

## 【第14话：经典面试题wait()和sleep()的区别】

Hello 小伙伴们，这节课给大家讲解一个经典面试题，也是出现频率比较高的面试题“wait()和sleep()的区别”。

我们先来说说sleep()方法。它是Thread中的一个静态方法。可以使用Thread.sleep();进行直接调用，表示让线程处于休眠状态。可以使用在任何线程的代码中，Thread.sleep()写在哪个线程就表示让哪个线程休眠，达到指定的休眠状态后线程自动恢复运行状态。

### wait()和sleep()的区别



在Thread中有重载了两个sleep()方法，其中使用的最多的是第一个sleep(long);参数表示休眠的毫秒数。

```
m sleep(long): void  
m sleep(long, int): void
```

下面代码就表示输出“程序开始执行”后休眠1秒，一秒后继续执行，输出“休眠结束”

```
1 public class Test {  
2     public static void main(String[] args) throws InterruptedException {  
3         System.out.println("程序开始执行");  
4         Thread.sleep(1000);  
5         System.out.println("休眠结束");  
6     }  
7 }
```

sleep()方法说完之后说一下wait()方法。

wait() 是Object中的方法。调用wait()后会让线程从运行状态变为阻塞状态。

在Object类中提供了wait()的重载方法

```
m 📁 notify(): void 唤醒某一个线程, 系统决定
m 📁 notifyAll(): void 唤醒所有线程
m 📁 wait(): void 等待, 执行 notify() | notifyAll() 唤醒线程 执行 interrupt() 中断, 结束线程
m 📁 wait(long): void 等待最长时间后唤醒线程
m 📁 wait(long, int): void 等待最长时间或最长时间 +1 后唤醒线程
```

wait()方法会让线程变为阻塞，阻塞的同时会**释放锁**。所以wait()必须要求被等待的线程**持有锁**，调用wait()后会把锁释放，其他线程竞争获取锁。当其他线程竞争获取到锁以后，如果达到某个条件后可以通过notify()唤醒，如果有多个wait的线程，系统判断唤醒其中一个。也可以使用notifyAll全部唤醒。唤醒后线程处于就绪状态。

需要注意的是：**一个线程唤醒其他线程时，要求当前线程必须持有锁**

最简易结论：

1. 使用wait()和notify() | notifyAll()要求必须有锁。所以wait()、notify()、notifyAll() 都是放入锁的代码中。

3. wait()后的线程需要通过notify() | notifyAll() 唤醒

下面给用生产者和消费者模式来演示wait()和notify()的效果。

```
1 public class Production {
2     int totalCount = 0;
3     String str = "库存锁";
4     /* 生产者 */
5     public class Producer extends Thread{
6         @Override
7         public void run() {
8             synchronized (str){
9                 while (true) {
10                     totalCount++;
11                     System.out.println("生产了"+totalCount+"个产品");
12                     try {
13                         Thread.sleep(500);
14                         if(totalCount==10){
15                             try {
16                                 //唤醒一个线程，Producer线程并没有等待，所以会继续执行
17                                 str.notify();
18                                 //Producer线程等待，释放锁。线程等待后续代码就不会继续执行，直到被唤
19                                 醒
20                                 str.wait();
21                             } catch (InterruptedException e) {
22                                 e.printStackTrace();
23                             }
24                         } catch (InterruptedException e) {
25                             e.printStackTrace();
26                         }
27                     }
28                 }
29             }
30         }
31     }
32 }
```

```

28     }
29 }
30 }
31 /* 消费者 */
32 public class Consumer extends Thread{
33     @Override
34     public void run() {
35         synchronized (str){
36             //库存为0, Consumer线程等待(不等待, Consumer线程先获得到锁会出现负消费)
37             if(totalCount==0){
38                 try {
39                     str.wait();
40                 } catch (InterruptedException e) {
41                     e.printStackTrace();
42                 }
43             }
44             while (true) {
45                 totalCount--;
46                 System.out.println("消费了" + (10 - totalCount) + "个产品");
47                 try {
48                     Thread.sleep(500);
49                     if(totalCount==0){
50                         try {
51                             //唤醒一个线程, Consumer线程并没有等待, 所以会继续执行
52                             str.notify();
53                             //Consumer线程等待, 释放锁. 线程等待后续代码就不会继续执行, 直到被唤
醒
54                             str.wait();
55                         } catch (InterruptedException e) {
56                             e.printStackTrace();
57                         }
58                     }
59                 } catch (InterruptedException e) {
60                     e.printStackTrace();
61                 }
62             }
63         }
64     }
65 }
66 public static void main(String[] args) {
67     Production production = new Production();
68     Consumer consumer = production.new Consumer();
69     consumer.start();
70     Producer producer = production.new Producer();
71     producer.start();
72 }
73 }
74

```

通过上面代码小伙伴应该知道了wait()和sleep()的使用方式, 下面给小伙伴们分门别类的总结一下wait()和sleep()的区别:

#### 1. 所属类不同

wait(long) 是Object中方法

sleep(long)是Thread的方法

## 2. 唤醒机制不同

wait() 没有设置最大时间情况下，必须等待notify() | notifyAll()

sleep()是到指定时间自动唤醒。

## 3. 锁机制不同

wait(long)释放锁

sleep(long)只是让线程休眠，不会释放锁

## 4. 使用位置不同

wait()必须在持有锁的线程代码中

sleep()可以使用在任意地方

## 5. 方法类型不同

wait()是实例方法

sleep()是静态方法