JAVA线程

**总结：**

**实现Runnable接口比继承Thread类所具有的优势：**

**1）：适合多个相同的程序代码的线程去处理同一个资源**

**2）：可以避免java中的单继承的限制**

**3）：增加程序的健壮性，代码可以被多个线程共享，代码和数据独立**

**提醒一下大家：main方法其实也是一个线程。在java中所有的线程都是同时启动的，至于什么时候，哪个先执行，完全看谁先得到CPU的资源。**

**在java中，每次程序运行至少启动2个线程。一个是main线程，一个是垃圾收集线程。因为每当使用java命令执行一个类的时候，实际上都会启动一个ＪＶＭ，每一个ｊＶＭ实习在就是在操作系统中启动了一个进程。**

**Obj.wait()，与Obj.notify()必须要与synchronized(Obj)一起使用**，也就是wait,与notify是针对已经获取了Obj锁进行操作，从语法角度来说就是Obj.wait(),Obj.notify必须在synchronized(Obj){...}语句块内。从功能上来说wait就是说线程在获取对象锁后，主动释放对象锁，同时本线程休眠。直到有其它线程调用对象的notify()唤醒该线程，才能继续获取对象锁，并继续执行。相应的notify()就是对对象锁的唤醒操作。但有一点需要注意的是notify()调用后，并不是马上就释放对象锁的，而是在相应的synchronized(){}语句块执行结束，自动释放锁后，JVM会在wait()对象锁的线程中随机选取一线程，赋予其对象锁，唤醒线程，继续执行。这样就提供了在线程间同步、唤醒的操作。Thread.sleep()与Object.wait()二者都可以暂停当前线程，释放CPU控制权，主要的区别在于**Object.wait()在释放CPU同时，释放了对象锁的控制**。

1. <http://baijiahao.baidu.com/s?id=1602265641578157555&wfr=spider&for=pc>

线程池：

1. <https://blog.csdn.net/u013541140/article/details/95225769>

*/\*  
1. setDaemon需要在start方法调用之前使用  
2. 线程划分为用户线程和后台(daemon)进程，setDaemon将线程设置为后台进程  
3. 如果jvm中都是后台进程，当前jvm将exit。（随之而来的，所有的一切烟消云散，包括后台线程啦）  
4. 主线程结束后，  
1） 用户线程将会继续运行  
2） 如果没有用户线程，都是后台进程的话，那么jvm结束  
另外：  
setDaemon方法把java的线程设置为守护线程，此方法的调用必须在线程启动之前执行。只有在当前jvm中所有的线程都为守护线程时，jvm才会退出。  
如果创建的线程没有显示调用此方法，这默认为用户线程。  
 \*/*