

## 433 Final Exam, May 19th

Don't panic.

You may also avail yourself of the *punt* rule. Write down *punt* for a question, and your grade is 1/5 of the points for the question.

There are two questions; check the back size of the paper.

### 1. (20 points) Bakery algorithm for first-in, first-out queuing

In many bakeries, when you enter the bakery to go to a machine gives out slips of paper with numbers on them. Behind the counter, there is a display of the last number being served. When one of the people behind the counter is ready for the next transaction, they increment the number being served, and then help the person with that number. In this way, people are served in the order in which they took pieces of paper.

This question requires you to reimplement the `OnePlaceBuffer` interface we used in our first day quiz. However, if multiple threads block trying to call `take()`, they should return in FIFO (first-in, first-out) order. Calls to `put(Object o)` do *not* have to complete in FIFO order.

We are going to be ignoring interrupts, so `InterruptedExceptions` have been removed from the interface. You are encouraged to use the Bakery algorithm to implement this, although any other valid implementation is OK.

Your design should be such that no CPU power is used when no activity is taking place, and methods return as quickly as possible. In other words, you should not depend upon spin waits, yielding or sleeping.

```
package cmsc433;
public interface OnePlaceBuffer {
    public Object take();
    public void put(Object o);
}
```

### 2. (10 points) Short answer:

- There are two basic approaches/APIs for dealing with XML documents. In class, we discussed these as the SAX and the DOM approaches. Briefly describe the differences and relative advantages/disadvantages of these approaches.
- Why might you wish to restrict the machines and ports that untrusted code can connect to?
- You've calculated/measured that the data structures in use by your program require  $100 + 10n$  Kbytes at any one time to handle  $n$  warehouses. Your program will be doing a reasonable amount of memory allocation (with some of the previously allocated memory becoming unreachable and therefore garbage), so you need to worry about the cost of garbage collection.

For 1000 warehouses, which number is best estimate for the amount of heap memory your program would need to run in so that garbage collection would be efficient? Explain your choice:

- 10 Megabytes
- 15 Megabytes
- 25 Megabytes
- 50 Megabytes

### 3. (10 points) Mix and match

For each item in the left column, match it with the problem or domain in the right column for which it would most likely be useful.

Each item should be paired no more than once. *There is one item in each column that should not be paired at all (in other words, there is one bogus entry in each column).*

Technology	Problem
1 Interrupts	a Abstract syntax trees
2 Multicast	b Debuggers and IDEs
3 Observer pattern	c Fault tolerance
4 Reflection	d Finding peers
5 Singleton pattern	e GUIs
6 Visitor pattern	f Saving state
7 XML	g Shutting down

4. (15 points) Distributed computing:

One of the design choices in RMI was to force every method in a remote interface to throw a RemoteException and make RemoteException a declared exception (so that if you call a method that could throw a remote exception, you must either catch the exception or declare that you could throw it).

An alternative would be to make it be a RuntimeException, like a NullPointerException.

Discuss whether the decision made in RMI is a good or bad decision, and why. Feel free to refer to the ideas discussed in the *A Note on Distributed Computing*.

5. (10 points) Factory methods There are two basic approaches to allowing users to create instances:

- public constructors
- public static factory methods and non-public constructors

Give the relative advantages/disadvantages of these two approaches.