CLASS NOTES:

I. Upon server socket’s accept(), no new port # allocated
   - tcp connection: translation of internal/external ip address
   - see handout

II. Jdk 1.5 concurrent package
    – contains many classes that help manage multiple threads
    – http://java.sun.com/j2se/1.5.0/docs/api/java/util/concurrent/package-summary.html#package_description

III. thread per connection:
    synchronized insert (socket s)

    head

    tail

    socket s

    Remove Synchronized {
        Remove tail;
        Return tail;
    }

    - 2 shared objects, thread count & work queue

IV. pool of threads
    let LockObject be an object to lock on
    let Q be the queue
    worker thread:
    while(true)
    {
        try {
            w = Q.remove();
            process w;
        } catch (EmpyQueueException E) {
            synchronized(LockObject) {
                LockObject.wait();
            }
        }
    }
main thread:
while(true)
{
    c = ss.accept();
    Q.insert( c );
    LockObject.notifyAll();
}

- wait/notify constructs enables threads to communicate with each other without busy-wait; notify or notifyAll
- wait/notify are properties of the object
- usually implement notifyAll because we do not know how many we may need at moment
- notifyAll because of race condition; main thread executes while loop and q starvation if only 1 thread notified and that is the only thread that processes all connections, if main thread continues with q insertions, therefore rendering other threads in q starved

V. use serversocket SS as a queue

main thread:
    enlarge SS's queue
    create pool of n worker threads;

worker thread:
    while(true)
    {
        synchronized(SO)
        {
            //gets server socket, not connected socket
            connectedSocket = ss.accept();
        }
        //handle connected socket
    }

SO is an object for synchronization;

- but, tcp clients wait certain amount for connection; ie. if all threads busy, then if new client connection arrives, connection is ignored

VI. Junit
Flow for test reading input-stream vice versa for writing to output stream
Test input -> pipedOutputStream -> pipedInputStream -> program being tested (see hand-out for code example)

VII. IIAP handout – “building on our threads dicussion from lsat time”
1.a.ii.Pool:
    put new connections on the queue & notify all waiting threads

1.a.iii.
    accept as queue of connections, sub category for pool of threads, simplifies creating own queue