同步代码依赖于一种简单的可重入锁。这种锁使用简单，但也有诸多限制。[java.util.concurrent.locks](http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/locks/package-summary.html)包提供了更复杂的锁。我们不会详细考察这个包，但会重点关注其最基本的接口，锁。

锁对象作用非常类似同步代码使用的隐式锁。如同隐式锁，每次只有一个线程可以获得锁对象。通过关联[Condition](http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/locks/Condition.html)对象，锁对象也支持wait/notify机制。

锁对象之于隐式锁最大的优势在于，它们有能力收回获得锁的尝试。如果当前锁对象不可用，或者锁请求超时（如果超时时间已指定），tryLock方法会收回获取锁的请求。如果在锁获取前，另一个线程发送了一个中断，lockInterruptibly方法也会收回获取锁的请求。

让我们使用锁对象来解决我们在[活跃度](http://docs.oracle.com/javase/tutorial/essential/concurrency/liveness.html)中见到的死锁问题。Alphonse和Gaston已经把自己训练成能注意到朋友何时要鞠躬。我们通过要求Friend对象在双方鞠躬前必须先获得锁来模拟这次改善。下面是改善后模型的源代码，Safelock。为了展示其用途广泛，我们假设Alphonse和Gaston对于他们新发现的稳定鞠躬的能力是如此入迷，以至于他们无法不相互鞠躬。

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

import java.util.Random;

public class Safelock {

static class Friend {

private final String name;

private final Lock lock = new ReentrantLock();

public Friend(String name) {

this.name = name;

}

public String getName() {

return this.name;

}

public boolean impendingBow(Friend bower) {

Boolean myLock = false;

Boolean yourLock = false;

try {

myLock = lock.tryLock();

yourLock = bower.lock.tryLock();

} finally {

if (! (myLock && yourLock)) {

if (myLock) {

lock.unlock();

}

if (yourLock) {

bower.lock.unlock();

}

}

}

return myLock && yourLock;

}

public void bow(Friend bower) {

if (impendingBow(bower)) {

try {

System.out.format("%s: %s has"

+ " bowed to me!%n",

this.name, bower.getName());

bower.bowBack(this);

} finally {

lock.unlock();

bower.lock.unlock();

}

} else {

System.out.format("%s: %s started"

+ " to bow to me, but saw that"

+ " I was already bowing to"

+ " him.%n",

this.name, bower.getName());

}

}

public void bowBack(Friend bower) {

System.out.format("%s: %s has" +

" bowed back to me!%n",

this.name, bower.getName());

}

}

static class BowLoop implements Runnable {

private Friend bower;

private Friend bowee;

public BowLoop(Friend bower, Friend bowee) {

this.bower = bower;

this.bowee = bowee;

}

public void run() {

Random random = new Random();

for (;;) {

try {

Thread.sleep(random.nextInt(10));

} catch (InterruptedException e) {}

bowee.bow(bower);

}

}

}

public static void main(String[] args) {

final Friend alphonse =

new Friend("Alphonse");

final Friend gaston =

new Friend("Gaston");

new Thread(new BowLoop(alphonse, gaston)).start();

new Thread(new BowLoop(gaston, alphonse)).start();

}

}

***原创文章，转载请注明：****转载自*[*并发编程网 – ifeve.com*](http://ifeve.com/)***本文链接地址:***[*Oracle官方并发教程之锁对象*](http://ifeve.com/lock-objects/)