

# บทที่ 8: การจัดการเหตุการณ์ (GUI Handling Events)

บรรยายโดย ผศ.ดร. ธราวิชษฐ์ ธิติจรรย์โรจน์

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

# เหตุการณ์ (Event)

- เป็นสถานการณ์ที่เกิดขึ้นในขณะรันโปรแกรม อาทิเช่น การใช้อินพุต (เมาส์หรือคีย์บอร์ด) ติดต่อกับโปรแกรม **GUI**  
โดยที่องค์ประกอบของเหตุการณ์ในจาวาจะประกอบด้วย 3 ส่วน ได้แก่



- ซึ่ง **Event Handler** จะรับเหตุการณ์ผ่านทางเมธอดและมีคำสั่งในการจัดการกับเหตุการณ์เพื่อโต้ตอบกับผู้ใช้ เช่น
  - การ**ขยับ**เมาส์ใน **JFrame** จะเกิดวัตถุของคลาส **MouseEvent** ขึ้นมา
  - การ**ปิด** **JFrame** จะเกิดวัตถุของคลาส **WindowEvent** ขึ้นมา
  - การ**กดปุ่ม**ใน **JFrame** จะเกิดวัตถุของคลาส **ActionEvent** ขึ้นมา
  - การ**พิมพ์ข้อความ**ใน **TextField** จะเกิดวัตถุของคลาส **KeyEvent** ขึ้นมา

# เหตุการณ์ (Event)



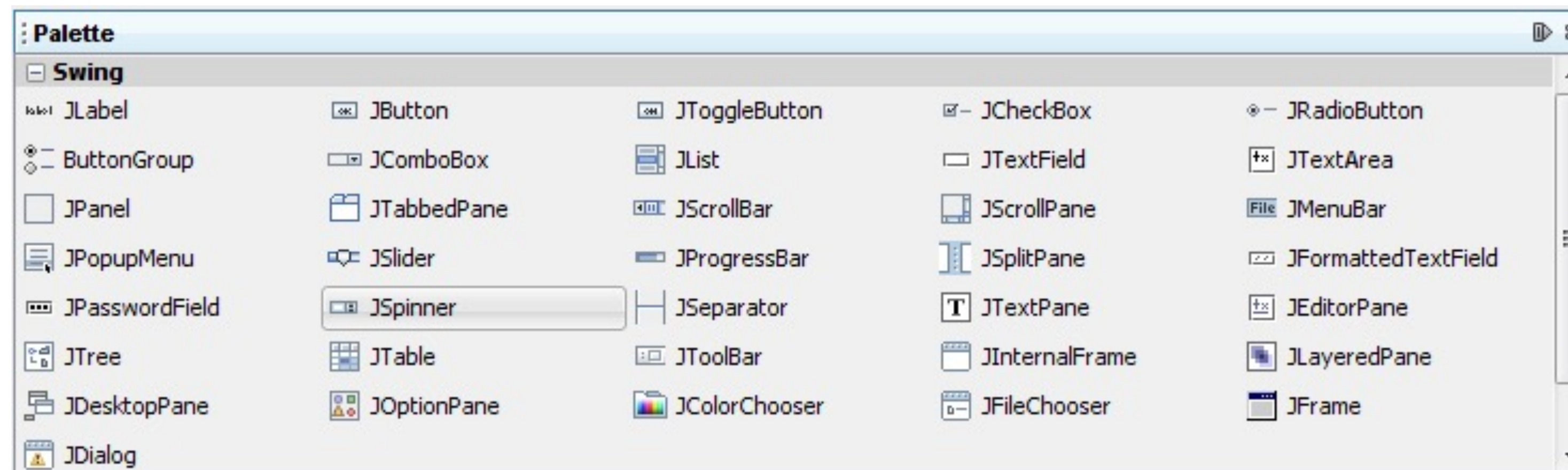
# เหตุการณ์ (Event)



# Event Source



- Event Source ได้แก่



# เหตุการณ์ (Event)

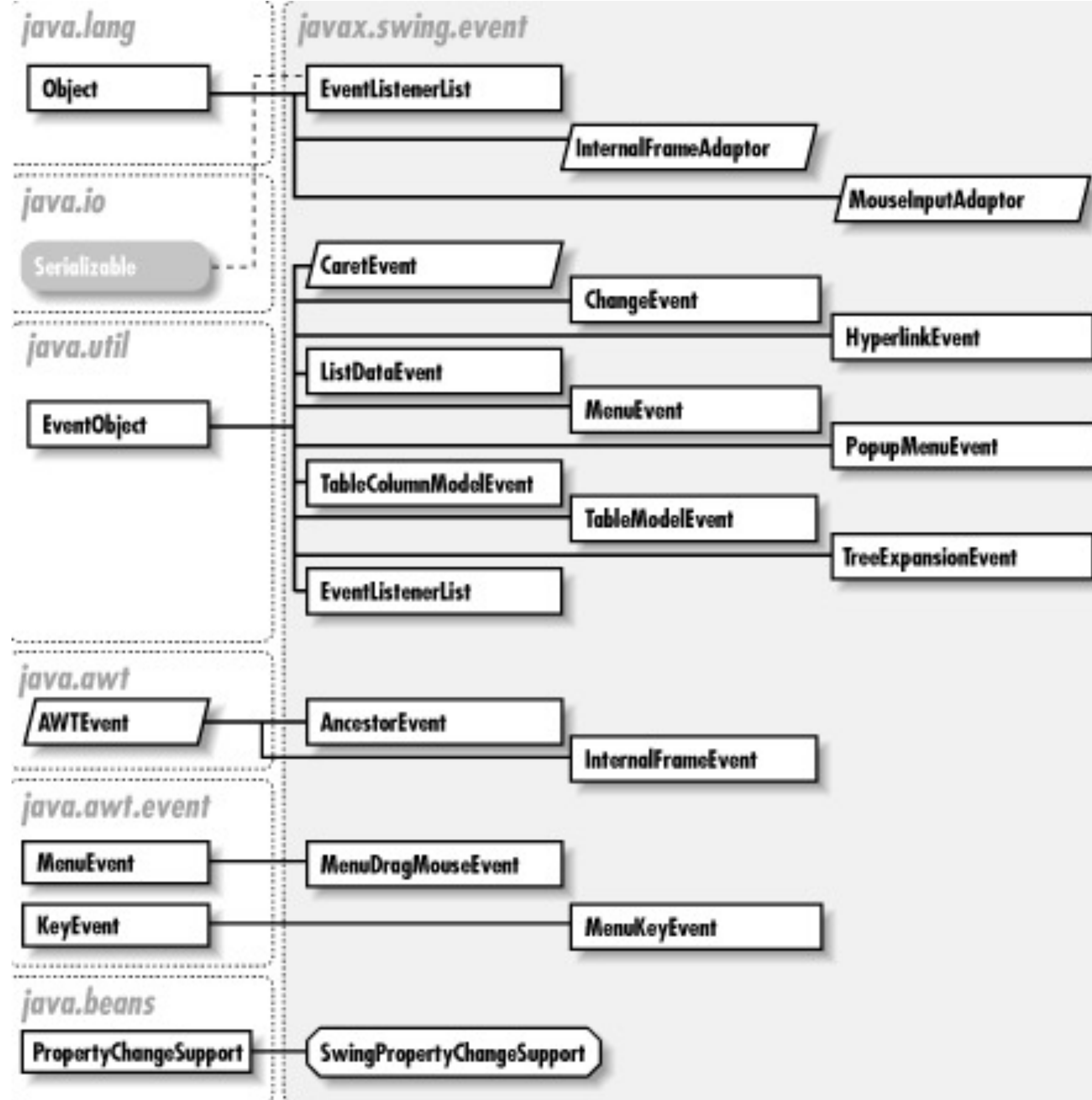
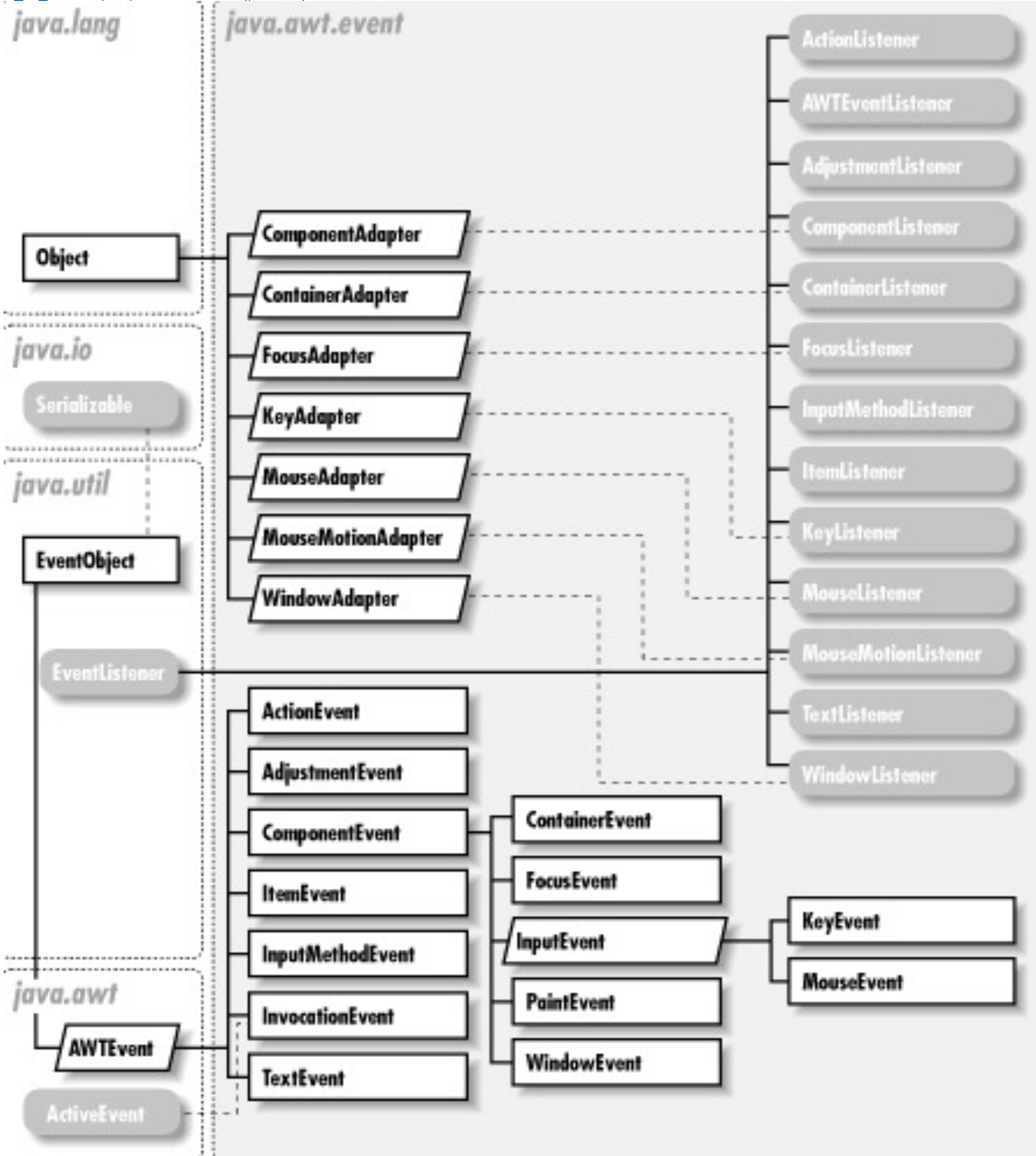




# คลาสประเภท Event

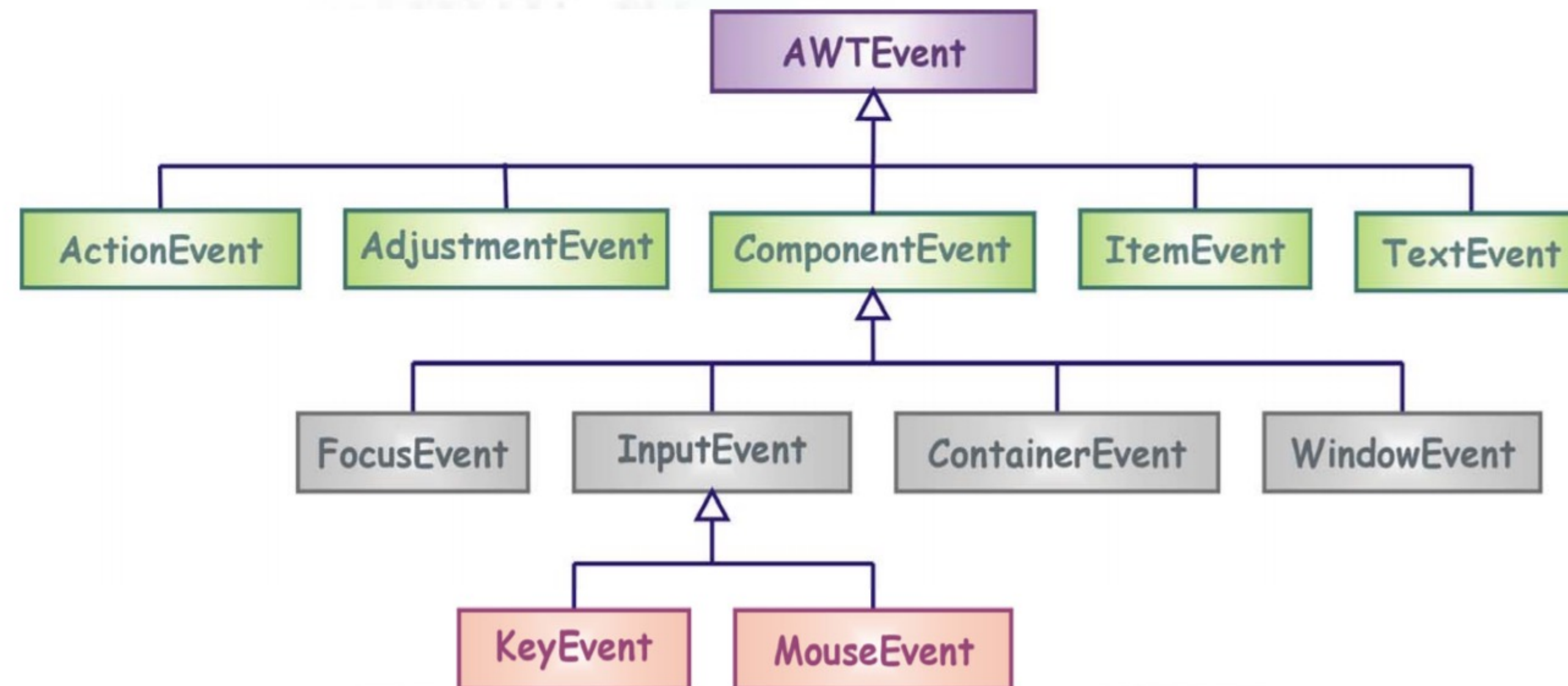


- ออปเจ็คของคลาสประเภท **Event** ที่เกิดขึ้นจะแตกต่างกันตามประเภทของเหตุการณ์
- คลาสประเภท **Event** ในภาษาจาวาจะสืบทอดจากคลาสที่ชื่อ **EventObject**
- คลาสประเภท **Event** สำหรับเหตุการณ์ทางกราฟิกจะสืบทอดจากคลาส **AWTEvent**
- คลาสประเภท **Event** สำหรับเหตุการณ์ทางกราฟิกจะอยู่ในแพ็คเกจ **java.awt.event**
- คลาสประเภท **Event** สำหรับเหตุการณ์ทางกราฟิกจะมีอยู่สองกลุ่มคือ คลาสในแพ็คเกจ **java.awt.event** และ **javax.swing.event**
- คลาสประเภท **Event** ที่ใช้โดยพื้นฐานคือคลาสในแพ็คเกจ **java.awt.event**
- ส่วนประกอบทางกราฟิกทั้งในแพ็คเกจ **AWT** และ **Swing** ต่างก็สามารถที่จะใช้คลาสเหล่านั้นได้
- **Java SE** ในเวอร์ชัน 1.4 ได้มีการเพิ่มคลาสประเภท **Event** และอินเตอร์เฟสต่างๆ ที่เกี่ยวข้องสำหรับใช้ใน อ็อบเจกต์ของคลาสประเภท **Swing** มากขึ้น โดยคลาสและอินเตอร์เฟสเหล่านี้จะอยู่ในแพ็คเกจ **javax.swing.event**





# Event



- วัตถุของคลาสประเภท **Event** ที่เกิดขึ้นจะแตกต่างกันตามประเภทของเหตุการณ์
- คลาสประเภท **Event** ในภาษาจาวาจะอยู่ในแพ็คเกจ **java.awt.event** และสืบทอดจากคลาสที่ **ObjectEvent**
- คลาสประเภท **Event** สำหรับเหตุการณ์ทางกราฟฟิกจะสืบทอดจากคลาส **AWTEvent** ซึ่งจะมีเมธอด **Object getSource()** เพื่อเรียกดูวัตถุประเภท **Event Source** และ **int getID()** เพื่อระบุชนิดของเหตุการณ์

# WindowEvent



- จะถูกสร้างขึ้นในกรณีที่มีเหตุการณ์เกิดขึ้นกับวัตถุคลาส **Window** ในรูปแบบต่อไปนี้

เหตุการณ์	คำอธิบาย
<b>Opened</b>	จะทำงานก็ต่อเมื่อแสดง ( <b>visible</b> กำหนดให้เป็น <b>true</b> ) หน้าต่างเป็นครั้งแรก
<b>Closed</b>	จะทำงานก็ต่อเมื่อหน้าต่างนั้นถูกปิดตัวลง
<b>Closing</b>	จะทำงานก็ต่อเมื่อผู้ใช้กดปุ่มปิด บริเวณเมนูของระบบปฏิบัติการ
<b>Iconified</b>	จะทำงานก็ต่อเมื่อหน้าต่างนั้นถูกย่ออยู่ที่ <b>window bar</b>
<b>Deiconified</b>	จะทำงานก็ต่อเมื่อหน้าต่างนั้นกลับมาสู่หน้าต่างปกติ
<b>Activated</b>	จะทำงานก็ต่อเมื่อหน้าต่างนั้นถูกเลือก (กำลังถูกพิจารณา)
<b>Deactivated</b>	จะทำงานก็ต่อเมื่อหน้าต่างนั้นไม่ถูกเลือก (ไม่ถูกพิจารณา)

- WindowEvent** จะมีเมธอดที่สำคัญดังนี้

เหตุการณ์	คำอธิบาย
<b>Object getWindow()</b>	จะส่งวัตถุที่เป็น <b>Event Source</b> ชนิด <b>Window</b> คืนมา

# MouseEvent



- จะถูกสร้างขึ้นในกรณีที่มีการใช้งานเมาส์ เพื่อติดต่อกับผู้ใช้โดยมีเหตุการณ์ที่เกิดขึ้นได้ ดังนี้

เหตุการณ์	คำอธิบาย
<b>Clicked</b>	การคลิกเมาส์ที่ <b>component</b>
<b>Entered</b>	การเลื่อนเมาส์ไปบน <b>component</b> ครั้งแรก
<b>Exited</b>	การเลื่อนเมาส์ไปออกจาก <b>component</b> ครั้งแรก
<b>Pressed</b>	การคลิกเมาส์
<b>Released</b>	การปล่อยเมาส์ที่คลิก
<b>Dragged</b>	การลากเมาส์
<b>Moved</b>	การขยับเมาส์

- **MouseEvent** จะมีเมธอดที่สำคัญดังนี้

เหตุการณ์	คำอธิบาย
<b>int getX()</b>	จะส่งพิกัดของเมาส์ ณ แกน <b>X</b>
<b>int getY()</b>	จะส่งพิกัดของเมาส์ ณ แกน <b>Y</b>
<b>int getClickCount()</b>	จะส่งจำนวนครั้งของการคลิกเมาส์ขึ้นมา

# ItemEvent



- จะถูกสร้างขึ้นในกรณีที่มีเหตุการณ์เกิดขึ้นกับวัตถุใน **GUI** ในรูปแบบต่อไปนี้
  - เลือก หรือ ยกเลิกรายการใน **List** หรือ **Checkbox**
  - คลิกเมาส์รายการใน **Choice** หรือ **RadioButton**
- **ItemEvent** จะมีเมธอดที่สำคัญดังนี้

เหตุการณ์	คำอธิบาย
<b>Object getItem()</b>	จะส่งวัตถุของรายการที่ถูกเลือกคืนมา
<b>ItemSelectable getItemSelectable()</b>	จะส่งวัตถุที่เป็น <b>Event Source</b> ชนิด <b>ItemSelectable</b> คืนมา
<b>Object getStateChange()</b>	จะส่งค่าคงที่ชนิด <b>int</b> ที่มีค่าเป็น <b>SELECTED</b> หรือ <b>DESELECTED</b> เพื่อระบุสถานการณ์เลือกของรายการคืนมา

# Event อื่น ๆ

- นอกจากนี้ยังมี **Event** อื่น ๆ ที่สำคัญ ได้แก่

เหตุการณ์	คำอธิบาย
<b>KeyEvent</b>	จะถูกสร้างขึ้นในกรณีที่เกิดเหตุการณ์กดคีย์บอร์ดขึ้น
<b>FocusEvent</b>	จะถูกสร้างขึ้นในกรณีที่เกิดเหตุการณ์ผู้ใช้เลื่อนอุปกรณ์อินพุท (เมาส์) มาชี้ยังวัตถุของ <b>GUI</b>
<b>ComponentEvent</b>	จะถูกสร้างขึ้นในกรณีที่เกิดเหตุการณ์วัตถุของ <b>GUI</b> มีการเปลี่ยนแปลง เช่น พิกัด, ตำแหน่ง , ขนาด เป็นต้น
<b>ContainerEvent</b>	จะถูกสร้างขึ้นในกรณีที่เกิดเหตุการณ์เพิ่มลด <b>component</b> ลงบน <b>Container</b>
<b>AdjustmentEvent</b>	จะถูกสร้างขึ้นในกรณีที่เกิดเหตุการณ์ปรับตำแหน่งของวัตถุในคลาส <b>ScrollBar</b> หรือ <b>ScrollPane</b>
<b>TextEvent</b>	จะถูกสร้างขึ้นในกรณีที่เกิดเหตุการณ์เปลี่ยนแปลงข้อความในวัตถุของคลาส <b>TextArea</b>



# ActionEvent



- จะถูกสร้างขึ้นในกรณีที่มีเหตุการณ์ (**action + object**) เกิดขึ้นกับ **component** บน **GUI** ต่อไปนี้
  - คลิกเมาส์บน **JButton**
  - ป้อนคีย์ **Enter** ใน **JTextField**
  - เลือกเมนูใน **JMenuItem**
  - กด **double** คลิก ใน **JList**

# เหตุการณ์ (Event)



# Event Handle โดยอาศัยอินเทอร์เฟซประเภท Listener



- ในภาษาจาวา **Event Handle** จะจัดการกับเหตุการณ์โดยอาศัยอินเทอร์เฟซประเภท **Listener** มาคอยรับฟังเหตุการณ์ (เช่น **ActionEvent**, **WindowEvent**, **MouseEvent**, **KeyEvent** เป็นต้น) ที่สอดคล้องกัน

Event	อินเทอร์เฟซ
MouseEvent	MouseListener
MouseMotionEvent	MouseMotionListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
KeyEvent	KeyListener
WindowEvent	WindowListener
FocusEvent	FocusListener
ItemEvent	ItemListener
TextEvent	TextListener
ActionEvent	ActionListener
AdjustmentEvent	AdjustmentListener

# เมธอดที่ใช้รับฟังเหตุการณ์

อินเทอร์เฟส	เมธอดที่ใช้รับฟังเหตุการณ์
ActionListener	addActionListener()
ItemListener	addItemListener()
KeyListener	addKeyListener()
MouseListener	addMouseListener()
MouseMotionListener	addMouseMotionListener()
TextListener	addTextListener()
FocusListener	addFocusListener()
AdjustmentListener	addAdjustmentListener()
ComponentListener	addComponentListener()
ContainerListener	addContainerListener()
WindowListener	addWindowListener()

# เมธอดที่ต้องลงทะเบียนสำหรับจัดการกับเหตุการณ์



อินเทอร์เฟส	เมธอดที่ต้อง implements
KeyListener	keyPressed (KeyEvent) keyReleased (KeyEvent) keyTyped (KeyEvent)
FocusListener	focusGained (FocusEvent) focusLost (FocusEvent)
AdjustmentListener	adjustmentValueChanged (AdjustmentEvent)
ComponentListener	componentMoved (ComponentEvent) componentHidden (ComponentEvent) componentResized (ComponentEvent) componentShown (ComponentEvent)



# เมธอดที่ต้องลงทะเบียนสำหรับการกับเหตุการณ์



อินเทอร์เฟส	เมธอดที่ต้อง implements
ActionListener	actionPerformed (ActionEvent)
ItemListener	itemStateChanged (ItemEvent)
MouseListener	mouseClicked (MouseEvent) mouseEntered (MouseEvent) mouseExited (MouseEvent) mousePressed (MouseEvent) mouseReleased (MouseEvent)
MouseMotionListener	mouseMoved (MouseEvent) mouseDragged (MouseEvent)

# เมธอดที่ต้องลงทะเบียนสำหรับการจัดการกับเหตุการณ์



อินเทอร์เฟส	เมธอดที่ต้อง implements
WindowListener	<code>windowOpened (WindowEvent)</code> <code>windowClosed (WindowEvent)</code> <code>windowClosing (WindowEvent)</code> <code>windowIconified (WindowEvent)</code> <code>windowDeiconified (WindowEvent)</code> <code>windowActivated (WindowEvent)</code> <code>windowDeactivated (WindowEvent)</code>
ContainerListener	<code>componentAdded (ContainerEvent)</code> <code>componentRemoved (ContainerEvent)</code>
TextListener	<code>textValueChanged (TextEvent)</code>

# การรับฟังเหตุการณั



การรับฟัง**หนึ่ง**เหตุการณั

การรับฟัง**หลาย**เหตุการณั

# การรับฟังเหตุการณั

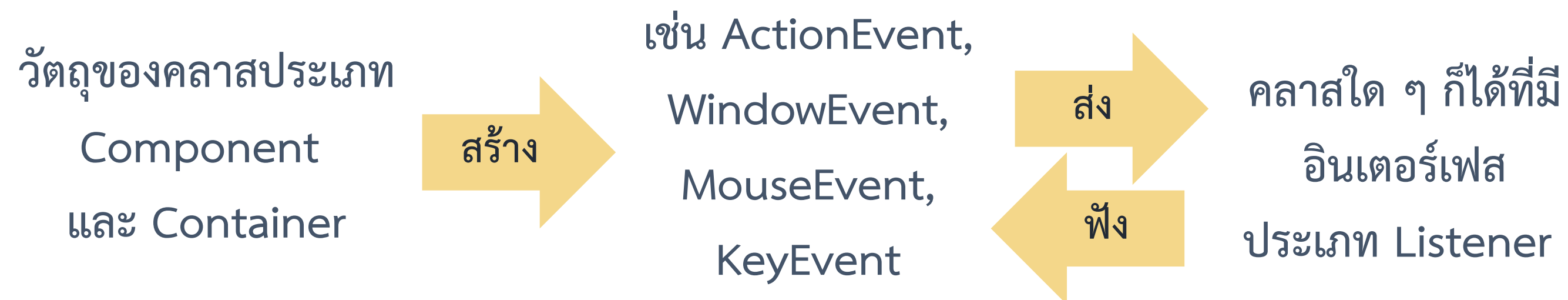


การรับฟังหนึ่งเหตุการณั

การรับฟังหลายเหตุการณั

# ขั้นตอนการจัดการกับเหตุการณ์

- ในภาษาจาวามีวิธีจัดการกับเหตุการณ์ที่เรียกว่า “Delegation Model” ซึ่งมีหลักการ ดังนี้



- Event Source ใด ๆ ต้องการที่จะจัดการกับเหตุการณ์ใดต้องลงทะเบียน เพื่อรับฟังเหตุการณ์ โดยมีรูปแบบต่อไปนี้

```
[eventSource].addXxxListener(listener);
```

ซึ่งสามารถทำได้ 4 รูปแบบ



# การจัดการกับเหตุการณ์ กรณีที่ 1 : คลาสภายนอก

```

import java.awt.FlowLayout;
import javax.swing.*;
public class EventDemo1 {
    public void EventDemo1() {
        JFrame fr = new JFrame("Outer Class Event");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JButton bn1 = new JButton("Exit");
        fr.setLayout(new FlowLayout());
        bn1.addActionListener(new ActionListener());
        fr.add(bn1);
        fr.setSize(200, 200);
        fr.setVisible(true);
    }
    public static void main(String[] args) {
        new EventDemo1();
    }
}

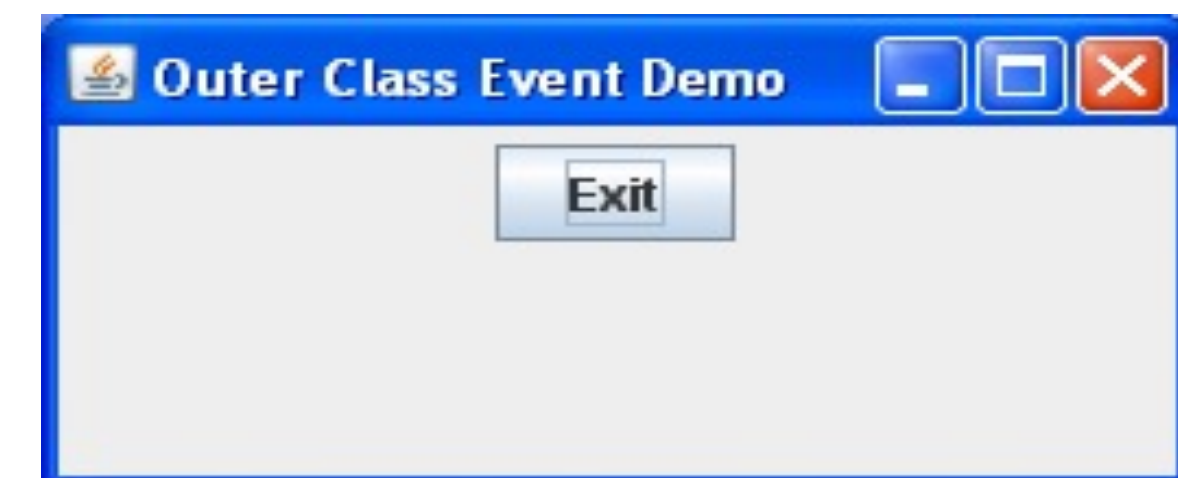
public class ActionListener implements ActionListener {
    public void actionPerformed(ActionEvent ev) {
        System.exit(0);
    }
}
  
```

- คลาส **EventDemo1** มีปุ่มที่ชื่อ **bn1** ซึ่งลงทะเบียนรับฟัง **Event** ประเภท **ActionEvent** จากคำสั่ง

```
bn1.addActionListener(new ActionListener());
```

- คำสั่ง **new ActionListener()** เป็นการสร้างวัตถุที่ใช้ในการจัดการกับ **Event** ประเภท **ActionEvent** โดยที่คลาส **ActionHandler** จะต้อง **implements** อินเตอร์เฟสชื่อ **ActionListener** ซึ่งเมธอดที่ต้อง **implements** คือ

```
public void actionPerformed (ActionEvent ev)
```



# การจัดการกับเหตุการณ์ กรณีที่ 2 : คลาสภายใน

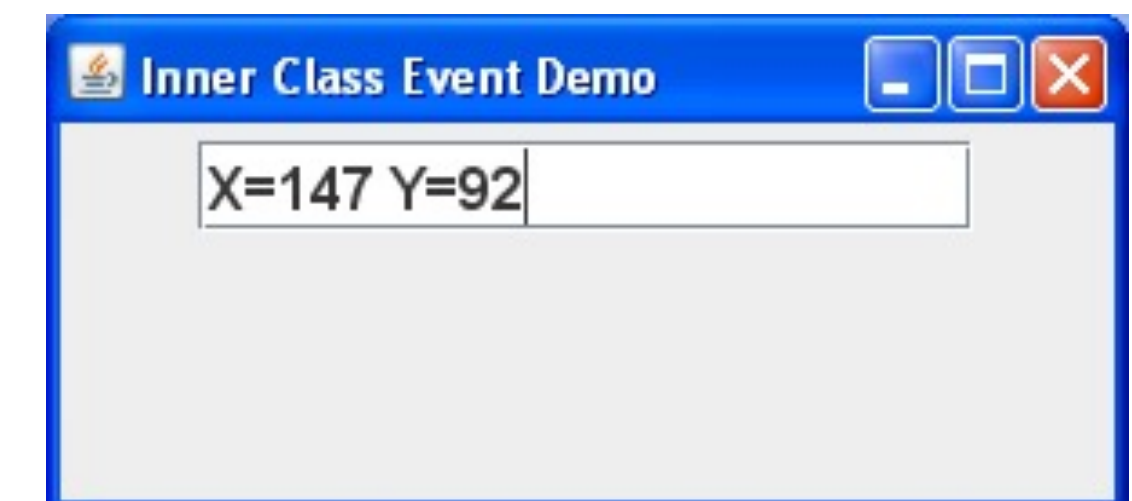
```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class EventDemo2 {
    private JFrame fr;
    private JTextField tf;
    public EventDemo2() {
        fr = new JFrame("Same Class Event Demo");
        tf = new JTextField(15);
        fr.setLayout(new FlowLayout());
        fr.add(tf);
        fr.addMouseMotionListener(new MouseHandler());
        fr.setSize(200, 200); fr.setVisible(true);
    }
    public class MouseHandler implements MouseMotionListener {
        public void mouseDragged(MouseEvent ev) {
            tf.setText("X=" + ev.getX() + " Y=" + ev.getY());
        }
        public void mouseMoved(MouseEvent ev) { }
    }
    public static void main(String[] args) {
        new EventDemo2();
    }
}
  
```

- คลาส **EventDemo2** มีการลงทะเบียนวัตถุ **Frame** เพื่อรับฟัง **MouseEvent** จากคำสั่ง

```
fr.addMouseMotionListener(
    new MouseHandler());
```

- โปรแกรมนี้จะแสดงตำแหน่งของเมาส์ เมื่อมีการเลื่อน
- คลาส **MouseHandler** เป็นคลาสภายใน ทำให้สามารถเรียกใช้คุณลักษณะของคลาส **EventDemo2** ได้เช่น วัตถุ **tf** ของคลาส **TextField**



# การจัดการกับเหตุการณ์ กรณีที่ 3 : คลาสเดียวกัน\*

```

import java.awt.event.*;
import javax.swing.JFrame;
public class EventDemo3 implements WindowListener{
    public EventDemo3() {
        JFrame fr = new JFrame("Same Class Event Demo");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr.addWindowListener(this);
        fr.setSize(200, 200);
        fr.setVisible(true);
    }
    public void windowClosing(WindowEvent ev) {
        System.exit(0);
    }
    public void windowOpened(WindowEvent ev) {}
    public void windowClosed(WindowEvent ev) {}
    public void windowIconified(WindowEvent ev) {}
    public void windowDeiconified(WindowEvent ev) {}
    public void windowActivated(WindowEvent ev) {}
    public void windowDeactivated(WindowEvent ev) {}
    public static void main(String[] args) {
        new EventDemo3();
    }
}
  
```

- **this** หมายถึง วัตถุในคลาสเดียวกัน

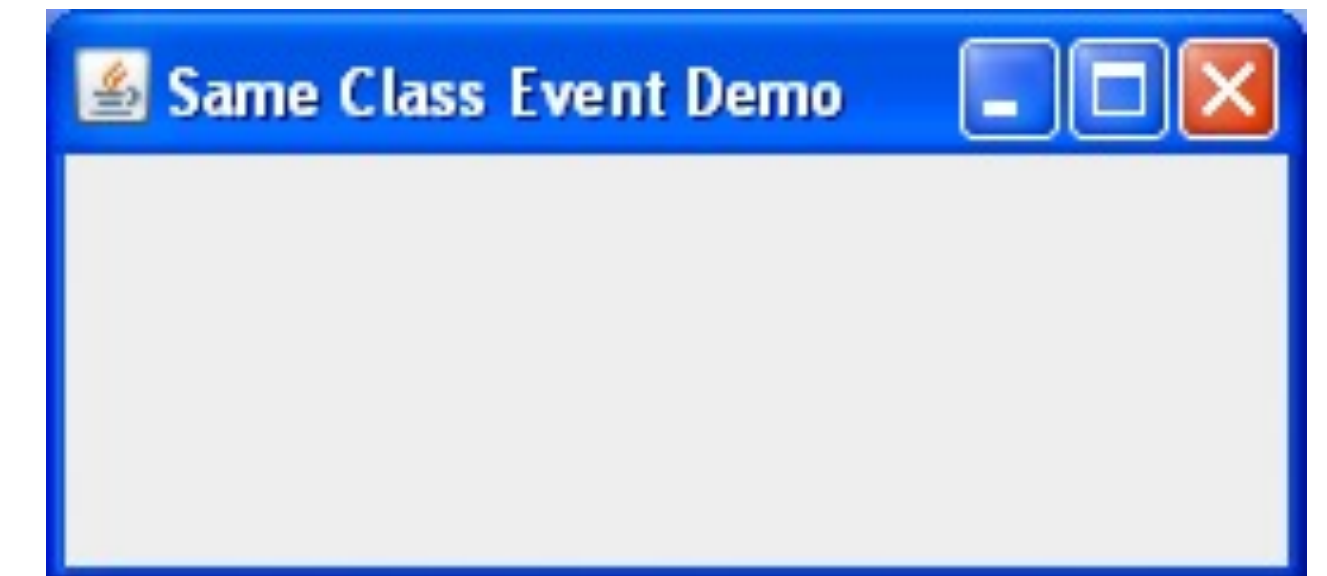
- การใช้วัตถุ **this** ในการจัดการกับ

**WindowEvent** จากคำสั่ง

```
fr.addWindowListener(this);
```

- คลาส **EventDemo3** ต้อง **implements** อินเตอร์เฟส **WindowListener**

- คำสั่ง **System.exit(0);** ทำให้จบโปรแกรม





# การจัดการกับเหตุการณ์ กรณีที่ 4 : คลาส **anonymous**

```
import java.awt.event.*;
import javax.swing.JFrame;
public class EventDemo6 {
    public EventDemo6() {
        JFrame fr = new JFrame("Anonymous Class Event");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent ev) {
                System.exit(0);
            }
        });
        fr.setSize(200, 200);
        fr.setVisible(true);
    }
    public static void main(String[] args) {
        EventDemo6 obj = new EventDemo6();
    }
}
```

กำหนดคลาสทั้งหมดให้อยู่ภายในเมธอดที่เรียกใช้งาน ซึ่งจะเรียกคลาสประเภทนี้ว่า **anonymous** นิยมใช้ในการจัดการกับ **event**

# การรับฟังเหตุการณั



การรับฟังหนึ่งเหตุการณั

การรับฟังหลายเหตุการณั



# การรับฟังหลายเหตุการณ์

```
import java.awt.*;
import java.awt.event.*;

public class EventDemo4 implements MouseMotionListener,
                                   WindowListener{

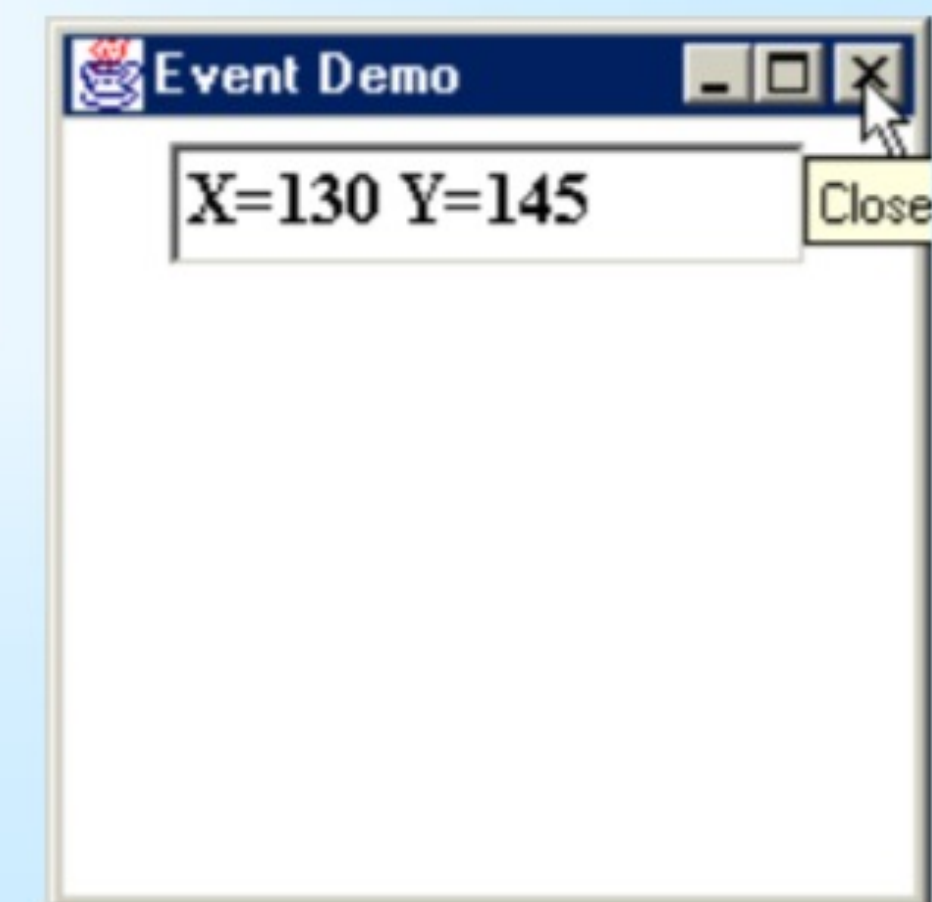
    private Frame fr;
    private TextField tf;
    public EventDemo4() {
        fr = new Frame("Event Demo");
        tf = new TextField(15);
        fr.setLayout(new FlowLayout());
        fr.add(tf);
        fr.setFont(new Font("TimesRoman",Font.BOLD,16));
        fr.addMouseMotionListener(this);
        fr.addWindowListener(this);
        fr.setSize(200,200);
        fr.setVisible(true);
    }
}
```



# การรับฟังหลายเหตุการณ์

```
public static void main(String args[]) {  
    EventDemo4 obj = new EventDemo4();  
}  
  
public void mouseDragged(MouseEvent ev) {  
    tf.setText("X="+ev.getX()+" Y="+ev.getY());  
}  
  
public void mouseMoved(MouseEvent ev) { }  
public void windowClosing(WindowEvent ev) {  
    System.exit(0);  
}  
  
public void windowOpened(WindowEvent ev) {}  
public void windowClosed(WindowEvent ev) {}  
public void windowIconified(WindowEvent ev) {}  
public void windowDeiconified(WindowEvent ev) {}  
public void windowActivated(WindowEvent ev) {}  
public void windowDeactivated(WindowEvent ev) {}  
}
```

ผลลัพธ์ที่ได้จากการรันโปรแกรม



# การรับฟังหลายเหตุการณ์



- คลาสประเภท Event Source สามารถที่จะขึ้นทะเบียนรับฟังได้ event หลายประเภท เช่น

**`fr.addMouseMotionListener(this);`** และ

**`fr.addWindowListener(this);`**

- คลาสใด ๆ สามารถ implements อินเตอร์เฟสได้หลายตัว เช่น

**`public class EventDemo4 implements MouseMotionListener,`**  
**`WindowListener`**

- Event Handler สามารถที่จัดการกับ Event Source ได้หลายตัวและวัตถุของคลาสประเภท Event จะมีเมธอดในการแยก Event Source ได้



# คลาสประเภท Event Adapter

```
import javax.swing.JFrame;
public class EventDemo5 {
    public EventDemo5() {
        JFrame fr = new JFrame("Adapter Class Event Demo");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr.addWindowListener(new WindowHandler());
        fr.setSize(200, 200);
        fr.setVisible(true);
    }
    public static void main(String[] args) {
        EventDemo5 obj = new EventDemo5();
    }
}
```

```
import java.awt.event.*;
public class WindowHandler extends WindowAdapter {
    public void windowClosing(WindowEvent ev) {
        System.exit(0);
    }
}
```

คลาสประเภท **Event Adapter** คือ  
คลาสที่ได้ implements อินเตอร์เฟส  
ประเภท Listener ไว้แล้ว และได้  
กำหนดเมธอดต่าง ๆ ของอินเตอร์เฟสที่ต้อง  
เขียนแล้ว แต่จะเป็นบล็อกคำสั่งที่ไม่มีคำสั่งใด  
ๆ อยู่ภายใน ซึ่งจะช่วยให้เขียนโปรแกรมคลาส  
ประเภท Event Handler ได้ง่ายขึ้น  
โดยลดจำนวนเมธอดที่ไม่ใช้ลง

# คลาสประเภท Event Adapter



```
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;

public class WindowAdapter implements WindowListener {
    @Override
    public void windowOpened(WindowEvent we) {}
    @Override
    public void windowClosing(WindowEvent we) {}
    @Override
    public void windowClosed(WindowEvent we) {}
    @Override
    public void windowIconified(WindowEvent we) {}
    @Override
    public void windowDeiconified(WindowEvent we) {}
    @Override
    public void windowActivated(WindowEvent we) {}
    @Override
    public void windowDeactivated(WindowEvent we) {}
}
```



# คลาสประเภท Event Adapter



คลาส Adapter	อินเทอร์เฟส
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
KeyAdapter	KeyListener
WindowAdapter	WindowListener
FocusAdapter	FocusListener

# ตัวอย่าง (Example)

```

public class App1 implements ActionListener {
    private String labelPrefix = "Number of button clicks: ";
    private int numClicks = 0;
    private JLabel label;
    private JFrame fr;
    private JButton button;
    private JPanel panel;
    public App1() {
        fr = new JFrame();
        label = new JLabel(labelPrefix + "0 ");
        button = new JButton("I'm a Swing button!");
        panel = new JPanel();

        panel.add(button); panel.add(label);
        fr.setContentPane(panel);
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        button.addActionListener(this);
        fr.pack();
        fr.setVisible(true);
    }
    public void actionPerformed(ActionEvent ev) {
        label.setText(labelPrefix + ++numClicks);
    }
    public static void main(String[] args) { new App1(); }
}

```



// ขั้นที่ 1 ประกาศ Component

// ขั้นที่ 2 สร้าง Component และกำหนดค่า

// ขั้นที่ 3 จัด Layout

// ขั้นที่ 4 กำหนดผู้จัดการเหตุการณ์

// ขั้นที่ 5 จัดการเหตุการณ์

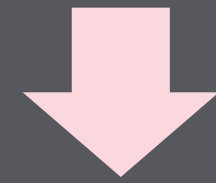
ตัวอย่างที่ 1

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class App2 implements ActionListener {
    private JButton b;
    private JFrame fr;
    public App2() {
        fr = new JFrame();
        b = new JButton("Click me!");

        fr.getContentPane().add(b);
        b.addActionListener(this);
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        fr.pack();
        fr.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        b.setBackground(Color.CYAN);
    }
    public static void main(String[] args) {
        new App2();
    }
}
```



ตัวอย่างที่ 2

```

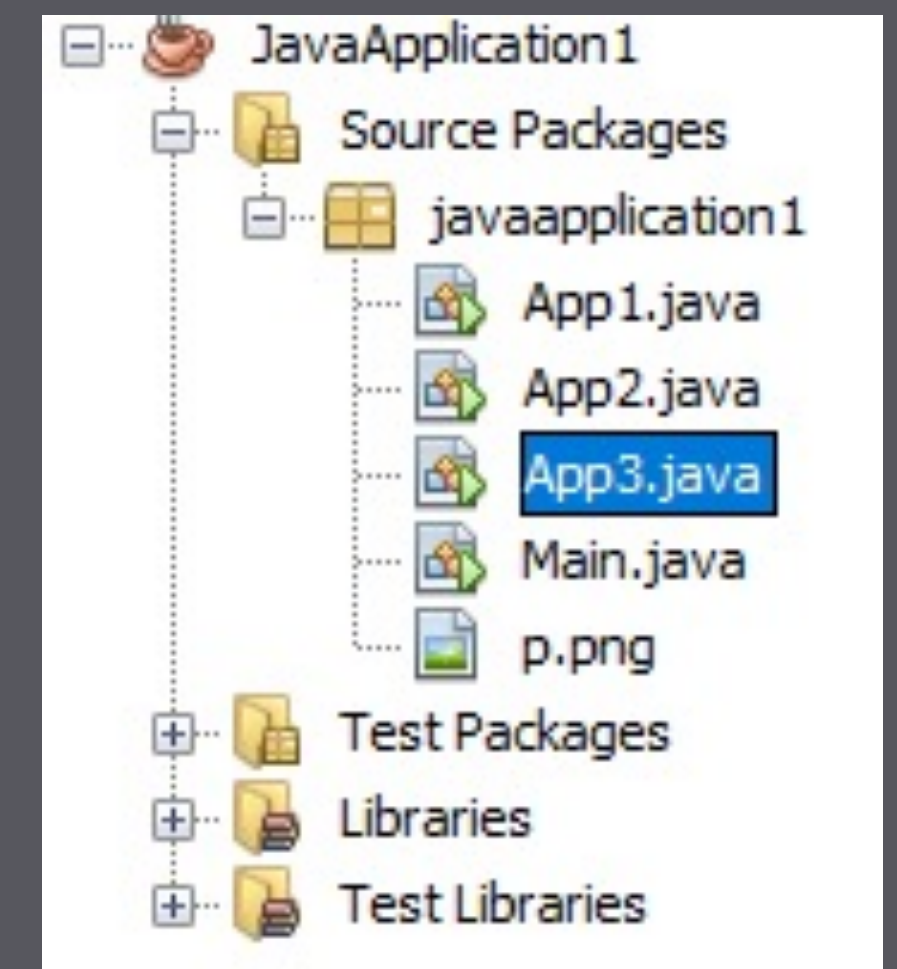
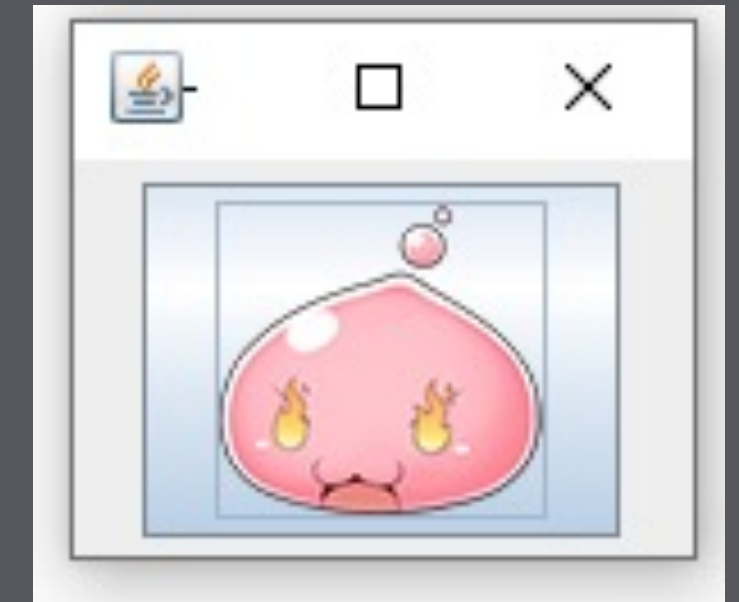
import javax.swing.*;
public class App3 {
    private JFrame frame;
    private ImageIcon icon;
    private JPanel panel;
    private JButton button;

    public App3() {
        frame = new JFrame("ImageIcon Example");
        icon = new ImageIcon("p.png");
        panel = new JPanel();
        button = new JButton(icon);

        panel.add(button);
        frame.getContentPane().add(panel);
        frame.pack();
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {
        new App3();
    }
}

```



ตัวอย่างที่ 3



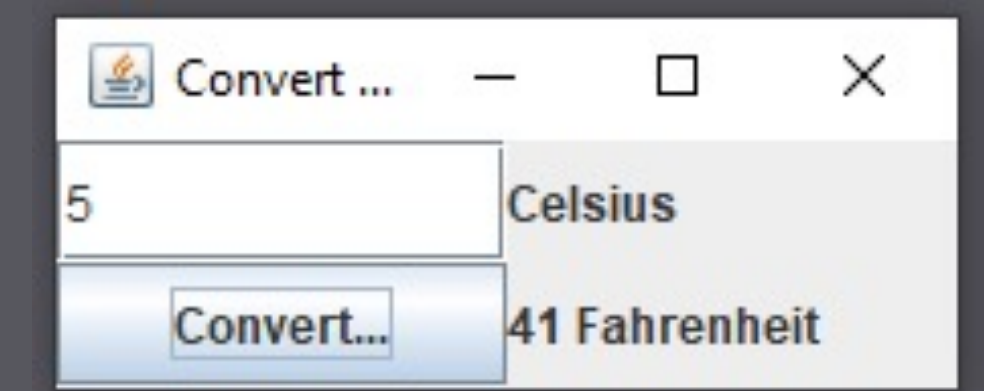
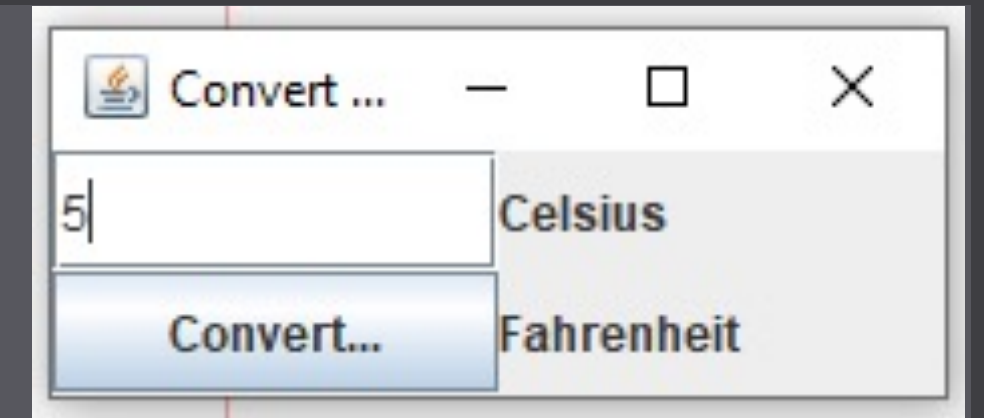
```

public class App4 implements ActionListener {
    private JFrame converterFrame;
    private JPanel converterPanel;
    private JButton convertTemp;
    private JTextField tempCelsius;
    private JLabel celsiusLabel, fahrenheitLabel;
    public App4() {
        converterFrame = new JFrame("Convert Celsius to Fahrenheit");
        converterPanel = new JPanel();
        convertTemp = new JButton("Convert...");    tempCelsius = new JTextField(2);
        celsiusLabel = new JLabel("Celsius", SwingConstants.LEFT);
        fahrenheitLabel = new JLabel("Fahrenheit", SwingConstants.LEFT);

        converterPanel.setLayout(new GridLayout(2, 2));
        converterPanel.add(tempCelsius);    converterPanel.add(celsiusLabel);
        converterPanel.add(convertTemp);    converterPanel.add(fahrenheitLabel);
        converterFrame.setContentPane(converterPanel);

        convertTemp.addActionListener(this);
        converterFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        converterFrame.pack();                converterFrame.setVisible(true);
    } public void actionPerformed(ActionEvent event) {
        int tempFahr = (int) ((Double.parseDouble(tempCelsius.getText())) * 1.8 + 32);
        fahrenheitLabel.setText(tempFahr + " Fahrenheit");
    } public static void main(String[] args) {    new App4(); }
}

```



ตัวอย่างที่ 4



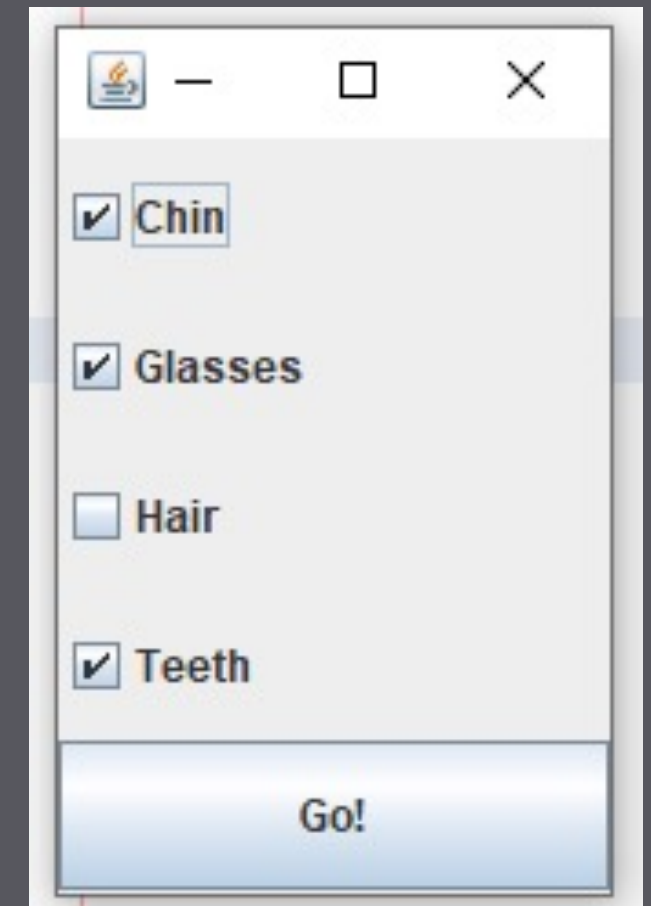
```

public class App5 implements ActionListener {
    private JFrame fr;
    private JPanel p;
    private JCheckBox chinButton, glassesButton, hairButton, teethButton;
    private JButton goButton;
    public App5() {
        fr = new JFrame();      p = new JPanel();      goButton = new JButton("Go!");
        chinButton = new JCheckBox("Chin");      glassesButton = new JCheckBox("Glasses");
        hairButton = new JCheckBox("Hair");      teethButton = new JCheckBox("Teeth");
        chinButton.setSelected(true);      glassesButton.setSelected(true);
        hairButton.setSelected(false);      teethButton.setSelected(true);

        p.setLayout(new GridLayout(0, 1));
        p.add(chinButton);      p.add(glassesButton);
        p.add(hairButton);      p.add(teethButton);      p.add(goButton);
        fr.setContentPane(p);

        goButton.addActionListener(this);
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr.pack();
        fr.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        if (glassesButton.isSelected()) { System.out.println("Glasses = true"); }
        else { System.out.println("Glasses = false"); }
        System.exit(0);
    }
    public static void main(String s[]) { new App5(); }
}

```



ตัวอย่างที่ 5

```

public class App6 implements ActionListener {
    private JFrame fr;
    private JRadioButton birdButton, catButton, dogButton, rabbitButton, pigButton ;
    private JButton goButton ;
    private ButtonGroup group;
    public App6() {
        fr = new JFrame();
        birdButton = new JRadioButton("Bird");
        catButton = new JRadioButton("Cat");
        dogButton = new JRadioButton("Dog");
        rabbitButton = new JRadioButton("Rabbit");
        pigButton = new JRadioButton("Pig");
        goButton = new JButton("Go!");
        birdButton.setSelected(true);
        group = new ButtonGroup();

        group.add(birdButton);          group.add(catButton);
        group.add(dogButton);          group.add(rabbitButton);      group.add(pigButton);
        fr.setLayout(new GridLayout(0, 1));
        fr.add(birdButton);             fr.add(catButton);         fr.add(dogButton);
        fr.add(rabbitButton);           fr.add(pigButton);         fr.add(goButton);

        goButton.addActionListener(this);
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr.pack(); fr.setVisible(true);
    }
}

```



ตัวอย่างที่ 6

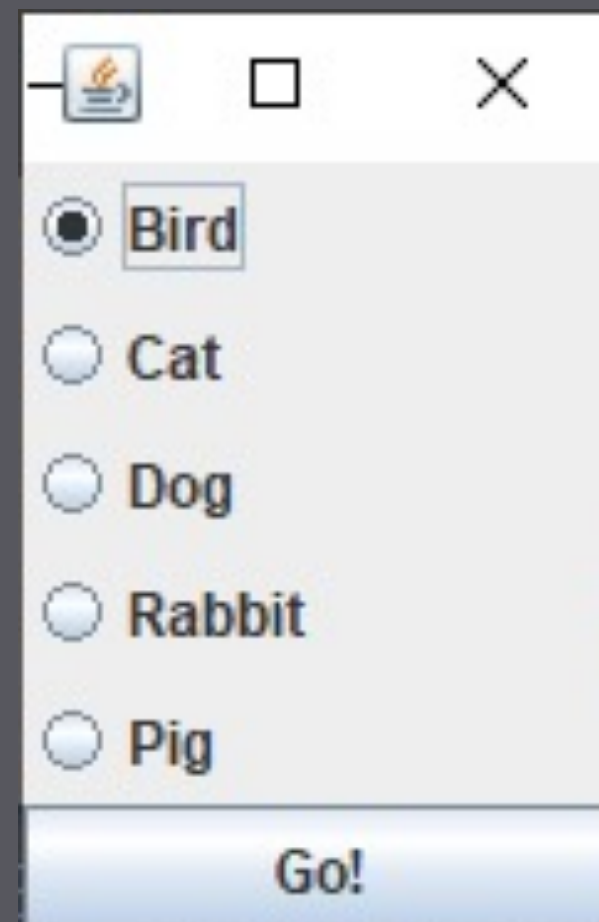
```

public class App6 implements ActionListener {
    private JFrame fr;
    private JRadioButton birdButton, catButton, dogButton, rabbitButton, pigButton ;
    private JButton goButton ;
    private ButtonGroup group;
    public App6() {

        .... // ดูจาก Slide ก่อนหน้า

    } public void actionPerformed(ActionEvent e) {
        if (birdButton.isSelected()) {          System.out.println("User finally selected bird.");    }
        else if (catButton.isSelected()) {        System.out.println("User finally selected cat.");      }
        else if (dogButton.isSelected()) {        System.out.println("User finally selected dog.");      }
        else if (rabbitButton.isSelected()) {     System.out.println("User finally selected rabbit."); }
        else if (pigButton.isSelected()) {        System.out.println("User finally selected pig.");    }
    } public static void main(String s[]) { new App6(); }
}

```



ตัวอย่างที่ 6 (ต่อ)