**FINAL STUDY GUIDE**

The final is cumulative.

The final is closed book. Do not log into your computers. No calculators, iPods, iPads or notes.

The final will be a mix of questions below and questions that you have not seen before.

Dec. 13 - Final exam - In classroom - 9:00-10:30

-------------

Review the midterm study guide:

<http://afodor.github.io/classes/prog2017/StudyGuideMidterm.docx>

I am happy to give feedback on any or all optional exercises in person or over email…

Optional Exercise 1:

Using FastaSequenceOneAtATime, use BlockingQueue to read a large Fasta file and output a file with the GC content of each sequence. Does the performance improve with multiple threads on your machine?

Change the program so that instead of using a BlockingQueue, it uses a Semaphore

(You can substitute any other task to run in parallel if you don't want to parse sequences and calculate GC content).

Resources:

FastaSequenceOneAtATime: https://github.com/afodor/metagenomicsTools/blob/master/src/parsers/FastaSequenceOneAtATime.java

A Fasta file with a bunch of sequences in it:

http://fodorlab.uncc.edu/sites/fodorlab.uncc.edu/files/media/Hamp\_Fodor\_090810.tar.gz

Optional Exercise 2: Make a simple javascipt GUI in Chrome; or, alternatively (or in addition), make a simple javascript server in Node.js

Optional Exercise 3:

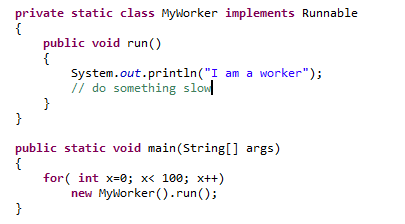
Find a binary dataset (for example there are a few described here) :

http://machinelearningmastery.com/standard-machine-learning-datasets-used-practice-weka/

and use WEKA to generate an ROC curve (using any classifier) via 10-fold cross-validation.

New questions:

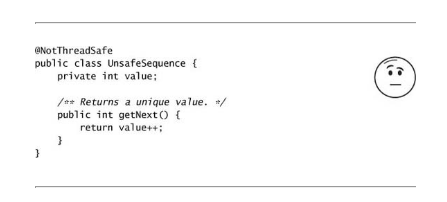
1. How many threads are run in parallel in the following code:

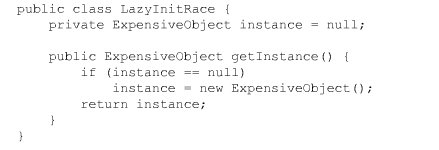
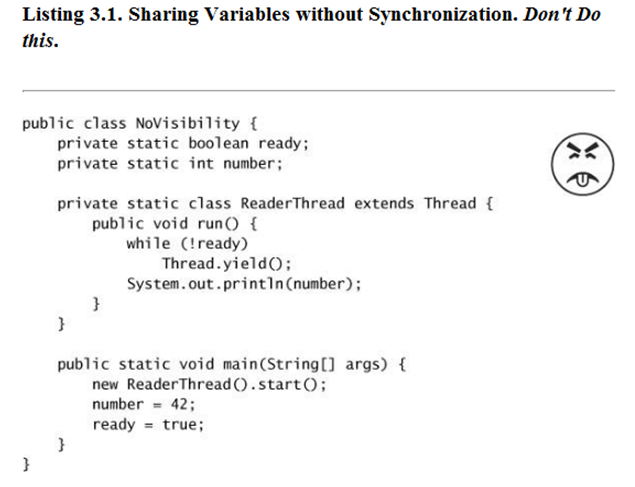
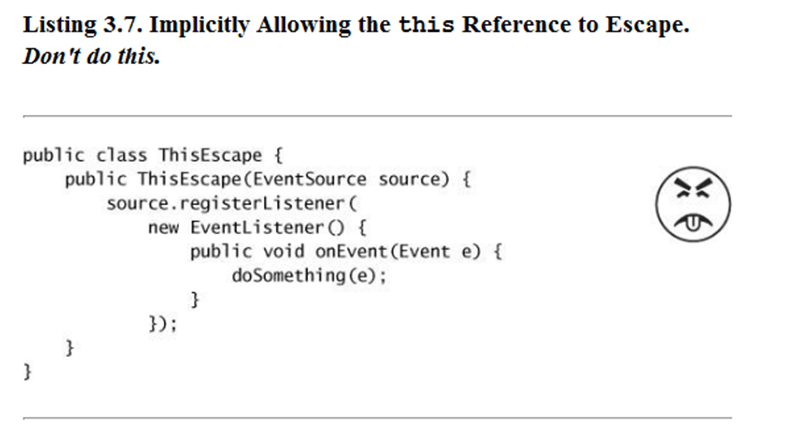
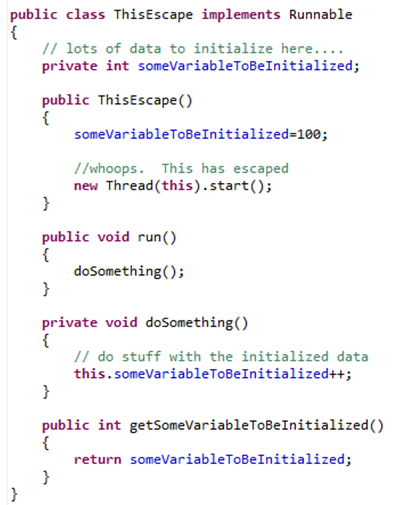
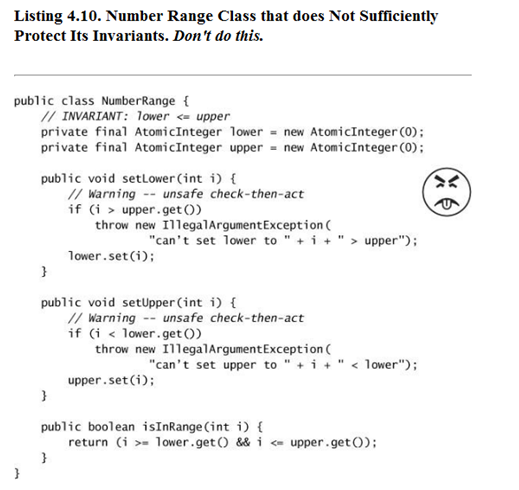


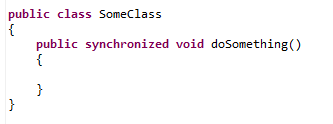
How would you modify the code to make it run more threads in parallel?

------------------------------------------------------------------------------------------------------------------------

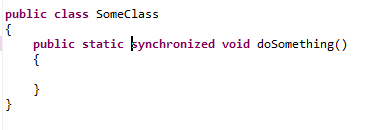
1. Why is the code below not thread safe? Modify the code so that it is thread safe



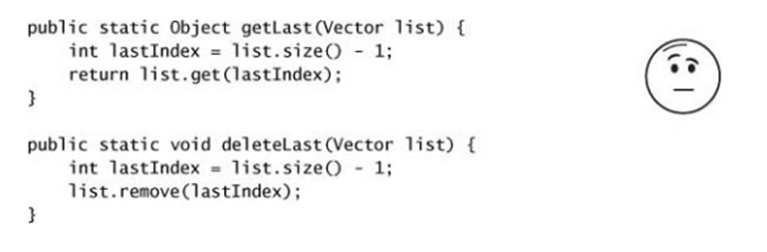
1. Is the following code thread safe? Why or why not? If it is not thread safe, modify it is so that is thread safe.
2. Why is the following code not thread safe? Write a version that is thread safe. 
3. What is the AWT thread. Why are you not supposed to put slow operations on the AWT thread. If you are not on the AWT thread, how do you place a non-thread safe GUI operation on the AWT thread? What is the difference between invokeLater() and invokeAndWait()?
4. Define the term race condition. What does the phrase “race conditions remove determinism” mean?
5. True or false: Stateless objects are always thread safe. Why or why not?
6. True or false: Immutable objects are always thread safe. Why or why not?
7. True or false: Mutable objects are never thread safe. Why or why not?
8. What is the difference between guarding mutable state with “synchronized” and having the keyword “volatile” modify mutable variables? What guarantees are made with synchronized vs. “volatile”?
9. What is the output of this program?
10. Why is this code not thread safe?
11. Why is this code not thread safe?
12. Why is this code not thread safe? Fix the code so that it is thread-safe
13. Why is this code not thread safe? Fix the code so that it is threadsafe.
14. What object is locked in this code?



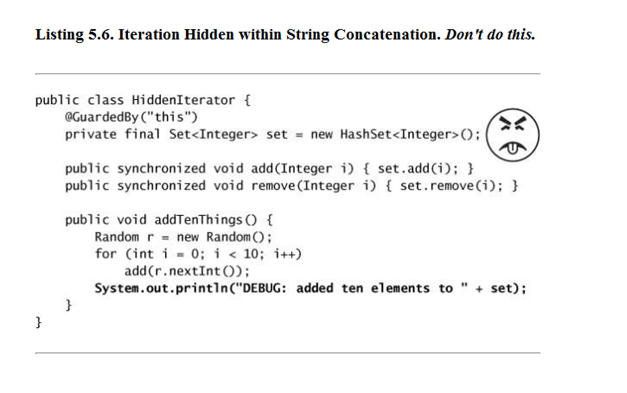
1. What object is locked in this code?



1. Is this code thread safe? What is the potential problem with this code?



1. Why is this code not thread safe? Modify the code so that it is thread safe.



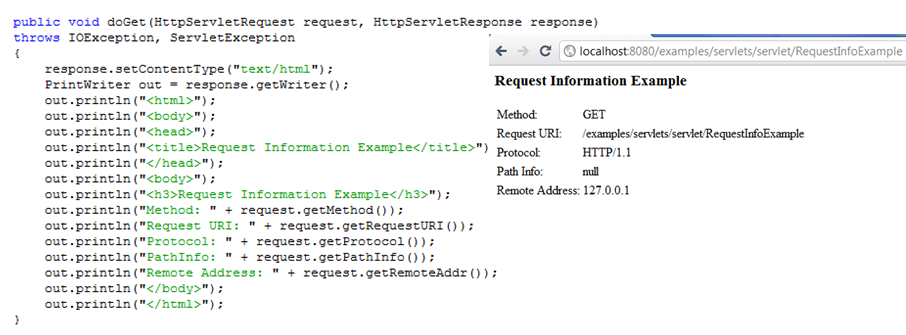
(20) In your own words, what is the difference between a HashMap created with Collections.syncrhonizedMap() and a ConcurrentHashMap. Why is .size() on a concurrent hash map only approximate? How does the ConcurrentHashMap produce better performance under load from multiple threads then the synchronized map? What is lock striping?

(21) In your own words, what is the difference between a List created with Collections.syncrhonizedList() and a CopyOnWriteArrayList(). Why is .size() on a CopyOnWriteArrayList only approximate? Why can the CopyOnWriteArrayList() be slow?

(22) What is the semaphore class? How does it work?

(23) In lecture, we demonstrated a multi-threaded FastaSequence parser and ran it on an 8 core machine, but we only saw a ~5.5X fold speedup. What are some possible reasons we didn't see an 8 fold speedup? What is Amdahl's law?

(24) Is this Servlet threadsafe:



(25) Why should you avoid starting threads from constructors?

(26) Define the monitor pattern. Does the ConcurrentHashMap use the monitor pattern?

(27) Define the decorator pattern.

(28) What is deadlock? Give an example of code that can deadlock? What are some solutions that will allow you to avoid deadlock?

(29) What is the difference between acquiring a lock with ReentrantLock and synchronized?

(30) What is a strategy for writing test code to determine if a method blocks (for example in trying to remove an element from a thread-safe stack when the stack is empty).

(31) What is an ROC curve. Draw an ROC curve for a perfect classifier and a classifier that does no better than random.

(32) What is the run time of the Needleman-Wunsch global alignment algorithm? What is the run time of the Smith-Waterman local alignment algorithm? What is the difference between the two algorithms?

(33) With a match score of 1, a mismatch of -3 and a gap penalty of -1, draw the Needlman-Wunsch grid for DNA sequences “ACG” and “AG”. What is the score of the final alignment?

(34) As an alternative to the Needlman-Wunsch algorithm, why not just evaluate every possible alignment between two sequences of length ~100 basepairs?

(35) Does a larger word size lead to a faster or slower search using BLAST?

(36) What is the BWT transform of the string CCAG ?

(37) Given the BWT transform gc$aaac find the original DNA sequence.

(38) What are closures in javascript? How do closures in javascript differ from closures in Java.

(39) What is the difference between prototype-based inheritance (such as in javascript) and classical object-oriented inheritance (such as in Java).

(40) Javascript is single-threaded but a server built in node.js can still handle multiple independent requests from different clients. How? What is the difference between javascript’s single-threaded event loop and java’s multi-threaded server model?