Analysis & Design, Implementation, and Test

Document

개체이(가) 표시된 사진

매우 높은 신뢰도로 생성된 설명

TEAM 16

**김은경 (20163927) 김소연 (20165417) 김수진 (20160342) 남유선 (20163228)**

**손승표 (20162581) 윤신영 (20163657)**

1. **Domain model (class diagram)**

텍스트이(가) 표시된 사진

매우 높은 신뢰도로 생성된 설명

(you can see more clear diagram images at Github ‘Diagram’ folder!)

1. **Software architecture & Design model**

텍스트이(가) 표시된 사진

매우 높은 신뢰도로 생성된 설명

텍스트, 지도이(가) 표시된 사진

높은 신뢰도로 생성된 설명[System sequence diagram]

1. **Major design decision**

[How to compare the texts?]

By using the algorithms for longest common subsequence(LCS) problem and sequence alignment, we compute the matrix C which stores the LCS length and compare two panels based on the matrix.

[How to distinguish two cases? 1. Different strings are in same line index in each panel 2. The string is in only one panel]

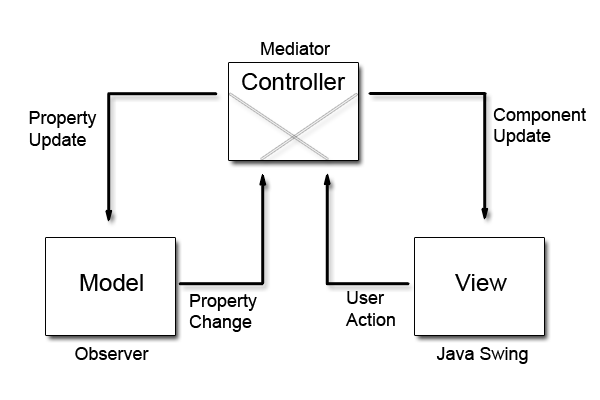
In the case 1, we multiply (-1) to the index of string. In the case 2, the string is stored in the panel which the string is in and the blank value(0) is stored in another panel.

[What data type would be returned to Merge class after comparing in FileComparator class?]

In FileComparator class, the difference of two panel is stored in each integer type arraylist and the block, which store the block’s start and end index, is stored in integer array type arraylist. When merge call getter functions by FileComparator object, all elements in left panel’s arraylist are changed to negative integer, and all in right panel’s are changed to positive integer.

1. **Explanation about how MVC concept was applied**

Here is the picture of MVC model. When we design this program, we referred to this picture. The point is that ‘Model’ can’t be directly connected to ‘View’.



We define ‘view’ classes as {MainView, PanelView, TextLineNumber, CompareTable, CompareTableRenderer}, ‘model’ classes as {Merge, TextEditorModel, FileComparator}, and ‘controller’ as {MergeController, TextEditorController}.

As you can see these two class diagrams (domain model and design model), all the classes in view are not directly connected to any class in model. In other words, controller is the only mediator between model and view. Also, when state of model is changed, controller receives the signal and sends it to the view so that the view can apply the change and show it to the user.

So, concept of MVC model is applied to our program.

1. **Explanation about how OO design principles were applied (with code examples)**

[Encapsulation]

We set member variables as private or protected (there are only few exceptions), also we made public getter & setter methods to access those private or protected member variables. For some methods which are used only within in one class, we set them as private, otherwise, we set methods as private.

[Inheritance]

We use inheritance when we design ‘view’ classes. Here is the pictures of code.

스크린샷이(가) 표시된 사진

매우 높은 신뢰도로 생성된 설명

1. **Explanation about how our program was designed to be testable by Unit-test tools**
2. **Usage of program & screen shots of examples**
3. **Functional unit test cases and results**
4. **System test cases and results**
5. **(If any) functional limitations**