经验分享:对Java中的线程感想(多线程)Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/569/2021_2022__E7_BB_8F_E9_AA_8C_E5_88_86_E4_c104_569010.htm 1.进程和线程的区别 通俗一点说，进程就是程序的一次执行，而线程可以理解为进程中的执行的一段程序片段。 用一点文词说就是，每个进程都有独立的代码和数据空间（进程上下文）；而线程可以看成是轻量级的进程。一般来讲（不使用特殊技术），同一进程所产生的线程共享同一块内存空间。 同一进程中的两段代码是不可能同时执行的，除非引入线程。 线程是属于进程的，当进程退出时该进程所产生的线程都会被强制退出并清除。 线程占用的资源要少于进程所占用的资源。 进程和线程都可以有优先级。 在线程系统中进程也是一个线程。可以将进程理解为一个程序的第一个线程。 多进程在操作系统中，能同时运行多个任务（程序）。 多线程在同一应用程序中，有多个顺序流同时执行。 2.通过铁路售票程序来理解实现多线程的两种方法：通过java.lang.Thread类和通过Runnable接口 java中有两种实现多线程的方式。一是直接继承Thread类，二是实现Runnable接口。那么这两种实现多线程的方式在应用上有什么区别呢？ 为了回答这个问题，我们可以通过编写一段代码来进行分析。我们用代码来模拟铁路售票系统，实现通过四个售票点发售某日某次列车的100张车票，一个售票点用一个线程表示。 我们首先这样编写这个程序： public class ThreadDome1{ public static void main(String[] args){ ThreadTest t = new ThreadTest(). t.start(). t.start(). t.start(). t.start(). } } class ThreadTest extends Thread{ private int ticket = 100. public

void run(){ while(true){ if(ticket gt. 0){ System.out.println(Thread.currentThread().getName() " is saling ticket" ticket--). }else{ break. } } } } 这下达到目的了吗？ 从结果上看每个票号都被打印了四次，即四个线程各自卖各自的100张票，而不去卖共同的100张票。这种情况是怎么造成的呢？ 我们需要的是，多个线程去处理同一个资源，一个资源只能对应一个对象，在上面的程序中，我们创建了四个ThreadTest对象，就等于创建了四个资源，每个资源都有100张票，每个线程都在独自处理各自的资源。 经过这些实验和分析，可以总结出，要实现这个铁路售票程序，我们只能创建一个资源对象，但要创建多个线程去处理同一个资源对象，并且每个线程上所运行的是相同的程序代码。在回顾一下使用接口编写多线程的过程。 public class ThreadDemo1{ public static void main(String[] args){ ThreadTest t = new ThreadTest(). new Thread(t).start(). new Thread(t).start(). new Thread(t).start(). new Thread(t).start(). } } class ThreadTest implements Runnable{ private int tickets = 100. public void run(){ while(true){ if(tickets &gt. 0){ System.out.println(Thread.currentThread().getName() " is saling ticket " tickets--). } } } } 上面的程序中，创建了四个线程，每个线程调用的是同一个ThreadTest对象中的run（）方法，访问的是同一个对象中的变量（tickets）的实例，这个程序满足了我们的需求。在Windows上可以启动多个记事本程序一样，也就是多个进程使用同一个记事本程序代码。 可见，实现Runnable接口相对于继承Thread类来说，有如下显著的好处：（1）适合多个相同程序代码的线程去处理同一资源的情况，把虚拟CPU（线程）同程序的代码，数据有效的分离，

较好地体现了面向对象的设计思想。（2）可以避免由于Java的单继承特性带来的局限。我们经常碰到这样一种情况，即当我们要将已经继承了某一个类的子类放入多线程中，由于一个类不能同时有两个父类，所以不能用继承Thread类的方式，那么，这个类就只能采用实现Runnable接口的方式了。（3）有利于程序的健壮性，代码能够被多个线程共享，代码与数据是独立的。当多个线程的执行代码来自同一个类的实例时，即称它们共享相同的代码。多个线程操作相同的数据，与它们的代码无关。当共享访问相同的对象时，即它们共享相同的数据。当线程被构造时，需要的代码和数据通过一个对象作为构造函数实参传递进去，这个对象就是一个实现了Runnable接口的类的实例。