Lecture 14
Midterm Review
Topics Covered

• Parallel / concurrent / distributed systems
• Nondeterminism
• Procedural abstraction
• Processes and threads
• Scheduling
• Context switching
• Testing multi-threaded programs
• Types of testing: functional / performance / stress / unit / integration / acceptance
• Interleavings and how to count them
• Forcing interleavings via `Thread.yield()` and `Thread.sleep()`
• Threads as objects in Java
• `Thread` class
• `Runnable` interface
• Thread states
• User vs. daemon threads
• Data races
• Race conditions
• Class specifications, correctness
• Thread safety
• Atomicity

• Locks
• Intrinsic / monitor locks
• Synchronized blocks, methods
• Reentrant locks
• Locks and performance
• Locking protocols
• Deadlock
• Waits-for graphs
• Deadlock prevention
• Built-in atomic memory access in Java
• Synchronization and visibility
• Volatile variables / fields
• Locking and visibility in Java
• Object publishing and escape
• Indirect publishing
• Improper object construction and escape of `this`
• Safe object construction via factory methods
• Thread confinement
• Stack confinement
• `ThreadLocal`
Topic Covered (cont.)

- Immutable objects and final fields
- Initialization safety
- Safe publication
- Effectively immutable objects
- State-dependent actions
- Balking / guarded suspension / optimistic retry
- wait() / notify() / notifyAll()
- notify() and deadlock
- Timed waiting
- Nested monitor lockout
- Collections.synchronizedXXX() (XXX is the name of a type of collection)
- Thread safety, compound actions and client-side locking
- ConcurrentModificationException and hidden iteration
- Concurrent collections
- ConcurrentHashMap and lock striping
- Fail-fast vs. weakly consistent iterators
- CopyOnWriteArrayList
- Blocking queues: bounded, unbounded, synchronous
- The Producer-Consumer pattern
- InterruptedException
- Synchronizers
- Explicit (reentrant) locks
- Conditions, await() / signal() / signalAll()
- Futures / FutureTasks