**Software Engineering Team Project (Assignment 2)**

21000369 Kyungbae Song, 21200525 Sangjun Lee,

21300212 Haneul Kim, 21300807 Goun Han**,** 21400549 Seunghwan Lee

**Current progress of our project: Implementing of IR, parser, document, code generator by applying the visitor pattern. We can generate the lists of node, but we cannot generate lists of token yet.**

1. Main class receive input the name of file on CMD than parsing method is executed, file content and Document object are the parameters.
2. Parsing method separates contents of the file line by line and make them to node list. After that, each node makes strings to token list.
3. Visit plain, fancy, and slide visitor depending on the type, and get HTML code.
4. Check the HTML syntax using jTidy.
5. Create completed HTML file.

**Below is the explanation of nodes currently implemented.**

* Header – this program is suitable for text-style headers which are “underlined’ using equal signs (for first-level header) and dashes (for second level headers). Any number of underlying “=” or “-“ will work.

\* Exception in using markdown is not considered. We developed this program under the premise that user has sufficient knowledge about markdown syntax and knows how to use the syntax properly. For example, the expression ‘===’ is classified as a header. However, if a user uses the expression like ‘==\*=’, this should be classified as a plain text, but our program for now cannot handle the exception like this. For now, the expression ‘==\*=’ is also classified as a header.

* Item List – if a sentence starts with asterisk and one space, it is Item List node.
* Code Block – if a sentence starts with 4 space or 1 tab, it is code block.
* Quoted Block – if a sentence starts with “> “, it is quoted block.

**Here is the explanation about MD Parser and Code Generator.**

* Node, Token: A class and attributes for making node to token are already built, but the function to do that is not implemented yet. The major problem to consider for making node to token is the nested situation. For instance, ‘\*\*\*abc\*\*\*’ should be converted as ‘<strong><em>abc</em></strong>’ or ‘<em><strong>abc</strong></em>’. We have to figure out how to treat the situation like this.
* Code Generator: If a node is successfully separated to token, Converter can conduce translation process. For now, plain, style, image, and link can be translated if there is no problem of nested situation. We expect that if the tokenize process is performed well, Code Generator will be implemented easily.

**Now, following content is about Ant and jTidy applied to our project.**

We added build.xml to java project. In this file, there are five targets; build, compile, copy, jar, and clean. Build is set as default target and copy, compile, jar, and clean is all performed in here. Copy target creates lib folder in current location, copies every file from sourceDir to targetDir. Compile target compiles all java files in sourceDir and store created class file in targetDir. Jar target compresses java files in targetDir and the name of the compressed file will be convert.jar. Clean target removes class files in targetDir. Each target is depended by the order according to the purpose.