**生产电脑：**

设计一个生产电脑和搬运电脑类，要求生产出一台电脑就搬走一台电脑，如果没有新的电脑生脑产出出来，则搬运工要等待新的电脑产出;如果生产出的电脑没有搬走，则要等待电脑搬走之后再生产，并统计出生产的电脑数量。

本程序就是一个标准的生产者与消费者的模型：

具体实现：

|  |
| --- |
| package com.company;  /\*\*  \* @author JackWen  \*/  public class ComputerAndConsumer {  public static void main(String[] args) {  Computer com = new Computer() ;  ComputerProduce produce = new ComputerProduce(com);  Porter porter = new Porter(com) ;  new Thread(produce,"厂商").start();  new Thread(porter,"电脑搬运工").start();  }  }  /\*\*  \* 电脑产商  \*/  class ComputerProduce implements Runnable{  private Computer com ;  public ComputerProduce (Computer com ){  this.com = com ;  }  @Override  public void run() {  while (true){  try {  this.com.produce();  } catch (Exception e) {  e.printStackTrace();  }  }  }  }  /\*\*  \* 电脑搬运工  \*/  class Porter implements Runnable {  private Computer com ;  public Porter(Computer com){  this.com = com ;  }  @Override  public void run() {  while(true){  try {  this.com.proter();  } catch (Exception e) {  e.printStackTrace();  }  }  }  }  /\*\*  \* 电脑资源  \* @id:computerNumber  \* @flag:When it is true, it means that the computer can be produced, but it is not allowed to be transported.  \* When it is false, it means it can be transported but cannot be produced.  \*/  class Computer{  private int id = 0 ;  private boolean flag = true ;  public synchronized void produce() throws Exception{  if (this.flag == false){  //线程等待  super.wait();  }  Thread.sleep(100);  this.id++ ;  System.out.println(Thread.currentThread().getName()+"产商生产出"+this.id+"号电脑，请及时搬运");  this.flag = false ;  super.notify();  }  public synchronized void proter() throws Exception{  if(this.flag){  super.wait();  }  Thread.sleep(100);  System.out.println(id + "号电脑已经被"+Thread.currentThread().getName()+"搬出，请及时生产电脑");  this.flag = true ;  super.notify();  }  } |