* Since version 5.0, the Java platform has also included high-level concurrency APIs.
* The high-level APIs in the java.util.concurrent packages.
* There are also new concurrent data structures in the Java Collections Framework.
  + - [Lock objects](http://docs.oracle.com/javase/tutorial/essential/concurrency/newlocks.html) support locking idioms that simplify many concurrent applications.
    - [Executors](http://docs.oracle.com/javase/tutorial/essential/concurrency/executors.html) define a high-level API for launching and managing threads. Executor implementations provided by java.util.concurrent provide thread pool management suitable for large-scale applications.
    - [Concurrent collections](http://docs.oracle.com/javase/tutorial/essential/concurrency/collections.html) make it easier to manage large collections of data, and can greatly reduce the need for synchronization.
    - [Atomic variables](http://docs.oracle.com/javase/tutorial/essential/concurrency/atomicvars.html) have features that minimize synchronization and help avoid memory consistency errors.
    - [ThreadLocalRandom](http://docs.oracle.com/javase/tutorial/essential/concurrency/threadlocalrandom.html) (in JDK 7) provides efficient generation of pseudorandom numbers from multiple threads.
* Synchronized code relies on a simple kind of reentrant lock. This kind of lock is easy to use, but has many limitations. More sophisticated locking idioms are supported by the[java.util.concurrent.locks](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/locks/package-summary.html) package