Assignment 3

Due date

- Due by 11.59 PM EST on October 28th.

Github link

The git links are the following:

- For individual assignments: https://classroom.github.com/a/MOvdXMTo
- For Group assignments: https://classroom.github.com/g/9QJy5jYZ

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

- Tue Oct 16 21:08:17 EDT 2018: Posted a link to MyLogger.java

Assignment Goal

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

Team Work

- CS 542: No team work is allowed. Work individually.
- CS 442: teams of two students.

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 *bingsuns* 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 you 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:
  - The name of the input file (referred to as input.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 5.
    - Validate that the DEBUG_VALUE is between 0 and 4.
  - The Driver code should create an instance of CreateWorkers and pass an instance of the FileProcessor, Results, and IsPrime 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 borrow NUM_THREADS threads (via the threaded class WorkerThread), start them and join on them. The instances fo FileProcessor, Results, and IsPrime should be passed to he constructor of the worker thread class.
  - Design a thread pool to borrow threads (it is ok in this assignment to not return threads to the pool). Implement the "borrow" method in the thread pool, and then any other method you need.
  - 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
      - Check if it is a prime number
      - Store the prime number in the data structure in the Results instance
  - The IsPrime class should consider all non-even numbers to be prime (for simplicity). The documentation should indicate that it is a placeholder for now.
  - 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 writeSumToScreen(...). The driver should invoke this method on the Results instance to print the sum of all the prime (just odd numbers, really) numbers.
  - The output should be "The sum of all the prime numbers is: XYZ".
  - 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 can be converted to integers. However, it is possible that the input file has no strings at all.
  - Except in the Logger class and the thread pool pattern, if you plan to use it), do not make any other method static. Here is code you can extend and use for MyLogger.java
  - 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~~ Results instance]
    - DEBUG_VALUE=0 [No output should be printed from the application, except the line "The sume of all the prime numbers is: XYZ"]
  - The Logger class should have a static method to write~~Ouput~~Message(...).
  - Place the FileProcessor.java in the util/ folder.

Code Organization

- Your directory structure should be similar to assignment-2.

Submission

- Same as before
- Both the team members should submit.

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.
- 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.
- Include javadoc style documentation for at least 3 classes. It is acceptable for this assignment to just have one parameter of a method described for each method's documentation.
- For the classes provided in the code template, add interfaces as appropriate

Design Requirements

Late Submissions

- The policy for late submissions is that you will lose 10% of the grade for each day that your submission is delayed.

Grading Guidelines

Grading guidelines have been posted here.

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