**Assignment**

**Assignment Goal**

Application of design principles for a simple multi-threaded application.

**Programming Language**

You are required to program using Java.

**Compilation Method**

* You are required to use ANT for compilation of code written in Java.

**Project Description**

* From the command line, accept the following as input:
  + The name of the input file (referred to as input.txt below)
  + The name of the output file (referred to as output.txt below)
  + The number of threads to be used: referred to as NUM\_THREADS below
  + Debug Value: an integer that controls what is printed on stdout
    - Validate that the correct number of command line argumets have been passed.
    - Validate that the value of NUM\_THREADS is between 1 and 3.
    - Validate that the DEBUG\_VALUE is between 0 and 4.
  + Update from Assignment-1: There are 7 courses (A, B, C, D, E, F, G) being offered in the summer session. The capacity for each course is 60. The total number of students is 80. Each student is required to register for 5 courses. The student is asked to provide a preference for each of the courses. Top preference is specified as "1", while the lowest preference is specified as "7".
  + The Driver code should create an instance of CreateWorkers and pass an instance of the FileProcessor, Results, other instances needed by the Worker to the constructor of CreateWorkers. The Driver should invoke the method startWorkers(...) in CreateWorkers and pass the NUM\_THREADS as an argument.
  + CreateWorkers' startWorkers(...) method should create NUM\_THREADS threads (via the threaded class WorkerThread), start them and join on them. The instances fo FileProcessor, Results, and classes needed for the scheduling should be passed to the constructor of the worker thread class.
  + The *run* method of the worker thread should do the following till the end of file is reached:
    - While the end of file has not been reached:
      * Invoke a method in the FileProcessor to read one line as a string
      * Run your algorithm to assign courses to this student.
      * Store the results in the data structure in the Results instance
  + The choice of data structure used in the Results class should be justified in the README.txt in terms of space and/or time complexity.
  + The Results class should implement an interface, StdoutDisplayInterface. This interface should have a method writeSchedulesToScreen(...). The driver should invoke this method on the Results instance to print the schedules, average preference score, etc. (similar format as Assignment-1) for all the students.
  + The Results class should implement an interface, FileDisplayInterface. This interface should have a method writeSchedulesToFile(...). The driver should invoke this method on the Results instance to print the schedules, average preference score, etc. (similar format as Assignment-1) for all the students to the output file.
  + Use an Object Pool instance to manage the 7 courses. If your algorithm requires you to return a spot in a course, then you should use the ObjectPool API for this purpose. You should ObjectPool API to request a spot in a course, check availability, etc.
  + Assume that the input file will have one unique string per line, no white spaces, and no empty lines. Also assume that the intput strings in the file do not have errors. The entries in every line are space delimited.
  + Except in the Logger, Object Pool class, do not make any other method static.
  + The DEBUG\_VALUE, read from the command line, should be set in the Logger class. Use the DEBUG\_VALUE in the following way:
    - DEBUG\_VALUE=4 [Print to stdout everytime a constructor is called]
    - DEBUG\_VALUE=3 [Print to stdout everytime a thread's run() method is called]
    - DEBUG\_VALUE=2 [Print to stdout everytime an entry is added to the Results data structure]
    - DEBUG\_VALUE=1 [Print to stdout the contents of the data structure in the store]
    - DEBUG\_VALUE=0 [No output should be printed from the application, except the line "The average preference value is X.Y" ]
  + The Logger class should have a static method to writeMessage(...).
  + Place the FileProcessor.java in the util/ folder.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_