Java 中的线程：对象锁 - III（为什么对象锁不起作用？）

切坦·库达尔卡

2021 年 6 月 15 日 警察

为什么对象锁不起作用？这是我上一篇关于“Java 中的线程：对象锁 - II”的文章的延续

本系列文章试图帮助读者了解 Java 中的锁概念。假设读者对 Java 和 Java 中的线程创建有一些先验知识。

介绍

这篇文章是上一篇文章“Java 中的线程：对象锁 II”的延续。在前面的例子中，我们从同一个处理器对象中创建了线程`t1`，并将它们与对象锁同步，对象锁是类处理器的成员变量，并被实例化为. 因此，两个线程和都可以访问相同的内容，因为两个线程和都是从以下代码片段中创建的：`t2p1objLockp1t1t2objLockt1t2p1`

C# 复制代码

```
processor p1 = new processor();
Thread t1 = new Thread(p1, "t1");
Thread t2 = new Thread(p1, "t2");
```

让我们挤一些柠檬来为这个程序添加一种浓郁的味道。想想如果我们用处理器实例化`t2`的新对象`p2`来创建会发生什么？该程序将如下所示：

C# 缩小▲ 复制代码

```
class processor implements Runnable{
    Object obj;
    public processor(){
        obj = new Object();
    }
    public void run() {
        display();
    }
    public void display() {
        synchronized(obj) {
            for (int i = 0 ; i<= 5; i++) {
                System.out.println("i = " + i + " In thread" + Thread.currentThread());
            }
        }
    }
}

class locks {
```

```
    public static void main(String[] args) {
        processor p1 = new processor();
        processor p2 = new processor();
        Thread t1 = new Thread(p1, "t1");
        Thread t2 = new Thread(p2, "t2");
        t1.start();
        t2.start();
    }
}
```

你能猜出程序的输出是什么吗？难道是有规律的输出或者这将是一个不规则的交错式输出全的printfs来自t1然后t2再t1等等等等?

好吧.....上面的程序编译并成功运行，但请检查我们得到了什么输出！

C# 复制代码

```
i = 0 In threadThread[t2,5,main]
i = 0 In threadThread[t1,5,main]
i = 1 In threadThread[t1,5,main]
i = 1 In threadThread[t2,5,main]
i = 2 In threadThread[t1,5,main]
i = 2 In threadThread[t2,5,main]
i = 3 In threadThread[t1,5,main]
i = 3 In threadThread[t2,5,main]
i = 4 In threadThread[t1,5,main]
i = 4 In threadThread[t2,5,main]
i = 5 In threadThread[t1,5,main]
i = 5 In threadThread[t2,5,main]
```

如果你仔细观察，我们会得到一个交错的、不规则的输出。好吧，如果我们有一个对象锁，为什么我们会得到一个交错的输出？答案很简单。好吧，我们确实有一个对象锁。但在这种情况下，objectLockforp1和objectLockforp2是不同的，并不常见。对象p1的创造出processor()有自己的实例objectLock是从完全不同objectLock的p2创作出来的processor()，因为p1和p2是两个不同的对象了。没有竞争条件来获取相同的锁。Thread t2获取自己的锁并thread t1获取自己的锁，然后继续。因此，我们得到一个交错的输出。

那么我们该怎么做才能获得同步输出呢？为了获得同步输出，我们需要确保两个线程都可以访问同一个锁。这是在以下示例中完成的:

C# 缩小▲ 复制代码

```
class processor implements Runnable{
    Object objLock;
    public processor(Object objLock){
        this.objLock = objLock;
    }
    public void run() {
        display();
    }
    public void display() {
        synchronized(objLock) {
            for (int i = 0 ; i<= 5; i++) {
                System.out.println("i = " + i + " In thread" + Thread.currentThread());
            }
        }
    }
}

class locks {
    public static void main(String[] args) {
        Object objLock = new Object();
        processor p1 = new processor(objLock);
        processor p2 = new processor(objLock);
        Thread t1 = new Thread(p1, "t1");
        Thread t2 = new Thread(p2, "t2");
        t1.start();
        t2.start();
    }
```

```
}

Output
=======
i = 0 In threadThread[t1,5,main]
i = 1 In threadThread[t1,5,main]
i = 2 In threadThread[t1,5,main]
i = 3 In threadThread[t1,5,main]
i = 4 In threadThread[t1,5,main]
i = 5 In threadThread[t1,5,main]
i = 0 In threadThread[t2,5,main]
i = 1 In threadThread[t2,5,main]
i = 2 In threadThread[t2,5,main]
i = 3 In threadThread[t2,5,main]
i = 4 In threadThread[t2,5,main]
i = 5 In threadThread[t2,5,main]
```

上述程序编译并成功运行以产生所需的同步输出。为了`thread t1`和`t2`具有相同的`objLock`，我们创建了它`main()`，然后将它传递给处理器对象`p1`并`p2`通过它的构造函数。现在两者`p1`和`p2`共享相同`objLock`。在`display()`forloop 方法中的竞争条件期间，`t1`获取到 的锁`objLock`。因此`t2`必须等到`t1`释放锁。`t2`只有在`t2`释放锁后才有机会进入 forloop 。这样，我们就得到了一个同步的输出。希望能说明对象锁是如何工作的。如果两个或多个线程只有一个且只有一个共同的对象锁，则它们将同步。

兴趣点

如果两个或多个线程只有一个且只有一个共同的对象锁，则它们将同步。

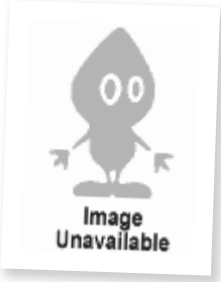
历史

- 2021 年 6 月15日：初步修订

执照

分享

关于作者



切坦·库达尔卡
软件开发人员（高级）
印度

手表
该会员

我是一名软件工程师，拥有大约 7 年以上的经验。我的大部分经验是在存储技术方面。

评论和讨论

添加评论或问题 电子邮件提醒 Search Comments

第一 页上一页 下一页

我的4票
Vincent Radio 17-Jun-21 1:24

回复：我投了 4 票
Chetan Kudalkar 17-Jun-21 10:38

刷新 1

一般 新闻 建议 问题 错误 答案 笑话 赞美 咆哮 管理员

使用Ctrl+Left/Right 切换消息，Ctrl+Up/Down 切换主题，Ctrl+Shift+Left/Right 切换页面。