基于 JAVA 2 swing. Timer 类的 Swing 组件界面刷新的讨论

JAVA Timer class with the module of Swing

曹大有 （郧阳师范高等专科学校 计算机科学系 湖北丹江口 442700）

摘要：JAVA 从内核上支持多线程编程技术，在 JAVA 的图形用户界面编程中，Swing 组件库是 JAVA 2 中很重要的系统类库，由于 Swing 组件库中的很多方法，在多线程程序运行中表现出的不同步性，使得在多线程程序中对由 Swing 组件元素组成的图形用户界面的刷新常常出现内容与显示不同步的问题，进而引起图形用户界面的损坏。本文则探讨了如何用 swing 系统包中的 Timer 类来解决这个问题的方法。

关键词：多线程技术 Swing 刷新 同步

1 问题提出

在 J2SE 1.4 API 文档的 javax. swing. JTextArea. insert( ) 方法前有一句话：This method is thread safe, although most Swing methods are not。它的大意是：该方法在线程中运行是安全的，不过大多数 Swing 方法是不安全的。这说明大部分 Swing 组件并不是个能安全运行的线程，Swing 类的大多数方法并不是同步执行的。如果我们试图通过多线程程序对用户界面中的各个 Swing 组件进行界面的刷新操作时，我们的用户界面或 JAVA 控制台窗口将可能受到损坏，因此我们在工作线程中执行 Swing 组件的刷新操作时，我们应该非常谨慎。例如当我们运行附带程序 SwingThreadTest1. java 时，当我们单击“开始”按钮时，一个新线程将被启动，以便对 JComboBox 组合框对象进行编辑、随机添加和删除相应的值。其主要过程如下：

```
class BadWorkerThread extends Thread{ //类首部
    public BadWorkerThread(JComboBox aCombo)
{ //构造方法
        combo = aCombo; generator = new Random( );//给两个属性赋值
    }
    public void run( ){ //线程主体
        try{ while (! interrupted( )){ //若该线程未被中断
            int i = Math. abs ( generator. nextInt ( ));//生成一个随机数
            if (i % 2 = = 0)//若该随机数为偶数
                combo. insertItemAt ( new Integer ( i ), 0);//添入组合框
            else if (combo. getItemCount( ) > 0)
                combo. removeItemAt (i % combo. getItemCount( ));//从组合框中删除一项
            sleep(10);//休眠一段时间
        }}catch (Exception exception) {}}
    private JComboBox combo; private Random generator;//两个私有成员
}
```

当该程序运行一段时间后（可能需要几分钟），我们将会从控制台中看到一个偶尔发生的异常事件报告，此异常主要是由 ArrayIndexOutOfBounds 异常引起的。这是为什么呢？由于该程序主要是对组合框进行插入、删除操作，我们从 JComboBox 类的定义过程中可知：当一个元素被插入组合框时，该组合框便产生一个事件来更新显示窗口，接着激活显示代码，读取组合

框的当前大小，并且准备显示各个值。但是工作线程仍然在继续运行，它常常会导致组合框中项的数量的变化。这时显示代码会认为组合框中存在着比它实际上拥有的值更多的值，因此要求显示那些实际上并不存在的值，这样就导致异常 ArrayIndexOutOfBounds 的发生。这实际上是 JComboBox 组件的内容与显示不同步的现象。

这种现象不只在 JComboBox 组件中会发生，在大部分 Swing 组件中，都存在这种现象发生。如：想定期更新 JLabel 组件的显示内容（附带程序 SwingLabelThreadTest1. java）、在 JButton 组件中显示运动的标题（附带程序 SwingButtonThreadTest1. java）都会出现这种 Swing 组件的内容与显示不同步的现象发生。如何避免这种现象的发生呢？一种方法是在显示代码中将 Swing 对象锁定，但是我们从 Swing 的实现代码中可以看出，Swing 的设计者并没有采取这种策略，主要的原因可能是线程的同步是有代价的，这样做会降低 Swing 组件的运行速度，再者由于 Swing 工具包是个可以扩展的工具包，这样其他人员可以加入自己的元素，结果若加进同步要求往往会容易造成死锁现象的发生。所以 Swing 的设计者将这一任务的解决交给了程序员自己来处理。下面我们就来探讨如何通过 Swing 中 Timer 定时器类来解决这一问题。

2　问题的解决

在提出解决上述现象的方法之前，我们可以先查看一下 J2SE 1.4 java. awt. Component. repaint( ) 方法的具体实现过程。

```
public void repaint(long tm, int x, int y, int width, int height) {
    if (this.peer instanceof LightweightPeer) {
  if (parent != null) {//若组件的父组件不这空
   int px = this.x + ((x < 0) ? 0 : x); int py = this.y + ((y < 0) ? 0 : y);
int pwidth = (width > this.width) ? this.width : width;
int pheight = (height > this.height) ? this.height : height;//求出该组件的大小
parent.repaint(tm, px, py, pwidth, pheight);//由父组件负责绘制它
}} else {//没有父组件
  if (isVisible( ) && (this.peer != null) && (width > 0) && (height > 0)) {
      PaintEvent e = new PaintEvent(this, PaintEvent.UPDATE,
        new Rectangle(x, y, width, height));//生成绘制事件 PaintEvent
      Toolkit.getEventQueue( ).postEvent(e);//将事件插入事件队列 EventQueue
}}}
```

该方法对于每个需要重新绘制的 AWT 组件，若该组件有父组件，那么将由它的父组件负责它的绘制工作。若没有父组件，该方法用于安排一个公共的绘制刷新事件 PaintEvent，然后将该事件通过系统事件调度线程队列 EventQueue 发送处理。这说明 repaint( ) 对 AWT 组件的刷新是通过另一线程：事件调度线程队列 EventQueue 进行的。对 Swing 组件界面的刷新操作，我们即然不能在包含 Swing 组件的线程中进行，我们能否也采取上述方法的思路，让另一个线程（事件调度线程队列 EventQueue）来完成这项工作呢？为此我们先查看实现 javax. swing. Timer 类的 API 文档，从 javax. swing. Timer 类实现的 API 文档中，我们可以发现 javax. swing. Timer 类用于按照规定的时间间隔对我们程序中的元素进行报警。javax. swing. Timer 是一个很容易使用的内置定时器类。若要建立定时器对象，我们可以提供一个类的对象，用于实现 ActionListener 接口和定时器报警之间的延迟，该延迟以毫秒为单位。如：Timer t = new Timer(1000, listener)；

调用下面这个方法：t. start( )；

以便启动该定时器。然后，每当经过了定时器指定的时间间隔时，便调用 listener 类的 actionPerformed( ) 方法。但是 actionPerformed( ) 方法是在事件调度线程队列 EventQueue 上按照指定的时间间隔自动调用的（这一点相当重要），而不是在定时器线程上调用的，因此我们也可以在该回调中随意调用 Swing 组件的方法。

若要停止定时器的工作，请调用下面这个方法：t. stop( )；

这时定时器便停止发送动作事件，直到重新启动定时器为止。

实际上，我们通过分析 javax. swing. Timer 类的 API 文档，可以发现 listener 类的 actionPerformed( ) 方法，是通过 EventQueue 类的 invokeLater( ) 方法执行的，该方法将动作事件移植到事件调度线程上去后，invokeLater( ) 方法便立即返回，相应的线程以异步的方式来执行。当然在使用时，要注意 SDK 1. 3 中配有一个不相关的 java. util. Timer 类，用于安排一个 TimerTask 类供以后执行。该 TimerTask 类用于实现 Runnable 接口，并且还提供一个 cancel( ) 方法，用于撤消该任务。不过 java. util. Timer 类不具备定期回调的功能。下面我们就来用 javax. swing. Timer 类解决 Swing 组件的刷新问题。

对于上面组合框的显示问题，我们可以重新构建下面一个类（详细见附带程序 SwingThreadTest2. java）：

```
class GoodWorkerThread extends Thread{
public GoodWorkerThread(JComboBox aCombo){
  combo = aCombo;generator = new Random
();}
public void run(){
 new Timer(1,new ActionListener(){
   public void actionPerformed(ActionEvent e-
vent){
     int i = Math.abs(generator.nextInt());
      if(i % 2 == 0)combo.insertItemAt(new
Integer(i), 0);
       else if(combo.getItemCount() > 0)
         combo.removeItemAt(i % combo.get-
ItemCount());}}).start();}
   private JComboBox combo; private Random
generator;}
```

该程序无论运行多长时间，在控制台窗口中都不会出现任何异常现象。对于定期更新 JLabel 组件的显示内容问题，我们构建下面的类（详细见附带程序 SwingLabelThreadTest2. java）：

```
class LabelUpdater extends Thread{
public LabelUpdater(JLabel aLabel, int aPercent-
age){
   label = aLabel;percentage = aPercentage;}
public void run(){ new Timer(10,new ActionListen-
er(){
     public void actionPerformed(ActionEvent e-
vent){
       label.setText(percentage + "% Com-
plete");}}).start();}
   private JLabel label;
   private int percentage;}
```

这样就可以在 JLabel 组件中定期更新显示内容。对于在 JButton 组件中显示运动的标题（详细见附带程序 SwingButtonThreadTest2. java），我们构建的类为：

```
class ButtonUpdater extends Thread{
 public ButtonUpdater(JButton aButton,int aPercent-
age){
   button = aButton;percentage = aPercentage;}
 public void run(){ new Timer(10,new ActionListener
(){
     public void actionPerformed(ActionEvent e-
vent){
       button.setText(percentage + "% Com-
plete");}}).start();}
   private JButton button;private int percentage;}
```

它不但能在 JButton 组件中显示运动的标题，而且也不影响 JButton 组件对其它事件的响应（如单击事件）。以上说明我们用 javax. swing. Timer 类，完全可以解决 Swing 组件的界面刷新问题。

当然，由于 javax. swing. Timer 类是通过定时间隔发送动作事件，在这种情况下我们可以不需要进行任何的线程编程，javax. swing. Timer 类对象将负责线程的具体执行。所以我们也可以通过 javax. swing. Timer 类来代替 java. Thread 类的部分工作，比如实现动画，动画显示一般是通过线程编程来实现的。在我们程序后附带的程序 Animation. java 中，我们就通过以下程序段在小程序中实现动画显示。

```
runner = new Timer(200,new ActionListener(){
  public void actionPerformed(ActionEvent event)
{
    repaint();current = (current + 1) % im-
ageCount;}});
runner.start();
```

（下转第 106 页）

(上接第 102 页)

3 总结

从上面的讨论,我们可以得出以下结论:当我们将多线程编程技术与 Swing 组件一起使用时,在多线程程序中不能接触 Swing 组件的界面,即对 Swing 组件不能进行界面刷新工作,这通常被称为 Swing 编程的单线程原则。对 Swing 组件界面的刷新工作应该通过事件调度线程来进行,而且这也不影响对 Swing 组件进行其它工作。当然我们不能说所有的 Swing 组件的方法在线程中都是不安全的,在 JAVA 中有少数几个 Swing 组件的方法是线程安全,它们是:

JtextComponent 的 setText( ) 方法,JtextArea 的 insert( ) 方法,JtextArea 的 append( ) 方法,JtextArea 的 replaceRange( ) 方法。以上程序是在 J2SE 1.4 下调试运行的。

参考文献

1 John Zukowski(美)著,邱仲潘译,Java 2 从入门到精通[M],北京:电子工业出版社,2005.

2 Cay S. Horstmann、Gary Cornell(美)著,京京工作室译,Java 2 核心 卷 I :基础知识[M],北京:机械工业出版社,2000.