## synchronized关键字是用来控制线程同步的，就是在多线程的环境下，控制synchronized代码段不被多个线程同时执行。synchronized既可以加在一段代码上，也可以加在方法上。

# 1、线程中new对象，再调用

## 1、加在方法上，其实没有成功

|  |
| --- |
| **public** **class** OneMain {    **public** **static** **void** main(String[] args) {  **for** (**int** i = 0; i < 3; i++) {  Thread thread = **new** MyThread();  thread.start();  }  }  }  **class** Sync {    **public** **synchronized** **void** test() {  System.*out*.println("test开始..");  **try** {  Thread.*sleep*(1000);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  System.*out*.println("test结束..");  }  }    **class** MyThread **extends** Thread {    **public** **void** run() {  Sync sync = **new** Sync();  sync.test();  }  } |

## 2、测试控制台结果



## 3、解释，上面的程序起了三个线程，同时运行Sync类中的test()方法，虽然test()方法加上了synchronized，但是还是同时运行起来，貌似synchronized没起作用。

## 1、每个线程中都new了一个Sync类的对象，也就是产生了三个Sync对象，由于不是同一个对象，所以可以多线程同时运行synchronized方法或代码段。

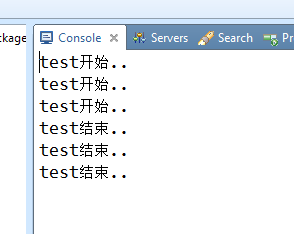
## 2、对于非static的synchronized方法，锁的就是对象本身也就是this。

# 2、将synchronized 写到方法里面，也是没有成功。

## 1、test方法如下

|  |
| --- |
| **public** **class** TwoMain {    **public** **static** **void** main(String[] args) {  **for** (**int** i = 0; i < 3; i++) {  TwoMyThread thread = **new** TwoMyThread();  thread.start();  }  }  }  **class** TeoSync {    **public** **void** test() {  **synchronized**(**this**){  System.*out*.println("test开始..");  **try** {  Thread.*sleep*(1000);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  System.*out*.println("test结束..");  }  }  }    **class** TwoMyThread **extends** Thread {    **public** **void** run() {  TeoSync sync = **new** TeoSync();  sync.test();  }  } |

## 2、测试结果



## 3、解释：

## 1、synchronized(this)以及非static的synchronized方法（至于static synchronized方法请往下看），只能防止多个线程同时执行同一个对象的同步代码段。很明显，线程中new了的是3个对象

## 2、synchronized锁住的是括号里的对象，而不是代码。

# 3、正确写法

## 1、对于上面第二种的修改

|  |
| --- |
| **public** **class** ThreeForTwoMain {    **public** **static** **void** main(String[] args) {  ThreeTeoSync threeTeoSync = **new** ThreeTeoSync();  **for** (**int** i = 0; i < 3; i++) {  ThreeTwoMyThread thread = **new** ThreeTwoMyThread(threeTeoSync);  thread.start();  }  }  }  **class** ThreeTeoSync {    **public** **void** test() {  **synchronized**(**this**){  System.*out*.println("test开始..");  **try** {  Thread.*sleep*(1000);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  System.*out*.println("test结束..");  }  }  }    **class** ThreeTwoMyThread **extends** Thread {    **public** ThreeTeoSync threeTeoSync;    **public** ThreeTwoMyThread(ThreeTeoSync threeTeoSync){  **this**.threeTeoSync = threeTeoSync;  }  **public** **void** run() {  threeTeoSync.test();  }  } |

## 2、对于上面第一种的操作

|  |
| --- |
| **public** **class** ThreeForOneMain {    **public** **static** **void** main(String[] args) {  ThreeOneSync threeTeoSync = **new** ThreeOneSync();  **for** (**int** i = 0; i < 3; i++) {  ThreeOneMyThread thread = **new** ThreeOneMyThread(threeTeoSync);  thread.start();  }  }  }  **class** ThreeOneSync {    **public** **synchronized** **void** test() {  System.*out*.println("test开始..");  **try** {  Thread.*sleep*(1000);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  System.*out*.println("test结束..");    }  }    **class** ThreeOneMyThread **extends** Thread {    **public** ThreeOneSync threeTeoSync;    **public** ThreeOneMyThread(ThreeOneSync threeTeoSync){  **this**.threeTeoSync = threeTeoSync;  }  **public** **void** run() {  threeTeoSync.test();  }  } |

# 4、非要new3个对象怎么解决，则是用JavaName.class, synchronized(FourNewMain.class){

## 解释：synchronized后的括号中锁同一个固定对象，这样就行了。这样是没问题，但是，比较多的做法是让synchronized锁这个类对应的Class对象。

|  |
| --- |
| **public** **class** FourNewMain {    **public** **static** **void** main(String[] args) {  **for** (**int** i = 0; i < 3; i++) {  FourMyThread thread = **new** FourMyThread();  thread.start();  }  }  }  **class** FourSync {    **public** **void** test() {  **synchronized**(FourNewMain.**class**){  System.*out*.println("test开始..");  **try** {  Thread.*sleep*(1000);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  System.*out*.println("test结束..");  }  }  }    **class** FourMyThread **extends** Thread {    **public** **void** run() {  FourSync sync = **new** FourSync();  sync.test();  } |

# 5、静态的static （单例模式有相关代码）

## static方法可以直接类名加方法名调用，方法中无法使用this，所以它锁的不是this，而是类的Class对象，所以，static synchronized方法也相当于全局锁，相当于锁住了代码段。