Concurrency

* Concurrent program: A program where multiple threads are executing at once
* Shared member vs message passing
* Process vs thread
  + Process: Program execution and will have its own resources
  + Thread: A part of a process, every process has at least one thread.
* Java.lang.Thread vs Runnable interface, run()
* Synchronization: to restrict the interleavings
  + Ensures safety: avoid race conditions
  + Coordinate actions of thread
* Race conditions
  + When differences in ordering lead to incorrect results, usually handled by mutual exclusion/serialization
* Mutual exclusion: Prevent more than one thread from accessing critical section at a given time, serialize.
* Atomicity
* Locks
* Block Synchronization: synchronized(object)
  + Takes as a parameter an object whose lock the system needs to obtain before it can continue
* Synchronized methods
  + Calling a synchronized method will lock the object until the method is finished.
  + Lets say we have public void synchronized m(){ } as a synchronized method in class A. When this method is called, then the instance of the object which called this method is locked.
* Deadlocks
  + Concurrent modules are stuck waiting from each other to do something
* Livelocks
* Starvation
* Lack of fairness
* Some locks never get a chance to run. This happens if the ordering of the locks stays the same and you have a lot of threads going in and out at the same time.

Processes and sockets

* Queue
* Happens before relationship
* Synchronized queue
* BlockQueue: interface java.util.concurrent.BlockingQueue
* Blocking
* Immutable object for thread safety
* Client-server programming
* Sockets

Event-Based Programming (GUI)

* Composite pattern
* View hierarchy
* Publish-Subscribe pattern
* Event/Listener
* Model-View-Controller
  + Model: Data, View: Output, Controller: Input
* Decoupling

Performance Engineering

* Performance measures
  + Response time, expandability, scability, power/resource usage
* Many room of improvement
  + Algorithm selection, archictertural details, parallelism, etc.

Equality

* Referential equality
* Behavioral equality
* Observational equality
* == vs .equals()
* Equivalence relations: symmetry, transitive, reflexivity
* Object class
* Overriding vs overloading
* Equals() and hashcode contract
* Inheritance
* Composition

Debugging Procedures/Strategies

Example Questions: