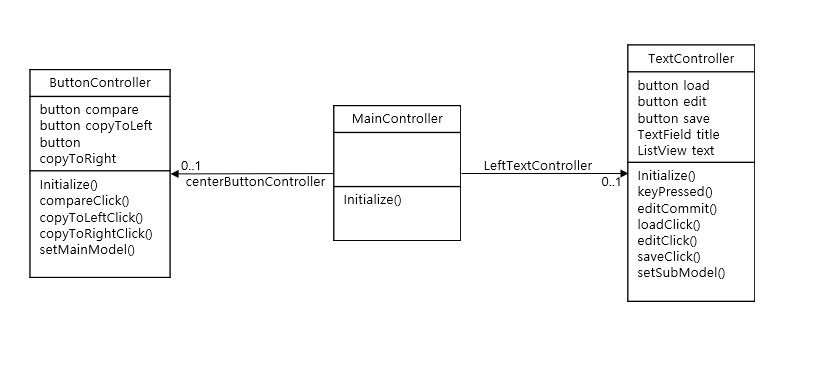
public class MainModel {

|  |
| --- |
|  |
|  | 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.