* **wait()与notify()和notifyAll()——在**
* **wait()**：令当前线程挂起并放弃CPU、同步资源，使别的线程可访问并修改共享资源，而当前线程排队等候再次对资源的访问。
* **notify()**：唤醒正在排队等待同步资源的线程中优先级最高者结束等待。
* **notifyAll()**：唤醒正在排队等待资源的所有线程结束等待。
* 注： Java.lang.Object提供的这三个方法只有在**synchronized方法或synchronized代码块中才能使用**，否则会报java.lang.IllegalMonitorStateException异常。
* **wait()方法**
* 在当前线程中调用方法：对象名.wait()。
* 使当前线程进入等待（某对象）状态，直到另一线程对该对象发出notify(或notifyAll)为止。
* 调用方法的必要条件：当前线程必须具有对该对象的监控权（加锁）。
* **调用此方法后，当前线程将释放对象监控权，然后进入等待**。
* 在当前线程被notify后，要重新获得监控权，然后从断点处继续代码的执行。
* **notify()/notifyAll()**
* 在当前线程中调用方法：对象名.notify()。
* 功能：唤醒等待该对象监控权的一个线程。
* 调用方法的必要条件：当前线程必须具有对该对象的监控权（加锁）。
* **线程通信 例题**

使用两个线程打印1-100.线程1,线程2交替打印。

public class TestCummunication {  
  
 public static void main(String[] args) {  
 PrintNum p = new PrintNum();  
 Thread t1 = new Thread(p);  
 Thread t2 = new Thread(p);  
  
 t1.setName("甲");  
 t2.setName("乙");  
  
 t1.start();  
 t2.start();  
 }  
}  
  
class PrintNum implements Runnable {  
 int num = 1;  
  
 @Override  
 public void run() {  
 while (true) {  
 synchronized (this) {  
 //唤醒  
 notify();  
  
 if (num <= 100) {  
 try {  
 Thread.currentThread().sleep(10);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.***out***.println(Thread.currentThread().getName() + +":" + num);  
 num++;  
 } else {  
 break;  
 }  
  
 //线程放弃对象锁  
 try {  
 wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 }  
 }  
 }  
}