**Java线程死锁demo**

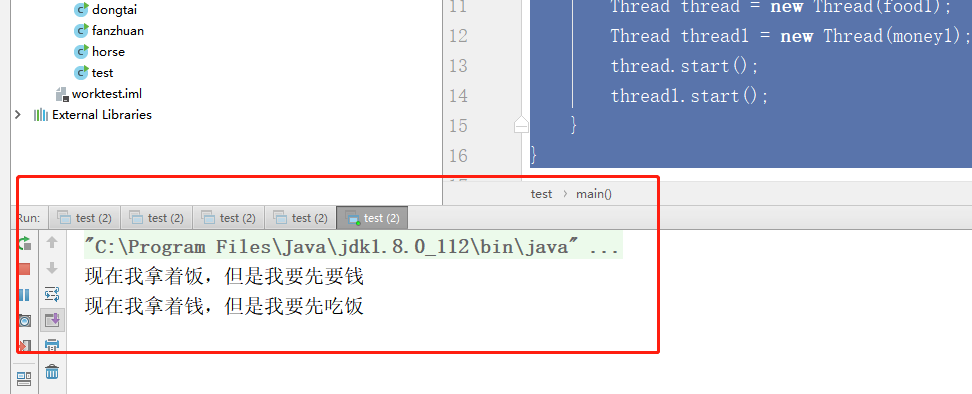
当线程中的同步代码有多个嵌套同步锁时容易发生死锁，下面的例子中2个线程公用2锁，一个拿着钱，要饭，一个拿着饭，要钱，互相不给，所以可能造成死锁现象：

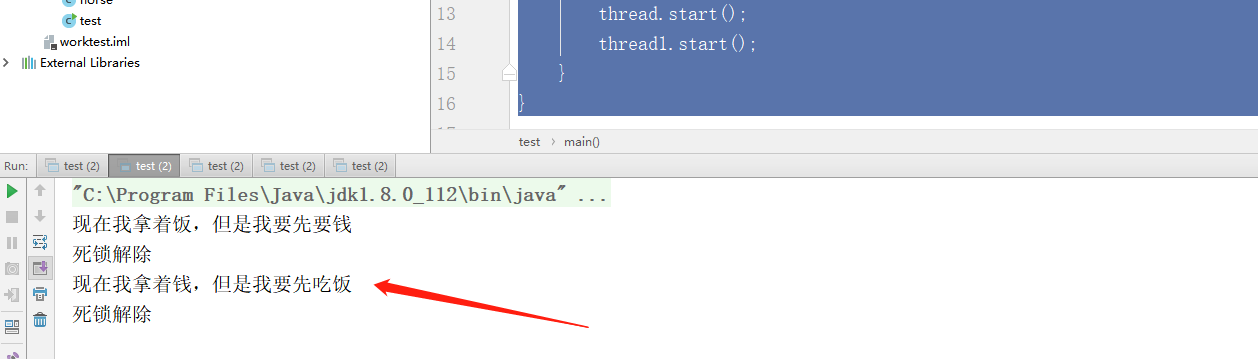
解决方法其实很简单，双方统一一下，先给饭，再要钱。这样就不会死锁了。

**package** dielock;  
  
**public class** Money **implements** Runnable{  
 **private** Object **money**;  
 **private** Object **food**;  
  
 **public** Money(Object money, Object food) {  
 **this**.**money** = money;  
 **this**.**food** = food;  
 }  
  
  
 @Override  
 **public void** run() {  
 **synchronized** (**money**) {  
 System.***out***.println(**"现在我拿着钱，但是我要先吃饭"**);  
 **try** {  
 Thread.*sleep*(0);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 **synchronized** (**food**) {  
 System.***out***.println(**"死锁解除"**);  
 }  
 }  
 }  
}

**package** dielock;  
  
**public class** Food **implements** Runnable{  
 **private** Object **money**;  
 **private** Object **food**;  
  
 **public** Food(Object money, Object food) {  
 **this**.**money** = money;  
 **this**.**food** = food;  
 }  
  
 @Override  
 **public void** run() {  
 **synchronized** (**food**) {  
 System.***out***.println(**"现在我拿着饭，但是我要先要钱"**);  
 **try** {  
 Thread.*sleep*(0);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 **synchronized** (**money**) {  
 System.***out***.println(**"死锁解除"**);  
 }  
 }  
 }  
  
  
}

**package** dielock;  
  
**public class** test {  
 **public static void** main(String[] args) {  
 Object money = **new** Object();  
 Object food = **new** Object();  
  
 Food food1 = **new** Food(money, food);  
 Money money1 = **new** Money(money, food);  
  
 Thread thread = **new** Thread(food1);  
 Thread thread1 = **new** Thread(money1);  
 thread.start();  
 thread1.start();  
 }  
}



可以看出锁住了，没有死锁解除的字样。但是有时候也可能会有，就是一个人同时拿到两个锁，弄完了释放2个资源给对方，如下：但是概率太小了。

**package** dielock;  
  
**public class** Money **implements** Runnable{  
 **private** Object **money**;  
 **private** Object **food**;  
  
 **public** Money(Object money, Object food) {  
 **this**.**money** = money;  
 **this**.**food** = food;  
 }  
  
  
 @Override  
 **public void** run() {  
 **synchronized** (**money**) {  
 System.***out***.println(**"现在我拿着钱，但是我要先吃饭"**);  
 **try** {  
 Thread.*sleep*(0);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 **synchronized** (**food**) {  
 System.***out***.println(**"饭吃完了"**);  
 }  
 }  
 }  
}

**package** dielock;  
  
**public class** Food **implements** Runnable{  
 **private** Object **money**;  
 **private** Object **food**;  
  
 **public** Food(Object money, Object food) {  
 **this**.**money** = money;  
 **this**.**food** = food;  
 }  
  
 @Override  
 **public void** run() {  
 **synchronized** (**money**) {  
 System.***out***.println(**"现在我拿着饭，但是可以给你先吃饭，再给钱"**);  
 **try** {  
 Thread.*sleep*(0);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 **synchronized** (**food**) {  
 System.***out***.println(**"收到钱了谢谢亲"**);  
 }  
 }  
 }  
  
  
}

**package** dielock;  
  
**public class** test {  
 **public static void** main(String[] args) {  
 Object money = **new** Object();  
 Object food = **new** Object();  
  
 Food food1 = **new** Food(money, food);  
 Money money1 = **new** Money(money, food);  
  
 Thread thread = **new** Thread(food1);  
 Thread thread1 = **new** Thread(money1);  
 thread1.start();  
 thread.start();  
  
 }  
}

