Synchronized block: the code inside the synchronized block only can be called by one thread at a time and another threads can't until that thread is finished all codes in the synchronized block.

Synchronized method: when a thread calls a synchronized method, other threads can't call this method from this kind of Object. So, this synchronized method only can be access by one thread at a time.

Synchronized static: Synchronized static means class-level lock rather than object-level lock such as synchronized method. Once a thread calls a synchronized static method, other static methods are all locked for multithreading.

Lock:

- lock(): this is the method to acquire a lock, if the lock is occupied, then the thread will be waited.
- unlock(): the method to release the lock.
- tryLock(): this is the method to acquired a lock, if the lock is occupied it will return false otherwise return true. So, the thread won't be waited.
- newCondition(): To generate a condition object bonded with the lock.
- lockInterruptibly():this is a special method to acquire a lock. If the lock is occupied, this thread could be interrupted rather than waited.
- ReentrantLock is the only class implemented the Lock interface.

ReadWriteLock Interface:

- readLock(): Lock read-operations by other multiple threads. Writing operations are not affected.
- writeLock(): Writing read-operations by other multiple threads. reading operations are not affected.
- ReentrantReadWriteLock: is the only class implemented the ReadWriteLock interface

Future (interface) / CompletableFuture

- Future is a class with the result of an asynchronous computation
- cancel(): cancel the execution of the task.
- get(): waiting for getting the result from the execution of the task.
- isCancelled():checked if the execution is cancelled now.
- isDone():checked if the execution is finished now.
- CompletableFuture can automatically run a given function when the execution is done.