**Software Requirements Specification**

**SimpleMerge Project**

**Team 17**

**Project team member:**

**Lukas Gužauskas**

**Table of contents**

1. Introduction project team 3

2. Introduction 4

3. USE CASE 5

4. Nonfunctional Requirements 8

5. Constraints 13

6. Development and Target Platforms 13

7. Project Glossary 13

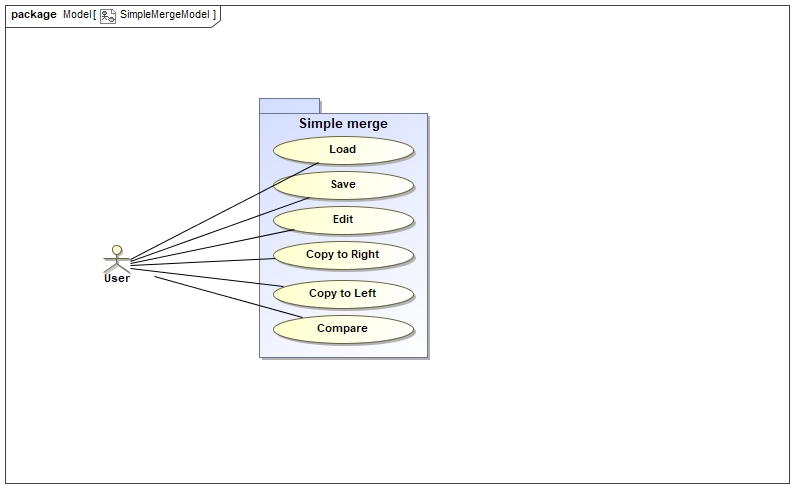
Introduction project team

**Project team 17:**

**Lukas Gužauskas**

1. Introduction

The goal of this project is to create a SimpleMerge information system using Java language. The SimpleMerge displays two text files next to edit panel (side by side) and the user can use several functionalities: load, edit, save, compare 2 files and merge. This document describes the requirements of this program.



1. USE CASE

UC1 Load

UC2 Edit

UC3 Save

UC4 Compare

UC5 Copy to Left

UC6 Copy to Right

1. Flow of Events for *Load* Use Case
   1. Preconditions:

The user must have text files.

* 1. Main flow:

The user can press “LOAD” button [S1-S2] [E1-E2] on the top of the main window. The edit panels on the left and on right display the contents in the corresponding edit panel.

* 1. Subflows:

[S1]. When the “LOAD” button is pressed, the small window shows up. The user selects two text files he wants to display on edit panel on the left and the right in the system. If the user wants to cancel the input file then he can press “Cancel” button [S2].

[S2]. When the Cancel button is pressed, the small window is closed.

* 1. Alternative flows:

[E1]. If user loads text files which is not supported file format, then the system reports error message.

[E2]. If user loads text files which is a blank text file, then the system could read it.

1. Flow of Events for *Edit* Use Case
   1. Preconditions:

The user has uploaded text files into the system and displays the contents in the corresponding edit panels.

* 1. Main flow:

The user can click Edit button [S1] on top of the main window. The user could edit the text files. After editing of content is finished, the user could save the edited content into the file [UC3].

* 1. Subflows:

[S1]. When the Edit button is clicked. The program unlocks the edit panel on the left or the right depending what side the user selects by pressing the left or the right button.

* 1. Alternative flows:

None.

1. Flow of Events for *Save* Use Case
   1. Preconditions
2. The user has uploaded text files into the system and displays the contents in the corresponding edit panels.
3. The user has edited text files.
   1. Main flow:

The user could click in two ways: Save As button [S1] or Save button [S2] on top of the main window [E1-E3], then the edited content of the file turns saved.

* 1. Subflows:

[S1]. If the Save As button is clicked, a directory window shows up asking the user for where the file could be saved. The user specifies name of the directory to place the file and clicks Save button.

[S2]. If the Save button is clicked, the edited content of the panel is saved to the same file.

* 1. Alternative flows:

[E1]. If the user specifies an invalid name (e.g. space symbol inserted) of the file, then the program shows up a pop-up window which is reporting an input error.

[E2]. If the user requests Save As that the file name is the same as the original file in the specified folder, the program rewrites the previous file.

[E3]. If the user hasn’t uploaded [UC1] the file into the panel and attempts requesting to run [S1-S2], then the program reports error message with pop-up window for the user.

1. Flow of Events for Compare Use Case
   1. Preconditions:

The user has uploaded text files into the system and displays the contents in the corresponding edit panels.

* 1. Main flow:

The user could click Compare button [S1] [E1-E3]. Then the program displays the different lines highlighted with color.

* 1. Subflows:

[S1]. When the Compare button is clicked, the program executes comparison of two text files

* 1. Alternative flow:

[E1]. If the user attempts to click Compare button without uploaded 2 files into both panels, nothing occurs, and none information is displayed.

[E2]. If the left panel is a blank text, other panel has some lines of the text then the comparison results will make up non-highlighted text on the right panel. So, nothing occurs, and none information is displayed.

[E3]. If the both panels have the same contents, pop-up window shows up that the new content is still the same as the previous content.

1. Flow of Events for *Copy to Left* Use Case
   1. Preconditions:
2. The user has uploaded text files into the system and displays the contents in the corresponding edit panels.
3. The compare feature must be executed to compare 2 contents and must be viewed different lines.
   1. Main flow:

The user could select to switch the lines that are different. Further, the user could click the Copy to Left button [S1] [E1] on top of the main window. Then the selected line will show up in the left panel.

* 1. Subflows:

[S1]. After the selected line, the marking is moving to the left panel of the next line by clicking Copy to Left button.

* 1. Alternative flows:

[E1]. If the panel on the right contents blank text, then [UC5] results to delete lines in the left panel.

1. Flow of Events for *Copy to Right* Use Case
   1. Preconditions:
2. The user has uploaded text files into the system and displays the contents in the corresponding edit panels.
3. The compare feature must be executed to compare 2 contents and must be viewed different lines.
   1. Main flow:

The user could select to switch the lines that are different. Further, the user could click the Copy to Right button [S1] [E1] on top of the main window. Then the selected line will show up in the right panel.

* 1. Subflows:

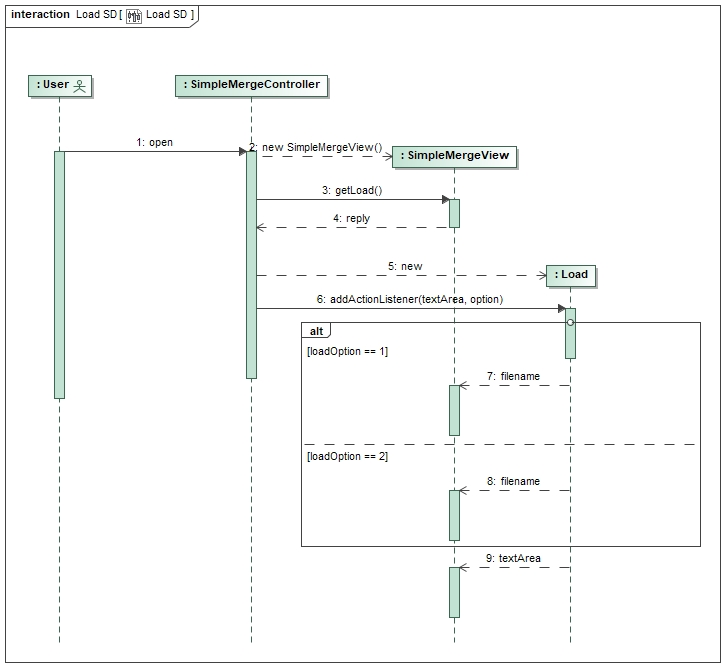
[S1]. After the selected line, the marking is moving to the right panel of the next line by clicking Copy to Right button.

* 1. Alternative flows:

[E1]. If the panel on the left contents blank text, then [UC5] results to delete lines in the right panel.

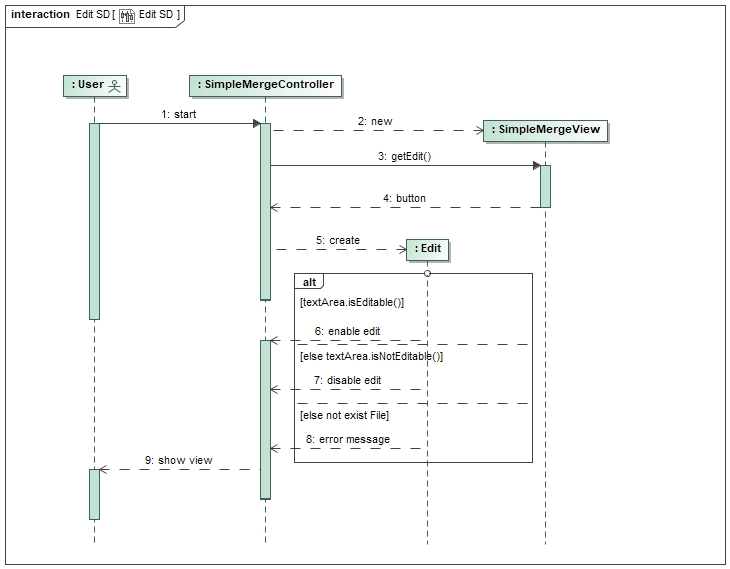
1. System Sequential Diagrams

Load SSD



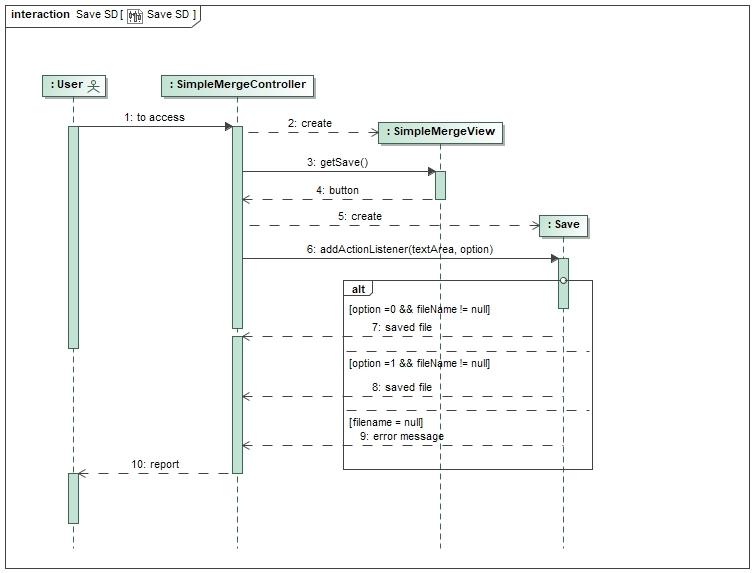
Description: The user is outside the boundary of the system. SimpleMergeController is an instance created by SimpleMergeController class. There are functions that create the SimpleMergeView main window from the controller. So, the user clicks a button from View class, then function creates Load class which is load method operator which indicates the panel area where the text file is loaded. Filename is the element which defines a name of the text file. So, there is alternative condition if loadOption is 1, then the file information transferred to the left panel area. If loadOption is 2 – to the right panel area.

Edit SSD



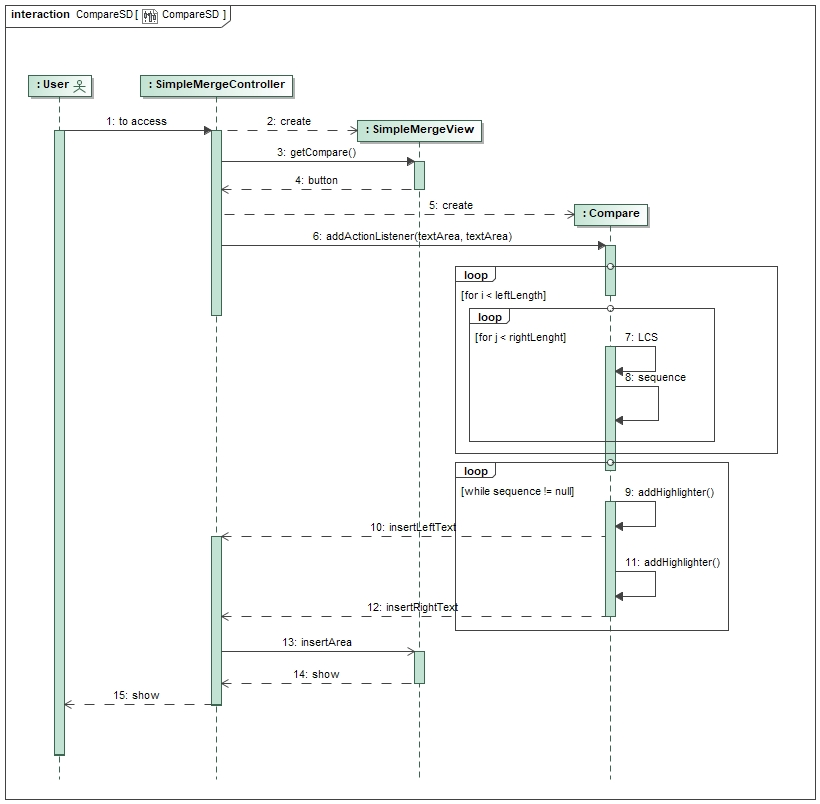
Description: By default, program still locks the text area. In this case, the user has to click Edit button, then the Controller calls method from the Edit class which is implemented for unlocking the text area. The controller shows text area which is unlocked, and the user could be able to edit some content. In the case, if the user hasn’t uploaded the file to the program and attempts to click Edit button, then the program will report error message, also the user can’t edit on the panel area.

Save SSD



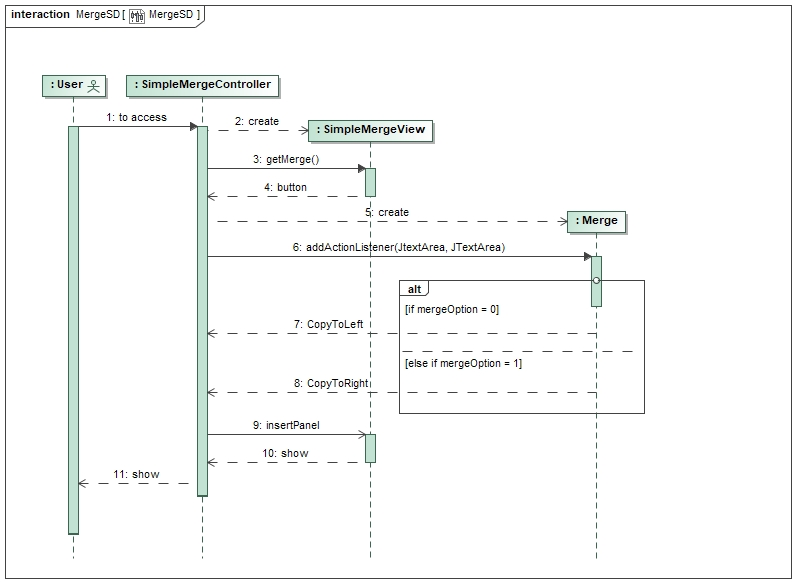
Description: The user has to click Edit button, then the Controller calls method from the Save class which is implemented for saving to the file. There is alternative condition, if file path exists for the file on the left or the right then the Controller saving to file is successful and reports to the user to be aware. In the case, if file path does not exist for the file on the left or the right then the Controller reports the user with error message.

Compare SD



Description: The user has to click Compare button, then the Controller calls method from the Compare class which is implemented for comparing two files on the left and the right. In the Compare class, there are 3 loop conditions. 2 loop conditions are implemented LCS algorithm, so some elements are inserted into arrays of LCS and sequence. After 2 loop conditions, coming to the last loop condition that is to compare between texts of the left and the right where the different sequences are. If texts find the different sequences between the left and the right, color highlight is added on some sequences by the text of the left or the right. After adding color highlight, the new contents are inserted to 2 TextArea panels on SimpleMergeView class. In the case, if texts not find any the difference sequence between the left and the right then it will be the same old contents which aren’t changed after clicking Compare button.

Merge SD



Description: The user has to click “Copy to Left” or “Copy to Right” button after indicating the differences found during the comparison, then the Controller calls method from the Merge class which is implemented for copying the selected block to the file shown in the left or the right panel. There is alternative condition, if merge option is equal 0 then the feature executes the process which copies the selected block to left panel or if merge option is 1 – copies to right panel. In the case, the user isn’t able to transfer any selected block in the panel on the left or the right, it will be the same old contents which indicate the differences after clicking “Copy to Left or Right” button.

1. Nonfunctional Requirements

**NR1. Performance**

All functions of the SimpleMerge program could be completed from the requested function by the User. All functions shall be completed quickly.

**NR2. Usability**

1. GUI (Graphical user interface): The system shall allow a user to interface with it through mouse events on button and view events on the text panels.
2. Language: All the contents of the system is supported English language.
3. Test framework: The SimpleMerge program is to able used Junit and EasyMock testing.

1. Constraints

All code development shall be done with the Java programming language.

All testing shall be done using Junit and EasyMock.

1. Requirements Dependency Traceability Table

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | UC1 | UC2 | UC3 | UC4 | UC5 | UC6 | NR1 | NR2 |
| UC1 |  |  |  |  |  |  |  |  |
| UC2 | X |  |  |  |  |  |  |  |
| UC3 | X | X |  |  |  |  |  |  |
| UC4 | X |  |  |  |  |  |  |  |
| UC5 | X |  |  | X |  |  |  |  |
| UC6 | X |  |  | X |  |  |  |  |
| NR1 |  |  |  |  |  |  |  |  |
| NR2 |  |  |  |  |  |  |  |  |

1. Development and Target Platforms
2. Eclipse development MVC
3. MagicDraws
4. JUnit 5
5. EasyMock
6. Project Glossary

Panel – a predefined display image in the program window which is to able loaded from text files. There are two panels of the left and the right in the program.

1. Document Revision History

|  |  |
| --- | --- |
| Version | 1.2 |
| Name(s) | Lukas Gužauskas |
| Date | 6/6/2018 |
| Change Description | Original creation of the SRS |