### 出处：http://blog.csdn.net/silencecarrot/article/details/52415704

### [同步方法与同步块、静态与非静态的区别](http://blog.csdn.net/SilenceCarrot/article/details/52415704)

分类：***学习日记***

 （410）  （0）  举报  收藏

java使用synchronized同步，分为四种情况：

  实例方法同步

  实例方法中同步块

  静态方法同步

  静态方法中同步块

我们从两个方面来说他们的不同，一个是同步方法和同步块的区别，一个是静态和非静态的区别。

同步方法就是在方法前加关键字synchronized，然后被同步的方法一次只能有一个线程进入，其他线程等待。而同步方法则是在方法内部使用大括号使得一个代码块得到同步。同步块会有一个同步的”目标“，使得同步块更加灵活一些（同步块可以通过”目标“决定需要锁定的对象）。一般情况下，如果此”目标“为this，那么同步方法和同步块没有太大的区别。

另外，通过反编译可以看出，同步块比同步方法多了两个指令。因此同步方法是比同步块要快一些。

非静态和静态的区别主要在于（以同步方法为例）： ****非静态的同步方法是锁定类的实例的，而静态的同步方法是锁定类的；****

也就是说，对于非静态的同步方法，在同一时刻，一个类的一个实例中，只有一个线程能进入同步的方法。但是对于多个实例，每一个实例的一个线程都可以进入同一同步的方法。

Demo1：一个实例的多个线程，一次只能有一个线程进入非静态同步的方法。

**[java]** [view plain](http://blog.csdn.net/silencecarrot/article/details/52415704" \o "view plain) [copy](http://blog.csdn.net/silencecarrot/article/details/52415704" \o "copy" \t "http://blog.csdn.net/silencecarrot/article/details/_blank) [IMG_256](https://code.csdn.net/snippets/1863653" \o "在CODE上查看代码片" \t "http://blog.csdn.net/silencecarrot/article/details/_blank)[IMG_257](https://code.csdn.net/snippets/1863653/fork" \o "派生到我的代码片" \t "http://blog.csdn.net/silencecarrot/article/details/_blank)

1. **package** SynchronizedTest;
3. /\*\*
4. \* Created by carrot on 16/8/31.
5. \*/
6. **public** **class** SyncFunc {
8. **public** **synchronized** **void** func1() {
9. System.out.println(Thread.currentThread().getName() + " is running");
10. **try** {
11. Thread.sleep(3000);
12. } **catch** (InterruptedException e) {
13. e.printStackTrace();
14. }
15. System.out.println(Thread.currentThread().getName() + " is stop");
16. }
18. **public** **static** **void** main(String[] args) {
20. NewThread newThread1 = **new** NewThread();
21. NewThread newThread2 = **new** NewThread();
22. NewThread newThread3 = **new** NewThread();
24. newThread1.start();
25. newThread2.start();
26. newThread3.start();
28. }
29. }
31. **class** NewThread **extends** Thread {
33. **static** SyncFunc syncFunc = **new** SyncFunc();
35. @Override
36. **public** **void** run() {
37. syncFunc.func1();
38. }
40. }

结果：

**[java]** [view plain](http://blog.csdn.net/silencecarrot/article/details/52415704" \o "view plain) [copy](http://blog.csdn.net/silencecarrot/article/details/52415704" \o "copy" \t "http://blog.csdn.net/silencecarrot/article/details/_blank) [IMG_258](https://code.csdn.net/snippets/1863653" \o "在CODE上查看代码片" \t "http://blog.csdn.net/silencecarrot/article/details/_blank)[IMG_259](https://code.csdn.net/snippets/1863653/fork" \o "派生到我的代码片" \t "http://blog.csdn.net/silencecarrot/article/details/_blank)

1. Thread-0 is running
2. Thread-0 is stop
3. Thread-2 is running
4. Thread-2 is stop
5. Thread-1 is running
6. Thread-1 is stop

从结果可见，每次只能有一个线程进入非静态同步的方法。

Demo2：多个实例的线程能同时进入非静态同步的方法。

**[java]** [view plain](http://blog.csdn.net/silencecarrot/article/details/52415704" \o "view plain) [copy](http://blog.csdn.net/silencecarrot/article/details/52415704" \o "copy" \t "http://blog.csdn.net/silencecarrot/article/details/_blank) [IMG_260](https://code.csdn.net/snippets/1863653" \o "在CODE上查看代码片" \t "http://blog.csdn.net/silencecarrot/article/details/_blank)[IMG_261](https://code.csdn.net/snippets/1863653/fork" \o "派生到我的代码片" \t "http://blog.csdn.net/silencecarrot/article/details/_blank)

1. **package** SynchronizedTest;
3. /\*\*
4. \* Created by carrot on 16/8/31.
5. \*/
6. **public** **class** SyncFunc {
8. **public** **synchronized** **void** func1() {
9. System.out.println(Thread.currentThread().getName() + " is running");
10. **try** {
11. Thread.sleep(3000);
12. } **catch** (InterruptedException e) {
13. e.printStackTrace();
14. }
15. System.out.println(Thread.currentThread().getName() + " is stop");
16. }
18. **public** **static** **void** main(String[] args) {
20. NewThread newThread1 = **new** NewThread();
21. NewThread newThread2 = **new** NewThread();
22. NewThread newThread3 = **new** NewThread();
24. newThread1.start();
25. newThread2.start();
26. newThread3.start();
28. }
29. }
31. **class** NewThread **extends** Thread {
33. SyncFunc syncFunc = **new** SyncFunc();
35. @Override
36. **public** **void** run() {
37. syncFunc.func1();
38. }
40. }

结果：

**[java]** [view plain](http://blog.csdn.net/silencecarrot/article/details/52415704" \o "view plain) [copy](http://blog.csdn.net/silencecarrot/article/details/52415704" \o "copy" \t "http://blog.csdn.net/silencecarrot/article/details/_blank) [IMG_262](https://code.csdn.net/snippets/1863653" \o "在CODE上查看代码片" \t "http://blog.csdn.net/silencecarrot/article/details/_blank)[IMG_263](https://code.csdn.net/snippets/1863653/fork" \o "派生到我的代码片" \t "http://blog.csdn.net/silencecarrot/article/details/_blank)

1. Thread-0 is running
2. Thread-1 is running
3. Thread-2 is running
4. Thread-0 is stop
5. Thread-2 is stop
6. Thread-1 is stop

从结果可以看出，多个实例的线程同时进入了同步的非静态方法。

Demo3：多个实例的线程进入静态的同步方法。

**[java]** [view plain](http://blog.csdn.net/silencecarrot/article/details/52415704" \o "view plain) [copy](http://blog.csdn.net/silencecarrot/article/details/52415704" \o "copy" \t "http://blog.csdn.net/silencecarrot/article/details/_blank) [IMG_264](https://code.csdn.net/snippets/1863653" \o "在CODE上查看代码片" \t "http://blog.csdn.net/silencecarrot/article/details/_blank)[IMG_265](https://code.csdn.net/snippets/1863653/fork" \o "派生到我的代码片" \t "http://blog.csdn.net/silencecarrot/article/details/_blank)

1. **package** SynchronizedTest;
3. /\*\*
4. \* Created by carrot on 16/8/31.
5. \*/
6. **public** **class** SyncFunc {
8. **public** **static** **synchronized** **void** func1() {
9. System.out.println(Thread.currentThread().getName() + " is running");
10. **try** {
11. Thread.sleep(3000);
12. } **catch** (InterruptedException e) {
13. e.printStackTrace();
14. }
15. System.out.println(Thread.currentThread().getName() + " is stop");
16. }
18. **public** **static** **void** main(String[] args) {
20. NewThread newThread1 = **new** NewThread();
21. NewThread newThread2 = **new** NewThread();
22. NewThread newThread3 = **new** NewThread();
24. newThread1.start();
25. newThread2.start();
26. newThread3.start();
28. }
29. }
31. **class** NewThread **extends** Thread {
33. SyncFunc syncFunc = **new** SyncFunc();
35. @Override
36. **public** **void** run() {
37. syncFunc.func1();
38. }
40. }

结果：

**[java]** [view plain](http://blog.csdn.net/silencecarrot/article/details/52415704" \o "view plain) [copy](http://blog.csdn.net/silencecarrot/article/details/52415704" \o "copy" \t "http://blog.csdn.net/silencecarrot/article/details/_blank) [IMG_266](https://code.csdn.net/snippets/1863653" \o "在CODE上查看代码片" \t "http://blog.csdn.net/silencecarrot/article/details/_blank)[IMG_267](https://code.csdn.net/snippets/1863653/fork" \o "派生到我的代码片" \t "http://blog.csdn.net/silencecarrot/article/details/_blank)

1. Thread-0 is running
2. Thread-0 is stop
3. Thread-2 is running
4. Thread-2 is stop
5. Thread-1 is running
6. Thread-1 is stop

从结果可以看出，对于同一个对象的多个实例，在进入静态的同步方法时，一次只能有一个类实例进入。

版权声明：技术就要分享才有意思，欢迎大家分享（注明出处），欢迎大家纠错。