Synchronization: a thread can take a lock only once. No mechanism of a waiting queue and after the exit of one thread, any thread can take the lock.

Reentrant Locks: gives a lock to the current working thread and blocks all other threads which are trying to take a lock on the shared resource.

ReadWriteLock:

1. One for read-only operations; and
2. one for writing.

a single thread at a time (a writer thread) can modify the shared data, in many cases any number of threads can concurrently read the data (hence reader threads).

CompletableFutures

can do something with the result of the operation without actually blocking a thread to wait for the result.

Future:

1. cannot be manually completed
2. don't have the ability to attach a callback function to the Future and have it get called automatically when the Future's result is available
3. Multiple Futures cannot be chained together
4. Not able to run multiple in parallel
5. No exception handing

CompletableFutures:

6. manually complete a Future
7. run some background tasks asynchronously and don't want to return anything from the task
8. run asynchronously and return result
9. handle work after task is completed