16.1

1. Its drawback is it involves the client in polling the server and the server in carrying out extra requests. It is also unfair, because other clients may make their requests before the waiting clients tries again.
2. The advantage is this alternative is safe, and the drawback is the execution speed is slow.
3. The advantage is servers and clients can be asynchronous and the working efficiency is high through callbacks.

16.2

T	U
X := read(j)	X := read(k)
Y := read(i)	
Write (j, 44)	
Write (i, 33)	
	Write (I, 55)
	Y := read(j)
	Write (k, 66)

The above example is equivalent to T before U.

T	U
X := read(j)	X := read(k)
	Write (I, 55)
	Y := read (j)
Y = read (i)	Write (k, 66)
Write (j, 44)	
Write (I, 33)	

The above example is equivalent to U before V

T	U
	X := read (k)
X := read (j)	Write (I, 55)
Y := read (i)	Y := read (j)

Write (j, 44)	Write (k, 66)
Write (I, 33)	

Concurrency Control and Java Apps
Explain the 3 different concurrency control methods (Ch 16 Coulouris Book – Locks based, Optimistic and Time-stamp Ordering) brifely. Compare and contrast how they achieve concurrency control. [20 points]

Locks based concurrency control means a server attempts to lock any object that is about to be used by any operations of a client's transaction. If a client requests to access to an object that is already locked due to another client's transaction, the request is suspended, and the client must wait until the object is unlocked. There is a common problem when using the locks-based concurrency control, which is called deadlock. The solution to prevent the deadlock is to lock all of the objects used by a transaction when it starts. Another solution to prevent the deadlock is called upgrade. There are two approaches to increase the concurrency. One is called two-version locking, another one is called hierarchic locks.

Optimistic concurrency control has three phases, working phase, validation phase and update phase. During the working phase, each transaction has a tentative version of each of the object. The use of tentative versions allows the transaction to abort with on effect on the objects. During the validation phase, the transaction needs to be validated to check to see whether or not its operations on objects conflict with operations of other transactions on the same objects. During the update phase, if a transaction is validated, all the changes recorded in tentative versions are made permanent.

In timestamp ordering, each transaction is assigned a unique timestamp value when it starts. The timestamp defines its position in the time sequence of transactions. Requests from transactions can be ordered according to their timestamps. Timestamp ordering ensures that each transaction accesses a consistent set of versions of the objects. Timestamp ordering has the write rule and the read rule. Write rule guarantees that a

transaction's timestamp is larger than the maximum read timestamp on D and is larger than the write timestamp on committed version of D. the improved version of the basic timestamp concurrency control is called multi-version timestamp ordering. This allows each transaction to write its own tentative versions of objects.

Java Concurrency In Practice by Brian goetz is the practice manual for Dist Systems. Study Chapter 6 (Task Execution) which is a small chapter.

Identify 5 different programming devices (e. g. Tasks, Exec Framework) used in Java for Task Execution. Then, briefly explain how you might use these to implement a basic Locks based concurrency control in a simple Web App. [40 points]

5 different programming devices are thread, TaskExecutionWebServer, Runnable, Callable, Future.

For a server, there are different kinds of operations that clients can do, such as GET/PUT/DELETE. Each request is independent. I plan to use Executor framework and create a thread pool for each server, and this thread pool has 100 threads. Each server can concurrently handle 100 client requests. When a client request is coming, this request is wrapped by a Callable object and push the object to the thread pool. If there no free threads in the thread pool, then the coming operations have to wait until some threads become available. If there are multiple operations accessing the same object, each operation should use lock to gain the access of the object. When an object is locked, other operations can access the object only after that object is released.