**鼠标监听器MouseListener**

监听鼠标事件MouseEvent。

**相应事件和处理方法**

鼠标事件         处理方法  
MOUSE\_CLICKED    MouseClicked (MouseEvent)    鼠标点击（单或双）  
MOUSE\_PRESSED    MousePressed (MouseEvent)    鼠标按下  
MOUSE\_RELEASED   MouseReleased(MouseEvent)    鼠标松开  
MOUSE\_ENTERED    MouseEntered (MouseEvent)    鼠标进入（某组件区域）  
MOUSE\_EXITED     MouseExited  (MouseEvent)    鼠标离开（某组件区域）

**鼠标事件MouseEvent常用方法**

int getClickCount()      得到点击次数1 OR 2；  
int getX(), int getY()   得到鼠标的（象素）位置。

**鼠标监听器MouseMotionListener**

对于鼠标的移动和拖放，另外用鼠标运动监听器MouseMotionListener。  
因为许多程序不需要监听鼠标运动，把两者分开可简化程序。

**相应事件和处理方法**

鼠标事件          处理方法

MOUSE\_MOVED      MouseMoved  (MouseEvent)    鼠标在移动

MOUSE\_DRAGGED    MouseDragged(MouseEvent)    鼠标被拖动

**范例Java程序**

我们把对这两个鼠标监听器综合运用。

**[java]** [view plain](http://blog.csdn.net/jueblog/article/details/9739627) [copy](http://blog.csdn.net/jueblog/article/details/9739627)

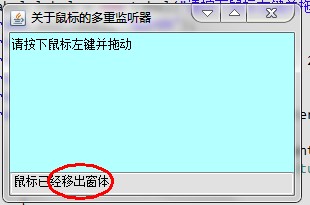
1. **package** com.test0803;
3. **import** java.awt.Color;
4. **import** java.awt.Frame;
5. **import** java.awt.Label;
6. **import** java.awt.TextField;
7. **import** java.awt.event.MouseEvent;
8. **import** java.awt.event.MouseListener;
9. **import** java.awt.event.MouseMotionListener;
10. **import** java.awt.event.WindowAdapter;
11. **import** java.awt.event.WindowEvent;
13. **public** **class** Test\_Mouse2  **implements** MouseMotionListener,MouseListener{
14. Frame frame = **new** Frame("关于鼠标的多重监听器");
15. TextField tField = **new** TextField(30);
16. **public** Test\_Mouse2(){
17. Label label = **new** Label("请按下鼠标左键并拖动");
18. frame.add(label, "North");
19. frame.add(tField, "South");
20. frame.setBackground(**new** Color(180, 255, 255));
21. frame.addMouseListener(**this**);
22. frame.addMouseMotionListener(**this**);
23. frame.addWindowListener(**new** WindowAdapter() {
24. @Override
25. **public** **void** windowClosing(WindowEvent e) {
26. // TODO Auto-generated method stub
27. System.exit(0);
28. }
29. });
30. frame.setSize(300, 200);
31. frame.setLocation(400, 250);
32. frame.setVisible(**true**);
33. }
34. **public** **static** **void** main(String[] args) {
35. **new** Test\_Mouse2();
36. }
38. @Override
39. **public** **void** mouseClicked(MouseEvent e) {
40. // TODO Auto-generated method stub
41. System.out.print("鼠标点击----" + "\t");
42. **if** (e.getClickCount()==1) {
43. System.out.println("单击！");
44. } **else** **if** (e.getClickCount()==2) {
45. System.out.println("双击！");
46. } **else** **if** (e.getClickCount()==3) {
47. System.out.println("三连击！！");
48. }
50. }
52. @Override
53. **public** **void** mousePressed(MouseEvent e) {
54. // TODO Auto-generated method stub
55. System.out.println("鼠标按下");
56. }
58. @Override
59. **public** **void** mouseReleased(MouseEvent e) {
60. // TODO Auto-generated method stub
61. System.out.println("鼠标松开");
62. }
64. @Override
65. **public** **void** mouseEntered(MouseEvent e) {
66. // TODO Auto-generated method stub
67. tField.setText("鼠标已经进入窗体");
68. }
70. @Override
71. **public** **void** mouseExited(MouseEvent e) {
72. // TODO Auto-generated method stub
73. tField.setText("鼠标已经移出窗体");
74. }
76. @Override
77. **public** **void** mouseDragged(MouseEvent e) {
78. // TODO Auto-generated method stub
79. String string = "鼠标拖动到位置：（" + e.getX() + "，" + e.getY() +"）";
80. tField.setText(string);
81. }
83. @Override
84. **public** **void** mouseMoved(MouseEvent e) {
85. // TODO Auto-generated method stub
86. }
88. }

**Java程序分析**

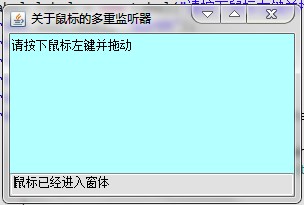
执行程序。

通过执行结果，我们可以得到监听器不同方法的含义。

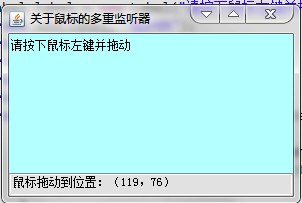
**鼠标未进入窗体时**



**鼠标进入窗体时**



**在窗体内拖动鼠标**



同时输出：

**[java]** [view plain](http://blog.csdn.net/jueblog/article/details/9739627) [copy](http://blog.csdn.net/jueblog/article/details/9739627)

1. 鼠标按下
2. 鼠标松开

**在窗体内单击鼠标**

输出：

**[java]** [view plain](http://blog.csdn.net/jueblog/article/details/9739627) [copy](http://blog.csdn.net/jueblog/article/details/9739627)

1. 鼠标按下
2. 鼠标松开
3. 鼠标点击----    单击！

由此可知监听器内部的执行顺序。

**在窗体内双击鼠标**

输出：

**[java]** [view plain](http://blog.csdn.net/jueblog/article/details/9739627) [copy](http://blog.csdn.net/jueblog/article/details/9739627)

1. 鼠标按下
2. 鼠标松开
3. 鼠标点击----    单击！
4. 鼠标按下
5. 鼠标松开
6. 鼠标点击----    双击！

**在窗体内三连击鼠标**

输出：

**[java]** [view plain](http://blog.csdn.net/jueblog/article/details/9739627) [copy](http://blog.csdn.net/jueblog/article/details/9739627)

1. 鼠标按下
2. 鼠标松开
3. 鼠标点击----    单击！
4. 鼠标按下
5. 鼠标松开
6. 鼠标点击----    双击！
7. 鼠标按下
8. 鼠标松开
9. 鼠标点击----    三连击！！