

433 Final Exam, Fall 2005

December 20th, 2005

Guidelines

Put your name on each page before starting the exam, and write your answer directly on the exam sheet.

Punting a Question

You may *punt* on any question. To *punt*, write *punt* for your answer, and you will receive 1/5 of the points for that question. If you already have written some material for the question when you decide to punt, be sure to make sure write down your decision to punt in such a way that it is very clear you have decided to punt the question. *The point of the punt rule is to avoid wasting both your time and my time when you realize that you don't think you can answer the question well.*

Blank extra pages

There are blank extra pages at the end of the exam. Feel free to use those if you need more space.

Grade

Question	Points	Grade
1	10	
2	15	
3	10	
4	15	
5	25	
6	25	

Questions

1. In his talk "Growing a Language", what did Guy Steele say about adding complex numbers to a programming language such as Java?

2. Say you have an interface A, and classes AImpl and B, where AImpl implements A and B uses A (e.g., B invokes methods on objects implementing the A interface) but not vice-versa (AImpl objects never invoke methods on B objects).

We want to change the system to allow A to be implemented both by AImpl and by remote proxies of AImpl (these might be hand written, as in our project 4, or automatically produced using something like Java RMI).

At a high level, describe the coding and design changes we might have to make to B in order to accommodate the fact that an A reference might now point to either a AImpl or a remote proxy for an AImpl. If you wish, you can also discuss any changes that should be made to the A interface.

3. What is the most fundamental difference between the SAX and DOM APIs? Under what circumstance would using SAX generally be a much better solution than DOM? Under what circumstance would DOM generally be a much better solution SAX?

4. `ConcurrentHashMap<K,V>` introduces a new function

```
public V putIfAbsent(K key,V value)
```

Provide a static method for a utility class

```
public static <K,V> V putIfAbsent(Map<K,V> map, K key, V value)
```

that tries to provide the same functionality as best as possible for any `Map`. Don't use any casts. If it does not provide exactly the same functionality as the `putIfAbsent` defined in `ConcurrentHashMap`, explain why.

5. Figure 1 gives the skeleton for a class `LockSet` that allows you to create tokens and get locks on multiple tokens at once. Provide an implementation of the 4 incomplete methods in `LockSet` class. For example, assuming `a` and `b` are both tokens made from the `LockSet s`, the call `s.lock(a,b)` will wait until no locks are held on `a` or `b`, and obtain locks on both of them.

Your class should be immune from deadlock (assuming the threads do not make nested lock calls). Operations on two distinct `LockSet`s shouldn't block each other, but it is OK if two operations on one `LockSet` momentarily block one another, even if they use disjoint tokens. Do not use spin waiting. You don't have to worry about reentrant behavior (obtaining a lock on a token you already have a lock on) or detecting errors (e.g., releasing a lock on a token that isn't locked, or passing a `Token` manufactured by one `LockSet` to a different `LockSet` instance).

The `LockSet` methods (`lock`, `tryLock`, ...) work as they do in the `Lock` interface. Note that `LockSet` uses Java varargs. For example, given the code below, the `lock` method will be passed an array of two `Token` objects as parameters.

```
LockSet s = new LockSet();
LockSet.Token a = s.makeToken();
LockSet.Token b = s.makeToken();
s.lock(a,b);
// do something
s.unlock(b);
s.unlock(a);
```

When considering deadlock, please consider what happens if one thread invokes `s.lock(a,b)` and another thread invokes `s.lock(b,a,c)`.

Feel free to use either 1.4 style synchronization (`synchronized`/`wait`/`notify`) or 1.5 style synchronization (e.g., using `ReentrantLock`). A refresher for the 1.5 synchronization APIs is provided in Figure 2.

```
public final class LockSet {
    public final class Token {
        Token() { }
        // define any methods you need here
    }
    public Token makeToken() {
        return new Token();
    }

    public boolean tryLock(Token... locks) {
        // implement this
    }
    public void lock(Token... locks) {
        // implement this
    }
    public void lockInterruptibly(Token... locks) throws InterruptedException {
        // implement this
    }
    public void unlock(Token... locks) {
        // implement this
    }
}
```

Figure 1: `LockSet` class

LockSet answer

If you want to use 1.5 style synchronization, here is a refresher on the Lock and Condition API's. You can use `new ReentrantLock()` to create a `ReentrantLock` that implements the Lock interface.

```
interface Lock {
    void lock();
    void lockInterruptibly() throws InterruptedException;
    boolean tryLock();
    boolean tryLock(long time, TimeUnit unit) throws InterruptedException;
    void unlock();
    Condition newCondition();
}

interface Condition {
    void await();
    boolean await(long time, TimeUnit unit);
    long awaitNanos(long nanosTimeout);
    void awaitUninterruptibly();
    boolean awaitUntil(Date deadline);
    void signal();
    void signalAll();
}
```

Figure 2: 1.5 synchronization APIs

6. XML: Here is an XML DTD named simple.dtd:

```
<!ELEMENT A (B, C*) >  
<!ELEMENT B ((D, B) | C+) >  
<!ELEMENT C (#PCDATA)>  
<!ELEMENT D (#PCDATA)>
```

- (a) Give a well-formed XML document that validates against simple.dtd that does not contain a D element.
- (b) Give a well-formed XML document that validates against simple.dtd that contains a D element.
- (c) Give an XPath query string that will match all C elements.
- (d) Give an XPath query string that will match all C elements contained inside a B element.
- (e) Give an XPath query string that will match all C elements not contained inside a B element.

