第十章 常用组件GUI设计

主要内容

1. 布局管理器
2. 窗口与面板容器
3. 事件响应原理
4. JLabel组件
5. JButton组件与JToggleButton组件
6. JcheckBox和JRadioButton组件
7. JComboBox组件
8. JList组件
9. JTextField与JTextArea组件

1 界面布局管理

- Java.awt包中共定义了五种布局管理类：

  (1) BorderLayout

  (2) FlowLayout

  (3) CardLayout

  (4) GridLayout

  (5) GridBagLayout

1 界面布局管理

- Java.Swing包中共定义了四种布局管理器：

  (1) BoxLayout

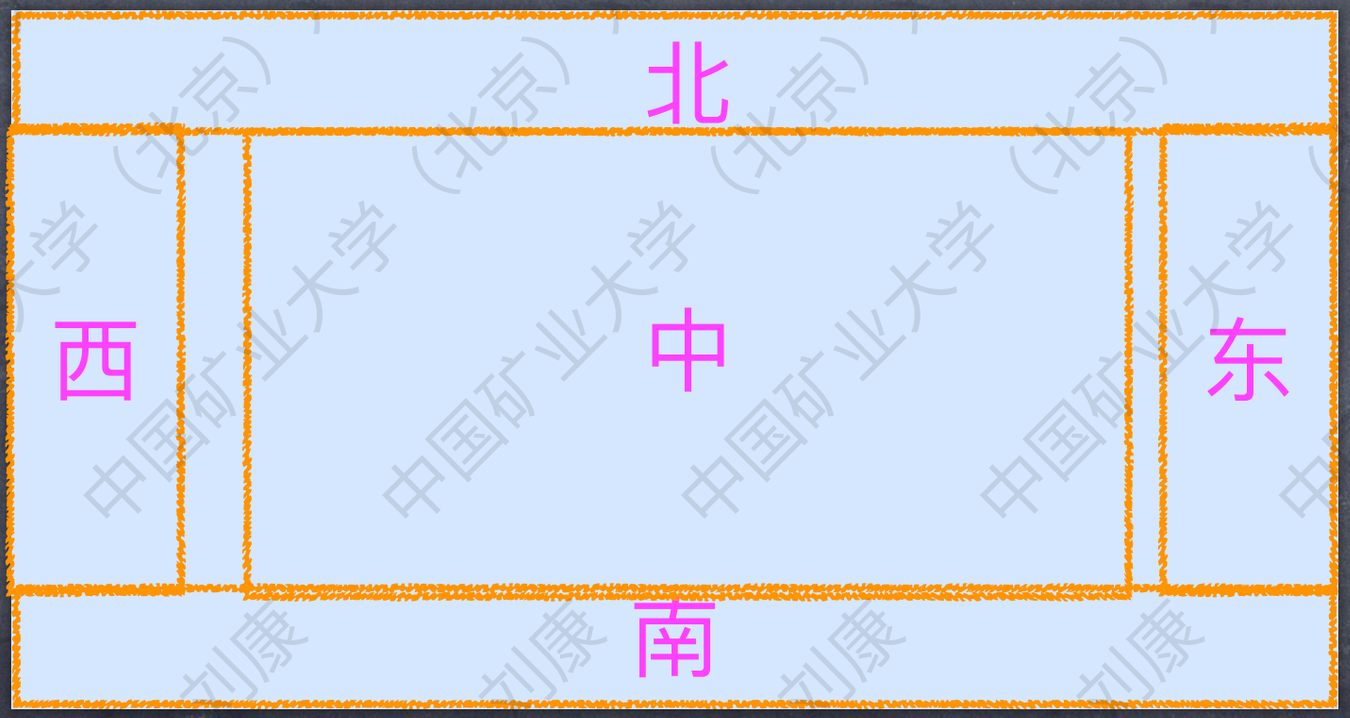
  (2) ScrollPaneLayout

  (3) ViewportLayout

  (4) OverlayLayout

1.1 BorderLayout

- 布局策略：把容器空间划分为东、西、南、北、中五个区域。



- 布局理念：向容器里每加入一个组件，都应指明把它放在容器的哪个区域中。如果某个区域没有分配组件，则其他组件可以占据它的空间。缺省布局策略。

- BorderLayout类有两个构造方法：

```
BorderLayout();
```

创建各组件间水平、垂直间隔为0的类对象。

```
BorderLayout(int hgap, int vgap);
```

创建各组件间的水平间隔为hgap，垂直间隔为vgap的类对象。

BorderLayout以Center作为默认值。

1.2 FlowLayout

- 布局理念

  容器中的组件按照加入的先后顺序从左向右排列，当一行排满之后就转到下一行继续从左至右排列，每一行的组件都居中排列。

- FlowLayout类有三种构造方法：

FlowLayout();

创建一个版面设定为居中对齐，各组件的水平及垂直间隔为5个像素点的类对象。

FlowLayout(int align);

创建一个版面，按给定的align值对齐，各组件的水平及垂直间隔为5个像素点。align值可为FlowLayout.LEFT(1)、RIGHT(2)、CENTER(3)。

FlowLayout(int align, int hgap, int vgap);

创建一个既指定对齐方式，又指定组件间间隔的FlowLayout类的对象。参数align作用同上；参数hgap指定组件间的水平间隔，参数vgap指定各组件间的垂直间隔。间隔单位为像素点。

1.3 CardLayout

- 布局理念

将每个组件看成一张卡片，如同扑克牌一样将组件堆叠起来，而在屏幕上只显示最上面的组件，被显示的组件将占据所有的容器空间。

当容器第一次显示时，第一个添加到CardLayout对象的组件为可见组件。

- CardLayout类有两个构造方法：

```
CardLayout();
```

使用默认间隔(间隔为0)，创建类对象。

```
CardLayout(int hgap,  int vgap);
```

使用hgap指定的水平间隔和vgap指定的垂直间隔创建类对象。

成员方法	说明
first (Container container)	显示container中第一个对象
last (Container container)	显示container中最后一个对象
next (Container container)	显示下一个对象
previous (Container container)	显示上一个对象

1.4 GridLayout

- 如果界面上需要放置的组件较多，且大小基本一致时，使用GridLayout布局是最佳选择。

- 布局理念

  把容器的空间划分为若干行、若干列的网格区域，每个组件按添加的顺序从左向右，从上向下占据网格。

- GridLayout类的三个构造方法：

GridLayout();

按默认(1行1列)方式创建一个GridLayout布局。

GridLayout(int rows, int cols);

创建一个具有rows行，cols列的GridLayout布局。

GridLayout(int rows, int cols, int hgap, int vgap);

按指定的行数rows，列数cols，水平间隔hgap和垂直间隔vgap创建一个GridLayout布局。

1.5 BoxLayout

- BoxLayout是Swing提供的布局管理器，继承关系如下：

  java.lang.Object

  javax.swing.BoxLayout

- BoxLayout只有两种排列方式：水平、垂直。使用BoxLayout提供的两个常量：X_AXIS、Y_AXIS指明。

- BoxLayout构造函数

```
BoxLayout(Container target, int axis);
```

target是容器对象；

axis指明target中组件的排列方式

- 水平排列(BoxLayout.X_AXIS)
- 垂直排列(BoxLayout.Y_AXIS)

- Box默认的布局是BoxLayout，且只能使用这个布局，否则编译会产生错误。
- Box构造函数：

```
Box(int axis);
```

构造函数只有一个参数axis，用于指定Box中组件的排列。

BoxLayout类常用成员方法

成员方法	说明
getLayoutAlignmentX (Container con)	返回值表示Container中对象X轴方向的对齐方式
getLayoutAlignmentY (Container con)	返回值表示Container中对象Y轴方向的对齐方式
setLayoutAlignmentX (Container con)	设置Container对象X轴方向的对齐方式
setLayoutAlignmentY (Container con)	设置Container对象Y轴方向的对齐方式
LayoutContainer (Container target)	设置target窗口容器的布局方式为BoxLayout

Box类常用的成员方法

成员方法	说明
Box.createHorizontalBox()	构造水平排列的Box组件
Box.createVerticalBox()	构造垂直排列的Box组件
Component.creatGlue()	构造可向水平与垂直方向延伸的Glue组件
Component.creatHorizontalGlue()	构造水平的Glue组件
Component.creatVerticalGlue()	构造垂直的Glue组件
Component.creatHorizontaStrut(int width)	构造水平的Strut组件
Component.creatVerticalStrut(int height)	构造垂直的Strut组件
Component.creatRigidArea(Dimension d)	构造Rigid组件
AecessibleContext.getAccessibleContext()	取得与JComponent相关联的AccessibleContext
setLayout(LayoutManager l)	抛出AWTError，Box只是用BoxLayout布局


bBox 容器
vBox1 容器
这是标签 1
这是按钮 1
vBox2 容器
这是文本框
vbox2h 容器
vBox2h1 容器
这是文本区域
vBox2h2 容器
这是按钮 2
这是按钮 4

- 使用BoxLayout的注意事项：

(1) 当组件按Boxlayout布局后，不管窗口缩小或放大都不会变动。

(2) 当使用水平排列方式时，若放置的组件不等高，系统将使所有的组件与最高组件等高。

(3) 当放在同一行的组件超出容器的宽度时，系统不会自动换行，需用户自行处理。

2 窗口与面板

2.1 JFrame容器

1）JFrame是Java Application程序的图形界面容器，是带有标题和边框的顶层窗口。默认布局为BorderLayout。

2）JFrame类包含支持任何通用的窗口特性的基本功能，如最小化窗口、移动窗口。

3）JFrame容器作为最顶层容器，不能被其他容器所包含，但可被其他容器创建，并弹出成为独立的容器。

JFrame的继承关系：

java.lang.Object

java.awt.Component

java.awt.Container

java.awt.Window

java.awt.Frame

javax.swing.JFrame

- 使用JFrame注意以下几点：

(1) 可使用JFrame()方法创建一个无标题的JFrame对象，也可以使用JFrame(String title)方法创建一个标签为title的JFrame对象。

(2) 使用setSize(int x, int y)设置JFrame容器大小

(3) 默认创建的JFrame不可见，用setVisible(true)显示。

(4)添加组件时，先取得ContentPane，然后再使用add()方法把组件加入到此ContentPane中，而不能像AWT中的Frame那样直接调用add()方法。

(5)每个JFrame在其右上角都有三个控制图标：最小化、最大化和关闭窗口的操作。

JFrame自动完成窗口的最小化和最大化操作，但关闭窗口的操作不能通过点击关闭图标来实现。

使用下述三个办法之一关闭窗口：

1)设置一个按钮，当用户点击此按钮时关闭窗口

2)WINDOWS_CLOSING事件做出响应，关闭窗口

3)使用菜单命令

无论使用哪一种办法，都需要用到System.exit(0)

2 窗口与面板

- 在设计用户界面时，为了更合理的安排各组件在窗口的位置，可以考虑将所需组件先排列在一个容器中，然后将其作为一个整体嵌入窗口。
- JPanel的继承关系：

java.lang.Object

  java.awt.Component

    java.awt.Container

      javax.swing.JComponent

        javax.swing.JPanel

使用JPanel的注意事项：

1. 不能把JPanel作为图形界面最顶层的容器，也不能指明JPanel大小。
2. JPanel总是作为一个容器组件被加入到JFrame、JApplet等顶层容器中，JPanel也可以加入到JPanel容器中。
3. JPanel的大小由包含的组件，容器的布局策略，以及容器中其他组件共同决定。

3 事件响应原理

- 设计和实现图形用户界面的工作有两个：

1. 创建组成界面的各种成分和元素，指定它们的属性和位置关系。

2. 定义图形界面的事件和各界面元素对不同事件的响应，实现图形界面与用户交互。

3.1 委托事件模型

- Java采用委托事件模型来处理事件，将事件的处理委托给独立的对象，而不是组件本身，从而将使用者界面与程序逻辑分开。
- 整个"委托事件模型"由产生事件的对象(事件源)，及事件源与监听者对象之间的关系所组成。

- 委托事件模型实现的步骤：

1. 建立事件源对象
2. 为事件源对象选择事件监听器
3. 为监听器添加处理方法
4. 建立监听器与事件源之间的关系

3.2 Swing事件及监听者

- 不同事件源上发生的事件种类不同，不同事件由不同的监听者处理。
- 每个事件监听者都有相应的成员方法，处理事件的代码需要写在对应的成员方法体中。

4 JLabel组件

- JLable组件称为标签，它是一个静态组件，也是标准组件中最简单的一个组件。
- 每个标签用一个标签类的对象表示，可以显示一行静态文本。标签只起信息说明的作用，而不接受用户的输入，也无事件响应。

构造方法	功能机参数说明
JLabel()	创建一个空标签
JLabel(Icon icon)	创建图标为icon的标签
JLabel(Icon icon, int halig)	创建图标为icon标签，指定水平排列方式（LEFT,CENTER,RIGHT,LEADING,TRAILING）
JLabel(String text)	创建一个含有文字的标签
JLabel(String text, int halig)	创建一个含有文字的标签，并指定排列方式
JLabel(String text, Icon icon, int halig)	创建一个含有文字及图标的标签，并指定水平排列方式

成员方法	功能说明
Icon getIcon()	获取此标签图标
void setIcon()	设置标签的图标
String getText()	获取此标签的文本
void setText(String label)	设置标签的文本
void setHorizontalAlignment(int alig)	设置标签内组件的水平对齐方式
void setVerticalAlignment(int alig)	设置标签内组件垂直对齐方式
void setHorizontalTextPosition(int tp)	设置标签内文字与图标水平相对位置
void setVerticalTextPosition(int tp)	设置标签内文字与图标垂直相对位置

5 JButton、JToggleButton

- JButton组件与JToggleButton组件通常被称为按钮，它是一个具有按下、抬起两种状态的组件。
- 用户可以指定按下按钮(单击事件)时所执行的操作(事件响应)。
- 按钮上通常有一行文字(标签)或一个图标以表明它的功能。

Swing组件中的按钮还可以实现下述效果：

(1) 改变按钮的图标，即一个按钮可以有多个图标，可根据Swing按钮所处的状态而自动变换不同的图标。

(2) 为按钮加入提示，即当鼠标在按钮上稍做停留时，在按钮边可出现提示，当鼠标移出按钮时，提示自动消失

(3) 在按钮上设置快捷键

(4)设置默认按钮，即通过回车键运行此按钮的功能

5.1 JButton类构造方法

构造方法	功能说明
JButton()	创建一个无标签的按钮
JButton(String text)	创建一个有标签的按钮
JButton(Icon icon)	创建一个有图标的按钮
JButton(String text, Icon icon)	创建一个有标签和图标的按钮

5.2 JToggleButton构造方法

- JToggleButton与JButton的区别：
  - 当按下JButton按钮并释放鼠标后，按钮会自动弹起；
  - 按下JToggleButton按钮并释放鼠标后，按钮不会自动弹起，除非再按一次。

构造方法	功能说明
JToggleButton()	创建一个无标签的按钮
JToggleButton(String text)	创建一个有标签的按钮
JToggleButton(String text, boolean selected)	创建一个有标签的按钮，初始状态为false
JToggleButton(Icon icon)	创建一个图标为icon的按钮
JToggleButton(Icon icon, boolean selected)	创建一个有图标的按钮，初始状态为false
JToggleButton(String text, Icon icon)	创建一个既有标签又有图标的按钮
JToggleButton(String text, Icon icon, boolean selected)	创建一个既有标签又有图标的按钮，初始状态为false

5.3 ActionEvent事件

- 按照Java的委托事件模型，按下按钮时会激发一个事件，称为动作事件。动作事件由AWT的ActionEvent类的方法处理。

1. 动作事件

ActionEvent类含有ACTION_PERFORMED事件，引发某个动作的执行事件。包括：单击按钮、双击列表选项、在文本框输入回车等。

2. ActionEvent类使用的主要方法

获取引发事件的对象名：

```
getSource();
```

获取对象的标签或为对象设置的命令名：

```
getActionCommand();
```

3、 事件响应

当用户点击对象时，会引发ActionEvent类代表的动作事件。

对象名.addActionListener(this)

注意事项：

注册事件源对象的监听者对象为this，要求this对象的类声明并实现ActionListener接口。

事件监听者引用actionPerformed(ActionEvent e)响应动作事件。在此方法中写入处理事件的代码。

6 JCheckBox、JRadioButton

- JCheckBox组件被称为复选框(也称检测框)，提供“选中”和“未选中”两种状态，实现按钮复选，按钮为方形图标。

- JRadioButton组件被称为选项按钮，该组件是单选按钮，按钮为圆形图标。

6.1 JCheckBox构造方法

构造方法	功能说明
JCheckBox()	创建一个无标签复选框对象
JCheckBox(String text)	创建一个有标签复选框对象
JCheckBox(String text, boolean selected)	创建一个有标签复选框对象，初始状态为false
JCheckBox(Icon icon)	创建一个有图标复选框对象
JCheckBox(Icon icon, boolean selected)	创建一个有图标复选框对象，初始状态为false
JCheckBox(String text, Icon icon)	创建一个标签和图标的复选框对象
JCheckBox(String text, Icon icon, boolean selected)	创建一个标签和图标的复选框对象，初始状态为false

6.2 JRadioButton构造方法

构造方法	功能说明
JRadioButton()	创建一个无标签的对象
JRadioButton(String text)	创建一个有标签的对象
JRadioButton(String text, boolean selected)	创建一个有标签的对象，且初始状态为false
JRadioButton(Icon icon)	创建一个有图标的对象
JRadioButton(Icon icon, boolean selected)	创建一个有图标的对象，且初始状态为false
JRadioButton(String text, Icon icon)	创建一个有标签和图标的对象
JRadioButton(String text, Icon icon, boolean selected)	创建一个有标签和图标的对象，且初始状态为false

6.3 ItemEvent事件

1、 选择事件

ItemEvent类只包含一个事件ITEM_STATE_CHANGED，引发这类事件的动作包括：

1）改变复选框JCheckbox对象的选中或不选中状态；

2）改变单选按钮JRadioButton对象的选中或不选中状态；

3）改变下拉列表框JComboBox对象中选项的选中或不选中状态；

4)改变菜单项JMenuItem对象中选项的选中或不选中状态；

5）改变JCheckboxMenuItem对象中选项的选中或不选中状态。

2. ItemEvent类主要方法有三个：

(1) ItemSelectable getItemSelectable( )

返回引发选中状态变化的事件源。

(2) Object getItem( )

返回引发选中状态变化事件的具体选择项。

(3) int getStateChange( )

返回此组件到底有没有被选中。它的返回值是一个整型值，通常用ItemEvent类的静态常量SELECTED(代表被选中)和DESELECTED(代表选项被放弃或不选)来表达。

3、事件响应

当用户点击对象使其选中状态发生变化时，引发ItemEvent类代表的选择事件。

对象名.addItemListener(this);

1）注册事件源对象的监听者对象为this，而且this对象的类必须声明该类实现ItemListener接口；

2）引发的事件将被此事件的监听者监听到；

3）引用ItemListener中的itemStateChanged(ItemEvent e)方法响应对象的状态改变。

在方法体中，引用ItemEvent事件的e.getItemSelectable( )方法获得引发事件源对象，再引用getStateChange( )方法获取选择事件之后的状态。

7 JComboBox

7.1 JComboBox构造方法和成员方法

构造方法	说明
JComboBox(Vector items)	使用向量表items构造一个JComboBox对象
JComboBox()	构造空对象，可使用addItem添加选项
JComboBox(ComboBoxModel aModel)	从已有aModel获取选项构造
JComboBox(Object[] items)	使用数组构造

成员方法	说明
void addActionListener(ActionListener e)	添加指定的ActionListener
void addItemListener(ItemListener aListener)	添加指定的ItemListener
void addItem(Object anObject)	给选项表添加选项
String getActionCommand()	获取动作命令
Object getItemAt(int index)	获取指定小标的列表项
int getItemCount()	获取列表中的选项数
int getSelectedIndex()	获取当前选择的下标
int getSelectedItem()	获取当前选择的项

7.2 事件响应

- JComboBox能够响应的事件为选择事件与动作事件。

1. 若用户选取下拉列表中的选择项时，则激发选择事件，使用ItemListener事件监听者进行处理；

2. 若用户在JComboBox上直接输入选择项并回车时，则激发动作事件，使用ActionListener事件监听者进行处理。

8 JList

- JList称为列表组件，将所有选项放入列表框中。如果将JList放入滚动面板(JScrollPane)中，则会出现滚动菜单效果。
- 利用JList提供的成员方法，用户可以指定显示在列表框中的选项个数，而多余的选项则可通过列表的上下滚动来显现。

- JList组件与JComboBox组件的区别

  1. JComboBox组件一次只能选择一项。

  2. JList组件一次可以选择一项或多项。

     选择多项时可以是连续区间选择(按住Shift键进行选择)，也可以是不连续的选择(按住Ctrl键进行选择)。

8.1 JList构造方法

构造方法	说明
JList(Vector listData)	使用包含元素的向量构造对象
JList()	构造空对象
JList(ListModel dataModel)	使用dataModel构造对象
JList(Object[] listData)	使用指定数组构造对象

8.2 JList成员方法

成员方法	说明
void addListSelectionListener (ListSelectionListener e)	添加指定的ListSelectionListener
int getSelectedIndex()	获取所选项的第一个下标
int getSelectedIndices()	获取所有选项的下标
void setSelection Background(Color c)	设置单元格的背景颜色
void setSelection Foreground(Color c)	设置单元格的前景颜色
int getVisibleRowCount()	获取可见的选项值
void setVisibleRowCount(int num)	设置可见的列表选项

8.3 ListSelectionEvent事件

- JList组件的事件处理一般可分为两种：
  1. 当用户单击列表框中的某个选项并选中它时，将产生ListSelectionEvent类的选择事件，此事件是Swing的事件；
  2. 当用户双击列表框中的某个选项时，则产生MouseEvent类的动作事件。JList类通过LocatToindex()方法来得知是单击还是双击。

9 JTextField、JTextArea

- JtextField被称为文本框。

  定义了一个单行条形文本区，可以输出任何基于文本的信息，也可以接受用户的输入。

- JTextArea被称为文本域。

  文本域可以输入/输出多行文本。

9.1 JTextField构造方法

构造方法	说明
JTextField()	创建一个JTextField对象
JTextField(int n)	创建列宽为n的空对象
JTextField(String s)	创建对象，并显示字符串s
JTextField(String s, int n)	创建对象，并指定字宽n显示字符串s
JTextField(Document doc, String s, int n)	使用指定文件存储模式创建对象，并以指定的字宽n显示字符串s

9.2 成员方法

成员方法	说明
int getColumns()	获取此对象列数
void setColumns(int Columns)	设置此对象列数
void addActionListener (ActionEvent e)	添加指定的动作事件监听程序
void setFont(Font f)	设置字体
void setHorizontalAlignment(int alig)	设置文本的水平对齐方式
void setActionCommand(String com)	设置动作事件使用的命令字符串

9.3 JTextArea构造方法

构造方法	功能说明
JTextArea()	创建空对象
JTextArea(int n, int m)	创建具有n行m列的空对象
JTextArea(String s)	创建显示字符串s的对象
JTextArea(String s, int n, int m)	创建以指定n行m列显示字符串s对象
JTextArea(String s, int n, int m, int k)	创建以指定n行m列和滚动条方向显示字符串s的对象
JTextArea(Document doc)	使用指定文件存储模式创建对象
JTextArea(Document doc, String s, int n)	使用指定文件存储模式创建对象，并以字宽n显示字符串s

9.4 成员方法

成员方法	说明
void setFont(Font f)	设置字体
void insert(String str, int pos)	在指定位置插入指定文本
void append(String str)	将指定文本添加到末尾
void replaceRange(String str, int start, int end)	将指定范围内的文本用指定的新文本替换
public int getRows()	返回此对象的行数
public void setRows(int rows)	设置此对象的行数
public int getColumns()	返回此对象的列数
public void setColumns(int Columns)	设置此对象的行数

9.5 事件处理

- JTextField只引发ActionEvent事件，当用户在文本框中按回车键时引发。
- JTextArea的事件响应由JTextComponent类决定。

可以引发两种事件：DocumentEvent事件与UndoableEditEvent事件。

1. 当用户修改了文本区域中的文本，如做文本的增、删、改等操作时，引发DocumentEvent事件；
2. 当用户在文本区域上撤消所做的增、删、改时，引发UndoableEditEvent事件。

(1) 实现文本框的ActionEvent事件

当监听者对象的类声明实现ActionListener接口，并通过addActionListener(this)注册文本框的监听者对象，监听程序内部的actionPerformed(ActionEvent e)响应动作事件。

(2)实现文本区域的DocumentEvent事件

当监听者对象的类声明实现DocumentEditListener接口，并通过textArea.addDocumentListener(this)语句注册文本区域的监听者对象，监听程序内部的DocumentListener接口的下列三个方法之一响应DocumentEvent事件。

```
changedUpdate(DocumentEvent e); //修改文本操作
insert Update(DocumentEvent e);   //增加文本操作
remove Update(DocumentEvent e);  //删除文本操作
```

(3) 实现文本区域的UndoableEditEvent事件

当监听者对象的类声明实现UndoableEditListener，并通过textArea.addUndoableEditListener(this)语句注册文本区域监听者对象，监听程序内部的UndoableEditListener接口的UndoableEditHappened(UndoableEditEvent e)方法响应UndoableEditEvent事件。