所有主程序中必然有main主线程和隐藏的后台线程（gc），其余的子线程都是由main主线程新建的。

implement方法实现的线程类没有start方法，需要按照如下方法开启线程：

*//        采用implement的方法是不能用start方法的*

*ThreadTestImpl threadImpl = new ThreadTestImpl();*

*Thread thread = new Thread(threadImpl);*

*thread.start();*

*线程的几个状态：*

*新建状态-》就绪状态-》运行状态-》（休眠状态）-》死亡状态*

*线程安全：*

*多个线程共享同一个全局变量数据，并需要对这个数据进行写操作的时候，需要对这个数据的读写操作进行控制。*

**课后面试题总结**

**问:什么是多线程安全？**

答:当多个线程同时共享，同一个**全局变量或静态变量**，做写的操作时，可能会发生数据冲突问题，也就是线程安全问题。做读操作是不会发生数据冲突问题。

**问:如何解决多线程之间线程安全问题?**

答:使用多线程之间**同步**或使**用锁(lock)**。

**问:为什么使用线程同步或使用锁能解决线程安全问题呢？**

答:将可能会发生数据冲突问题(线程不安全问题)，**只能让当前一个线程进行执行**。被包裹的代码执行完成后释放锁，让后才能让其他线程进行执行。这样的话就可以解决线程不安全问题。

**问:什么是多线程之间同步？**

答:当多个线程共享同一个资源,不会受到其他线程的干扰。

**问:什么是同步代码块？**

答:就是将可能会发生线程安全问题的代码，给包括起来。只能让当前一个线程进行执行，被包裹的代码执行完成之后才能释放所，让后才能让其他线程进行执行。

**问:多线程同步的分类？**

1. 使用同步代码块？

**synchronized(同一个数据){**

**可能会发生线程冲突问题**

**}**

|  |
| --- |
| **private Object mutex = new Object();// 自定义多线程同步锁**  **public void sale() {**  **synchronized (mutex) {**  **if (trainCount > 0) {**  **try {**  **Thread.*sleep*(10);**  **} catch (Exception e) {**  **}**  **System.*out*.println(Thread.*currentThread*().getName() + ",出售 第" + (100 - trainCount + 1) + "张票.");**  **trainCount--; }**  **}**  **}** |

1. 使用同步函数

在方法上修饰synchronized 称为同步函数

|  |
| --- |
| **public synchronized void sale() {**  **if (trainCount > 0) {**  **try {**  **Thread.*sleep*(40);**  **} catch (Exception e) {**  **}**  **System.*out*.println(Thread.*currentThread*().getName() + ",出售 第" + (100 - trainCount + 1) + "张票.");**  **trainCount--;**  **}**  **}** |

3.静态同步函数

方法上加上static关键字，使用synchronized 关键字修饰 为静态同步函数

静态的同步函数使用的锁是 该函数所属字节码文件对象

**问:同步代码块与同步函数区别？**

答:

同步代码使用自定锁(明锁)

同步函数使用this锁

**问:同步函数与静态同步函数区别?**

注意:有些面试会这样问：例如现在一个静态方法和一个非静态静态怎么实现同步？

答:

同步函数使用this锁

静态同步函数使用字节码文件，也就是类.class

**问:什么是多线程死锁?**

**答：**

同步中嵌套同步

解决办法:同步中尽量不要嵌套同步