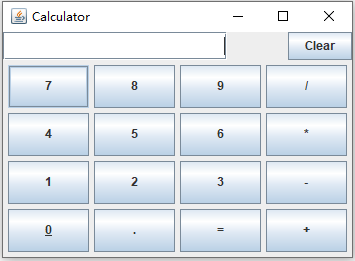
【案例11-2】 简易计算器

**【案例介绍】**

**1.案例描述**

本案例要求利用Java Swing 图形组件开发一个可以进行简单的四则运算的图形化计算器。

**2.运行结果**



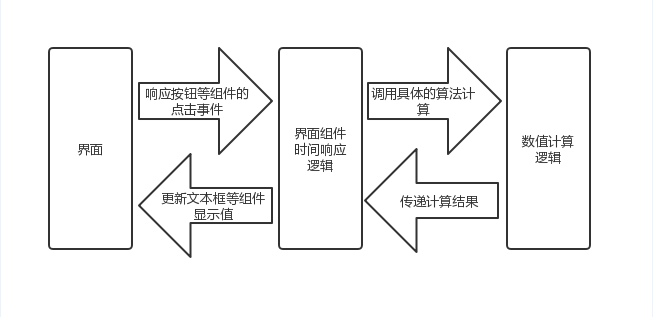
运行结果

**【案例目标】**

* 学会分析“简易计算器”程序实现的逻辑思路。
* 能够独立完成“简易计算器”程序的源代码编写、编译及运行。
* 掌握Java Swing界面编程的应用。
* 了解计算器逻辑运算实现。

**【案例分析】**

要制作一个计算器，首先要知道它由哪些部分组成，如下图所示：



一个简单的图形化界面计算器，需要由界面组件、组件的时间监听器和具体的事件处理逻辑组成。

实现一个简易图形化界面计算器可分以下几个步骤：

1.UI组件创建和初始化：包括窗体的创建，显示计算结果文本框、清除按钮、数字按钮、运算符等按钮的初始化。

2.在窗体中添加UI组件：包括放置数字键及运算符面板、放置清除框面板。

3.布局结束之后，就是计算器的难点：编写事件处理程序。

4.按键的响应实现。

5.计算逻辑的实现。

6.注册监听器

**【案例实现】**

1. import java.awt.\*;
2. import java.awt.event.\*;
3. import javax.swing.\*;
4. import java.util.Vector;
5. import java.math.BigDecimal;
6. public class Calculator {
7. // 操作数1，为了程序的安全，初值一定设置，这里我们设置为0。
8. String str1 = "0";
9. // 操作数2
10. String str2 = "0";
11. // 运算符
12. String signal = "+";
13. // 运算结果
14. String result = "";
15. // 以下k1至k5为状态开关
16. // 开关1用于选择输入方向，将要写入str1或str2
17. // 为 1 时写入 str1，为 2 时写入 str2
18. int k1 = 1;
19. // 开关 2 用于记录符号键的次数
20. // 如果 k2>1 说明进行的是 2+3-9+8 这样的多符号运算
21. int k2 = 1;
22. // 开关3用于标识 str1 是否可以被清 0
23. // 等于 1 时可以，不等于1时不能被清0
24. int k3 = 1;
25. // 开关4用于标识 str2 是否可以被清 0
26. // 等于 1 时可以，不等于1时不能被清0
27. int k4 = 1;
28. // 开关5用于控制小数点可否被录入
29. // 等于1时可以，不为1时，输入的小数点被丢掉
30. int k5 = 1;
31. // store的作用类似于寄存器，用于记录是否连续按下符号键
32. JButton store;
33. //vt 存储之前输入的运算符。
34. @SuppressWarnings("rawtypes")
35. Vector vt = new Vector(20, 10);
36. // 创建一个 JFrame 对象并初始化。JFrame 可以理解为程序的主窗体。
37. JFrame frame = new JFrame("Calculator");
38. //创建一个JTextField对象并初始化。JTextField用于显示操作和计算结果的文本框。
39. // 参数 20 表明可以显示 20 列的文本内容
40. JTextField result\_TextField = new JTextField(result, 20);
41. // 清除按钮
42. JButton clear\_Button = new JButton("Clear");
43. // 数字键0到9
44. JButton button0 = new JButton("0");
45. JButton button1 = new JButton("1");
46. JButton button2 = new JButton("2");
47. JButton button3 = new JButton("3");
48. JButton button4 = new JButton("4");
49. JButton button5 = new JButton("5");
50. JButton button6 = new JButton("6");
51. JButton button7 = new JButton("7");
52. JButton button8 = new JButton("8");
53. JButton button9 = new JButton("9");
54. // 计算命令按钮，加减乘除以及小数点等
55. JButton button\_Dian = new JButton(".");
56. JButton button\_jia = new JButton("+");
57. JButton button\_jian = new JButton("-");
58. JButton button\_cheng = new JButton("\*");
59. JButton button\_chu = new JButton("/");
60. JButton button\_dy = new JButton("=");
61. public Calculator() {
62. button0.setMnemonic(KeyEvent.VK\_0);
63. result\_TextField.setHorizontalAlignment(JTextField.RIGHT);
64. // 创建一个 Jpanel 对象并初始化
65. JPanel pan = new JPanel();
66. // 设置该容器的布局为四行四列，边距为5像素
67. pan.setLayout(new GridLayout(4, 4, 5, 5));
68. // 将用于计算的按钮添加到容器内
69. pan.add(button7);
70. pan.add(button8);
71. pan.add(button9);
72. pan.add(button\_chu);
73. pan.add(button4);
74. pan.add(button5);
75. pan.add(button6);
76. pan.add(button\_cheng);
77. pan.add(button1);
78. pan.add(button2);
79. pan.add(button3);
80. pan.add(button\_jian);
81. pan.add(button0);
82. pan.add(button\_Dian);
83. pan.add(button\_dy);
84. pan.add(button\_jia);
85. // 设置 pan 对象的边距
86. pan.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
87. // 按照同样的方式设置第二个JPanel
88. JPanel pan2 = new JPanel();
89. pan2.setLayout(new BorderLayout());
90. pan2.add(result\_TextField, BorderLayout.WEST);
91. pan2.add(clear\_Button, BorderLayout.EAST);
92. // 设置主窗口出现在屏幕上的位置
93. frame.setLocation(300, 200);
94. // 设置窗体可以调大小
95. frame.setResizable(true);
96. //窗体中可以放置 JPanel，这里我们将面板pan和面板pan2让如窗体
97. frame.getContentPane().setLayout(new BorderLayout());
98. frame.getContentPane().add(pan2, BorderLayout.NORTH);
99. frame.getContentPane().add(pan, BorderLayout.CENTER);
100. frame.pack();
101. frame.setVisible(true);
102. //Listener类中编写的是数字键的响应逻辑。
103. class Listener implements ActionListener {
104. @SuppressWarnings("unchecked")
105. public void actionPerformed(ActionEvent e) {
106. String ss = ((JButton) e.getSource()).getText();
107. store = (JButton) e.getSource();
108. vt.add(store);
109. if (k1 == 1) {
110. if (k3 == 1) {
111. str1 = "";
112. k5 = 1;
113. }
114. str1 = str1 + ss;
115. k3 = k3 + 1;
116. result\_TextField.setText(str1);
117. } else if (k1 == 2) {
118. if (k4 == 1) {
119. str2 = "";
120. k5 = 1;
121. }
122. str2 = str2 + ss;
123. k4 = k4 + 1;
124. result\_TextField.setText(str2);
125. }
126. }
127. }
128. //Listener\_signal类中编写了运算符号键的响应逻辑
129. class Listener\_signal implements ActionListener {
130. @SuppressWarnings("unchecked")
131. public void actionPerformed(ActionEvent e) {
132. String ss2 = ((JButton) e.getSource()).getText();
133. store = (JButton) e.getSource();
134. vt.add(store);
135. if (k2 == 1) {
136. k1 = 2;
137. k5 = 1;
138. signal = ss2;
139. k2 = k2 + 1;
140. } else {
141. int a = vt.size();
142. JButton c = (JButton) vt.get(a - 2);
143. if (!(c.getText().equals("+"))
144. && !(c.getText().equals("-"))
145. && !(c.getText().equals("\*"))
146. && !(c.getText().equals("/")))
147. {
148. cal();
149. str1 = result;
150. k1 = 2;
151. k5 = 1;
152. k4 = 1;
153. signal = ss2;
154. }
155. k2 = k2 + 1;
156. }
157. }
158. }
159. //Listener\_clear类中编写了清除键的响应逻辑
160. class Listener\_clear implements ActionListener {
161. @SuppressWarnings("unchecked")
162. public void actionPerformed(ActionEvent e) {
163. store = (JButton) e.getSource();
164. vt.add(store);
165. k5 = 1;
166. k2 = 1;
167. k1 = 1;
168. k3 = 1;
169. k4 = 1;
170. str1 = "0";
171. str2 = "0";
172. signal = "";
173. result = "";
174. result\_TextField.setText(result);
175. vt.clear();
176. }
177. }
178. //Listener\_dy类中编写的是等于号键的响应逻辑
179. class Listener\_dy implements ActionListener {
180. @SuppressWarnings("unchecked")
181. public void actionPerformed(ActionEvent e) {
182. store = (JButton) e.getSource();
183. vt.add(store);
184. cal();
185. k1 = 1;
186. k2 = 1;
187. k3 = 1;
188. k4 = 1;
189. str1 = result;
190. }
191. }
192. //Listener\_xiaos类中编写的是小数点键的相应逻辑
193. class Listener\_xiaos implements ActionListener {
194. @SuppressWarnings("unchecked")
195. public void actionPerformed(ActionEvent e) {
196. store = (JButton) e.getSource();
197. vt.add(store);
198. if (k5 == 1) {
199. String ss2 = ((JButton) e.getSource()).getText();
200. if (k1 == 1) {
201. if (k3 == 1) {
202. str1 = "";
203. k5 = 1;
204. }
205. str1 = str1 + ss2;
206. k3 = k3 + 1;
207. result\_TextField.setText(str1);
208. } else if (k1 == 2) {
209. if (k4 == 1) {
210. str2 = "";
211. k5 = 1;
212. }
213. str2 = str2 + ss2;
214. k4 = k4 + 1;
215. result\_TextField.setText(str2);
216. }
217. }
218. k5 = k5 + 1;
219. }
220. }
221. // 监听等于键
222. Listener\_dy jt\_dy = new Listener\_dy();
223. // 监听数字键
224. Listener jt = new Listener();
225. // 监听符号键
226. Listener\_signal jt\_signal = new Listener\_signal();
227. // 监听清除键
228. Listener\_clear jt\_c = new Listener\_clear();
229. // 监听小数点键
230. Listener\_xiaos jt\_xs = new Listener\_xiaos();
231. button7.addActionListener(jt);
232. button8.addActionListener(jt);
233. button9.addActionListener(jt);
234. button\_chu.addActionListener(jt\_signal);
235. button4.addActionListener(jt);
236. button5.addActionListener(jt);
237. button6.addActionListener(jt);
238. button\_cheng.addActionListener(jt\_signal);
239. button1.addActionListener(jt);
240. button2.addActionListener(jt);
241. button3.addActionListener(jt);
242. button\_jian.addActionListener(jt\_signal);
243. button0.addActionListener(jt);
244. button\_Dian.addActionListener(jt\_xs);
245. button\_dy.addActionListener(jt\_dy);
246. button\_jia.addActionListener(jt\_signal);
247. clear\_Button.addActionListener(jt\_c);
248. // 窗体关闭事件的响应程序
249. frame.addWindowListener(new WindowAdapter() {
250. public void windowClosing(WindowEvent e) {
251. System.exit(0);
252. }
253. });
254. }
255. // calc()方法中编写了计算逻辑的实现。
256. public void cal() {
257. double a2;
258. double b2;
259. String c = signal;
260. double result2 = 0;
261. if (c.equals("")) {
262. result\_TextField.setText("Please input operator");
263. } else {
264. if (str1.equals("."))
265. str1 = "0.0";
266. if (str2.equals("."))
267. str2 = "0.0";
268. a2 = Double.valueOf(str1).doubleValue();
269. b2 = Double.valueOf(str2).doubleValue();
270. if (c.equals("+")) {
271. result2 = a2 + b2;
272. }
273. if (c.equals("-")) {
274. result2 = a2 - b2;
275. }
276. if (c.equals("\*")) {
277. BigDecimal m1 = new BigDecimal(Double.toString(a2));
278. BigDecimal m2 = new
279. BigDecimal(Double.toString(b2));
280. result2 = m1.multiply(m2).doubleValue();
281. }
282. if (c.equals("/")) {
283. if (b2 == 0) {
284. result2 = 0;
285. } else {
286. result2 = a2 / b2;
287. }
288. }
289. result = ((new Double(result2)).toString());
290. result\_TextField.setText(result);
291. }
292. }
293. @SuppressWarnings("unused")
294. public static void main(String[] args) {
295. try {
296. //通过 UIManager 来设置窗体的 UI 风格
297. UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");
298. } catch (Exception e) {
299. e.printStackTrace();
300. }
301. Calculator cal = new Calculator();
302. }
303. }

上述代码中，第7-32行代码，定义了一些成员变量，方便响应的逻辑实现。第34-35行代码，创建了一个Vector对象，存储之前输入的运算符。第37行代码，创建了一个 JFrame 对象并初始化。JFrame 可以理解为程序的主窗体。第40行代码，创建了一个JTextField对象并初始化。JTextField用于显示操作和计算结果的文本框。参数 20 表明可以显示 20 列的文本内容。第42行代码，创建了一个清除按钮对象。第44-53行，创建数字键0-9按钮对象。第55-60行，创建计算命令按钮，加减乘除以及小数点等按钮对象。第65-101行代码，是对计算器进行布局。Listener类中编写的是数字键的响应逻辑。Listener\_xiaos类中编写的是小数点键的相应逻辑。Listener\_signal类中编写了运算符号键的响应逻辑。Listener\_dy类中编写的是等于号键的响应逻辑。Listener\_clear类中编写了清除键的响应逻辑。calc()方法中编写了计算逻辑的实现。第222-247行代码，创建监听对象，斌对各个按键进行监听。第249-253行代码，为窗体关闭事件的响应程序。第297行代码，通过 UIManager 来设置窗体的 UI 风格。