CSI 3131 – Summer 2014 Professor: Martin Scaiano
Date: Wednesday, July 9, 2014 Due: July 21st, 2014

Assignment 4 (*Apply operating system concepts to software development*)

In this assignment you are given a very simple webserver that returns pages/articles as clients request them. Clients can also create or update pages. The webserver provided is single threaded and you are required to run it in less than 2 MB memory. You must improve the performance of the webserver, while observing all the restrictions.

Restrictions:

- You may not change Database class
- You must use the database class to read and write files from disk. You may not implement your own file access.
- You may not modify the client or script code, nor can you modify the script generator.
- the server must run in under 2 MB of memory
- Two clients may not write to same article at the same time
- clients may not read an article from the database while it is being written

Grading

	Marks
Readability Code	10
Multi-threading (correct and functions)	10
Correct synchronization	20
Implementing a cache	10
Performance	10
Discussion Questions (break down below)	24 (8 per question)
Explaining your solution	2 per question
Providing an alternate approach	2 per question
Explaining the trade offs between approaches	2 per question
Justifying your values (cache size, thread count)	2 per question
Total	**/84**
Read Errors	-5 (1-5 errors) -10(more)
Write Errors	-5 (1-5 errors) -10(more)
Requiring 1500 KB	-5
Requiring 2 MB	-10
Failing Read Response (depends on the number)	-5 to -20
Performance Better than baseline	+5 (Bonus)
Other Cleverness	+5 (Bonus)

Running the assignment

Testing the assignment requires running two programs: the webserver and the client. The webserver is the program you, the student, are responsible for. The client runs a script simulating clients connecting to the server and making requests.

To run the server use the following command line

```
java -server -Xmx2m Server  4444 db
```

The “-server” option tells java to run the program optimized as a server
The “-Xmx2m” option tells java to limit the virtual machine memory to 2 Megabytes
“Server” is the class to start the server
4444 is the port number that the server operates on.
“db” is the directory where server files will be stored.

The server automatically stops after 1 minute of inactivity. It then reports on any read/write errors. ***Your solution should not have any of these errors.***

```
ERRORS: Reading during write 0
ERRORS: Writing during write 0
```

To run the client use the following command line:

```
java -server Script 4444 commands.txt
```

Notice we do not limit the memory of the client. This uses the default limit, which is usually between 200MB – 1GB
“4444” is the port to connect to the server on. This must match the port that the server is using.
“commands” a file containing the script to execute.

When the client finishes running it will report on the average response time (read and write times). Failed read response refers to request for a file that did not return a textual response. ***Failed read responses are errors and marks can be lost for them***. This can happen if you forget to flush the socket before closing it.

```
Average read response: 86.537413
Average write response: 86.269725
Failed read response: 0.000000
```

The Webserver

There are three files in the web server: Server, Database, HandleRequest. You can only modify Server and HandleRequest; you may NOT modify Database.

It is recommended that you use a cache of results from the Database. Since the database is designed and implemented to be quite slow, a cache can provide significant performance gains. Remember to keep your cache within the memory restrictions.

The implementation of the webserver is single threaded. It is recommended that you make the program multi-threaded to maximize performance.

For comparison, the performance on the dual core laptop that I used during testing using the original code (single threaded, non-cached) is:

```
Average read response: 86.537413
Average write response: 86.269725
Failed read response: 0.000000
```

Baseline (less that 20 minutes of work)- Using a simple cache and multi-threading the performance is:

```
Average read response: 12.487380
Average write response: 19.528440
Failed read response: 0.000000
```

You can run the baseline using the following command line. ***You are not allowed to reverse engineer, decompile, or examine the baseline.***

```
java -Xmx2m -server -jar baseline.jar 4444 db
```

Upper bound performance (ignoring some restrictions) was:

```
Average read response: 5 ms
Average write response: 5 ms
```

For full marks on performance you must achieve performance close to or better than the provided baseline. I have provide the baseline system because performance may differ from machine to machine. All assignments will be tested on the same machine along with the baseline systems.

Generating more scripts

You may generate more test scripts using:

```
java GenerateScript commands.txt 1000
```

"GenerateScript" is the class to run to generate a script. It can be change to produce different distributions.
"commands.txt" is the file where the script is written.
"1000" is the amount of time the script should run for.

Evaluating your performance

You should evaluate your systems performance using the provided commands.txt, as this will be the one used to grade the assignment. You have been provided the original code and jar file containing the baseline code. Since performance will differ from machine to machine you can test your performance relative to this baseline.

Questions

1. Discuss the how you implemented multithreading and why you choose that technique?

   What other techniques could you have used?

   Did you compare results to see what configuration gave the best performance?

2. Discuss your cache implementation?

   What technique did you use?

   What other techniques did you try?

   What was the performance improvement?

   How difficult was it to limit the amount of memory used?

3. Describe how you implemented reader/writer synchronization?

   How else could you have done it?

   Did synchronization affect performance?

   If so how and why is this acceptable?