CS4218 Milestone 1 Report

Team\_15 BF+EF2

CAO SHENGZE

KIM HYUNG JON

YAP HAN CHIANG

**Plans for implementation and testing**

**Implementation**

* **main**
  + The role of the main function is to read inputs, process each input line to obtain an array of string, and pass each string in the array to the **parseAndEvaluate** function.
  + Notice that the semicolon operator is evaluated inside the main function. Therefore, the test for the semicolon operator need to relay on the static function instead of using the parseAndEvaluate.
* **parseAndEvaluate**
  + Here, each string is processed for pipes and transformed into an array of string, and each string is passed to **CallCommand**.
  + parseAndEvaluate is the function mostly used in test cases. The behavior of the shell can be performed with the parseAndEvaluate function.
  + The parseAndEvaluate function is also responsible for the processing of the pipeline.
* **CallCommand** helper class
  + We have a **CallCommand** helper class to parse each line into their individual components, which are the application name, the arguments, input file redirection and output file redirection.
  + CallCommand also do the processing of all three kinds of Quotes: The double quotes, the single quotes as well as the back quotes, which is the command substitution.
  + CallCommands will call the respective’s application, which are defined it their own individual files.
* **CatApplication**
  + Can use both stdin or the file reader to read the file.
  + Output to the given output stream
* **CdApplication**
  + Would influence the global variable: directory.
  + We handle the special cases with something like ../
* **EchoApplication**
  + Can read from stdin only. Cannot read from files directly.
* **HeadApplication**
  + Similar to cat.
* **TailApplication**
  + Similar to head.
  + Count the number of lines of the input first, then decide which lines to print and which lines not.
* **SedApplication**
  + All the interface methods are remaining the same. The command would be parsed twice: The first time it would decide which certain api should be used (read from stdin or file, all or the first one). The second time of parsing is implemented inside each certain interface function.
* **WcApplication**
  + **We make some changes to the interface here: Previously, the argument should represent the given whole command. But then we find out the given interface cannot deal with the two output situation. For example, -lm. Therefore, we give up the given logic where the args is the original command. Instead, our args is still a string but it represents the content of the given file or stdin. This also makes our unit test much easier.**
* **The date**
  + One of the easiest application among all applications. Just use a Java.date class and all problems solved.
* **Semicolon Functionalities:**
  + For semicolon, what we do is before we process the command line, we look for unquoted semicolon. If found ,then the command would be split to different commands and executed one by one.
* **Quotes Functionalities:**
  + Similar to Semicolon, first look for all special symbols, then for the ones not quoted by other quotes( for example, back quotes would not work when it is inside a pair of single quotes.
* **Pipe Functionalities:**
  + Before passing the commands to the CallCommand function, the parseAndEvaluate functions would detect the valid | and then get a list of commands. With the help of pipeStream, we can easily use the output from one command become the input to the next command.

**Testing**

* We initialise the application to be tested in the **@BeforeClass**, and the application test related variables such as input and output stream in the **@Before**

**Test cases (**To addon forshell stuff like pipe, call)

* The test cases we came up with tested the results of calling the method in boundary cases. E.g. for cat, head application, the cases are testing it with empty arguments, testing it with file only, and testing it with invalid file.