线程同步技术--wait-notify

笔记本:	java-多线程		
创建时间:	2019/5/3 19:42	**更新时间:**	2019/5/4 10:32
作者:	ANNER		
URL:	https://blog.csdn.net/luckyzhoustar/article/details/48179161		

对于每一个对象的锁对象，可以实现唯一的调用，那么为了对象的线程同步，需要动态的释放锁和分配锁对象

notify（）

用来唤醒在此对象上等待的单个线程。说的有点太专业。打个比方，现在有十栋大房子，里面有很多被上了锁的房间，奇怪的是锁都是一样的，更不可思议的是，现在只有一把钥匙。而此时，张三用完钥匙后，就会发出归还钥匙的提醒，就相当于发出notify（）通知，但是要注意的是，此时钥匙还在张三手中，只不过，当张三发出notify（）通知后，JVM从那些整个沉睡的线程，唤醒一个。对应本例子，就是从其余的九栋大房子中唤醒一家，至于提醒谁来拿这把钥匙，就看JVM如何分配资源了。等到张三把钥匙归还后，那个被提醒的哪家，就可以使用该把钥匙来开房间门了。与此相对应的，还有一个notifyAll（）方法。这是什么意思呢，还是本例，张三嗓门大，这时吼了一嗓子，即notifyAll（），**所有沉睡的线程全都被吵醒了，当张三归还钥匙后，他们就可以竞争**了，注意，刚才是JVM自动分配，而此时是线程之间竞争，比如优先级等等条件，是有区别的。

notify（）方法执行后，并不是立即释放锁，而是等到加锁的代码块执行完后，才开始释放的，相当于本例中，张三只是发出了归还的通知，但是钥匙还没有归还，需要等到代码块，执行完后，才可以归还。

wait（）

执行到这个方法时，就把钥匙归还，开始睡觉了。Thread.sleep()与Object.wait()二者都可以暂停当前线程，释放CPU控制权，主要的区别在于**Object.wait()在释放CPU同时，释放了对象锁的控制。**

顺序打印ABC：

```
public ABCSyn(String name, Object prev, Object self) {
    this.name = name;
    this.prev = prev;
    this.self = self;
}

public void run() {
    int count=10;
    //避免虚假唤醒，不可以使用条件判断
    while (count>0){
        synchronized (prev){
            synchronized (self){
                System.out.print(name);
                count--;
                //全部唤醒等待进程
                self.notifyAll();
            }
            try {
                prev.wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```