**多线程案例分析一：**

设计四个线程对象，两个执行减法操作，两个执行加法操作。

|  |
| --- |
| package com.company;  /\*\*  \* @author Jackwen  \*/  public class AddAndSub {  public static void main(String[] args) {  Resource res = new Resource() ;  SubThread st = new SubThread(res) ;  AddThread at = new AddThread(res) ;  new Thread(at,"加法线程A").start();  new Thread(at,"加法线程B").start() ;  new Thread(st,"减法线程X").start() ;  new Thread(st,"减法线程Y").start() ;  }  }  /\*\*  \* 加法线程  \*/  class AddThread implements Runnable{  private Resource res ;  public AddThread(Resource res) {  this.res = res ;  }  @Override  public void run() {  for (int i = 0; i < 50; i++) {  try {  this.res.add();  } catch (Exception e) {  e.printStackTrace();  }  }  }  }  /\*\*  \* 减法线程  \*/  class SubThread implements Runnable{  private Resource res ;  public SubThread(Resource res) {  this.res = res ;  }  @Override  public void run() {  for (int i = 0; i < 50; i++) {  try {  this.res.subtraction();  } catch (Exception e) {  e.printStackTrace();  }  }  }  }  /\*\*  \* 线程访问资源  \*/  class Resource {  private int num = 0 ;  private boolean flag = true ;  //flag=true 表示执行加法  //flag=false 表示执行减法  public synchronized void add() throws Exception {  while(this.flag == false){  //线程等待  super.wait();  }  Thread.sleep(100);  this.num++ ;  System.out.println("执行加法操作，-"+Thread.currentThread().getName()+":num ="+ num);  //加法执行完成，执行减法操作  this.flag = false ;  super.notifyAll(); //唤醒线程  }  public synchronized void subtraction() throws Exception{  while(this.flag == true){  super.wait();  }  Thread.sleep(100);  num-- ;  System.out.println("执行减法操作，-"+Thread.currentThread().getName()+":num ="+ num);  this.flag = true ;  super.notifyAll();  }  } |