**实现多线程的两种方法：Thread与Runable。**

Thread:

1，定义类继承Thread。

2，复写Thread类中的run方法

3，调用线程的start方法

Runnable

1.类A实现Runnable接口

2.覆盖run方法。

3.通过Thread类建立线程对象。

4.将类A对象作为实际参数传递给Thread类的构造函数。

不同：

Thread继承自runnable

使用runnable好处：

避免点继承的局限，一个类可以继承多个接口。

适合于资源的共享（同一个对象可作为多个thread的构造参数）

**ThreadPool用法与优势。**

ThreadPoolExecutor(corePoolSize,maximumPoolSize,keepAliveTime,milliseconds,runnableTaskQueue, handler);

corePoolSize（线程池的基本大小）

runnableTaskQueue（任务队列）：用于保存等待执行的任务的阻塞队列。

maximumPoolSize（线程池最大大小）：线程池允许创建的最大线程数。RejectedExecutionHandler（饱和策略）：当队列和线程池都满了，说明线程池处于饱和状态



优点：减少创建线程消耗的系统资源

Java通过Executors提供四种线程池，分别为：  
newCachedThreadPool创建一个可缓存线程池，如果线程池长度超过处理需要，可灵活回收空闲线程，若无可回收，则新建线程。  
newFixedThreadPool 创建一个定长线程池，可控制线程最大并发数，超出的线程会在队列中等待。  
newScheduledThreadPool 创建一个定长线程池，支持定时及周期性任务执行。  
newSingleThreadExecutor 创建一个单线程化的线程池，它只会用唯一的工作线程来执行任务，保证所有任务按照指定顺序(FIFO, LIFO, 优先级)执行。

四种与ThreadPoolExecutor都是executorservice实现类

**wait()和sleep()的区别**

Object->wait thread->sleep

sleep()方法是Thread类的方法，因此它不能改变对象的机锁。

当在一个Synchronized方法中调用sleep（）时，线程虽然休眠了，但是对象的机锁没有被释放，其他线程仍然无法访问这个对象。

wait()方法则会在线程休眠的同时释放掉机锁，其他线程可以访问该对象(的**其他**synchronized方法)。

Synchronized、ReentrantLock、lock和读写锁介绍与原理

锁的等级：方法锁、对象锁、类锁。

ThreadLocal的设计理念与作用

Concurrent包里的其他东西：ArrayBlockingQueue、CountDownLatch等等。

Wait

Join

线程同步机制（voliate,synchronized,重入锁，threadlocal），线程间通信（wait,notify）