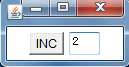
Java实验报告十

10.1

（1）使用AWT实现的代码如下：

|  |
| --- |
| import java.awt.\*;  import java.awt.event.\*;  public class JustForTest {  public static void main(String[] args) {  Frame frame = new Frame("");  frame.setSize(100,50);  // Panel pane = new Panel(); //创建一个面板  frame.setLayout(new FlowLayout()); //设置布局管理器  // frame.add(pane); //将面板添加上去  frame.addWindowListener(new WindowAdapter(){  public void windowClosing(WindowEvent e){  System.exit(0);  }//此处开始的时候没有加上该步骤，导致关闭失败  });//AWT的实现方法和SWING不同，所以需要进行监听  frame.add(new Button("INC"));  frame.add(new TextField("2"));  frame.pack();  frame.setVisible(true);  }  } |

以下是截图：

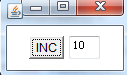
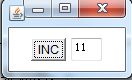
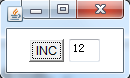


总结的问题就是两点：首先是没有添加那个监听器实现默认的关闭动作，于是第一次实验的时候点击关闭按钮没有反应。然后就是开始的时候添加了一个面板，然后再frame中添加的按钮和其他，导致了显示不正常。改正后就可以显示了。这里改正的办法是干脆不用panel，直接在frame中添加，可以成功。后来测试的时候，我还是用了panel，然后后面的代码就要改成panel.add，添加按钮和编辑框，这样也是可以正常显示的（测试过了）。于是用AWT实现就完成了。

（2）实现监听后的代码如下：

|  |
| --- |
| **import** java.awt.\*;  **import** java.awt.event.\*;  **public** **class** JustForTest {  **public** Frame frame;  Panel pane;  Button button;  TextField text;  **public** **static** **int** *num* = 0; //设置显示的数字的变量    **public** JustForTest(){  frame = **new** Frame("");  frame.setSize(100,50);  pane = **new** Panel(); //创建一个面板  frame.setLayout(**new** FlowLayout()); //设置布局管理器  frame.add(pane); //将面板添加上去  frame.addWindowListener(**new** WindowAdapter(){  **public** **void** windowClosing(WindowEvent e){  System.*exit*(0);  }//此处开始的时候没有加上该步骤，导致关闭失败  });//AWT的实现方法和SWING不同，所以需要进行监听  button = **new** Button("INC");  text = **new** TextField("0");  pane.add(button);  pane.add(text);  button.addActionListener(**new** ActionListener(){  **public** **void** actionPerformed(ActionEvent e){  ++*num*;  String newString = String.*valueOf*(*num*);  text.setText(newString);  }  });  frame.pack();  frame.setVisible(**true**);  }  **public** **static** **void** main(String[] args) {  **new** JustForTest();  }  } |

实现监听之后响应响应的动作，运行结果如图：

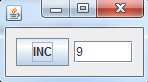
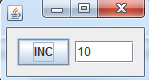
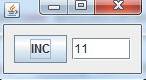
  

总结：这个代码较上面一个有很大的改变，虽然只是添加了一个监听的动作，但是由于需要相关的改变文本框的显示数据，所以费了些功夫。首先我还是按照原来的方法放在main中，不过遇到了问题，就是设置的这个num必须为final，因为是在类方法中，所以碰到这个问题，结合一起的经验，我干脆将这些代码放在构造函数中，然后再main中仅仅创建一个实例就可以了。至于对类成员的操作，都在里面进行，而且还将各个组件的声明放在外部作为成员，这样实现了该功能。由于全部是AWT组件，所以都是一些Button，Frame等等的类，不过基本的思想都是差不多。

（3）利用SWING实现如下：

|  |
| --- |
| **import** java.awt.\*;  **import** java.awt.event.\*;  **import** javax.swing.\*;  **public** **class** JustForTest {  **public** JFrame frame;  JPanel pane;  JButton button;  JTextField text;  **public** **static** **int** *num* = 0; //设置显示的数字的变量  **public** JustForTest(){  frame = **new** JFrame("");  frame.setSize(100,50);  pane = **new** JPanel(); //创建一个面板  frame.setLayout(**new** FlowLayout()); //设置布局管理器  frame.add(pane); //将面板添加上去  frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*); //对于SWING直接调用这个函数就可以了  button = **new** JButton("INC");  text = **new** JTextField("0",5);//为了让文本框好看而设置了长度  pane.add(button);  pane.add(text);  button.addActionListener(**new** ActionListener(){  **public** **void** actionPerformed(ActionEvent e){  ++*num*;  String newString = String.*valueOf*(*num*);//数字转字符串  text.setText(newString);  }  });  frame.pack();A  frame.setVisible(**true**);  }  **public** **static** **void** main(String[] args) {  **new** JustForTest();  }  } |

运行后的结果的截图如下：

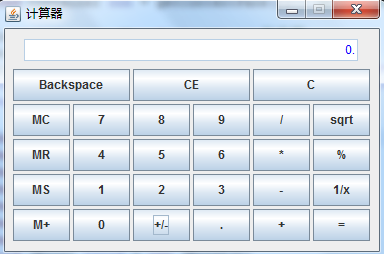
从结果可以看到，SWING的界面风格比较好看，而且这里我的实现基本代码都是从AWT上面修改来的，具体就是，1，将相应的组件类名该为J开头的。2，将JTextField构造的时候设置了个长度，便于观察，开始的太小。3.去掉了前面AWT中的那个关于监听的代码，因为那个事监听关闭按钮的，而这里直接调用函数setDefaultCloseOperation就可以很好的实现了。

上面，就是AWT和SWING在实现本题时候的区别。

10.2.（1）第一步创建这个计算器的布局，完成后的计算器的程序如下：

|  |
| --- |
| import java.awt.\*;  import javax.swing.\*;  public class calculator extends JFrame {  protected Container con = getContentPane();// 指向内容面板  JTextField jtf = new JTextField(30);// 文本框  JButton[] jb = new JButton[27];  String[] arr = { "Backspace", "CE", "C", "MC", "7", "8", "9", "/", "sqrt",  "MR", "4", "5", "6", "\*", "%", "MS", "1", "2", "3", "-", "1/x",  "M+", "0", "+/-", ".", "+", "=" };  protected JPanel jp = new JPanel();// 主面板  protected JPanel pane = new JPanel();// 次面板  protected JPanel pane1= new JPanel();  protected JPanel pane2 = new JPanel();  protected JPanel pane3 = new JPanel();  protected JPanel pane4 = new JPanel();  protected JPanel pane5 = new JPanel();  protected JPanel pane6 = new JPanel();  protected GridLayout glo = new GridLayout(6, 1, 3, 3);// 主网格布局  protected GridLayout glo1 = new GridLayout(1, 3, 3, 3);// 次网格布局  protected GridLayout glo2 = new GridLayout(1, 6, 3, 3);// 次网格布局  public calculator(String s) {  super(s);  for (int i = 0; i < 27; i++) {  jb[i] = new JButton(arr[i]);  }  jp.setLayout(glo);  jp.add(pane1);  pane1.add(jtf);  jtf.setEditable(false);  jtf.setText("0.");  jtf.setForeground(Color.BLUE);  jtf.setBackground(Color.WHITE);  jtf.setHorizontalAlignment(JTextField.RIGHT);//文本显示在右边  jp.add(pane2);  pane2.setLayout(glo1);  pane2.add(jb[0]);  pane2.add(jb[1]);  pane2.add(jb[2]);  jp.add(pane3);  pane3.setLayout(glo2);  pane3.add(jb[3]);  pane3.add(jb[4]);  pane3.add(jb[5]);  pane3.add(jb[6]);  pane3.add(jb[7]);  pane3.add(jb[8]);  jp.add(pane4);  pane4.setLayout(glo2);  pane4.add(jb[9]);  pane4.add(jb[10]);  pane4.add(jb[11]);  pane4.add(jb[12]);  pane4.add(jb[13]);  pane4.add(jb[14]);  jp.add(pane5);  pane5.setLayout(glo2);  pane5.add(jb[15]);  pane5.add(jb[16]);  pane5.add(jb[17]);  pane5.add(jb[18]);  pane5.add(jb[19]);  pane5.add(jb[20]);  jp.add(pane6);  pane6.setLayout(glo2);  pane6.add(jb[21]);  pane6.add(jb[22]);  pane6.add(jb[23]);  pane6.add(jb[24]);  pane6.add(jb[25]);  pane6.add(jb[26]);  pane.add(jp);  con.add(pane);  this.setResizable(false);// 不能用鼠标拉伸窗体  this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);//可以关闭窗体  this.setSize(380, 250);  this.setVisible(true);  }  public static void main(String[] args) {  new calculator("计算器");  }  } |

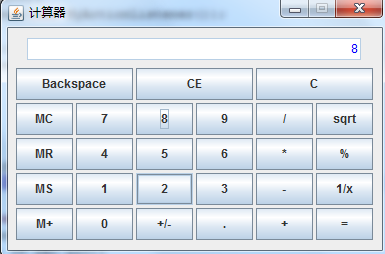
运行后的截图如下：

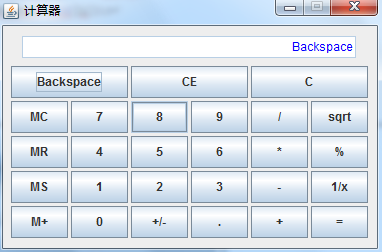


（2）.实现相应的键盘响应的代码如下：

|  |
| --- |
| **import** java.awt.\*;  **import** javax.swing.\*;  **import** java.awt.event.\*;  **public** **class** calculator **extends** JFrame {  **protected** Container con = getContentPane();// 指向内容面板  JTextField jtf = **new** JTextField(30);// 文本框  JButton[] jb = **new** JButton[27];  String[] arr = { "Backspace", "CE", "C", "MC", "7", "8", "9", "/", "sqrt",  "MR", "4", "5", "6", "\*", "%", "MS", "1", "2", "3", "-", "1/x",  "M+", "0", "+/-", ".", "+", "=" };  **protected** JPanel jp = **new** JPanel();// 主面板  **protected** JPanel pane = **new** JPanel();// 次面板  **protected** JPanel pane1= **new** JPanel();  **protected** JPanel pane2 = **new** JPanel();  **protected** JPanel pane3 = **new** JPanel();  **protected** JPanel pane4 = **new** JPanel();  **protected** JPanel pane5 = **new** JPanel();  **protected** JPanel pane6 = **new** JPanel();  **protected** GridLayout glo = **new** GridLayout(6, 1, 3, 3);// 主网格布局  **protected** GridLayout glo1 = **new** GridLayout(1, 3, 3, 3);// 次网格布局  **protected** GridLayout glo2 = **new** GridLayout(1, 6, 3, 3);// 次网格布局  **public** calculator(String s) {  **super**(s);  **for** (**int** i = 0; i < 27; i++) {  jb[i] = **new** JButton(arr[i]);  }  jp.setLayout(glo);  jp.add(pane1);  pane1.add(jtf);  jtf.setEditable(**false**);  jtf.setText("0.");  jtf.setForeground(Color.*BLUE*);  jtf.setBackground(Color.*WHITE*);  jtf.setHorizontalAlignment(JTextField.*RIGHT*);//文本显示在右边  jp.add(pane2);  pane2.setLayout(glo1);  pane2.add(jb[0]);  pane2.add(jb[1]);  pane2.add(jb[2]);  jp.add(pane3);  pane3.setLayout(glo2);  pane3.add(jb[3]);  pane3.add(jb[4]);  pane3.add(jb[5]);  pane3.add(jb[6]);  pane3.add(jb[7]);  pane3.add(jb[8]);  jp.add(pane4);  pane4.setLayout(glo2);  pane4.add(jb[9]);  pane4.add(jb[10]);  pane4.add(jb[11]);  pane4.add(jb[12]);  pane4.add(jb[13]);  pane4.add(jb[14]);  jp.add(pane5);  pane5.setLayout(glo2);  pane5.add(jb[15]);  pane5.add(jb[16]);  pane5.add(jb[17]);  pane5.add(jb[18]);  pane5.add(jb[19]);  pane5.add(jb[20]);  jp.add(pane6);  pane6.setLayout(glo2);  pane6.add(jb[21]);  pane6.add(jb[22]);  pane6.add(jb[23]);  pane6.add(jb[24]);  pane6.add(jb[25]);  pane6.add(jb[26]);  pane.add(jp);  con.add(pane);    **for**(**int** i = 0; i <= 26; i++){  jb[i].addActionListener(**new** MyActionListener());  }  **this**.setResizable(**false**);// 不能用鼠标拉伸窗体  **this**.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);//可以关闭窗体  **this**.setSize(380, 250);  **this**.setVisible(**true**);  }  **public** **static** **void** main(String[] args) {  **new** calculator("计算器");  }  **class** MyActionListener **implements** ActionListener{  **public** **void** actionPerformed(ActionEvent e){  **for**(**int** i = 0; i <= 26; i++){  **if**(e.getSource()== jb[i])  jtf.setText(arr[i]);  }  }  }  } |

截图如下：（实现了点击相应的项都显示出来的效果）





总结：计算器的布局花费了很多的功夫，查阅了相关的资料，开始的时候由于只用了一个panel，不管怎么样都出不来结果，后来发现了问题，并及时的解决了。其次就是监听的时候花费了功夫，开始的时候我定义的是一个内部类，但是这样中间就有一个问题，就是我要访问到外部的变量i，此时必须内部类中访问的是常量，于是不能用这种方法，自己写了个类实现ActionListener接口，然后成功的监听，用了个for循环，而且判断的时候直接用原来的那个String，比较简单，此处比较优化，之前本来想直接一个个的从0~26号button都添加方法，太麻烦了，于是想到了这个方法。完成。