Assignment 4

Due date

- 11.59 PM EST, November TBA

Submit your code as per the provided instructions. A signup sheet will be provided to you during class to setup an appointment with the TA to provide a demo of your project.

Updates

Assignment Goal

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

Team Work

- CS 442: team of two students each.
- CS 542: team of two students each.

Programming Language

You are required to program using Java.

Compilation Method

- You are required to use ANT for compilation of code written in Java.
- Your code should compile and run on *remote.cs.binghamton.edu* or the *debian-pods* in the Computer Science lab in the Engineering Building.

Policy on sharing of code

- EVERY line of code that you submit in this assignment should be written by your team or be part of the code template provided for this assignment. Do NOT show your code to any other group. Our code-comparison software can very easily detect similarities.
- Post to the listserv if you have any questions about the requirements. Do NOT post your code to the listserv asking for help with debugging.

Project Description

- From the command line, accept the following as input, in this order:
  - 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
  - The words that need to be deleted
  - 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 an integer between 1 and 3.
    - Validate that he number of delete words is equal to NUM_THREADS
    - Validate that the DEBUG_VALUE is an integer between 0 and 4.
  - The input.txt file will have words that are delimited by whitespace or newline.
  - Here is an example of input.txt file:

```
A quick brown fox jumps over the lazy dog
The question of whether a computer can think is no more interesting than the question of whether a submarine can swim Edsger W Dijk

The best programs are written so that computing machines can perform
them quickly and so that human beings can understand them clearly A
programmer is ideally an essayist who works with traditional aesthetic
and literary forms as well as mathematical concepts, to communicate
the way that an algorithm works and to convince a reader that the
results will be correct Donald Ervin Knuth Selected Papers on Computer
Science
```

  - Here is a sample output file (does not correspond to the input file above:

```
The total number of words: 50
The total number of characters: 200
The total number of distinct words: 30
```

  - It is possible for the delete words to be repeated. For example, "XYZ1", "XYZ2", "XYZ1".
  - The input file should be parsed and stored in a tree. Each distinct word should be stored in a distinct node. Choose the tree (BST, AVL, etc.) to optimize the operations to determine the total number of words, total number of characters, and total number of distinct words. Justify your choice of tree in the README.txt file in terms of time complexity for each of the three operations, and for the best, worst, and average case. You cannot use any data structure outside the tree to make these determinations. It is acceptable to have a data structure inside the Node.
  - Start all the *populate* threads (total NUM_THREADS) to populate the tree. Your code should ensure that a word is ready only once.
  - Next, start the *delete* threads (total NUM_THREADS) to search for the occurrences of the delete word in the tree. A thread should only reduce the occurrence of the delete word by 1. So, there are 5 occurences in the tree, a thread should reduce it to 4. However, if another thread also has the same delete word, then

there should be 3 occurrences remaining in the end. If the number of occurrences is 0, then a thread should skip updating that word. If a Node ends up with 0 words, then the node should NOT be deleted. As the NUM_THREADS and number of delete threads are equal, assign a deleted word to each search thread.
- The total number of words, characters, and distinct words should be determined after the populate and delete operations have completed.
- 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 startPopulateWorkers(...) in CreateWorkers and pass the NUM_THREADS as an argument.
- The Driver should invoke a method on a reference (you decide which method and reference are appropriate) to determine the total number of words, characters, and distinct words.
- The Driver should invoke the method startDeleteWorkers(...) in CreateWorkers.
- You can pass arguments, as necessary, to the methods in CreateWorkers.
- CreateWorkers' startPopulateWorkers(...) method should create NUM_THREADS threads (via the threaded class PopulateThread), start them and join on them. The instances fo FileProcessor, Results, and classes needed for the scheduling should be passed to he constructor of the worker thread class.
- CreateWorkers' startDeleteWorkers(...) method should create NUM_THREADS threads (via the threaded class DeleteThread), start them and join on them. The instances fo FileProcessor, Results, and classes needed for the scheduling should be passed to he constructor of the worker thread class.
- The *run* method of the worker threads 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 insert/search/delete
- The choice of data structure used in the Results class should also be justified in the README.txt in terms of time complexity (average, best, worst case).
- The Results class should implement an interface, StdoutDisplayInterface. This interface should have a method writeToScreen(...). The driver should invoke this method on the Results instance whenever the appropriate debug level is enabled (for your own debugging purpose).
- 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 to output.txt, irrespective of the debug level.
- Assume that the input file will not have any punctuations or special characters. It is possible for words to be separated by multiple white spaces and empty lines. It is possible for the input file to be empty.
- Except in the Logger, 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 [YOU DECIDE and explain in README.txt]
  - DEBUG_VALUE=1 [YOU DECIDE and explain in README.txt]
  - DEBUG_VALUE=0 [No output should be printed from the application, except the line "The average preference value is X.Y"]

Sample Input Files sent by students in this course

Please note that I have not verfied these input files.

Code Organization

```
firstName_lastName_firstName_lastName_assign_4/wordTree
firstName_lastName_firstName_lastName_assign_4/wordTree/src
firstName_lastName_firstName_lastName_assign_4/wordTree/src/BUILD
firstName_lastName_firstName_lastName_assign_4/wordTree/src/BUILD/classes
firstName_lastName_firstName_lastName_assign_4/wordTree/src/wordTree
firstName_lastName_firstName_lastName_assign_4/wordTree/src/wordTree/driver
firstName_lastName_firstName_lastName_assign_4/wordTree/src/wordTree/driver/Driver.java
firstName_lastName_firstName_lastName_assign_4/wordTree/src/wordTree/store
firstName_lastName_firstName_lastName_assign_4/wordTree/src/wordTree/store/Results.java
firstName_lastName_firstName_lastName_assign_4/wordTree/src/wordTree/threadMgmt
firstName_lastName_firstName_lastName_assign_4/wordTree/src/wordTree/threadMgmt/CreateWorkers.java
firstName_lastName_firstName_lastName_assign_4/wordTree/src/wordTree/threadMgmt/PopulateThread.java
firstName_lastName_firstName_lastName_assign_4/wordTree/src/wordTree/threadMgmt/DeleteThread.java
firstName_lastName_firstName_lastName_assign_4/wordTree/src/wordTree/util
firstName_lastName_firstName_lastName_assign_4/wordTree/src/wordTree/util/FileProcessor.java
firstName_lastName_firstName_lastName_assign_4/wordTree/src/wordTree/util/Logger.java
firstName_lastName_firstName_lastName_assign_4/wordTree/src/wordTree/util/StdoutDisplayInterface.java
firstName_lastName_firstName_lastName_assign_4/wordTree/src/wordTree/util/FileDisplayInterface.java
firstName_lastName_firstName_lastName_assign_4/wordTree/src/build.xml
firstName_lastName_firstName_lastName_assign_4/README.txt
```

- If you are working on a team project, change the top level directory to firstName_lastName_firstName_lastName_assign4

Submission

- Same as before.

- Both the team members should submit to blackboard.

General Requirements

- Start early and avoid panic during the last couple of days.
- Submit a README.txt file (placed at the top level). The README.txt file should be filled out as described in that file.
- Apply all design principles (wherever applicable).
- Separate out code appropriately into methods, one for each purpose.
- You should document your code. The comments should not exceed 72 coloums in width. Use javadoc style comments if you are coding in Java. For at least 10 methods, document one parameter in Javadoc style.
- Do not use "import XYZ.*" in your code. Instead, import each required type individually.
- Every class that has data members, should have corresponding accessors and mutators (unless the data member(s) is/are for use just within the method.).
- If a class does not implement an interface, you should have a good justification for it. For example, it is ok to have an abstract base class and a derived class, wherein both do not implement interfaces. Note that the Driver code is written by end-users, and so the Results class must implement the interface, or else the source code for Results will have to be exposed to the end-user.

Design Requirements

Late Submissions

- Same as Assignment-1.

Grading Guidelines

TBA

mgovinda at cs dot binghamton dot edu
Back to CSX42: Programming Design Patterns