线程

进程：正在运行的应用程序，负责了内存空间的划分

线程：线程在一个进程中负责了代码的执行，就是进程中一个执行路径

多线程：在一个进程中有多个线程同时在执行不同的任务

一个java应用程序至少有两个线程，一个负责main方法的执行，一个负责垃圾回收器

多线程的好处：

1. 可以在一个进程中执行多个任务
2. 提高资源的利用率

多线程的弊端：

1. 降低一个进程中线程执行的概率
2. 增加cpu的负担
3. 引发线程安全问题
4. 出现死锁

如何创建线程：

方式一：

1. 自定义个一个类继承thread类
2. 重写Thread的run方法
3. 创建Thread的子类对象，并且调用start方法启动线程

自定义线程安全问题的解决方案

方式一：同步代码块

方式二:同步函数(静态和非静态函数)

使用synchronized修饰方法

1. 非静态的锁对象是this对象, 静态锁的是当前函数所属的类的字节码文件(class对象)
2. 同步函数的锁对象是固定的，不能由你指定

推荐使用同步代码块:因为方便我们控制需要同步的部分，同步函数是整个函数都被控制

如果不存在线程安全问题，千万不要使用同步代码块，因为会降低效率

死锁：

条件：1.多线程2.每个线程控制多个相同的共享资源3.一个锁里面套着一个锁，导致线程进去后出不来

方式二：

1. 自定义一个类实现Runnable 接口
2. 实现接口的run方法
3. 创建类的实例，new Thread(),实例名作为参数，调用start( )方法

为什么要把Runnable实现类的对象作为实参传递给Thread对象？

作用：因为只有Thread 或Thread的子类才能调用start()方法，而实现Runnable接口的对象只是一个普通的对象，并不是线程对象，所以将它传入Thread（），将它的run（）方法让给Thread()去执行

推荐使用第二种（因为java是单继承多实现的，可以实现接口的同时再继承别的父类）

并且他可以共享线程类的实例变量

线程的常用方法：

静态方法：sleep()

currentThread()返回线程对象，在哪个线程中调用就返回哪个线程

非静态方法：setName()

setPriority()

getName()

getPriority()