1. 多线程共享消息队列的实现

(wait会释放synchronized时加的锁)

1. 锁的实现

读写锁

1. 线程池的实现：

(通过子线程的轮询消息队列实现(消息队列是同步的))

主函数：Channel channel = **new** Channel(5); // Worker Thread数量

channel.startWorkers();

**new** ClientThread("Alice", channel).start();

**new** ClientThread("Bobby", channel).start();

**new** ClientThread("Chris", channel).start();

1. 多线程模拟异步 (一开始给你的对象不实际处理，戴你要结果的时候才处理)



(返回FutureData，在其中有RealData，真正的处理数据)

1. 二阶段线程终止



GracefulThread提供最基本的线程终止方法：interrupt()方法

小知识整理：

Long 和 double只有使用volatile进行修饰才是原子操作

通过CPU的CAS指令，形成原子操作：

 Integer的实现

还有AtomicIntegerArray LongArray ReferenceArray数组的原子操作

通过反射作用的 AtomicReferenceFiledUpdater Integer Long

例子：

关键代码：

**public** **boolean** compareAndSetParent(TreeNode expect, TreeNode update) {

**return** ***parentUpdater***.compareAndSet(**this**, expect, update);

}

锁机制：

ReentrantLock可以设置线程的公平性：不会出现饥饿问难题

TryLock(true)

创建锁的过程：

Lock lock =new Lock();

Lock.lock();

Finally{

Lock.unlock();

}

重点：尽可能使用可重入锁，这样可以减少锁在线程之间的传递

Object的Wait，notify，notifyall方法与synchronized想对应

Lock与Condition的await，single，singleall方法

通过AbstractQueuedSynchronizer作为有限资源的管理器实现



其中的重点方法：

**protected** **int** tryAcquireShared(**int** acquires) {

**for** (;;) {

**int** available = getState();

**int** remaining = available - acquires;

**if** (remaining < 0 ||

compareAndSetState(available, remaining)) {

**return** remaining;

}

}

}

**protected** **boolean** tryReleaseShared(**int** releases) {

**for** (;;) {

**int** available = getState();

**int** next = available + releases;

**if** (compareAndSetState(available, next)) {

**return** **true**;

}

}

}