这两个类使用起来非常方便，可以完成我们对定时器的绝大多数需求

Timer类是用来执行任务的类，它接受一个TimerTask做参数

Timer有两种执行任务的模式,最常用的是schedule,它可以以两种方式执行任务:1:在某个时间(Data)，2:在某个固定的时间之后(int delay).这两种方式都可以指定任务执行的频率

|  |
| --- |
| TimerTest.java:  [java] view plain copy  package com.cn;  import java.io.IOException;  import java.util.Timer;  public class TimerTest{  public static void main(String[] args){  Timer timer = new Timer();  timer.schedule(new MyTask(), 1000, 2000);//在1秒后执行此任务,每次间隔2秒执行一次,如果传递一个Data参数,就可以在某个固定的时间执行这个任务.  while(true){//这个是用来停止此任务的,否则就一直循环执行此任务  try{  int in = System.in.read();  if(in == 's'){  timer.cancel();//使用这个方法退出任务  break;  }  } catch (IOException e){  // TODO Auto-generated catch block  e.printStackTrace();  }  }  }    static class MyTask extends java.util.TimerTask{  public void run(){  System.out.println("\_\_\_\_\_\_\_\_");  }  }  } |

此类运行时：

程序启动1秒后在控制台打印“————”

间隔两秒后接着执行MyTask的run()方法，打印“————”

这样一直循环

当在控制台输入s字符时，timer定时器取消工作

跳出整个循环

程序运行结束！

|  |
| --- |
| TimerTest2.java:  [java] view plain copy  package com.cn;    import java.io.IOException;  import java.util.Date;  import java.util.Timer;    public class TimerTest2{    public static void main(String[] args){  Timer timer = new Timer();  MyTask myTask1 = new MyTask();  MyTask myTask2 = new MyTask();  myTask2.setInfo("myTask-info-2");    timer.schedule(myTask1, 1000, 2000); //任务1 一秒钟后执行，每两秒执行一次。  timer.scheduleAtFixedRate(myTask2, 2000, 3000); //任务2 2秒后开始进行重复的固定速率执行（3秒钟重复一次）    while (true){  try{  //用来接收键盘输入的字符串  byte[] info = new byte[1024];  int len = System.in.read(info);    String strInfo = new String(info, 0, len, "GBK");//从控制台读出信息    if (strInfo.charAt(strInfo.length() - 1) == ' '){  strInfo = strInfo.substring(0, strInfo.length() - 2);  }    if (strInfo.startsWith("Cancel-1")){  myTask1.cancel();//退出任务1  // 其实应该在这里判断myTask2是否也退出了,是的话就应该break.但是因为无法在包外得到  // myTask2的状态,所以,这里不能做出是否退出循环的判断.  } else if (strInfo.startsWith("Cancel-2")){  myTask2.cancel(); //退出任务2  } else if (strInfo.startsWith("Cancel-All")){  timer.cancel();//退出Timer  break;  } else{  // 只对myTask1作出判断,偷个懒^\_^  myTask1.setInfo(strInfo);  }  } catch (IOException e){  // TODO Auto-generated catch block  e.printStackTrace();  }  }  }    static class MyTask extends java.util.TimerTask{    String info = "INFO";    @Override  public void run(){  // TODO Auto-generated method stub  System.out.println(new Date() + " " + info);  }    public String getInfo(){  return info;  }  public void setInfo(String info){  this.info = info;  }  }    } |

此类创建了两个定时任务mytask1和mytask2

mytask1任务和上面的TimerTest类中的例子用法一样。即安排指定的任务从指定的延迟后开始进行重复的固定延迟执行。

mytask2任务就不同于上面的用法了，timer.scheduleAtFixedRate它是用的timer定时器的scheduleAtFixedRate()方法来执行。

scheduleAtFixedRate()方法在API1.6.0中是这样定义的：

安排指定的任务在指定的时间开始进行重复的固定速率执行。以近似固定的时间间隔（由指定的周期分隔）进行后续执行。

近似固定的时间间隔的意思是说：在固定速率执行中，相对于已安排的初始执行时间来安排每次执行。如果由于任何原因（如垃圾回收或其他后台活动）而延迟了某次执行，则将快速连续地出现两次或更多次执行，从而使后续执行能够赶上来。

Timer类的常用其他方法：

cancel()

终止此计时器，丢弃所有当前已安排的任务。

purge()

从此计时器的任务队列中移除所有已取消的任务。

schedule(TimerTask task, Date time)

安排在指定的时间执行指定的任务。

TimerTask类的常用其他方法：

cancel()

取消此计时器任务。

run()

此计时器任务要执行的操作。

scheduledExecutionTime()

返回此任务最近实际 执行的已安排 执行时间。