# 多线程

## 阻塞队列

### BlockingQueue

根据jdk1.7，查看源码注释得到下面的记录。

提供的方法如下：

#### boolean add(E e)

在队列中插入一个元素。如果队列**没有容量限制**或者**有可用的空间**，插入成功，并且返回true，否则，抛出异常IllegalStateException。

这个方法是返回一个boolean类型的返回值，但是注释里只说失败了会抛出异常，没说失败了会返回false，应该就是不返回false。

假如队列存在容量限制，我们推荐优先使用offer方法。

#### boolean offer(E e)

在队列中插入一个元素。如果队列**有可用的空间**，插入成功，并且返回true；如果没有可用的空间，返回false。

当队列存在容量限制的时候，我们推荐优先使用该方法。

上面两个插入方法，注释里并没有提及没有可用空间会阻塞线程的问题，所以认为是如果没有可用空间，立即返回了。

#### void put(E e)

在队列中插入一个元素，当没有可用的空间的时候，会阻塞插入线程，直到有可用空间成功插入元素为止。

#### boolean offer(E e, long timeout, TimeUnit unit)

这个和前面的offer一样，只是加了一个等待时间，当没有可用空间的时候，会等待一个时间。成功插入，返回true；如果等待时间结束，队列仍然没有可用空间，返回false。

#### E take()

返回并且一处队列的头元素，如果队列当前为空，会阻塞取元素的线程，直到队列有可用的元素为止。

#### E poll(long timeout, TimeUnit unit)

返回并且一处队列的头元素，如果队列当前为空，会阻塞取元素的线程tiemout的时间。如果成功取到，返回对应的元素；否则，返回null。

#### ArrayBlockingQueue

该类实现了接口BlockingQueue。

类注释翻译整理如下：

使用一个有界的数组缓存数据，是一个先进先出(FIFO)的队列。其它的说明，无非就是插入和取出元素时阻塞之类的。

重点说一下关于等待策略：

当有很多线程来取元素的时候，如果使用公平策略，能严格执行先进先出策略，但是这样会降低系统的吞吐量。如果使用不公平策略，则不能完全保证先进先出策略，比如消费者线程，当有可用的元素的时候，只能保证这些线程一起去争夺这个可用元素，但是不能保证先进先出。

对于添加元素和删除元素都使用**同一个锁**。

## 可重入锁

字面意思就是当一个线程获取到了某个锁以后，该线程可以多次获得该锁。

可重入锁的意义就在于可以防止死锁。