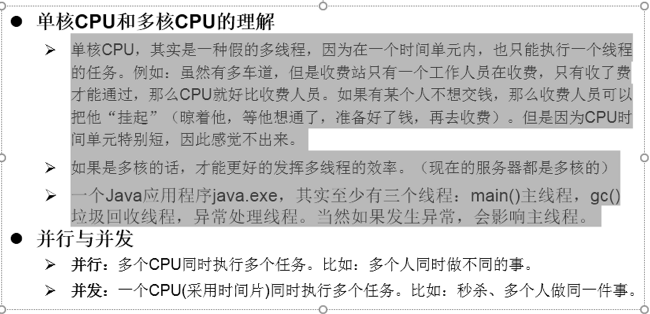


New出来的对象都是存放在堆中。

进程可以细化为多个线程。

每个线程，拥有自己的独立的：栈、程序计数器。

多个线程，共享同一个进程中的结构：方法区，堆。



**面试问题：1、有几种方式创建多线程？**

**总共有4种方式创建多线程！！！**

**1、继承Thread**

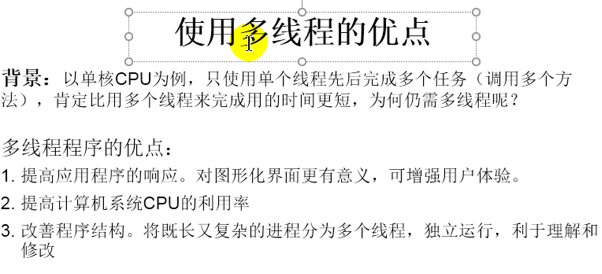
**2、实现Runnable**

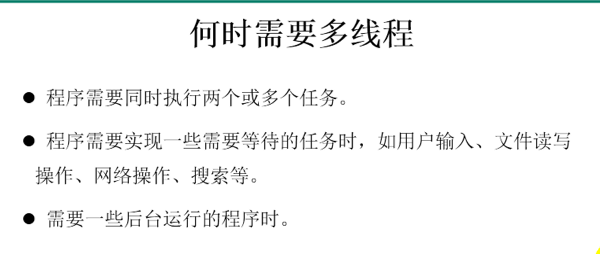
**3、实现Cellable接口，该方式会有返回值**

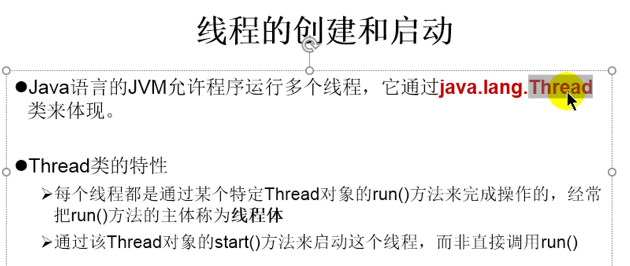
**4、使用线程池（该方式在开发中经常使用，一般在框架中实现）**

**2、有几种方式解决线程安全问题？**

**3种：1.同步方法，2.同步代码块,3.使用lock方式**

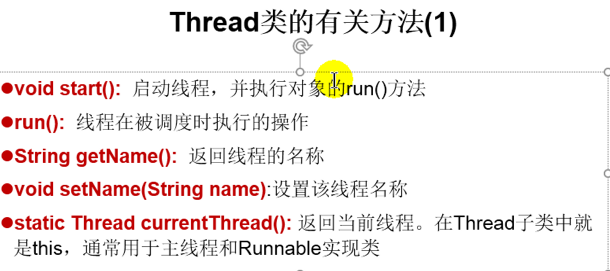


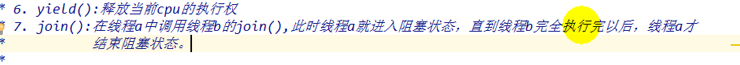




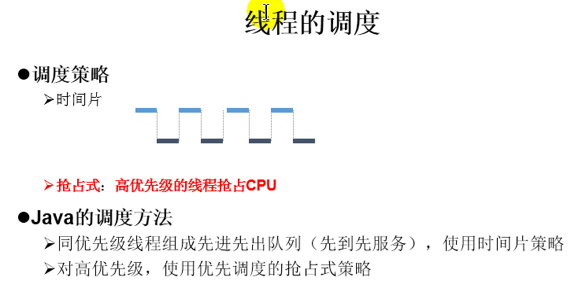




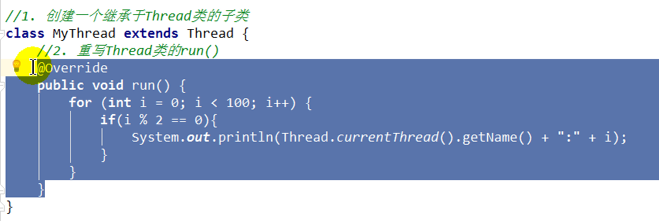




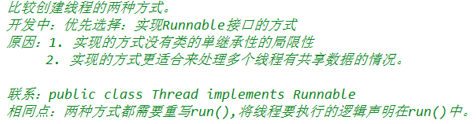




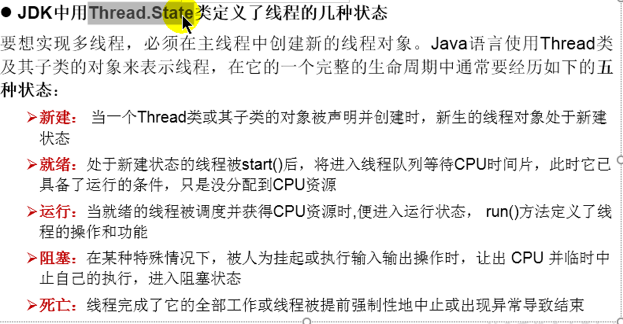


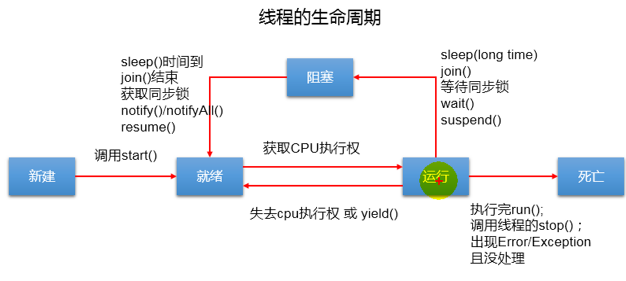


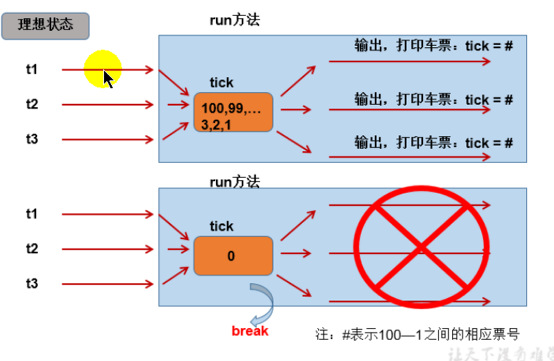


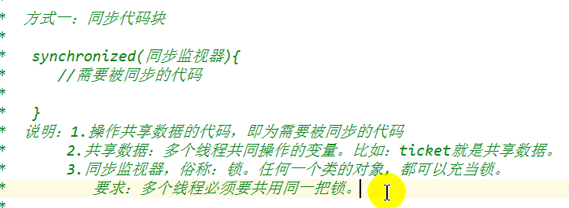


共享数据可以放在实现runnable接口类中，达到同步的效果。



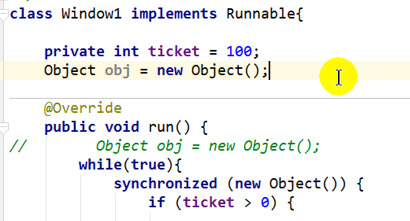


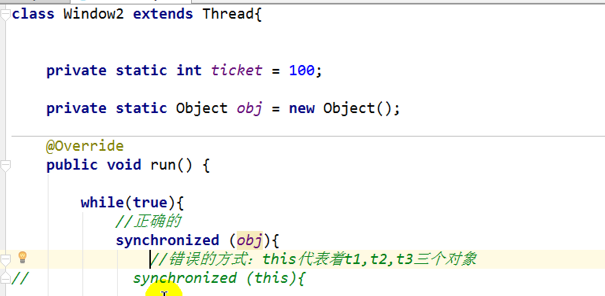






**此方法需要实现Runnable接口才能成功解决安全问题**

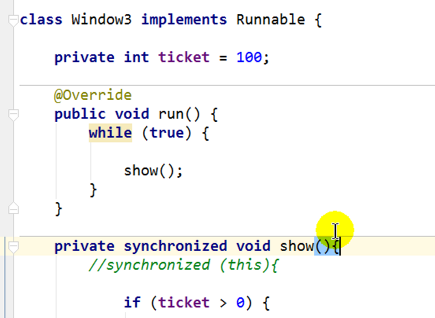


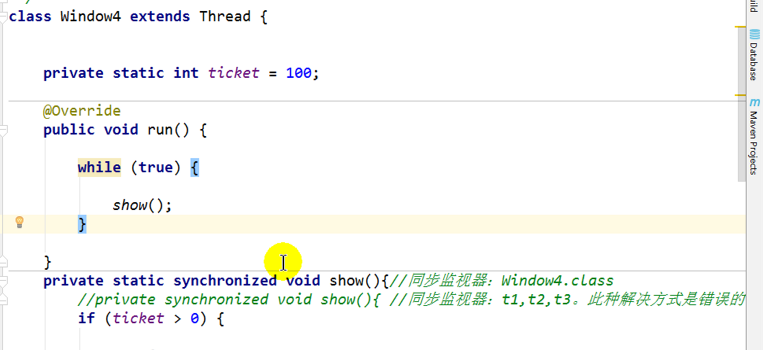


继承Thread创建多线程方法解决线程安全问题需要操作同一个对象，因此同步监视器需要是静态的才会是只有一个对象。

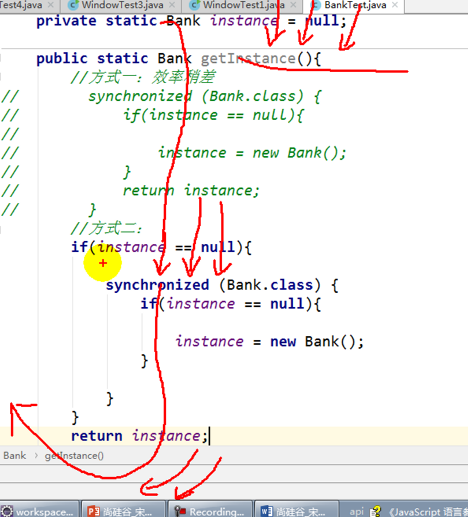
使用同步方法解决线程安全问题：

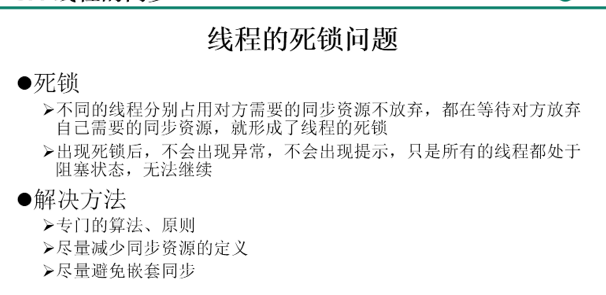
实现Runnable方法创建多线程直接在需要同步的方法中加上synchronized关键字即可解决线程安全问题。



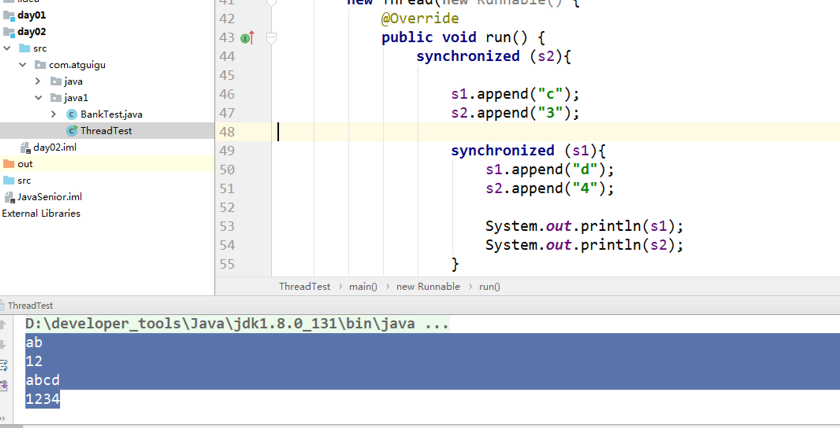
在继承Thread类创建多线程的方式下，需要同步的方法不可以直接写上synchrodnized关键字，因为他们操作的不是同一个同步监视器（锁），因此需要在此方法前加上static静态关键字。

懒汉式单例模式同步代码块解决线程安全：





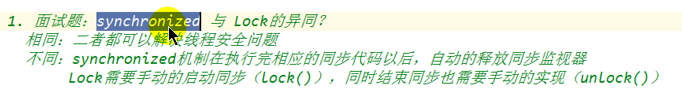
面试可能出现的问题：



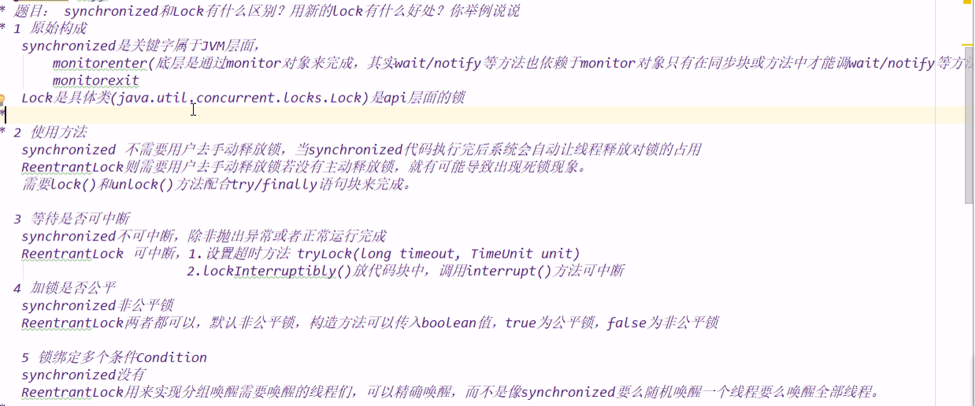
什么时候会发生死锁问题？在多个线程互相调用互相的同步监听器的时候会可能发生死锁问题。

问程序执行之后出现什么样的结果？这个程序会出现死锁吗？会

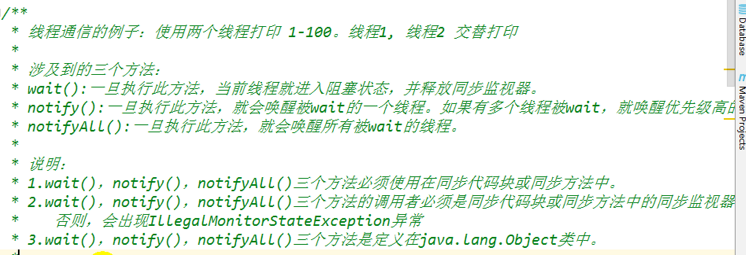


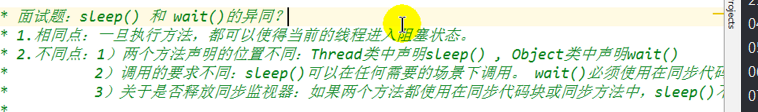






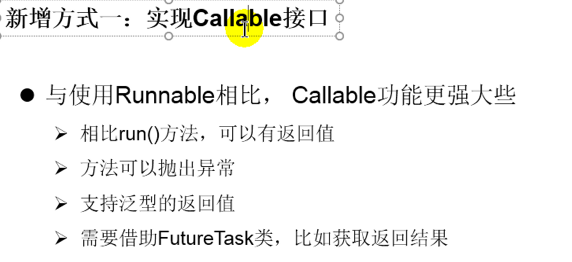
执行wait()方法会释放锁



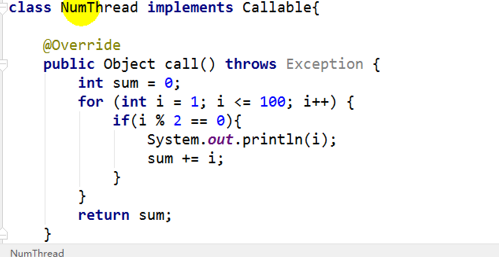


Sleep()方法不会释放同步监视器，wait()方法会释放同步监视器

第三种创建多线程方式：











第四种创建多线程方式：





