Name three thread design patterns

1. Thread pool (Boss/Workers)  
2. Peer (Work crew)  
3. Pipeline

Explain how a thread pool works

One thread dispatches other threads to do useful work which are usually part of a worker thread pool. This thread pool is usually pre-allocated before the boss (or master) begins dispatching threads to work. Although threads are lightweight, they still incur overhead when they are created.  
  
If the boss thread becomes a worker thread once it's done starting other worker threads then this is a Peer Thread Design Pattern.

Define: critical section

The code between lock and unlock calls to a mutex.

What are four mutex types? (also, try to explain each one)

- Recursive: allows a thread holding the lock to acquire the same lock again which may be necessary for recursive algorithms.  
  
- Queuing: allows for fairness in lock acquisition by providing FIFO ordering to the arrival of lock requests. Such mutexes may be slower due to increased overhead and the possibility of having to wake threads next in line that may be sleeping.  
  
- Reader/Writer: allows for multiple readers to acquire the lock simultaneously. If existing readers have the lock, a writer request on the lock will block until all readers have given up the lock. This can lead to writer starvation.  
  
- Scoped: RAII-style semantics regarding lock acquisition and unlocking.

Define: deadlock

Two (or more) threads have stopped execution or are spinning permanently. For example, a simple deadlock situation: thread 1 locks lock A, thread 2 locks lock B, thread 1 wants lock B and thread 2 wants lock A.

How can you prevent deadlocking from occurring?

You can prevent deadlocks from happening by making sure threads acquire locks in an agreed order (i.e. preservation of lock ordering). Deadlock can also happen if threads do not unlock mutexes properly.

Define: race condition

A race condition is when non-deterministic behavior results from threads accessing shared data or resources without following a defined synchronization protocol for serializing such access.

How can you prevent race conditions from occurring?

Take steps within your program to enforce serial access to shared file descriptors and other external resources.

Define: priority inversion

A higher priority thread can wait behind a lower priority thread if the lower priority thread holds a lock for which the higher priority thread is waiting.  
  
As happened on the Mars Pathfinder.

Define: Condition Variables (what is/are the analogous Java structure(s)?)

Condition variables allow threads to synchronize to a value of a shared resource. Typically, condition variables are used as a notification system between threads.  
  
In Java wait() and notify() are used.

Define: (thread) barriers

Barriers are a method to synchronize a set of threads at some point in time by having all participating threads in the barrier wait until all threads have called the said barrier function. This, in essence, blocks all threads participating in the barrier until the slowest participating thread reaches the barrier call.

Define: Semaphores

Semaphores are another type of synchronization primitive that come in two flavors: binary and counting. Binary semaphores act much like simple mutexes, while counting semaphores can behave as recursive mutexes. Counting semaphores can be initialized to any arbitrary value which should depend on how many resources you have available for that particular shared data. Many threads can obtain the lock simultaneously until the limit is reached. This is referred to as lock depth.   
  
Semaphores are more common in multiprocess programming (i.e. it's usually used as a synch primitive between processes).

Define: Spinlocks

Spinlocks are locks which spin on mutexes. Spinning refers to continuously polling until a condition has been met. In the case of spinlocks, if a thread cannot obtain the mutex, it will keep polling the lock until it is free. The advantage of a spinlock is that the thread is kept active and does not enter a sleep-wait for a mutex to become available, thus can perform better in certain cases than typical blocking-sleep-wait style mutexes. Mutexes which are heavily contended are poor candidates for spinlocks.   
  
Spinlocks should be avoided in uniprocessor contexts.

List six synchronization primitives.

1. Mutex  
2. Join  
3. Condition variables  
4. Barriers  
5. Spin lock  
6. Semaphore

Define: livelock

A livelock is similar to a deadlock, except that the states of the processes involved in the livelock constantly change with regard to one another, none progressing.

What does the term 'lock-free' mean?

Multithreaded code written such that the threads can never permanently lock up.

Define: "Busy waiting" and how it can be avoided

When one thread is waiting for another thread using an active looping structure that doesn't do anything. Example:   
Thread task = new TheTask();  
task.start();  
while( task.isAlive() ){  
; // do nothing  
}  
  
This can be avoided using mutexes.