**进程：**

**进程是程序在某个数据集合上的一次运行活动，也是操作系统进行资源分配和调度的基本单位**

* **一个进程包括由操作系统分配的内存空间，包含一个或多个线程**
* **一个线程不能独立的存在，它必须是进程的一部分**
* **一个进程一直运行，直到所有的非守护线程都结束运行后才能结束**

　进程：每个进程都有独立的代码和数据空间（进程上下文），进程间的切换会有较大的开销（进程是资源分配的最小单位）

　线程：同一类线程共享代码和数据空间，每个线程有独立的栈（保存其运行状态和局部变量）和程序计数器(记录某个线程的执行位置)，线程切换开销小

**创建一个线程**

**三种创建线程的方法：**

* **通过实现 Runnable 接口；**
* **通过继承 Thread 类本身；**
* **通过 Callable 和 Future 创建线程**

**实现Runnable接口比继承Thread类所具有的优势：**

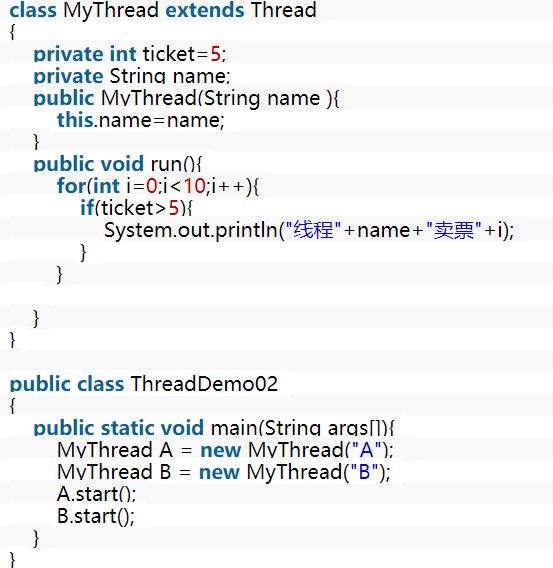
**1）：适合多个相同的程序代码的线程去处理同一个任务**

**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**

Runnable可以实现资源共享但Thread不能实现资源共享的原因

因为一个线程只能启动一次，通过Thread实现线程时，线程和线程所要执行的任务是捆绑在一起的。也就使得一个任务只能启动一个线程，不同的线程执行的任务是不相同的，所以没有必要，也不能让两个线程共享彼此任务中的资源



一个任务可以启动多个线程，通过Runnable方式实现的线程，实际是开辟一个线程，将任务传递进去，由此线程执行。可以实例化多个 Thread对象，将同一任务传递进去，也就是一个任务可以启动多个线程来执行它。这些线程执行的是同一个任务，所以他们的资源是共享

## **具体方法：**

**线程同步以及线程调度相关的方法**

**- wait()：使一个线程处于等待（阻塞）状态，并且释放所持有的对象的锁；**

**- sleep()：使一个正在运行的线程处于睡眠状态，是一个静态方法，此方法抛出InterruptedException异常；**

public static void sleep(long millis) throws InterruptedException

**- notify()：唤醒一个处于等待状态的线程，当然在调用此方法的时候，并不能确切的唤醒某一个等待状态的线程，而是由JVM确定唤醒哪个线程，而且与优先级无关；**

**- notityAll()：唤醒所有处于等待状态的线程，该方法并不是将对象的锁给所有线程，而是让它们竞争，只有获得锁的线程才能进入就绪状态**

**Thread类的sleep()方法和对象的wait()方法都可以让线程暂停执行，它们有什么区别?**

* sleep()方法是线程类（Thread）的静态方法，调用此方法会让当前线程暂停执行指定的时间，将执行机会（CPU）让给其他线程，但是对象的锁依然保持，因此休眠时间结束后会继续执行
* wait()是Object类的方法，调用对象的wait()方法导致当前线程放弃对象的锁（线程暂停执行），进入对象的等待池（wait pool），只有调用对象的notify()方法（或notifyAll()方法）时才能唤醒等待池中的线程进入等锁池（lock pool），如果线程重新获得对象的锁就可以进入就绪状态

join方法，等待调用该方法的线程执行完毕后 再往下继续执

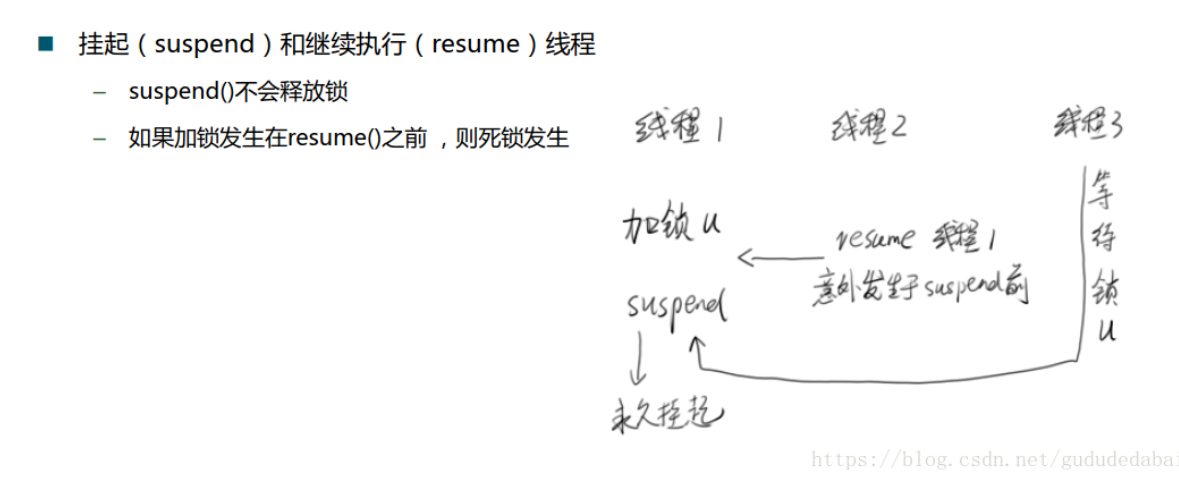
**线程的sleep()方法和yield()方法有什么区别？**

* sleep()方法给其他线程运行机会时不考虑线程的优先级；yield()方法只会给相同优先级或更高优先级的线程运行的机会；
* 线程执行sleep()方法后转入阻塞（blocked）状态，而执行yield()方法后转入就绪状态；

当一个线程使用了这个方法之后，它就会把自己CPU执行的时间让掉，使当前线程从执行状态（运行状态）变为可执行态（就绪状态），让自己或者其它的线程运行。cpu调度器会从众多的可执行态里选择，也就是说，当前也就是刚刚的那个线程还是有可能会被再次执行到的，并不是说一定会执行其他线程而该线程在下一次中不会执行到了

* sleep()方法声明抛出InterruptedException(中断异常，休眠被中断时)，而yield()方法没有声明任何异常；



后

