CSCD 467/567 Assignment 3

Search Patterns Using I/O Prefetching

Rules: Your code must use Java Language. If your program shows a compilation error, you get a zero credit for this lab assignment. To avoid compatibility issues, I encourage you to upgrade your JRE to latest version of SE 1.7 or 1.8.

Submission: Wrap up all your java files and a ReadMe text file into a single zip file. Name your zip file as FirstInitialYourLastNameCSCD467hw3.zip. For example, if your legal name is Will Smith, you should name your zip file as wSmithCSCD467hw3.zip. Please do NOT include the provided text file in your submission.

Please submit your single zip file on EWU Canvas by following CSCD467-01 Course → Assignments → Assignment3 → Submit Assignment to upload your single zip file.

You are required to submit the ReadMe text file along with all your java code. In the ReadMe file you should put your legal full name, description about how to compile and how to run your program. If you overlook the ReadMe file or provide wrong information regarding how to compile and run your program on **command line tool**, 5% of penalty will be deducted from your total grade for this assignment. An example of ReadMe file should look like the following:

(This only serves as an Example. Your ReadMe file should contain the similar content.)
Name: Will Smith
Description: unzip the submitted wSmithCSCD467hw3.zip, you get a folder named smithhw3.
To Compile: cd into folder smithhw3,
javac *.java
To Run:
java yourMainClass 4 8

Problem Description

You are required to write a program that can return the number of pattern found in a given text file. Please do **NOT** make any changes to the provided text file in which your program will search for patterns.

You have to use the Java Pattern and Matcher class to search the pattern in

the given file. You are required to apply the Producer-Consumer paradigm to this problem. Your program has to implement the following features.

1, In addition to the main thread, you are required to use another two threads to solve this problem.
2, A **Reader** thread is responsible for reading **one line** each time from the text file into a shared queue structure **Q,** till the whole file has been read through (end of file has been met). The size of **Q** is limited to a *MaxSize*, by default 100 lines of strings.
3, A **Searcher** thread fetches one line from **Q,** then searches the pattern in that line and accumulates the total number of occurrence for the pattern it finds so far.
4, When Q reaches its limits of *MaxSize* lines, **Reader** thread has to wait. It wakes up until the Q has at least one spot available. If no line of string is available in Q, **Searcher** thread has to wait, until Reader reads in at least one line of text.
5, You have to implement your **own** thread-safe queue class (not java build-in safe queue). But you are allowed to create your queue class by wrapping up any existing classes that implements the Queue interface. In that case, you have to implement your own thread-safe enqueue and dequeue methods.
6, Reader and Searcher threads have to work in parallel, you are required to use the **Q** structure as the monitor to synchronize the two threads.
7, The Reader thread first knows when the programs should terminate, i.e. an end of file has been reached. The Reader thread has to tell the Searcher thread to terminate at an appropriate time.

ExtraCredits
Since the Reader only enqueue at the end of the queue structure and Searcher dequeue at the front of the queue structure, it is not very efficient for them to lock the whole queue structure when accessing the queue. If your program only lock part of the queue(e.g. the front of the queue) when accessing, this allows both the Reader and the Searcher to access the queue at the same time while maintains data safety. If you can implement this feature, and obtains correct output results, you will get **20** points of extra credits. Please clearly state that you did this part in your readme file and describe how you accomplish this feature.

What is provided?
1, A zip file hw3.zip includes a **SerialSearchPattern.java** file and this description file. The **SerialSearchPattern.java** provides the sequential code to solve this problem.
2, A large text file is provided, which is a subset of Wikipedia articles and contains around 10 million words. You can find the description here, http://www.evanjones.ca/software/wikipedia2text.html

Test cases

```
Total number of lines searched is 299355.
Total occurence of pattern '(John) (.+?) ' is 3230.
Total time cost for serial solution is 4750ms.
```