

# Event Driven & GUI Programming

การโปรแกรมแบบขับเคลื่อนด้วยอีเวนต์  
และส่วนต่อประสานกราฟิกกับผู้ใช้

*Compiled by Kanjana Eiamsaard, 2022-01-31*

# Agenda

- Introduction to event-driven programming
- Introduction to GUI
- AWT & Swing
- Frame/Window
- Swing components
- Event handler with inner class and anonymous inner class
- Layout
  - BorderLayout, FlowLayout, GridLayout and CardLayout

# Introduction to event-driven programming

- พิจารณาสถานการณ์ต่อไปนี้: ในช่วงก่อนการสอบกลางภาคอาจารย์มอบหมายให้นิสิตเขียนโปรแกรมคำนวณผลลัพธ์จากชุดข้อมูลขนาดใหญ่ ที่ใช้เวลาในการประมวลผลนาน และในระหว่างที่โปรแกรมคำนวณผลอยู่นั้น โปรแกรมต้องคอยรับข้อมูลเข้าจากผู้ใช้อย่างไร เช่น การยกเลิกการคำนวณ การเพิ่มข้อมูลเข้า เป็นต้น
- นิสิตจะทำอย่างไรกับโจทย์ในข้างต้น ด้วยความรู้ที่เรียนก่อนสอบกลางภาค ?

## Introduction to event-driven programming (ต่อ)

- จากตัวอย่าง “การยกเลิกการคำนวณ” ถือเป็นเหตุการณ์ที่อาจเกิดขึ้นหรือไม่เกิดขึ้นก็ได้ เราเรียกเหตุการณ์ลักษณะนี้ว่า “อีเวนต์ (event)”
- หากภาษาที่ใช้พัฒนาโปรแกรม มีความสามารถในการทำให้โปรแกรมรับรู้ถึงเหตุการณ์ที่มากระทำกับโปรแกรม ในขณะที่โปรแกรมเองยังทำงานอื่นอยู่ เราเรียกความสามารถนี้ว่า “การจัดการอีเวนต์ “event handling”
- พบการเขียนโปรแกรมที่ต้องมีการจัดการอีเวนต์ กับ โปรแกรมประเภท GUI

# ระบบอีเวนต์ของ Java

- การทำงานร่วมกันระหว่าง ส่วนประกอบ 3 ส่วนคือ **ผู้สร้างอีเวนต์** **ตัวอีเวนต์** และ**ผู้จัดการอีเวนต์**
- **ผู้สร้างอีเวนต์ (event creator)** คือ อ็อบเจกต์ใด ๆ ที่กำลังทำงานอยู่และเกิดเหตุการณ์บางอย่างขึ้นกับตัวเอง และต้องการแจ้งเหตุการณ์ที่เกิดขึ้นออกไปให้ระบบทราบ
- **ตัวอีเวนต์ (event)** คือ อ็อบเจกต์ที่รวบรวมข้อมูลของเหตุการณ์เอาไว้
- **ผู้จัดการอีเวนต์ (event handler)** คือ อ็อบเจกต์ที่ทำหน้าที่จัดการอ็อบเจกต์อีเวนต์ให้เป็นไปในรูปแบบที่นักพัฒนาต้องการ ซึ่งจะจัดการอ็อบเจกต์อีเวนต์ได้ ต้องทำการลงทะเบียนกับผู้สร้างอีเวนต์นั้นไว้ก่อน

## ระบบอีเวนต์ของ Java (ต่อ)

- อ็อบเจกต์ผู้สร้างฯ มักเกิดจากคลาสในกลุ่มคอมโพเนนต์ (component) มีหน้าที่จัดการ GUI
- อ็อบเจกต์อีเวนต์ เกิดจากคลาสกลุ่มอีเวนต์ ซึ่งเป็นซับคลาสของคลาส AWTEvent มีหน้าที่เก็บข้อมูลต่าง ๆ ที่เกี่ยวข้องกับอีเวนต์แต่ละประเภท
- อ็อบเจกต์ผู้จัดการอีเวนต์ เกิดจากคลาสที่ implement interface กลุ่มที่เป็น sub interface ของ EventListener
- อีเวนต์แต่ละประเภทมักจะมี sub interface ของ EventListener ที่ใช้งานคู่กัน เช่น
  - ActionEvent <-> ActionListener
  - KeyEvent <-> KeyListener
  - MouseEvent <-> MouseListener + MouseMotionListener + MouseWheelListener

# ตัวอย่างโปรแกรมที่มีระบบอีเวนต์

```
1 import javax.swing.Timer;
2 import java.awt.event.ActionListener;
3 import java.awt.event.*;
4
5 public class TimerEventTest {
6     private Timer timer; // ผู้สร้างอีเวนต์
7     private int count = 3;
8
9     private class TimerListener implements ActionListener {
10         @Override
11         // เมทอดที่ถูกเรียกใช้โดยผู้สร้างอีเวนต์เมื่อเกิดอีเวนต์ขึ้น
12         // โดยจะส่งอีเวนต์ที่เกิดขึ้นมาเป็นอาร์กิวเมนต์
13         public void actionPerformed(ActionEvent event) {
14             System.out.println("Timer event occurred....." + count +
15                 " at time " + event.getWhen());
16             count--;
17             if (count < 0) {
18                 timer.stop();
19             }
20         }
21     }
22
23     public TimerEventTest() {
24         // สร้างอีเวนต์ทุก 5 วินาที พร้อมการลงทะเบียนผู้จัดการอีเวนต์
25         timer = new Timer(5000, new TimerListener());
26         timer.start();
27     }
28 }
```

- Timer เป็นตัวอย่างอ็อบเจกต์ผู้สร้างอีเวนต์
- ActionEvent เป็นตัวอย่างอ็อบเจกต์อีเวนต์ชนิดหนึ่ง
- TimerListener เป็นตัวอย่างอ็อบเจกต์ผู้จัดการอีเวนต์
- ActionListener คือ interface คลาสที่ทำงานร่วมกับอีเวนต์ประเภท ActionEvent

## ตัวอย่างโปรแกรมที่มีระบบอีเวนต์ (ต่อ)

```
29     public static void main(String[] args) {  
30         TimerEventTest timerTest = new TimerEventTest();  
31         // ตรวจสอบที่ timer ยังทำงานอยู่ ให้วนลูปต่อไป  
32         while (timerTest.timer.isRunning()) {  
33               
34         }  
35           
36         // สิ้นสุดการทำงานในลูปคือจบโปรแกรม ถ้าไม่วนลูปโปรแกรมจะจบทันที  
37         // โดยไม่มีโอกาสด้กักอีเวนต์ที่จะเกิดขึ้นกับโปรแกรม  
38     }  
39 }
```


ส่วน main ต้องมีส่วนที่ตรวจว่ามีอีเวนต์ทำงานอยู่อย่างต่อเนื่อง

- หากไม่มีอีเวนต์ที่ทำงานแล้ว ให้ออกจากลูปและจบโปรแกรม
- หากมีอีเวนต์ที่ทำงานอยู่ ให้ตรวจการมีอยู่ของอีเวนต์ต่อไป





## ● ส่วนต่อประสานกราฟิกกับผู้ใช้

- องค์ประกอบของ GUI
  - Swing & AWT
  - GUI components
- 

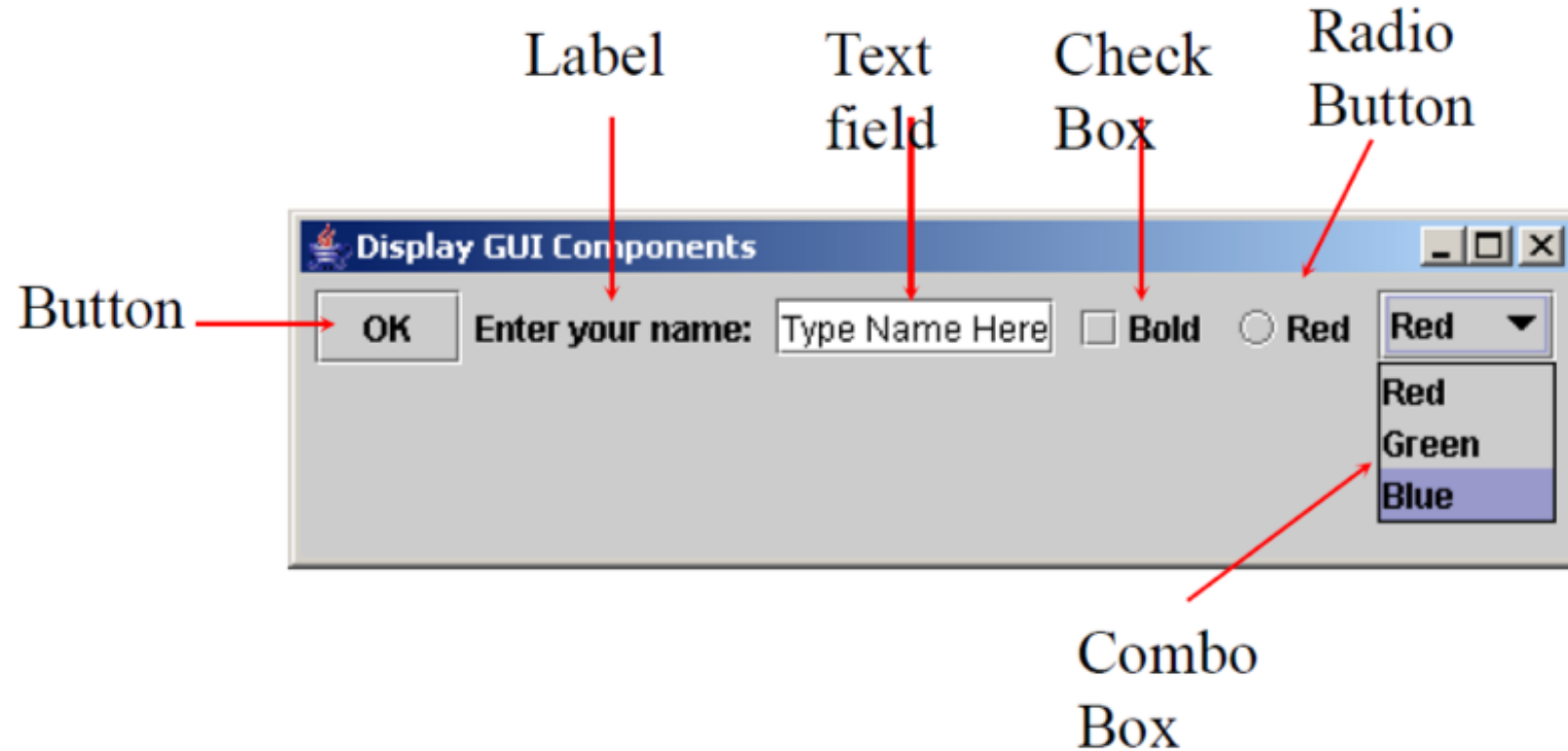
# ส่วนต่อประสานกราฟิกกับผู้ใช้

- Graphical User Interface (GUI ออกเสียงว่า กุย กูอี้ หรือ จียูไอ)
- คือ กลไกการโต้ตอบระหว่างผู้ใช้งานกับโปรแกรมโดยใช้ภาพเป็นสื่อกลางร่วมกับเครื่องมือนำข้อมูลเข้า เช่น เมาส์ คีย์บอร์ด เป็นต้น
- มักประกอบด้วย หน้าต่าง (window/frame) และ ส่วนประกอบย่อย (component)
- ตัวอย่าง component เช่น ปุ่มกด (button) ช่องข้อความ (text field) หรือเมนู เป็นต้น

## ส่วนต่อประสานกราฟิกกับผู้ใช้ (ต่อ)

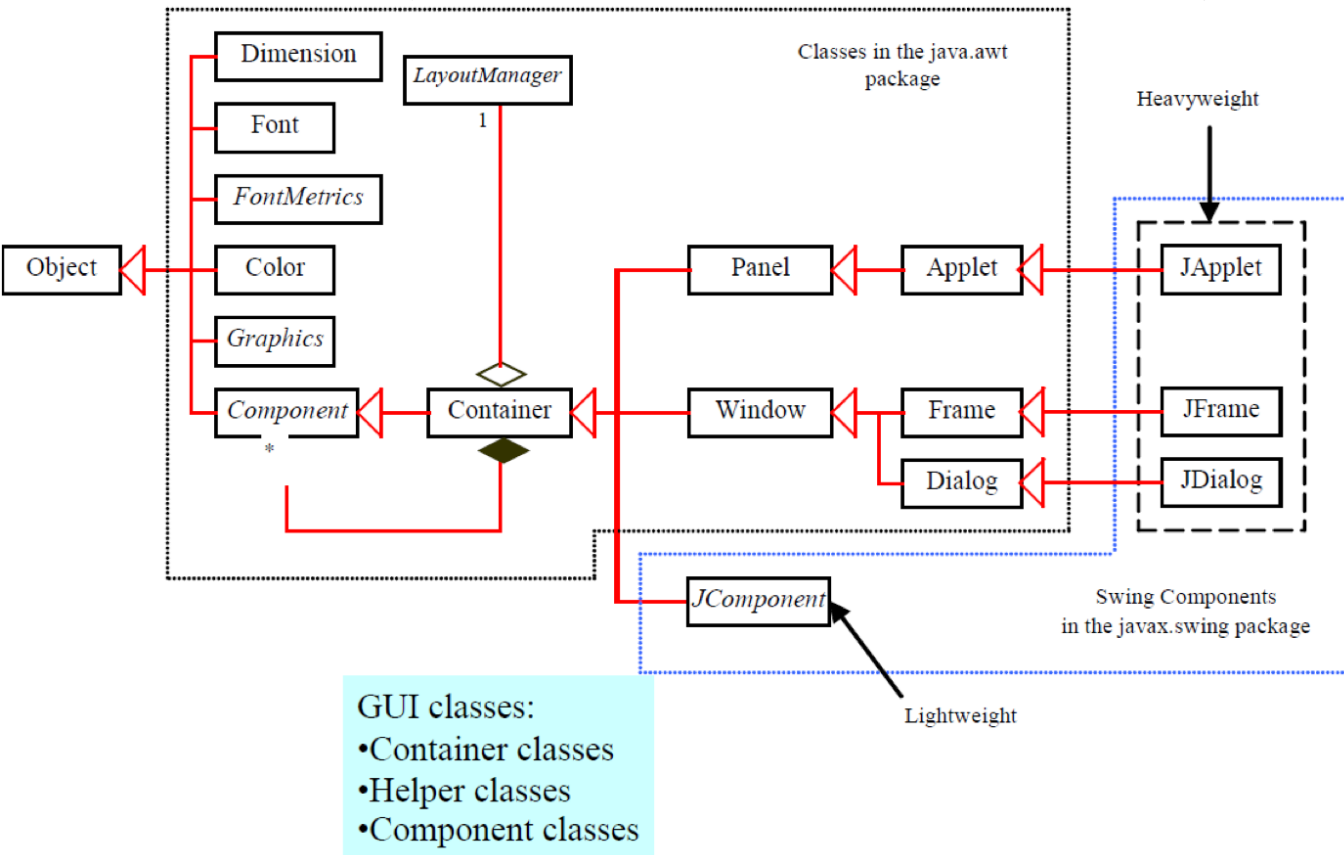
- ไลบรารีที่ใช้จัดการ GUI ใน Java รุ่นแรก ๆ มีชื่อว่า Abstract Window Toolkit (AWT)
  - มีหน้าต่างและส่วนประกอบย่อยเหมือนของระบบปฏิบัติการ (OS)
  - ข้อเสียคือ พฤติกรรมการทำงานยึดติดกับ OS มากเกินไป จึงเกิดความไม่เหมาะสมในบางกรณี
- Java 1.2 ปรับไลบรารีที่ใช้จัดการ GUI เป็น Swing
  - มีหน้าต่างและส่วนประกอบเฉพาะแบบของ Swing ไม่ขึ้นกับ OS
  - บางส่วนของ Swing ยังเรียกใช้ AWT อยู่ ยกเว้น Swing component
- ปัจจุบันเจ้าของ Java ประกาศให้ใช้ JavaFx แทน Swing
- คอร์สนี้ยังคงใช้ Swing เพราะเป็นพื้นฐานในการใช้ JavaFx ในอนาคตได้

# องค์ประกอบของ GUI



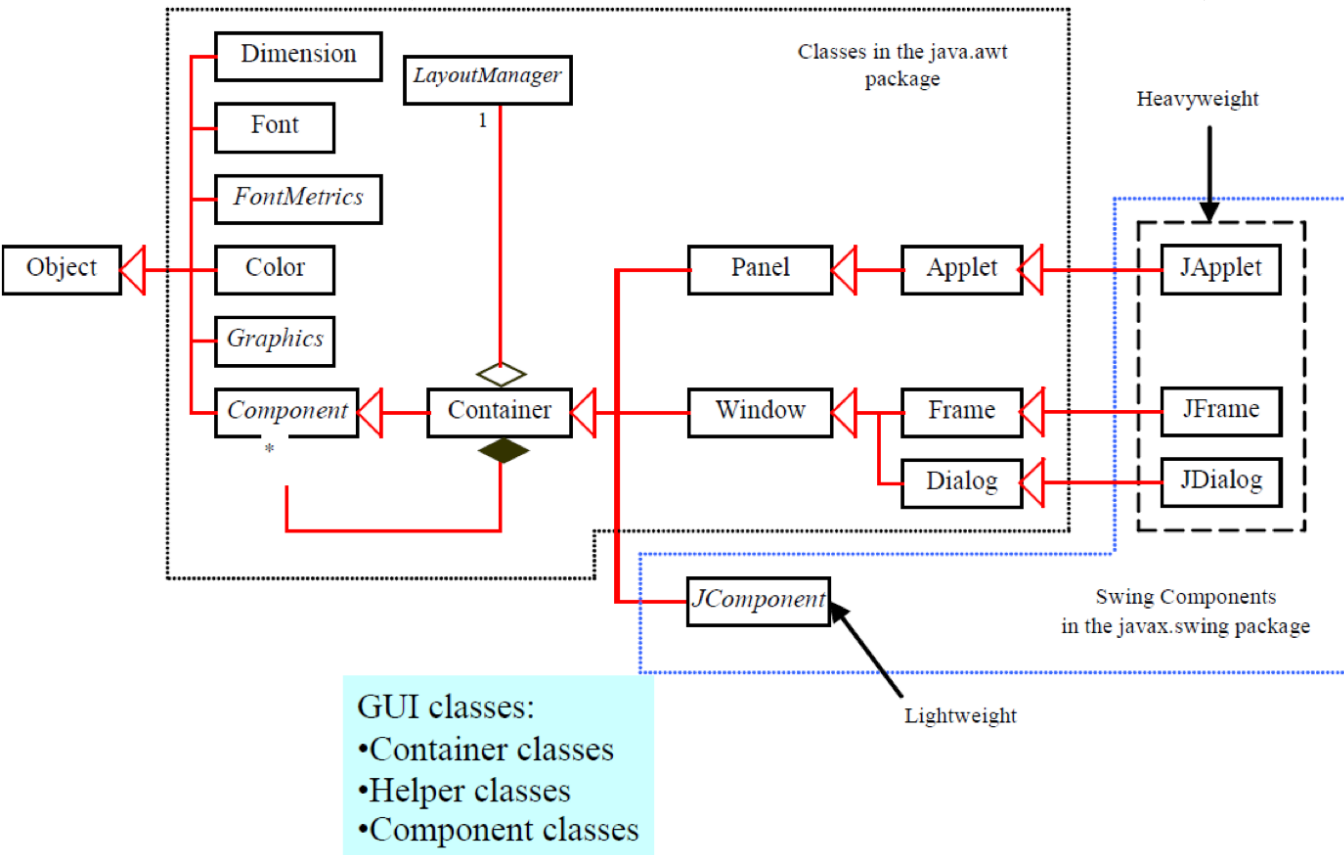
GUI	CLASS
หน้าต่าง/เฟรม (Window/Frame)	JFrame
พื้นที่ในหน้าต่าง นำมาใช้รวมส่วนประกอบย่อย (Panel)	JPanel

# ความสัมพันธ์ระหว่าง Swing และ AWT



- สิ่งที่เราได้เห็นทางจอภาพเป็น subclass ของคลาส Component
- สิ่งที่เราบรรจุ component เอาไว้ได้คือ container
- Container ใช้การจัดวางตำแหน่ง component ผ่านคลาส LayoutManager
- คลาสที่สืบทอดจาก Panel และ Window จะเกาะติดอยู่กับ OS เรียก “heavyweight component”

# ความสัมพันธ์ระหว่าง Swing และ AWT (ต่อ)



- ในส่วนของ Swing ทุก ๆ component ที่ไม่ใช่ (J)Frame, (J)Dialog และพาเนล ไม่เกาะติดกับ component ของ OS เรียก “Lightweight component”
- “Lightweight component” เป็นคลาสที่สืบทอดจาก container โดยตรง ทำให้ component ของ Swing สามารถบรรจุอ็อบเจกต์คอมโพเนนต์อื่น ๆ ในตัวได้

# Component AWT & Swing

AWT Comp.	Swing Comp.
Applet	JApplet
Component	JComponent
Container	-
Button	JButton
Canvas	-
CheckBox	JCheckBox
Dialog	JDialog
Frame	JFrame
Label	JLabel

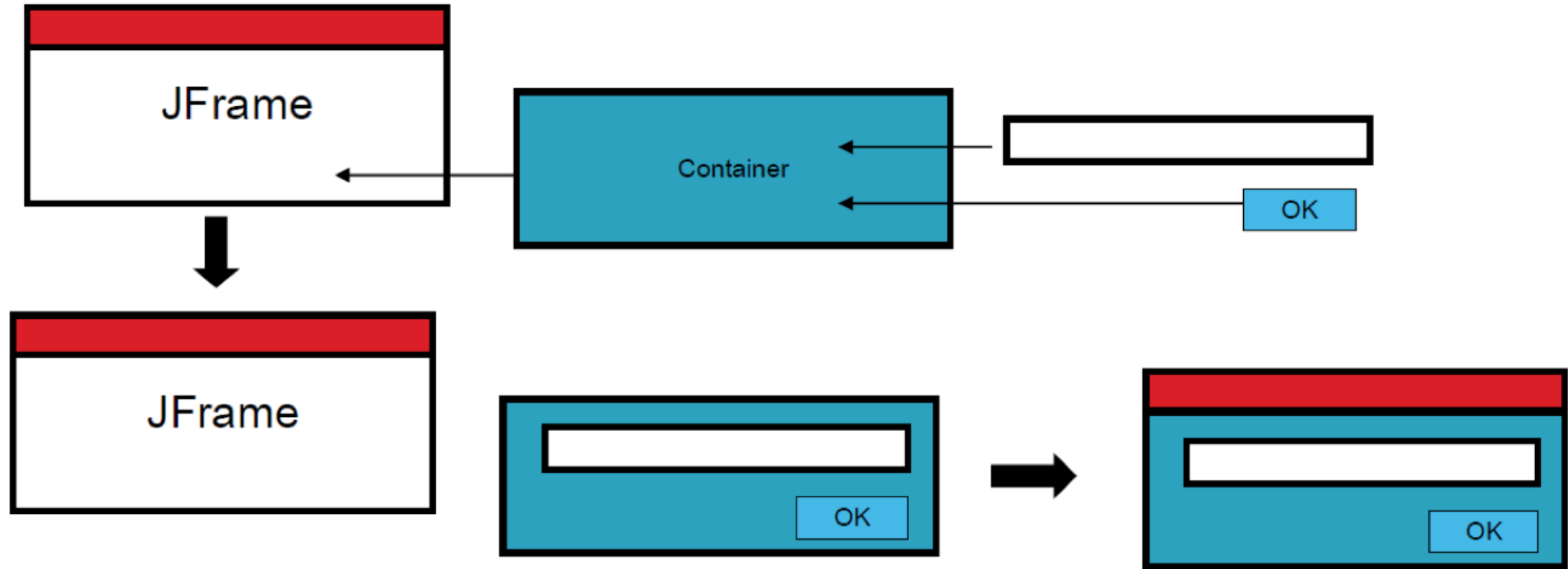
AWT Comp.	Swing Comp.
List	JList
Menu	JMenu
MenuBar	JMenuBar
MenuItem	JMenuItem
Panel	JPanel
ScrollBar	JScrollBar
TextArea	JTextArea
TextComponent	JTextComponent
TextField	JTextField

# Swing Component Class

Swing Component class	Component
javax.swing.JButton	Button
javax.swing.JCheckbox	Checkbox
javax.swing.JDialog	Dialog
javax.swing.JFrame	Frame
javax.swing.JLabel	Label
javax.swing.JMenu	Menu
javax.swing.JPanel	Panel
javax.swing.JScrollbar	Scrollbar
javax.swing.JScrollPane	Scroll pane
javax.swing.JTextArea	Text Area
javax.swing.JTextField	Text Field
javax.swing.JComboBox	Combo Box
javax.swing.JList	List



# โปรแกรมแบบ GUI เบื้องต้น



เริ่มต้นจากการสร้างเฟรม จากนั้นนำ component มาวางลงในเฟรมโดยตรง  
หรือ วางบน container ก่อนแล้วค่อยวาง container ในเฟรม

# ตัวอย่างการสร้างเฟรมเปล่าด้วย JFrame

```
1 import javax.swing.JFrame;
2
3 public class BlankFrame{
4     public static void main(String[] args) {
5         JFrame frame = new JFrame("Blank frame demo");
6         frame.setSize(400, 100);
7         frame.setVisible(true);
8     }
9 }
```



- สร้างหน้าต่างด้วย new JFrame พร้อมกำหนดชื่อ title bar
- setSize() กำหนดขนาดหน้าต่าง
- setVisible(true) สั่งให้หน้าต่างบนจอภาพ
- ถึงแม้สิ้นสุดการทำงานใน main แต่หน้าต่างยังอยู่ เพราะมีการแตก thread ที่ทำงานเป็น background ชื่อว่า  
“Event Dispatch Thread (EDT)”

# JFrame Class

`javax.swing.JFrame`

`+JFrame()`

`+JFrame(title: String)`

`+setSize(width: int, height: int): void`

`+setLocation(x: int, y: int): void`

`+setVisible(visible: boolean): void`

`+setDefaultCloseOperation(mode: int): void`

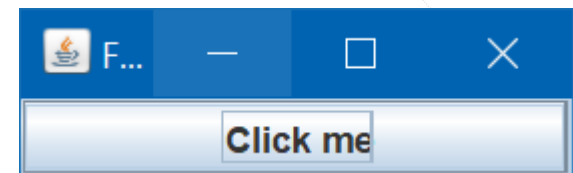
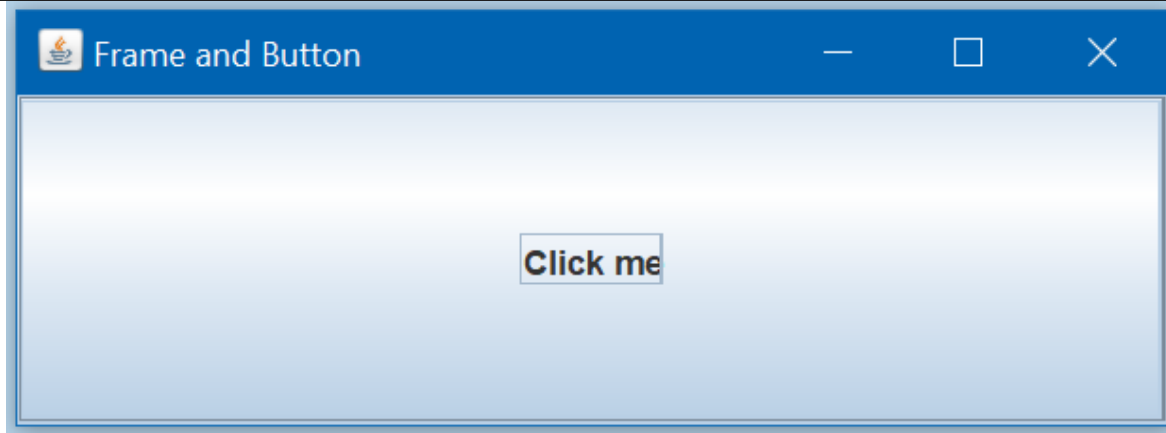
`+setLocationRelativeTo (c: Component):  
void`

# JFrame mostly used method

- `public JFrame` – ใช้สร้าง object แบบ JFrame
- `public JFrame(String s)` – ใช้สร้าง object JFrame โดยกำหนดชื่อ title ได้
- `public void setSize(int w, int h)` – ใช้กำหนดขนาดของ JFrame โดย w คือความกว้าง h คือความสูง หน่วยเป็น pixel
- `public void setTitle(String s)` – ใช้กำหนดชื่อบน Title Bar
- `public void setVisible(boolean b)` – ใช้กำหนดค่าการแสดง JFrame
- `public void setDefaultCloseOperation(int operation)` – ใช้กำหนดว่าเมื่อกดปุ่มปิด แล้วให้โปรแกรมเป็นอย่างไร ●

# ตัวอย่างการสร้างปุ่มกดลงในเฟรมเปล่า

```
1 import javax.swing.JButton;
2 import javax.swing.JFrame;
3
4 public class ButtonFrameDemo {
5     public static void main(String[] args) {
6         JFrame frame = new JFrame("Frame and Button");
7         JButton button = new JButton("Click me");
8         frame.add(button);
9         frame.setSize(400, 100);
10        frame.setVisible(true);
11    }
12 }
```



- ใช้ JButton ในการสร้างปุ่ม
- แต่ได้ปุ่มกดที่มีขนาดใหญ่เท่ากับหน้าต่าง
- แก้ไขโดยเรียกเมทอด pack() แทน setSize
- เพื่อให้ JFrame คำนวณขนาดของหน้าต่างที่เหมาะสมกับคอมโพเนนต์ที่มีอยู่ได้

# ตัวอย่างการเพิ่ม Lable ลงเฟรม และให้อยู่เหนือปุ่ม

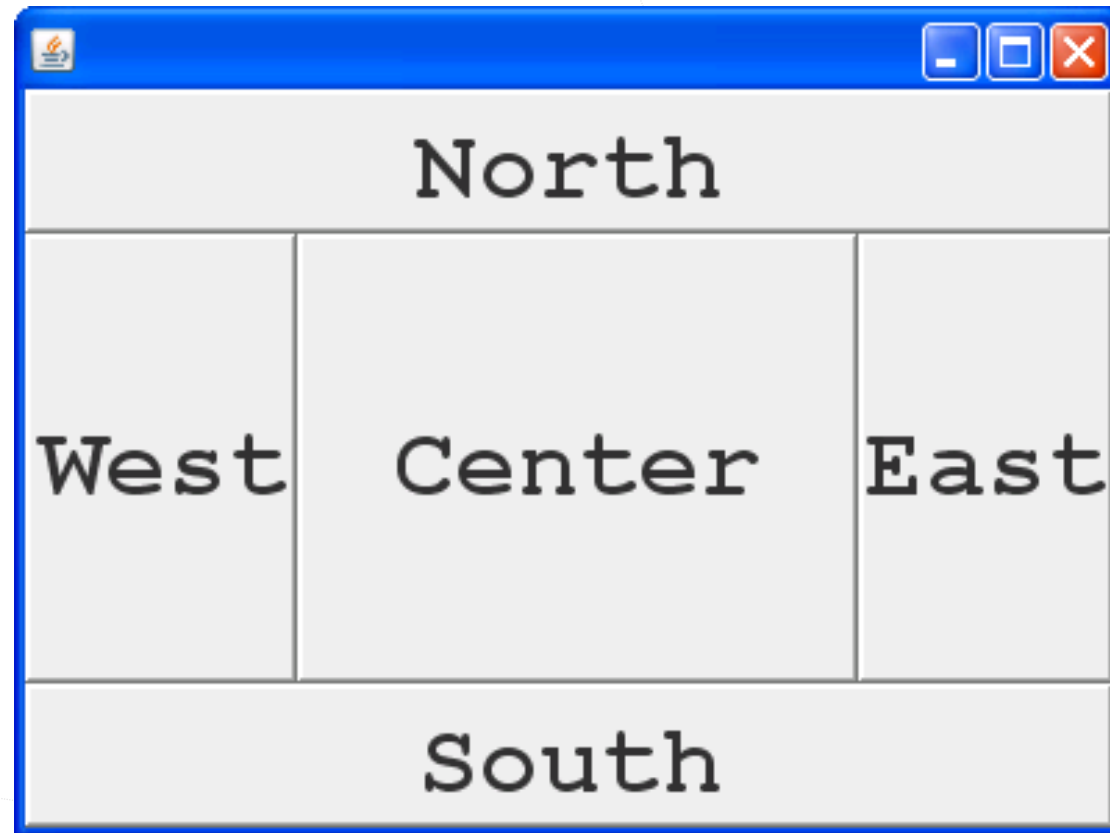
```
1 import javax.swing.JFrame;  
2 import javax.swing.JLabel;  
3 import javax.swing.JButton;  
4  
5 public class BorderLayoutDemo {  
6     public static void main(String[] args) {  
7         JFrame frame = new JFrame("Frame Demo");  
8         JLabel label = new JLabel("Click below:");  
9         JButton button = new JButton("Click!");  
10        frame.add(label);  
11        frame.add(button);  
12        frame.pack();  
13        frame.setVisible(true);  
14    }  
15 }
```



- ไม่แสดง Lable เพราะถูกทับด้วยปุ่ม
- เพราะ JFrame มีตำแหน่งในการจัดวางปริยายอยู่ที่ด้านบนสุด (North)
- ดังนั้นจึงต้องศึกษาพื้นที่การวาง component (BorderLayout)

# ตำแหน่งการจัดวางคอมโพเนนต์ในเฟรม (BorderLayout)

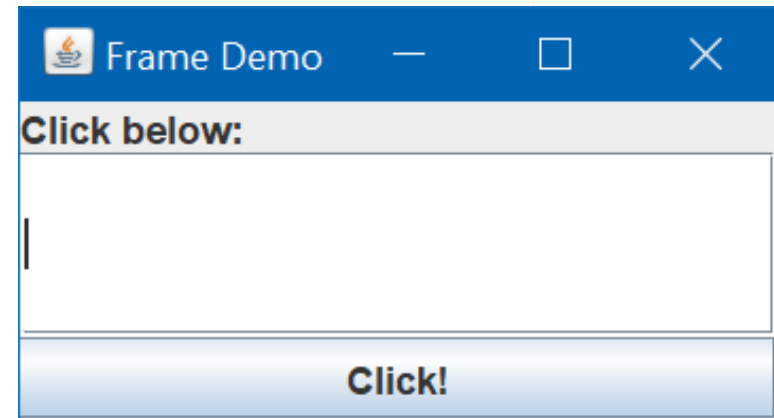
- มี 5 ตำแหน่ง ได้แก่ BorderLayout.NORTH, BorderLayout.WEST, BorderLayout.CENTER, BorderLayout.EAST, BorderLayout.SOUTH



# ตัวอย่างการจัดวางคอมโพเนนต์ในเฟรม

- สร้างคอมโพเนนต์แล้วนำมาจัดวางด้วย  
`add(COMP_NAME, LAYOUT)`

```
1  import javax.swing.JFrame;
2  import javax.swing.JLabel;
3  import javax.swing.JTextField;
4
5  import java.awt.BorderLayout;
6
7  import javax.swing.JButton;
8
9  public class BorderLayoutDemo {
10     public static void main(String[] args) {
11         JFrame frame = new JFrame("Frame Demo");
12         JLabel label = new JLabel("Click below:");
13         JButton button = new JButton("Click!");
14         JTextField text = new JTextField();
15         frame.add(label, BorderLayout.NORTH);
16         frame.add(text, BorderLayout.CENTER);
17         frame.add(button, BorderLayout.SOUTH);
18         frame.pack();
19         frame.setVisible(true);
20     }
21 }
```





## คลาส JPanel

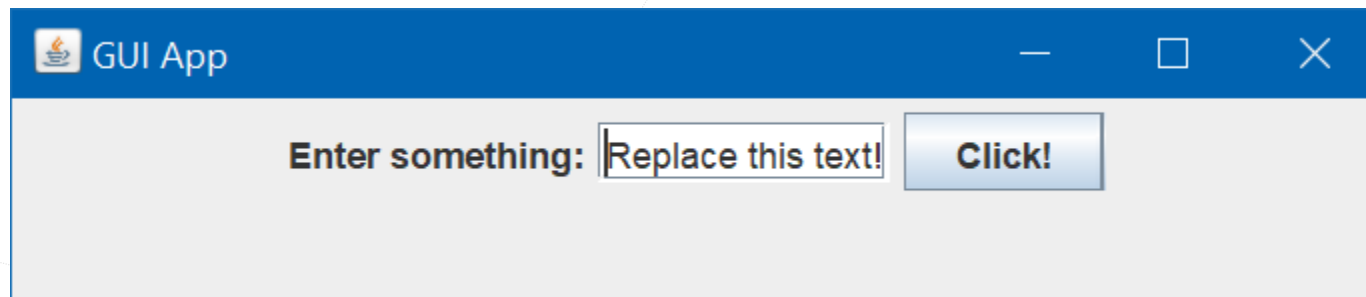
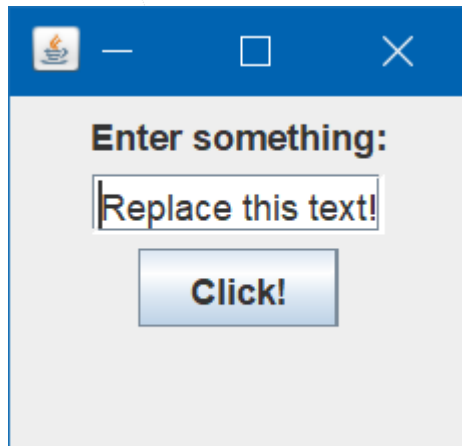
- ในทางปฏิบัติ ไม่ควรใส่และจัดหน้าตาของคอมโพเนนต์ลงไป ใน JFrame โดยตรง
- ทำให้การเปลี่ยนแปลงเกิดขึ้นได้ยาก
- นิยมจัดวางหน้าตาบน JPanel ก่อน แล้วจึงนำ JPanel มาวางลงใน JFrame
- เหตุผลที่ JPanel เปลี่ยนได้ง่ายกว่าคือ เราสามารถสร้าง JPanel ไว้หลาย ๆ แบบ แล้วเลือกมาวางลงใน JFrame เมื่อต้องการเปลี่ยนหน้า
- JPanel ไม่ได้ใช้การจัดวางผ่าน BorderLayout แต่ทำผ่าน **FlowLayout** ซึ่งเป็นการวางคอมโพเนนต์ตามลำดับการ add จากซ้ายไปขวา หากเต็มเฟรม จะวางต่อด้านล่างไปเรื่อย ๆ

# ตัวอย่างการใช้ JPanel

```
1 import javax.swing.JPanel;
2 import javax.swing.JLabel;
3 import javax.swing.JTextField;
4 import javax.swing.JButton;
5
6 public class AppPanel extends JPanel {
7
8     public AppPanel() {
9         JLabel label = new JLabel("Enter something:");
10        JTextField text = new JTextField("Replace this text!");
11        JButton button = new JButton("Click!");
12        add(label);
13        add(text);
14        add(button);
15    }
16 }
```

```
1 import javax.swing.JFrame;
2
3 public class MainApp {
4     public static void main(String[] args) {
5         JFrame frame = new JFrame("GUI App");
6         frame.add(new AppPanel());
7         frame.setSize(400, 300);
8         frame.setVisible(true);
9     }
10 }
```

“เปลี่ยนตำแหน่งการจัดวางอัตโนมัติตามขนาดเฟรม”



## คลาส Color

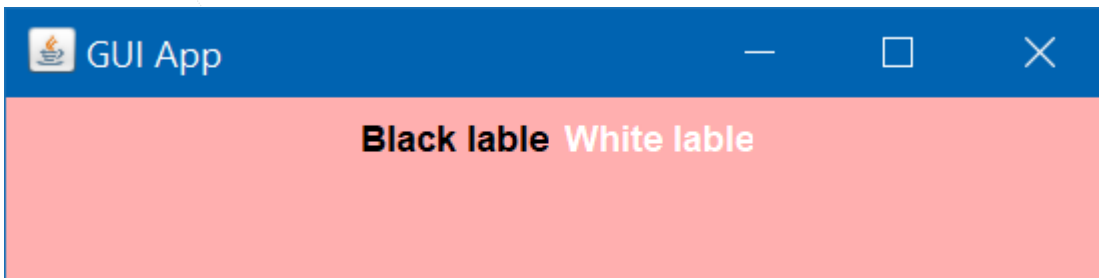
- ใช้กำหนดสีให้กับตัวอักษรและพื้นหลังของคอมโพเนนต์ รวมไปถึงใช้ในการกำหนดสีของจุดหรือเส้นเวลาที่มีการวาดลงบนคอมโพเนนต์ต่าง ๆ
- Java สร้างอ็อบเจกต์สีไว้ให้อยู่แล้ว โดยที่เราไม่ต้องสร้างใหม่ คือ Color.BLACK, Color.BLUE, Color.CYAN, Color.DARK\_GRAY, Color.GRAY, Color.GREEN, Color.LIGHT\_GRAY, Color.MAGENTA, Color.ORANGE, Color.PINK, Color.RED, Color.WHITE และ Color.YELLOW
- สร้างด้วยการผสมสี RGB ใหม่ได้ โดยค่ารหัสสี เป็น float (ช่วงของค่า 0.0-1.0) หรือ int (ช่วงของค่า 0-255) ก็ได้

```
Color myColor = new Color(red, green, blue);
```

# ตัวอย่างการใช้คลาส Color

```
6  public class ColorDemo extends JPanel {
7      private static final long serialVersionUID = 1L;
8  public ColorDemo(){
9      this.setBackground(Color.PINK);
10     JLabel label = new JLabel("Black lable");
11     label.setForeground(Color.black);
12     Color myColor = new Color(255,255,255);
13     JLabel label2 = new JLabel("White lable");
14     label2.setForeground(myColor);
15
16     add(label);
17     add(label2);
18 }
19 }
```

- กำหนดพื้นหลัง panel ด้วย setBackground()
- กำหนดสีป้ายกำกับ ด้วย setForeground()
- ผสมสีเองด้วยการสร้างอ็อบเจกต์ color



## คลาส Font

- สามารถกำหนดฟอนต์ที่จะใช้แสดงผลได้โดยการสร้างอ็อบเจกต์จากคลาส Font

**Font myFont = new Font(name, style, size);**

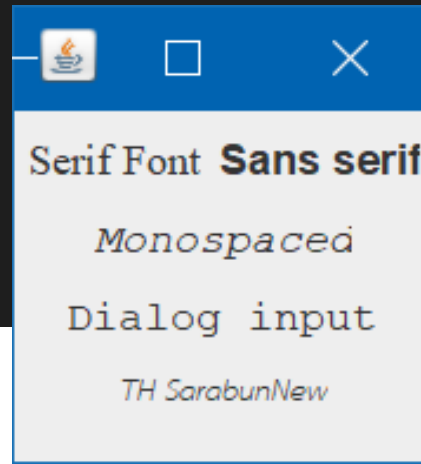
- **name** เป็นชื่อของฟอนต์ที่มีอยู่ในระบบ หรือเราสามารถระบุเป็นตระกูลของฟอนต์ได้ เช่น Font.SERIF, Font.SANS\_SERIF, Font.MONOSPACED, Font.DIALOG และ Font.DIALOG\_INPUT
- **style** คือรูปแบบของฟอนต์ สามารถกำหนดได้ดังนี้ Font.PLAIN, Font.BOLD, Font.ITALIC และรูปแบบผสม Font.BOLD+Font.ITALIC
- **size** คือขนาดของฟอนต์

# ตัวอย่างการใช้คลาส Font

```
public FontDemo(){
    JLabel seriJLabel = new JLabel("Serif Font");
    JLabel sansserifLabel = new JLabel("Sans serif");
    JLabel monoLabel = new JLabel("Monospaced");
    JLabel dialogInputLabel = new JLabel("Dialog input");
    JLabel thaiLabel = new JLabel("TH SarabunNew");

    Font serif = new Font(Font.SERIF,Font.PLAIN,12);
    seriJLabel.setFont(serif);
    Font sans = new Font(Font.SANS_SERIF,Font.BOLD,12);
    sansserifLabel.setFont(sans);
    Font mono = new Font(Font.MONOSPACED,Font.ITALIC,12);
    monoLabel.setFont(mono);
    Font dialog = new Font(Font.DIALOG_INPUT,Font.PLAIN,12);
    dialogInputLabel.setFont(dialog);
    Font th = new Font("TH Sarabun New",Font.BOLD+Font.CENTER_BASELINE,12);
    thaiLabel.setFont(th);

    add(seriJLabel);
    add(sansserifLabel);
    add(monoLabel);
    add(dialogInputLabel);
    add(thaiLabel);
}
```



- กำหนด Font ด้วยกลุ่ม เช่น Font.DIALOG\_INPUT
- กำหนด Font ด้วยชื่อ Font เช่น TH Sarabun New
- กำหนด style แบบเดียว เช่น Font.PLAIN
- กำหนด Style มากกว่า 1 แบบ เช่น Font.BOLD+Font.CENTER\_BASELINE
- กำหนดขนาดด้วยตัวเลข

# วิธีดูรายชื่อฟอนต์ที่มีในระบบ

- import java.awt.GraphicsEnvironment;

```
GraphicsEnvironment env = GraphicsEnvironment.getLocalGraphicsEnvironment();  
String[] fontNames = env.getAvailableFontFamilyNames();  
  
System.out.println(Arrays.toString(fontNames));
```

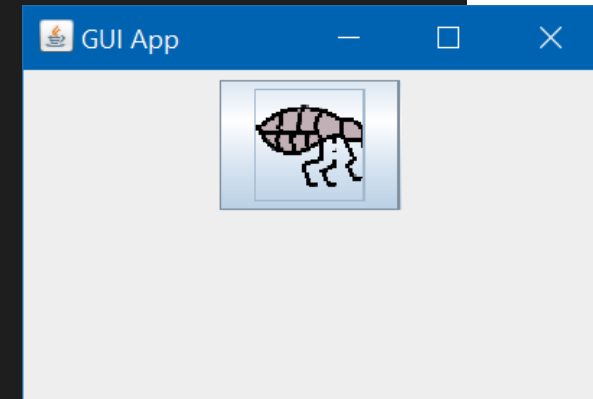


```
E:\playground\fund2\week08>java MainApp  
[Angsana New, AngsanaUPC, Arial, Arial Black, Arial Narrow, Bahnschrift, Book Antiqua, Bookman Old Style, Bookshelf Symbol  
7, Browallia New, BrowalliaUPC, Calibri, Calibri Light, Cambria, Cambria Math, Candara, Candara Light, Century, Comic San  
s MS, Consolas, Constantia, Corbel, Corbel Light, Cordia New, CordiaUPC, Courier New, Dialog, DialogInput, DilleniaUPC, Eb  
rima, EucrosiaUPC, Franklin Gothic Medium, FreesiaUPC, Gabriola, Gadugi, Garamond, Georgia, HoloLens MDL2 Assets, Impact,  
Ink Free, IrisUPC, JasmineUPC, Javanese Text, jdFontAwesome, jdFontCustom, jdIcoFont, jdIcoMoonFree, jdiconfontA, jdiconfo  
ntB, jdiconfontC, jdiconfontD, JdIonicons, KodchiangUPC, Leelawadee, Leelawadee UI, Leelawadee UI Semilight, LilyUPC, Luci  
da Console, Lucida Sans Unicode, Malgun Gothic, Malgun Gothic Semilight, Marlett, Microsoft Himalaya, Microsoft JhengHei,  
Microsoft JhengHei Light, Microsoft JhengHei UI, Microsoft JhengHei UI Light, Microsoft New Tai Lue, Microsoft PhagsPa, Mi  
crosoft Sans Serif, Microsoft Tai Le, Microsoft YaHei, Microsoft YaHei Light, Microsoft YaHei UI, Microsoft YaHei UI Light  
, Microsoft Yi Baiti, MingLiU-ExtB, MingLiU_HKSCS-ExtB, Mongolian Baiti, Monospaced, Monotype Corsiva, MS Gothic, MS PGoth  
ic, MS Reference Sans Serif, MS Reference Specialty, MS UI Gothic, MV Boli, Myanmar Text, Nirmala UI, Nirmala UI Semilight  
, NSimSun, Palatino Linotype, PMingLiU-ExtB, SansSerif, Segoe MDL2 Assets, Segoe Print, Segoe Script, Segoe UI, Segoe UI B  
lack, Segoe UI Emoji, Segoe UI Historic, Segoe UI Light, Segoe UI Semibold, Segoe UI Semilight, Segoe UI Symbol, Serif, Si  
mSun, SimSun-ExtB, Sitka Banner, Sitka Display, Sitka Heading, Sitka Small, Sitka Subheading, Sitka Text, Sylfaen, Symbol,  
Tahoma, TH Sarabun New, Times New Roman, Trebuchet MS, Verdana, Webdings, Wingdings, Wingdings 2, Wingdings 3, Yu Gothic,  
Yu Gothic Light, Yu Gothic Medium, Yu Gothic UI, Yu Gothic UI Light, Yu Gothic UI Semibold, Yu Gothic UI Semilight]
```

# คลาส Imgelcon

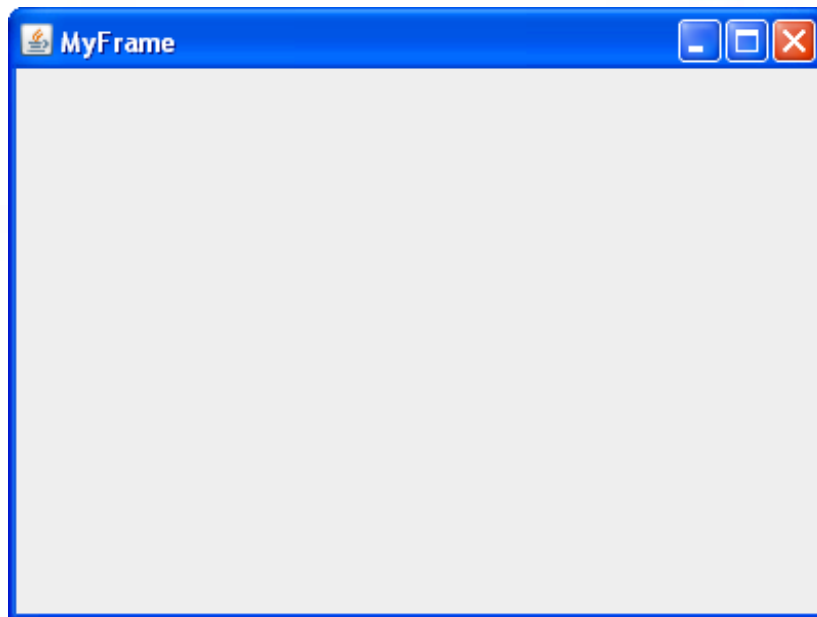
- ไอคอนคือรูปขนาดเล็กที่มักจะใช้ประกอบในคอมโพเนนต์อื่น

```
1  import javax.swing.ImageIcon;
2  import javax.swing.JButton;
3  import javax.swing.JPanel;
4
5  import java.awt.Image;
6
7  public class IconDemo extends JPanel{
8      private static final long serialVersionUID = 1L;
9      public IconDemo() {
10         ImageIcon icon = new ImageIcon(new ImageIcon("icon/flea.png").
11             |         |         getImage().getScaledInstance(50, 50, Image.SCALE_DEFAULT));
12         JButton button = new JButton(icon);
13         add(button);
14     }
15 }
```

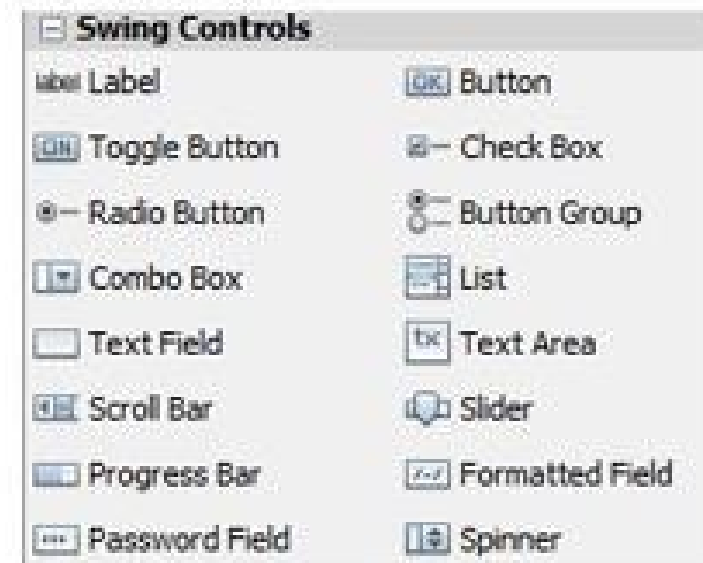




# คอมโพเนนต์ใน Swing



JFrame

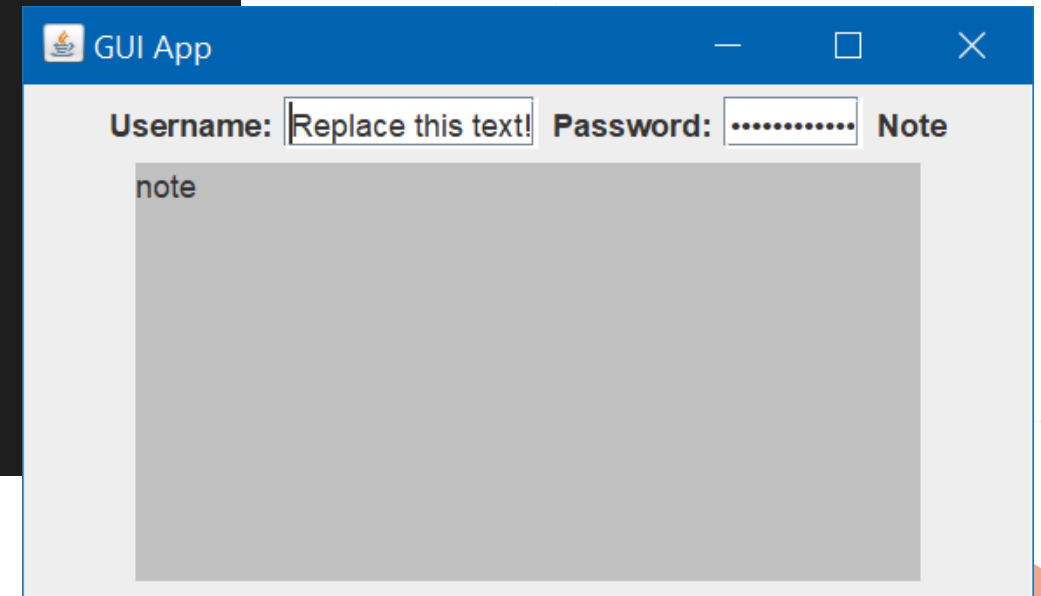


Swing Component

# คลาสสำหรับข้อความ

- JTextField, JPasswordField และ JTextArea
- สร้าง(ActionEvent) ถ้ามีการกดปุ่ม Enter ขณะป้อนข้อความ

```
13 public TextDemo() {  
14     JLabel usernameLabel = new JLabel("Username:");  
15     JLabel passwdLabel = new JLabel("Password:");  
16     JLabel noteLabel = new JLabel("Note");  
17     JTextField user = new JTextField("Replace this text!");  
18     JPasswordField pass = new JPasswordField("12characters");  
19     JTextArea note = new JTextArea("note", 10, 30);  
20     note.setBackground(Color.lightGray);  
21     add(usernameLabel);  
22     add(user);  
23     add(passwdLabel);  
24     add(pass);  
25     add(noteLabel);  
26     add(note);  
27 }
```

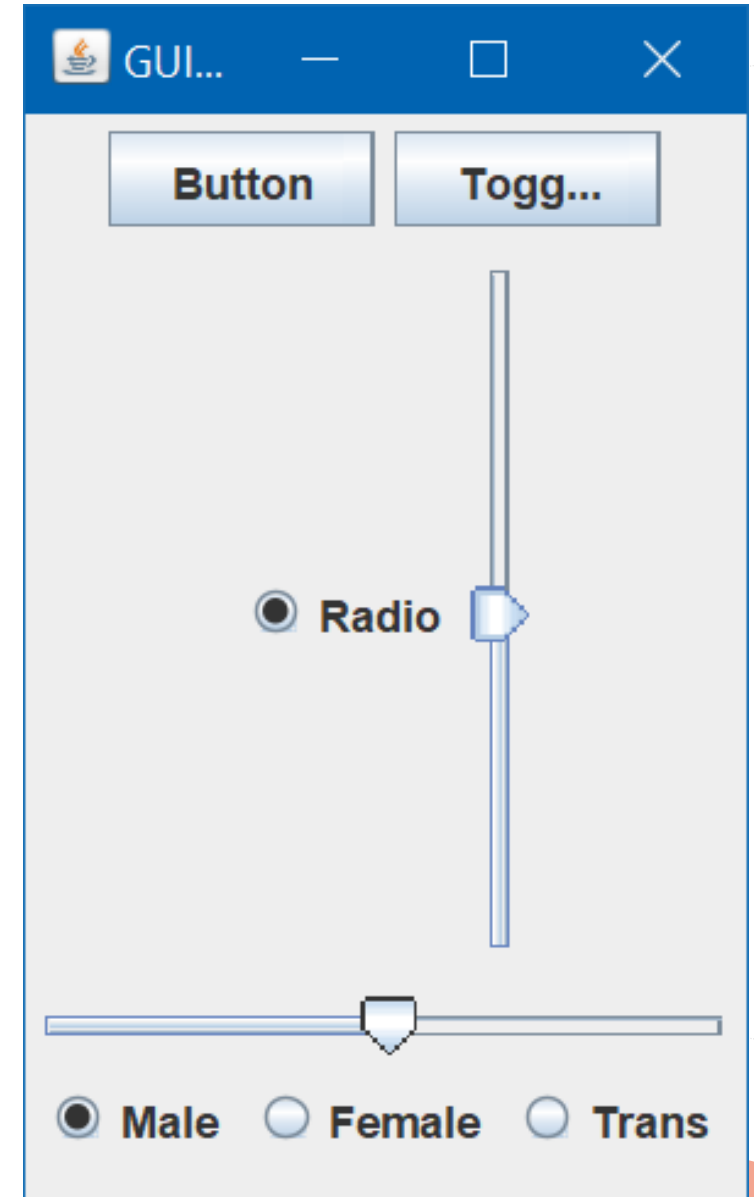


# คลาสสำหรับปุ่มกด/ประยุกต์ให้เหมือนปุ่มกด

- Button
- Toggle Button
- Radio Button
- ButtonGroup
- Vertical Slider
- Horizontal Slider

# ตัวอย่างการสร้างปุ่มกด/ประยุกต์ให้เหมือนปุ่มกด

```
11 public ButtonDemo() {  
12     JButton button = new JButton("Button");  
13     JToggleButton toggle = new JToggleButton("Toggle", false);  
14     JRadioButton rbutton = new JRadioButton("Radio", true);  
15     JSlider vslider = new JSlider(JSlider.VERTICAL);  
16     JSlider hslider = new JSlider(JSlider.HORIZONTAL);  
17  
18     JRadioButton rmale = new JRadioButton("Male", true);  
19     JRadioButton rfemale = new JRadioButton("Female", true);  
20     JRadioButton rtran = new JRadioButton("Trans", true);  
21     ButtonGroup gender = new ButtonGroup();  
22     gender.add(rmale);  
23     gender.add(rfemale);  
24     gender.add(rtran);  
25  
26     add(button);  
27     add(toggle);  
28     add(rbutton);  
29     add(vslider);  
30     add(hslider);  
31     add(rmale);  
32     add(rfemale);  
33     add(rtran);  
34 }
```



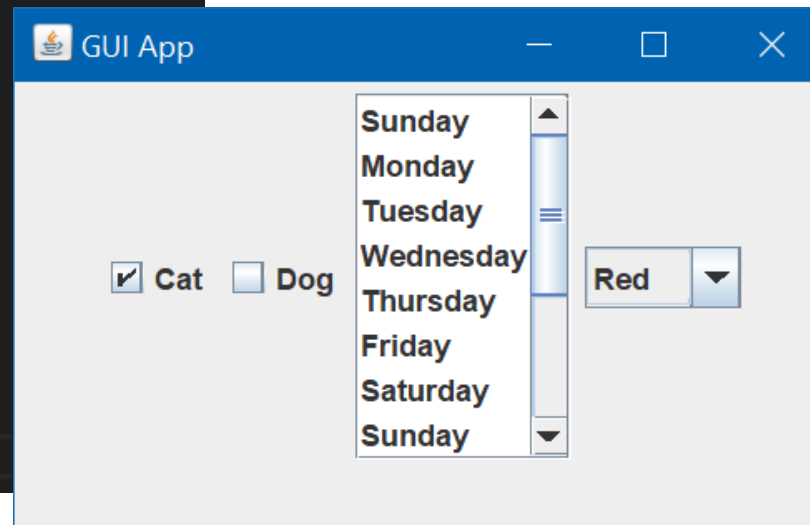
# คลาสสำหรับตัวเลือก

- CheckBox
- List + ScrollBar
- ComboBox

# ตัวอย่างการใช้งานคลาสสำหรับตัวเลือก

```
9 public ListDemo(){
10
11     JCheckBox chkChoice1 = new JCheckBox("Cat",true);
12     JCheckBox chkChoice2 = new JCheckBox("Dog");
13
14     String[] days = new String[] { "Sunday", "Monday",
15         "Tuesday", "Wednesday", "Thursday",
16         "Friday", "Saturday", "Sunday", "Monday",
17         "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday" };
18     JList<String> dayList = new JList<String>(days);
19     JScrollPane scrollPane = new JScrollPane(dayList);
20     scrollPane.setVerticalScrollBarPolicy(20); //20 = as need
21     scrollPane.setHorizontalScrollBarPolicy(
22         JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
23
24     JComboBox<String> comboBox = new JComboBox<String>
25         (new String[] { "Red", "Green", "Blue" });
26
27     add(chkChoice1);
28     add(chkChoice2);
29     add(scrollPane);
30     add(comboBox);
31 }
32
```

- กำหนด true ในการสร้างอ็อบเจกต์ JCheckBox เพื่อแสดงการเช็ค
- JScrollPane ในแนวตั้งกำหนดให้แสดงเมื่อมีข้อมูลมาก ด้วยการกำหนดเป็นจำนวนเต็ม
- JScrollPane ในแนวนอนกำหนดให้แสดงเมื่อมีข้อมูลมาก ด้วยการกำหนดเป็นค่าคงที่
- สร้าง combo box ด้วยอ็อบเจกต์สตริง





## การจัดการอีเวนต์บน GUI

- Inner class
- คลาสนิรนาม (anonymous inner class)

# การจัดการอีเวนต์บน GUI

- องค์ประกอบที่เกี่ยวข้องกับระบบอีเวนต์ใน Java มี 3 ส่วนด้วยกันคือ
  - Event creator = GUI Component object
  - Event = Object ที่เก็บข้อมูลเหตุการณ์ที่เกิดขึ้นกับ GUI Component
  - **Event handler = Object สำหรับจัดการอีเวนต์**
- ขั้นตอนในการจัดการอีเวนต์บน GUI มีดังนี้
  1. สร้างคลาสที่ implement interface ในกลุ่ม “Listener” ที่ตรงกับประเภทของอีเวนต์ เช่น สร้างคลาสที่ implement interface “ActionListener” เมื่อมีอีเวนต์ประเภท “ActionEvent”



## การจัดการอีเวนต์บน GUI (ต่อ)

- ขั้นตอนในการจัดการอีเวนต์บน GUI มีดังนี้
  - 2. implement method ภายในคลาสที่สร้างในขั้นตอนที่ 1 คือ actionPerformed() ซึ่งเป็นเมทอดที่จะถูกเรียกเมื่อเกิดอีเวนต์ขึ้น
  - 3. ลงทะเบียนอ็อบเจกต์ของคลาสในขั้นตอนที่ 1 ด้วยเมทอดของ GUI component มักจะตั้งชื่อตามประเภทของอีเวนต์คือ addXXXListener เช่น GUI ที่มีอีเวนต์เป็น action จะมีเมทอดชื่อ addActionListener() เป็นเมทอดสำหรับลงทะเบียน

## การจัดการอีเวนต์บน GUI (ต่อ)

- คลาสที่เกิดมาเพื่อจัดการอีเวนต์มักสร้างด้วยเทคนิค **inner class** เนื่องจากการจัดการอีเวนต์ของแต่ละส่วนเป็นกระบวนการเฉพาะที่ไม่น่าใช้ซ้ำในที่อื่นได้
- การใช้คลาสภายในจึงเป็นการป้องกันคลาสจัดการอีเวนต์นี้จากการถูกเรียกใช้ภายนอก

# ตัวอย่างการจัดการอีเวนต์ด้วยเทคนิค inner class

```
9 public class SimpleInnerclass extends JPanel {
10     private static final long serialVersionUID =
11     private JLabel label;
12     private JTextField txt;
13     private JButton addX;
14     private JButton addY;
15     public SimpleInnerclass() {
16         label = new JLabel("Enter X or Y");
17         txt = new JTextField(10);
18         addX = new JButton("X");
19         addY = new JButton("Y");
20
21         /* Event handler zone */
22         ButtonListener listener = new ButtonListener();
23         addX.addActionListener(listener);
24         addY.addActionListener(listener);
25         /* End event handler*/
26
27         add(label);
28         add(txt);
29         add(addX);
30         add(addY);
31     }
32
33     private class ButtonListener implements ActionListener {
34         @Override
35         public void actionPerformed(ActionEvent event) {
36             if(event.getSource() == addX)
37                 txt.setText(txt.getText()+"X");
38             else
39                 txt.setText(txt.getText()+"Y");
40         }
41     } //end class
```

- ลงทะเบียน listener กับ component ผ่านเมทอด addXXXXListener
- ButtonListener เป็น inner class ที่ implement มาจากคลาสกลุ่ม Listener ที่รองรับอีเวนต์แบบ action

## คลาสภายในแบบนิรนาม (Anonymous inner class)

- สังเกตการสร้างคลาส ButtonListener จากตัวอย่างก่อนหน้านี้ พบว่า “มีการใช้อ็อบเจกต์ของตัวจัดการร่วมกันระหว่าง component”
- ในทางปฏิบัติแล้ว การใช้อ็อบเจกต์หรือแม้แต่คลาสจัดการอีเวนต์ร่วมกันเป็นสิ่งที่ควรหลีกเลี่ยง (เพราะจะทำให้โค้ดซับซ้อนโดยไม่จำเป็น)
- นิยมใช้หนึ่งคลาสต่อหนึ่งอีเวนต์ แล้วโค้ดจะเยอะและเยิ่นเย้อเกินไปหรือไม่ ?
  - Java มีกลไกที่ทำให้การเขียนคลาสลักษณะนี้กระชับขึ้นโดยใช้

**“คลาสนิรนาม”**

# ตัวอย่างการใช้คลาสนิรนาม

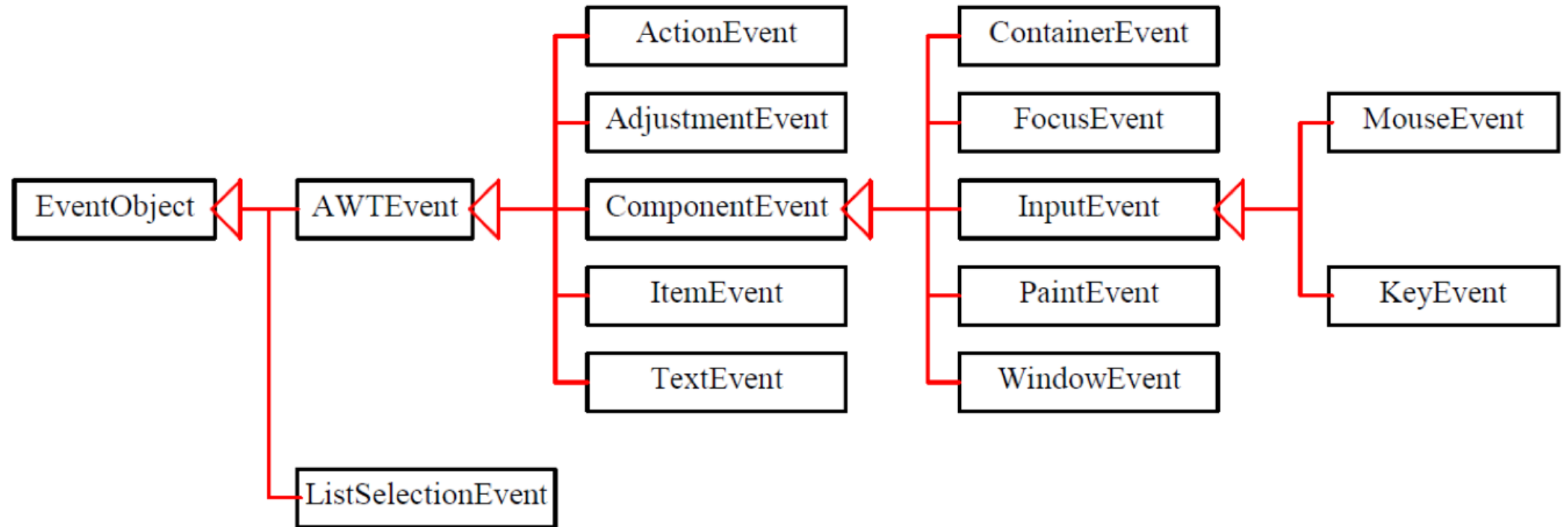
```
16 public AnonymousClass() {  
17     label = new JLabel("Enter X or Y");  
18     txt = new JTextField(10);  
19     addX = new JButton("X");  
20     addY = new JButton("Y");  
21     /* Event handler zone */  
22     addX.addActionListener(new ActionListener(){  
23         @Override  
24         public void actionPerformed(ActionEvent e) {  
25             txt.setText(txt.getText()+"X");  
26         }  
27     });  
28     addY.addActionListener(new ActionListener(){  
29         @Override  
30         public void actionPerformed(ActionEvent e) {  
31             txt.setText(txt.getText()+"Y");  
32         }  
33     });  
34     /* End event handler */  
35     add(label);  
36     add(txt);  
37     add(addX);  
38     add(addY);  
39 }  
40 }
```

- จากโค้ดก่อนหน้านี้ เราลบคลาส ButtonListener ออก
- สร้าง anonymous inner class ในฐานะ argument ของ method addActionListener

# เปรียบเทียบการใช้คลาสนิรนามกับคลาสนิรนามแบบ Lambda

```
/* Event handler zone */
addX.addActionListener(new ActionListener(){
    @Override
    public void actionPerformed(ActionEvent e) {
        txt.setText(txt.getText()+"X");
    }
});
addY.addActionListener(e -> txt.setText(txt.getText()+"Y"));
/* End event handler */
```

# อีเวนต์ใน Java



# ตัวอย่างของอีเวนต์ของแต่ละคอมโพเนนต์

User Action	Source Object	Event Type Generated
Click a button	JButton	ActionEvent
Press return on a text field	TextField	ActionEvent
Select a new item	JComboBox	ItemEvent, ActionEvent
Select item(s)	JList	ListSelectionEvent
Click a check box	JCheckBox	ItemEvent, ActionEvent
Click a radio button	JRadioButton	ItemEvent, ActionEvent
Select a menu item	JMenuItem	ActionEvent
Move the scroll bar	JScrollBar	AdjustmentEvent
Window opened, closed, etc.	Window	WindowEvent
Mouse pressed, released, etc.	Component	MouseEvent
Key released, pressed, etc.	Component	KeyEvent
Component added or removed etc.	Container	ContainerEvent
Component moved, etc.	Component	ComponentEvent
Component gained or lost focus	Component	FocusEvent

☞ Table lists external user actions, source objects, and event types generated.





## GUI components (ต่อ)

- JMenuBar & Jmenu & JMenuItem
- การกำหนดรูปแบบการจัดวาง



## รายการเลือก (Menu)

- JMenuBar คือ แถบรายการเลือกที่อยู่ด้านบนของหน้าต่างโปรแกรม
- JMenu คือ รายการเลือกที่เกิดขึ้นเมื่อคลิก JMenuBar และเป็นที่ยอมรับของ JMenuItem และ/หรือ JMenu
- JMenuItem คือ รายการเลือกขั้นสุดท้าย

# ตัวอย่างการใช้ menu

```
public class MenuDemo extends JFrame {
    private static final long serialVersionUID = 1L;

    public MenuDemo() {
        super("Menu demo");
        JMenuBar menuBar = new JMenuBar();
        JMenu fileMenu = new JMenu("File");
        JMenu abMenu = new JMenu("About");

        JMenu newMenu = new JMenu("New");
        JMenuItem newFileItem = new JMenuItem("File");
        JMenuItem newFolItem = new JMenuItem("Folder");
        JMenuItem exMenu = new JMenuItem("Exit");
        newMenu.add(newFileItem);
        newMenu.add(newFolItem);

        fileMenu.add(newMenu);
        fileMenu.addSeparator();
        fileMenu.add(exMenu);

        JMenuItem vMenu = new JMenuItem("Version");
        abMenu.add(vMenu);
```

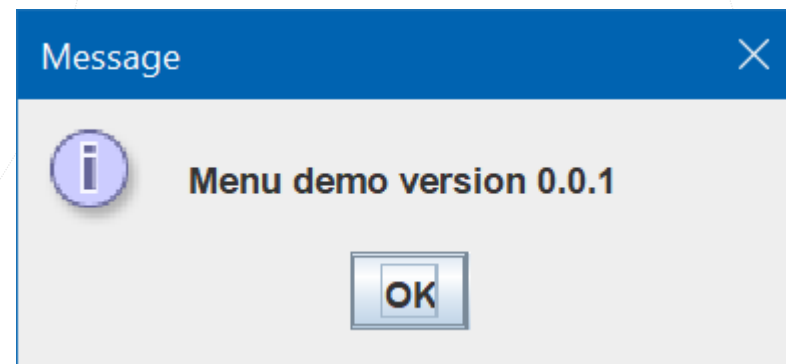
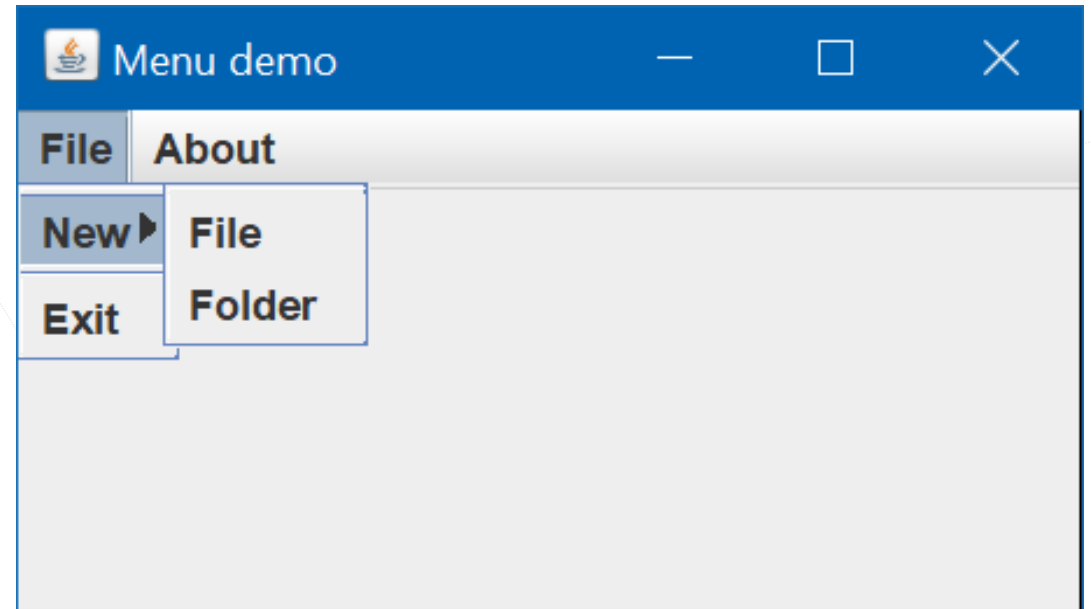
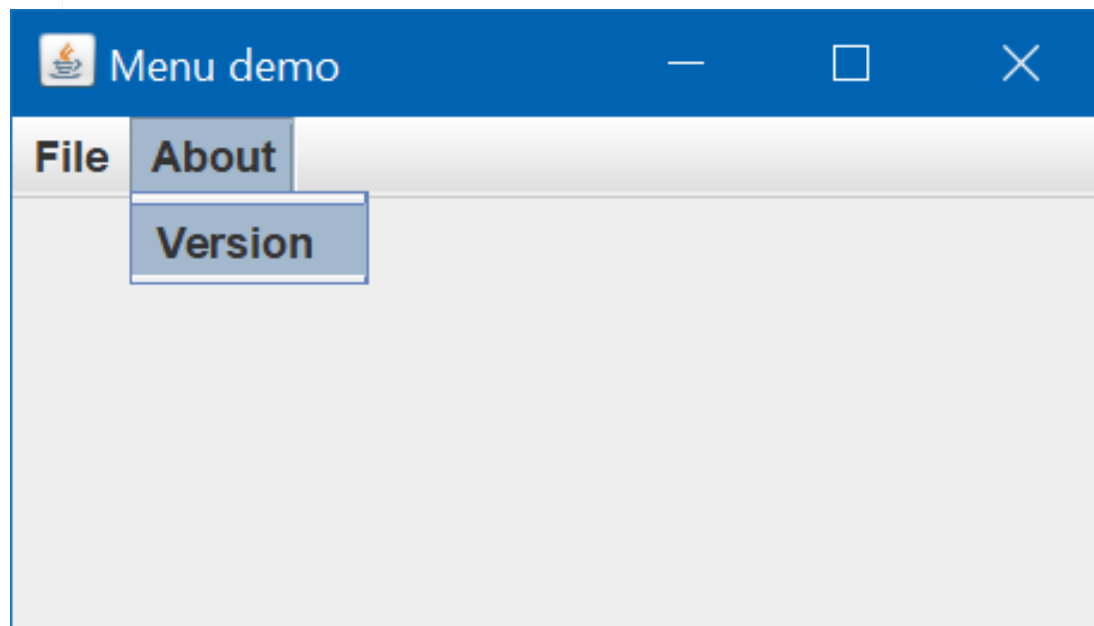
```
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55

        vMenu.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(MenuDemo.this,
                    "Menu demo version 0.0.1");
            }
        });
        exMenu.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });

        menuBar.add(fileMenu);
        menuBar.add(abMenu);
        setJMenuBar(menuBar);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 300);
    }
```

## ตัวอย่างการใช้ menu (ต่อ)

```
public static void main(String[] args) {  
    MenuDemo app = new MenuDemo();  
    app.setVisible(true);  
}
```

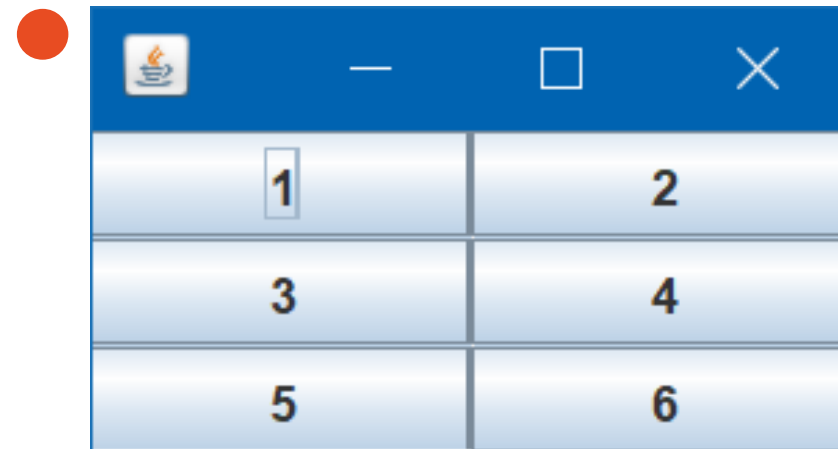


## การกำหนดรูปแบบการจัดวาง

- นอกเหนือจาก BorderLayout และ FlowLayout แล้ว ยังมีรูปแบบการจัดเรียงอีก 2 แบบ คือ GridLayout และ CardLayout
- GridLayout คือ การแบ่งส่วนพื้นที่ภายในหน้าต่างเป็นช่องขนาดเท่า ๆ กันตามมิติที่นักพัฒนามกำหนด
- CardLayout คือ การแบ่งส่วนพื้นที่ภายในด้วย JPanel ที่นำมาประกอบในหน้าต่าง ๆ เหมาะกับโปรแกรมที่มีการแสดงผลหลายรูปแบบ

# คลาส GridLayout

```
5 public class SimpleGrid extends JFrame {
6
7     private static final long serialVersionUID = 1L;
8
9     public SimpleGrid() {
10         setLayout(new GridLayout(3,2));
11         add(new JButton("1"));
12         add(new JButton("2"));
13         add(new JButton("3"));
14         add(new JButton("4"));
15         add(new JButton("5"));
16         add(new JButton("6"));
17
18         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19         pack();
20     }
21
22     public static void main(String[] args) {
23         SimpleGrid app = new SimpleGrid();
24         app.setVisible(true);
25     }
26 }
```



- ได้ผลลัพธ์เป็นกริดขนาด 3x2 ช่อง
- เรียงจากซ้ายไปขวาและจากบนลงล่าง

# การจัดวางที่ซับซ้อน

```
16  ✓ public ComplexLayout() {  
17      super("Complex Layout");  
18  
19      JPanel mainPanel = new JPanel();  
20      mainPanel.setLayout(new BorderLayout());  
21  
22      JPanel keypad = new JPanel();  
23      keypad.setLayout(new GridLayout(3, 3));  
24      keypad.add(new JButton("7"));  
25      keypad.add(new JButton("8"));  
26      keypad.add(new JButton("9"));  
27      keypad.add(new JButton("4"));  
28      keypad.add(new JButton("5"));  
29      keypad.add(new JButton("6"));  
30      keypad.add(new JButton("1"));  
31      keypad.add(new JButton("2"));  
32      keypad.add(new JButton("3"));
```

- สร้าง mainPanel แบบ BorderLayout (เหนือ กลาง ออก ตก ใต้)
- สร้าง Panel สำหรับเก็บตัวเลข 1-9 แบบ Grid ขนาด 3x3

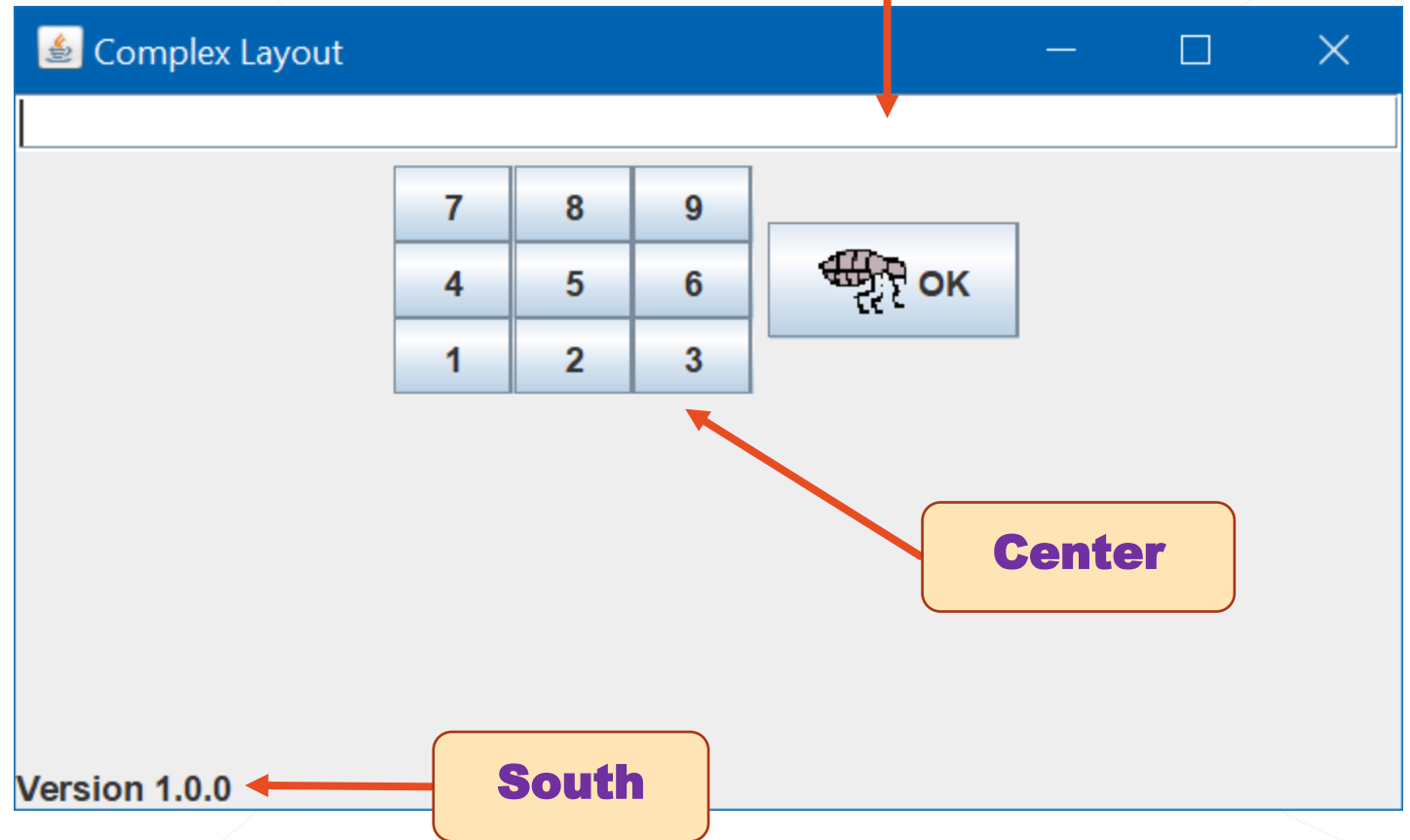
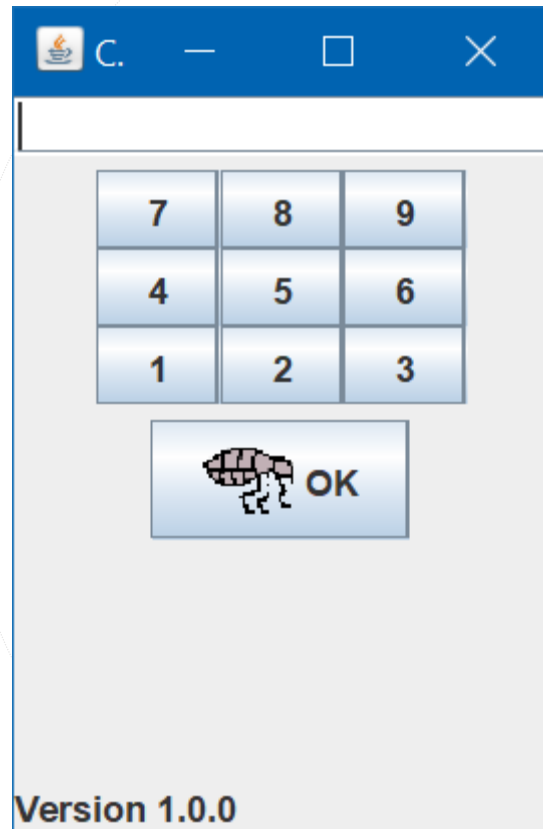
## การจัดวางที่ซับซ้อน (ต่อ)

```
34     JPanel allKeys = new JPanel();
35     ImageIcon icon = new ImageIcon(new ImageIcon("icon/flea.png").
36         |         |         getImage().getScaledInstance(30, 30, Image.SCALE_DEFAULT));
37
38     allKeys.add(keypad);
39     allKeys.add(new JButton("OK", icon));
40
41     mainPanel.add(new JTextField(), BorderLayout.NORTH);
42     mainPanel.add(allKeys, BorderLayout.CENTER);
43     mainPanel.add(new JLabel("Version 1.0.0"), BorderLayout.SOUTH);
44
45     add(mainPanel);
46     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
47     pack();
48 }
```

- สร้าง Panel allKeys สำหรับเก็บ Panel ตัวเลข 1-9 และ image icon ซึ่งมี default เป็น FlowLayout
- ใน mainPanel กำหนด component แต่ละส่วนให้อยู่ในตำแหน่ง north, center และ south ตามลำดับ โดยให้ panel allKeys อยู่ตรงกลาง

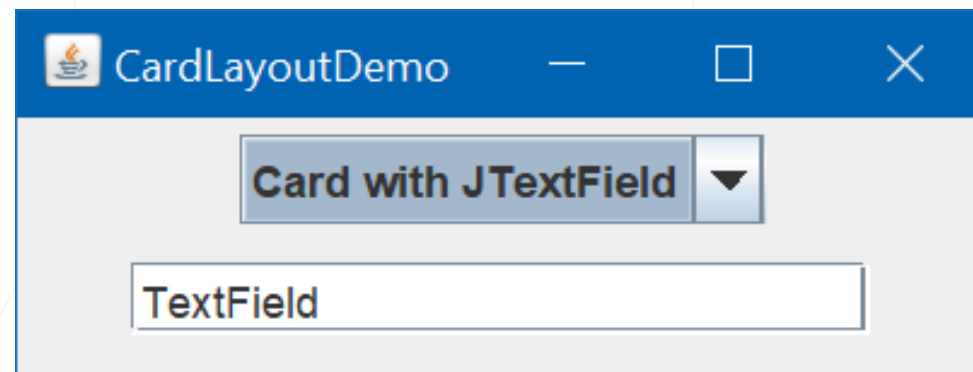
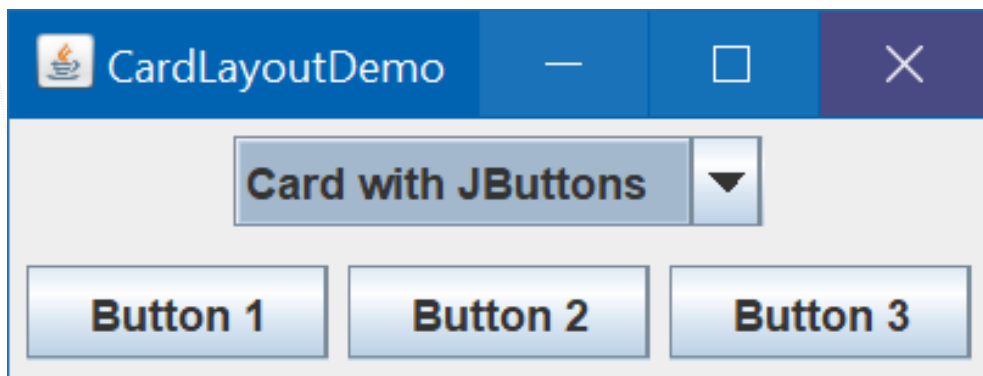


# การจัดวางที่ซับซ้อน (ต่อ)



# คลาส CardLayout

- เหมาะกับโปรแกรมที่มีการเปลี่ยนหน้าแสดงผล และมีการแสดงผลหลายรูปแบบ
- โดยสร้างแต่ละหน้าเป็น JPanel แล้วนำมาบรรจุในหน้าต่างโปรแกรมโดยกำหนดให้ใช้การจัดวางแบบ CardLayout



## คลาส CardLayout (ต่อ)

```
public static void main(String[] args) {  
    javax.swing.SwingUtilities.invokeLater(new Runnable() {  
        public void run() {  
            createAndShowGUI();  
        }  
    });  
}
```

```
private static void createAndShowGUI() {  
    //Create and set up the window.  
    JFrame frame = new JFrame("CardLayoutDemo");  
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
    //Create and set up the content pane.  
    CardLayoutDemo demo = new CardLayoutDemo();  
    demo.addComponentToPane(frame.getContentPane());  
  
    //Display the window.  
    frame.pack();  
    frame.setVisible(true);  
}
```

## คลาส CardLayout (ต่อ)

```
public void itemStateChanged(ItemEvent evt) {  
    CardLayout cl = (CardLayout)(cards.getLayout());  
    cl.show(cards, (String)evt.getItem());  
}
```

การสลับหน้าทำได้โดยการเรียกเมทอด  
show() ของ CardLayout

```
5 public class CardLayoutDemo implements ItemListener {  
6     JPanel cards; //a panel that uses CardLayout  
7     final static String BUTTONPANEL = "Card with JButtons";  
8     final static String TEXTPANEL = "Card with JTextField";  
9  
10    public void addComponentToPane(Container pane) {  
11        //Put the JComboBox in a JPanel to get a nicer look.  
12        JPanel comboBoxPane = new JPanel(); //use FlowLayout  
13        String comboBoxItems[] = { BUTTONPANEL, TEXTPANEL };  
14        JComboBox<String> cb = new JComboBox<String>(comboBoxItems);  
15        cb.setEditable(false);  
16        cb.addItemListener(this);  
17        comboBoxPane.add(cb);  
18        //Create the "cards".  
19        JPanel card1 = new JPanel();  
20        card1.add(new JButton("Button 1"));  
21        card1.add(new JButton("Button 2"));  
22        card1.add(new JButton("Button 3"));  
23  
24        JPanel card2 = new JPanel();  
25        card2.add(new JTextField("TextField", 20));  
26  
27        //Create the panel that contains the "cards".  
28        cards = new JPanel(new CardLayout());  
29        cards.add(card1, BUTTONPANEL);  
30        cards.add(card2, TEXTPANEL);  
31  
32        pane.add(comboBoxPane, BorderLayout.PAGE_START);  
33        pane.add(cards, BorderLayout.CENTER);  
34    }
```

# การสื่อสารข้ามพาด้าน

- กรณีที่แอปพลิเคชันที่มีความซับซ้อน ต้องสร้างพาด้านสำหรับแต่ละหน้าแยกไปเป็นคลาสต่างหาก
- ทำให้เกิดข้อจำกัดในการแลกเปลี่ยนข้อมูลระหว่างหน้า เนื่องจากไม่ได้อยู่ในคลาสเดียวกันจึงไม่มีตัวแปรที่อ้างอิงร่วมกันได้
- แก้ไขโดย สร้างคลาสสำหรับเก็บข้อมูลเฉพาะเรื่องขึ้นมา (ไม่ใช่คลาส GUI) ซึ่งเป็นคลาสธรรมดาที่ใช้แทนข้อมูลที่ต้องการแลกเปลี่ยนระหว่างหน้า
- เมื่อโปรแกรมเริ่มทำงาน ให้สร้างอ็อบเจกต์จากคลาสข้อมูล แล้วส่งให้พาด้านแต่ละหน้าผ่านคอนสตรักเตอร์ของพาด้าน

# เธรดและกลไกของระบบ GUI

- โปรแกรม GUI มีการทำงานแบบหลายเธรด (main จบ โปรแกรมยังทำงานต่อ)
- main thread จะเริ่มต้นทำงานก่อน และเมื่อใดที่คอมโพเนนต์ GUI ถูกสั่งให้แสดงผล เธรด event dispatch thread (EDT) จะเริ่มทำงานไปควบคู่กับ main thread
- EDT ทำหน้าที่จัดการโค้ดที่เกี่ยวข้องกับ GUI ทั้งหมด
- ข้อควรระวังที่สำคัญที่สุดข้อหนึ่ง คือ อย่าเรียกคำสั่งของ GUI จาก main thread
- ดังนั้นในหลายตัวอย่างที่ผ่านมา ไม่ใช่ตัวอย่างที่ดีนัก เพราะมีการเรียกเมทอดของ JFrame ใน main thread

## เทรดและกลไกของระบบ GUI (ต่อ)

- การเรียก GUI ควรเรียกใน EDT ทำได้โดยการใช้เมทอด  
SwingUtilities.invokeLater() หรือ EventQueue.invokeLater()

```
public static void main(String[] args) {  
    javax.swing.SwingUtilities.invokeLater(new Runnable() {  
        public void run() {  
            createAndShowGUI();  
        }  
    });  
}
```

# อ้างอิง

- <https://nbviewer.jupyter.org/github/Poonna/java-book/>