**SE 2018 Spring Term Project:**

**Development of SimpleMerge**

**Team size:** At most **six students** in a team (You should indicate the role of each participant)

**1st Due: May 3 Thurs.**

1. Join a team at e-class

(2) Construct GitHub project and post the address at the e-class team board

**2nd Due: May 18 Fri.**

Upload Software Requirement Specification(SRS) to both e-class and github

**Final Submission Due: June 8 Fri.**

**Upload All the source/data files** **+ updated SRS + Analysis/Design/Implementation/Test Report + GitHub project management report to e-class**

**Note: All of your project activities (source & document update etc) should be trackable at GitHub!**

**Demo Day: June 9 Sat. 11AM**

**Evaluation Criteria:** Demo Evaluation + Software Requirement Specification(SRS) + Analysis/Design/Implementation/Test Report + GitHub project management report

Caution: We have accumulated more than 4 year’s artifcats for this project. PLEASE DO NOT ATTEMPT TO STEAL THE ARTIFACTS(Code & Documents) FROM THE PREVIOUSLY SUBMITTED PROJECTS since there will be a severe penalty for academic dishonesty.

**Project Description:**

The goal of this project is to create *SimpleMerge* of which main functionality is to compare and merge the files. You may want to refer to the similar extant products such as WinMerge(http://www.winmerge.org, WinDiff, and BeyondCompare to understand the general concepts and usage of this kind of software.

The **MUST features** of the software are shown in the following:

(1) Functionality of **viewing/editing/saving** files

At start-up, a main window with two edit panels (side-by-side) is displayed. On top of each edit panel, there are buttons labeled with "Load", "Edit", "Save".

If the user presses a "Load" button, then the program should allow the user to choose a file in the file system, load the contents of the file, and displays the contents in the corresponding edit panel.

If the user presses a "Edit" button, then the program should allow the user to edit the strings shown in the edit panel.

If the user presses a "Save" button, then the program should save the edited content into the file

(2) Functionality of **comparing** two files

The main window has a "Compare" button. If the button is pressed, then the program should display the different lines with a colored font/background. The comparison is done line by line. Your comparison result should be same as the one generated from "diff" program. The "diff" program is a utility comparing two files using "Longest common subsequence(LCS)" algorithm.

(3) Functionality of **merging** two files

The user should be able to traverse the blocks indicating the differences after the comparison and merge the differences. There are two basic merge buttons. **“Copy to Right”** button **copies the selected block** in the left panel to the file shown in the right panel. Similarly, **“Copy to Left”** button does to the file shown in the left panel. Note that series of executions of the merge operations will make the corresponding blocks be identical.

Note that you should try to use **Testing Frameworks** such as **JUnit** and **EasyMock** as much as possible during the development. Not only the functionalities of your program, but also documentation, analysis & design, test efforts will be counted as major evaluation criteria. In order to fully utilize JUnit and EasyMock, your GUI components should be designed by using MVC architecture pattern. You should learn the **concept of MVC** before designing your program. For test cases, you need to include both correct behavior checking in normal situations and reaction checking in abnormal situations.

**Project Summary:**

Create SimpleMerge program. Design your program to be unit-testable. Use Testing Frameworks such as Junit and EasyMock during the development.

**Final Submissions should include the following materials:**

**(1) All the executable files/source code/configuration data/test code/test data files**

* All the executable files/source codes, configuration, test codes, test data should be submitted
* Include a document “README” explaining how to execute and build your software

**(2) (Use Case-based) Software Requirement Specification (See attachment for samples)**

* Introduction to the system
* Use case diagrams
* Use case descriptions
* **System sequence diagrams**
* Non-functional requirements (Quality requirements, Constraints, etc)
* Requirement Dependency Traceability
* Development and Target Platforms
* Project Glossary
* Document Revision History

**(3) Analysis & Design, Implementation, and Test Document:**

* Domain model (utilize class diagrams)
* Software Architecture + Design Model (utilize Class diagrams, Sequence diagrams, Statechart, etc)
  + **Clearly indicate major design decisions!**
  + **must include the explanation that how MVC concept was applied to your design**
  + **must include explanation that how your program was designed to be testable by Unit-test tools.**
  + explain how **object-oriented design principles** were applied to your design **along with diagrams/code examples**
* usage of your program and the screen shots of examples
* Functional **unit test cases** and **their results**
* System test cases and their results
* NOTE: In case your program is not fully functional, you should mention those limitations in detail.

**(4) Project Management Report**

* **Indicate the address of your github project repository!**
* Briefly explain your project progress history
* Briefly explain your experience

**Hint:**

You may read the following articles in wikipedia to understand the algorithms for longest common subsequence problem and the "diff" utility.

http://en.wikipedia.org/wiki/Longest\_common\_subsequence\_problem

http://en.wikipedia.org/wiki/Diff