

บทที่ 10: การสร้างส่วนต่อประสานกราฟิกกับผู้ใช้งาน (Build Graphical User Interface: GUI)

บรรยายโดย ผศ.ดร. ธราวิเชษฐ์ ธิติจรรย์โรจน์

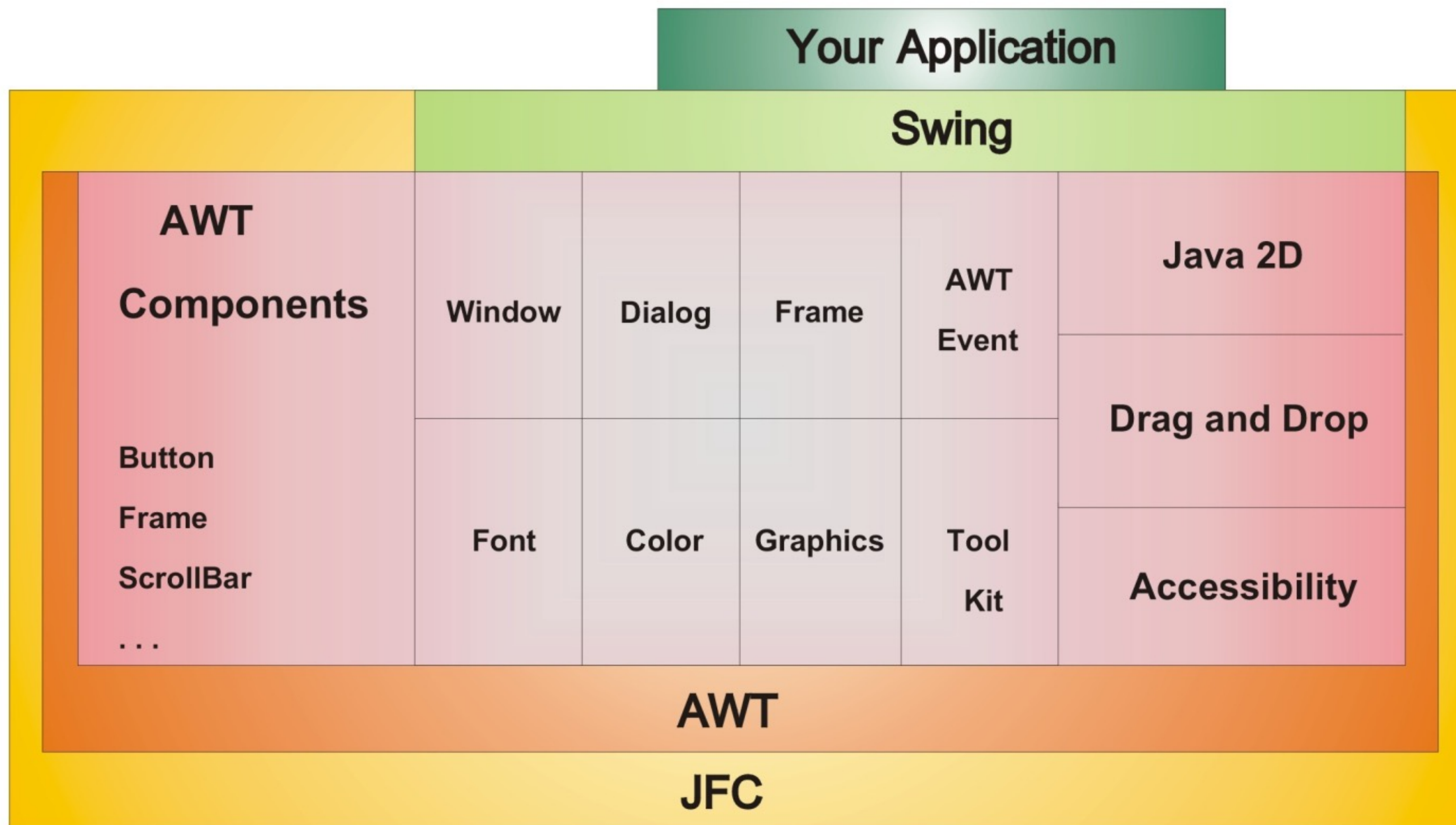
คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

Java Foundation Class (JFC)

- ระบบปฏิบัติการส่วนใหญ่จะมีส่วนติดต่อกับผู้ใช้เป็นแบบกราฟิก (Graphical User Interface: GUI)
- ภาษาจาวาจะสนับสนุนการพัฒนาโปรแกรม GUI ที่สามารถใช้งานได้หลายแพลตฟอร์ม โดยจะใช้ชุดคำสั่งเดียวกัน
- Java Foundation Class (JFC) ประกอบด้วยแพ็คเกจต่างๆดังนี้
 - Abstract Window Toolkit (AWT)
 - Swing
 - Java 2D
 - Accessibility
 - Drag and Drop

ส่วนประกอบของ Java Foundation Class (JFC)³



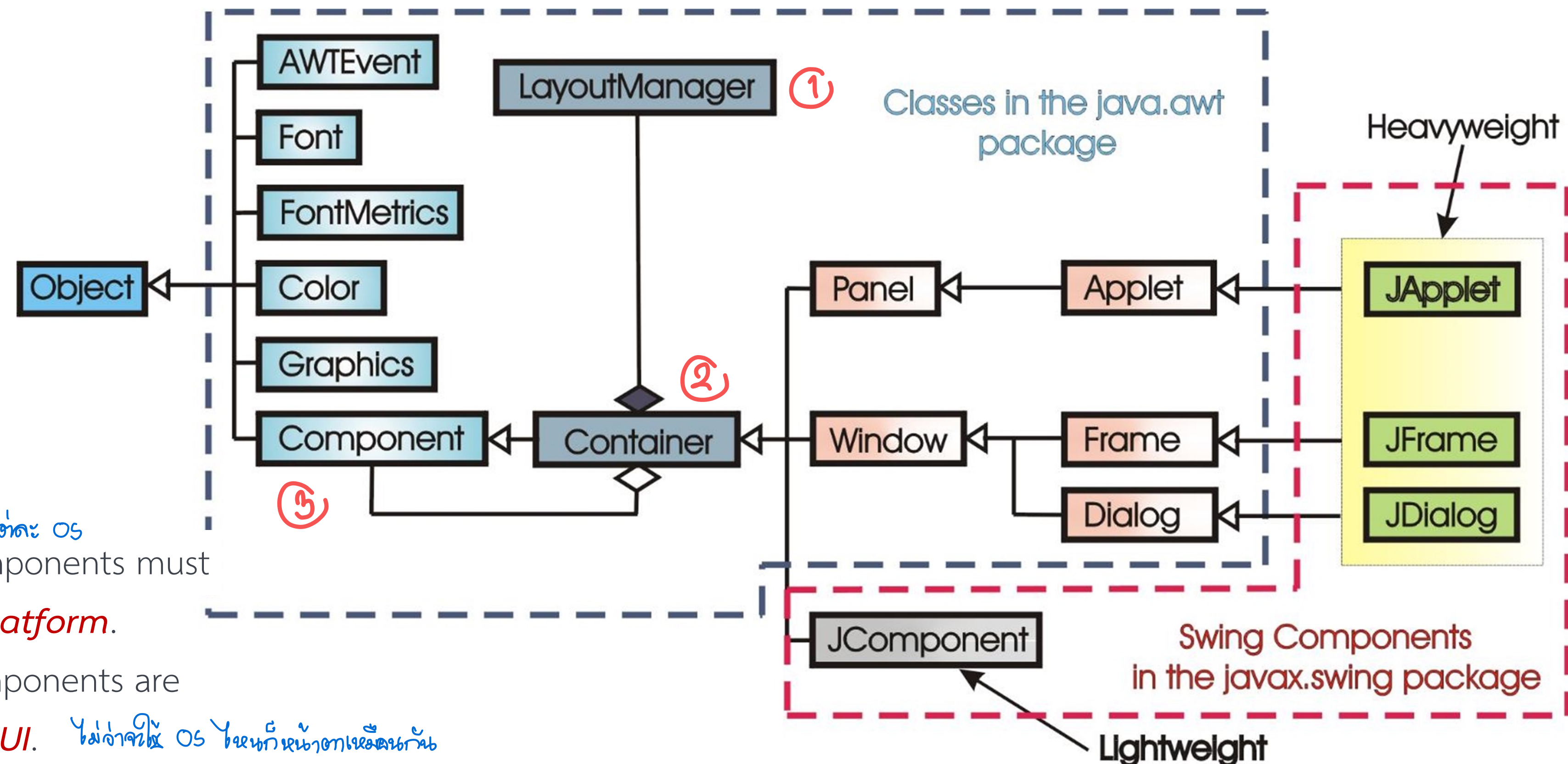
Abstract Window Toolkit (AWT)

4

Abstract Window Toolkit (AWT) เป็นแพ็คเกจที่ใช้พัฒนา Graphic User Interface **พื้นฐาน** ซึ่งจะให้รูปร่าง GUI ที่ได้ ขึ้นอยู่กับระบบปฏิบัติการ (look and feel) ซึ่งภาษาจาวาได้กำหนดแพ็คเกจ AWT ขึ้นไว้ตั้งแต่โปรแกรมจาวาเวอร์ชันแรก (JDK 1.0) โดยอยู่ในแพ็คเกจที่ชื่อ java.awt โดยจะประกอบไปด้วยคลาสและอินเทอร์เฟสต่างๆ เพื่อใช้ในการพัฒนาโปรแกรม GUI

- ③ • Component
- ② • Container
- ① • LayoutManager *
- Graphics
- Color
- Font

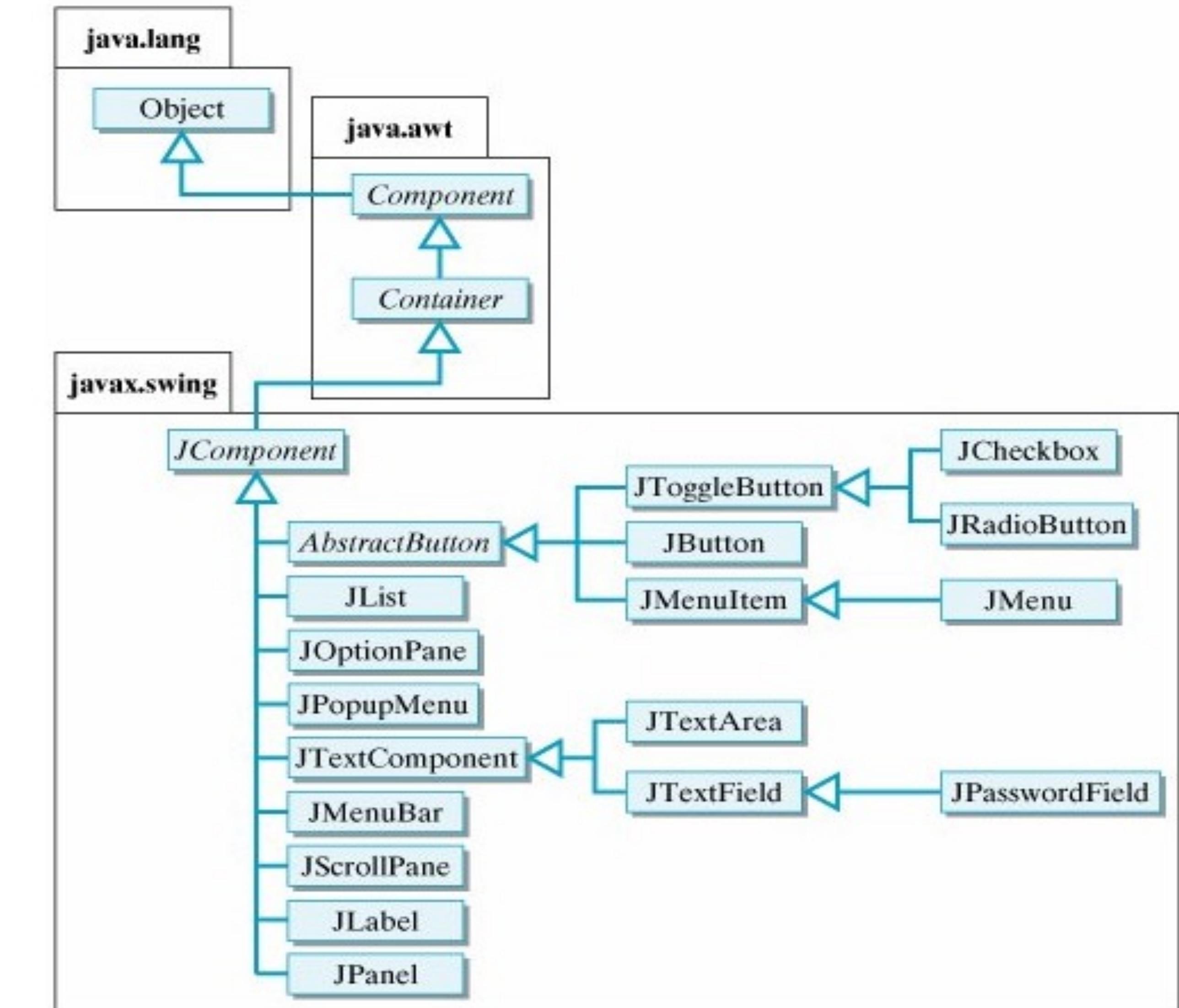
- **Heavyweight** components like "AWT" components must be drawn using native GUI on a **specific platform**.
ภาษาต่าง GUI ให้ใช้หน้าต่างต่างกันตามแต่ละ OS
- **Lightweight** components like "Swing" components are drawn by java and **don't rely on native GUI**.
ไม่จำเป็นต้องใช้ OS ให้หน้าต่างเหมือนกัน



ส่วนประกอบของ Java Foundation Class (JFC)

5

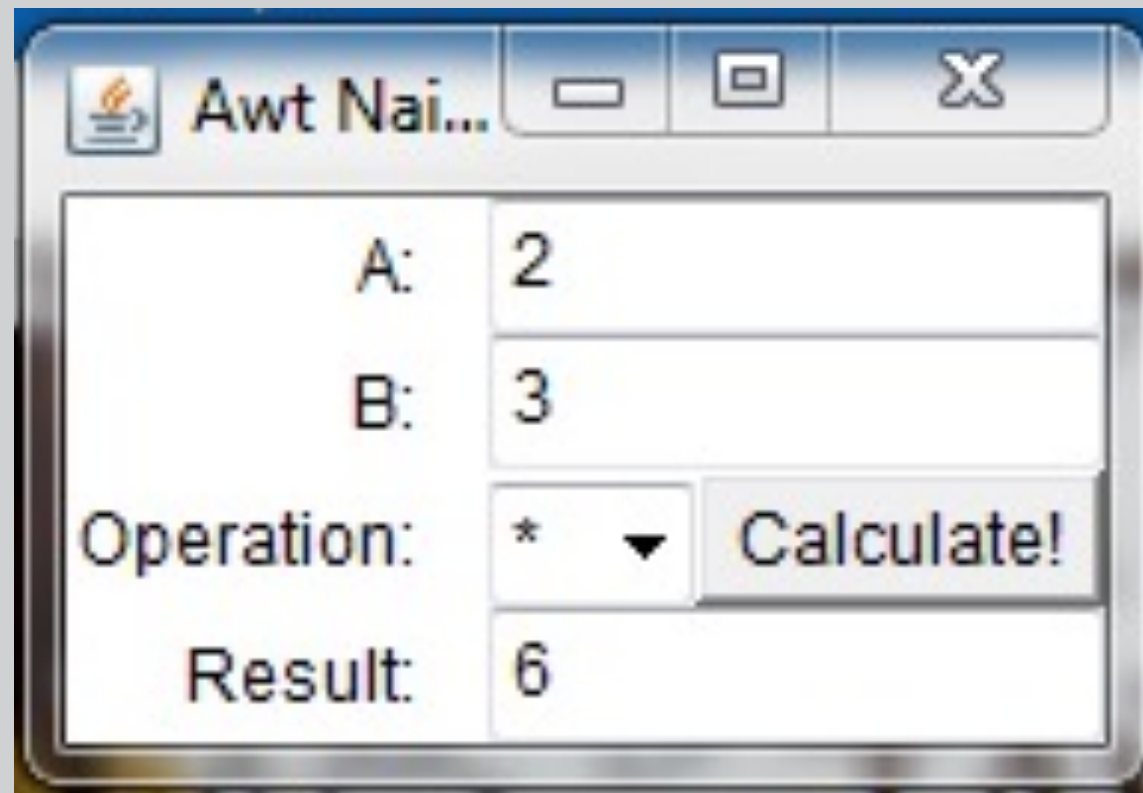
Swing เป็นแพ็คเกจที่มีส่วนประกอบกราฟฟิคที่มี
 คุณลักษณะและรูปแบบที่ **ดีกว่า (สวยงาม)** ส่วนประกอบ
 กราฟฟิคของแพ็คเกจ AWT แต่ Swing จะทำงาน **ช้ากว่า**
 AWT



ความแตกต่างระหว่าง AWT และ Swing

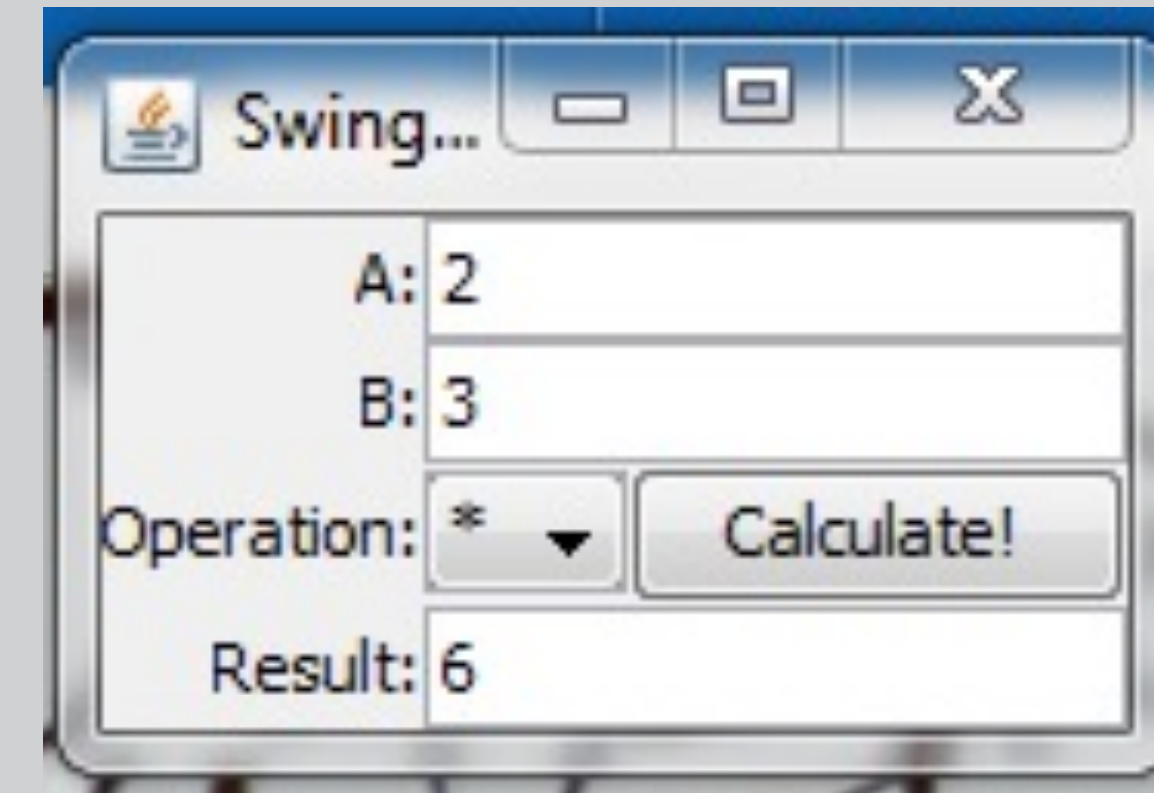
6

AWT



- อยู่บนพื้นฐานของ Abstract Window Toolkit
- **ไม่รองรับ**การทำงานแบบ Model View Controller
- รูปร่างหน้าต่างของแต่ละ Components **ขึ้นอยู่กับ**ระบบปฏิบัติการ
- **ต้องนำ**เข้าแพ็คเกจ java.awt
- เร็วกว่า Swing

Swing



- เป็นส่วนหนึ่งของ Java Foundation Class
- **รองรับ**การทำงานแบบ Model View Controller
- รูปร่างหน้าต่างของแต่ละ Components **ไม่ขึ้นอยู่กับ**ระบบปฏิบัติการ
- **ต้องนำ**เข้าแพ็คเกจ javax.swing
- ช้ากว่า AWT

ความแตกต่างระหว่าง AWT และ Swing

AWT	JAVA SWING
<ul style="list-style-type: none"> •Applet •Frame •Window •Dialog •Component •Panel •Button •Canvas ★ •Checkbox •Choice ★ •Label •List •TextArea •TextField •Menu •MenuItem 	<ul style="list-style-type: none"> •JApplet •JFrame •JWindow •JDialog •JComponent •JPanel •JButton •Panel •JCheckBox JRadioButton ★ •JComboBox ★ •JLabel •JList •JTextArea •JTextField •JMenu •JMenuItem

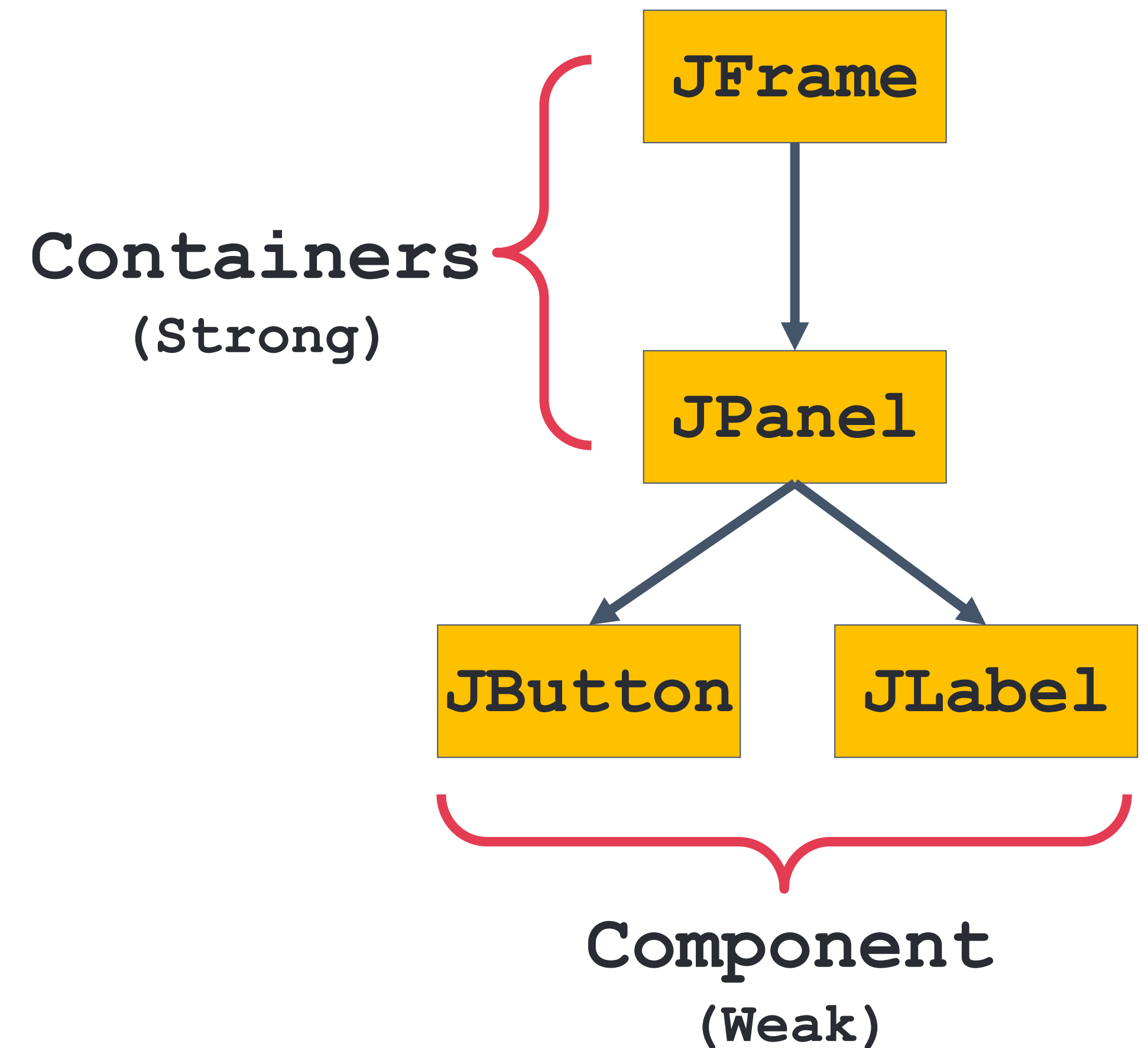
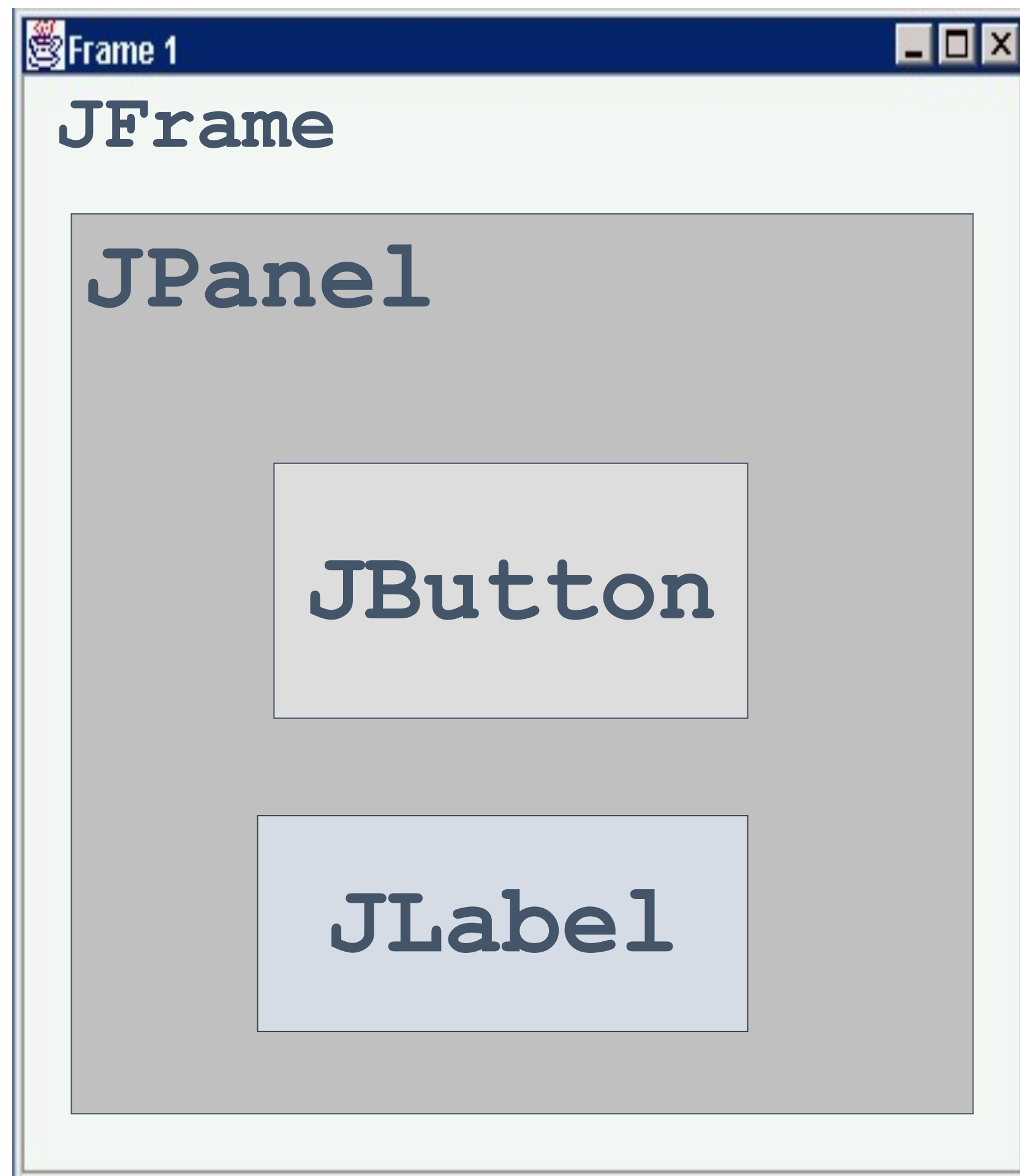
ส่วนประกอบของ Java Foundation Class (JFC)

- **Java 2D** เป็นชุดคำสั่งกราฟิกที่มีคลาสที่ช่วยในการพัฒนากราฟิกสองมิติและจัดการรูปภาพ *(สร้างเกม)*
- **Accessibility** เป็นชุดคำสั่งที่มีคลาสในการพัฒนาโปรแกรมที่มี input และ output ในลักษณะพิเศษ อาทิเช่น screen reader, screen magnifier และ audio text reader *ออกแบบมาสำหรับผู้ที่มีความผิดปกติทางกาย*
- **Drag and Drop** เป็นชุดคำสั่งของเทคโนโลยีที่ใช้ในการแลกเปลี่ยนข้อมูลระหว่างโปรแกรมที่พัฒนาด้วยภาษาจาวากับภาษาอื่น ๆ

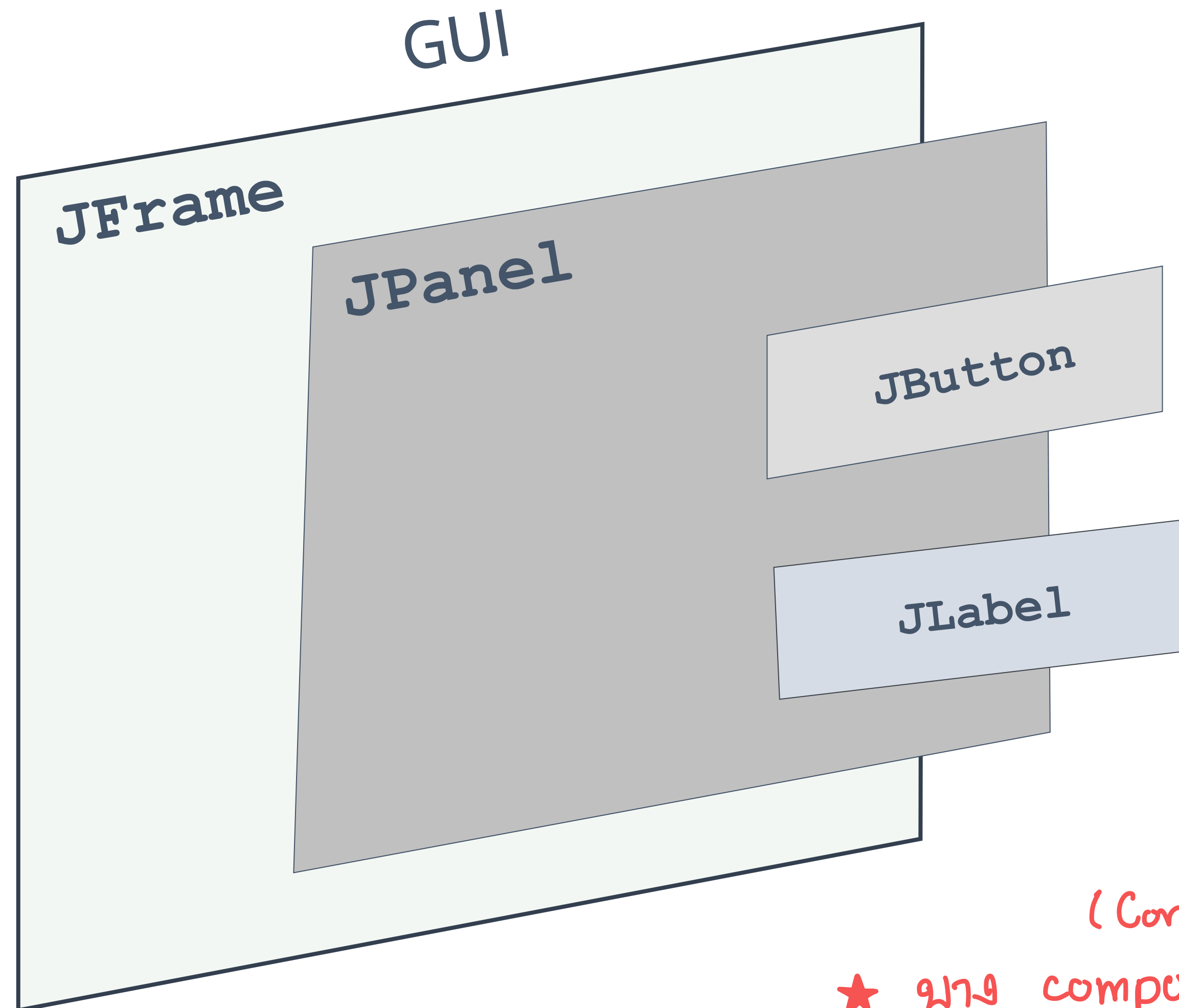
โครงสร้างของโปรแกรม GUI

GUI

ลำดับโครงสร้างภายใน

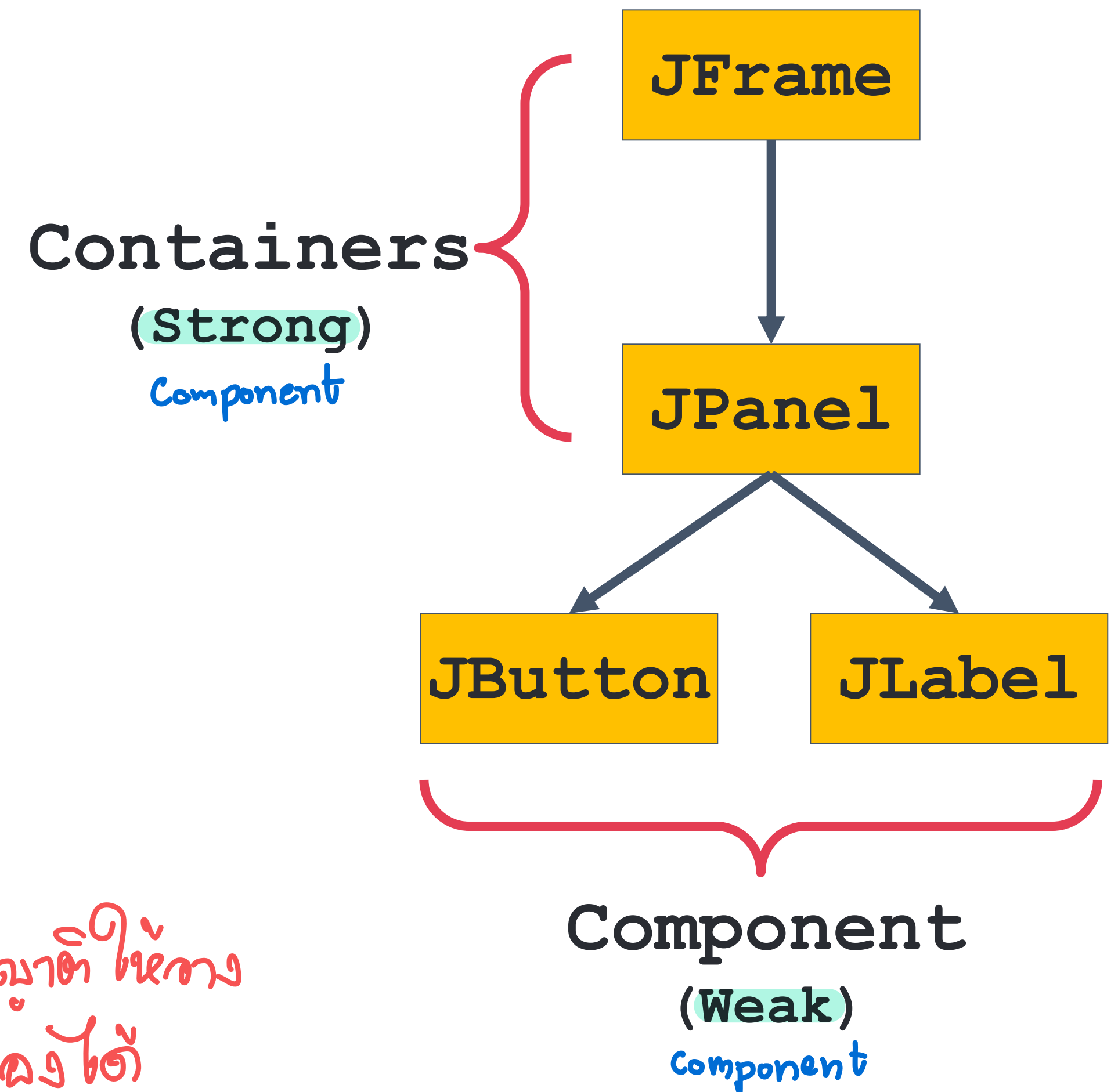


โครงสร้างของโปรแกรม GUI

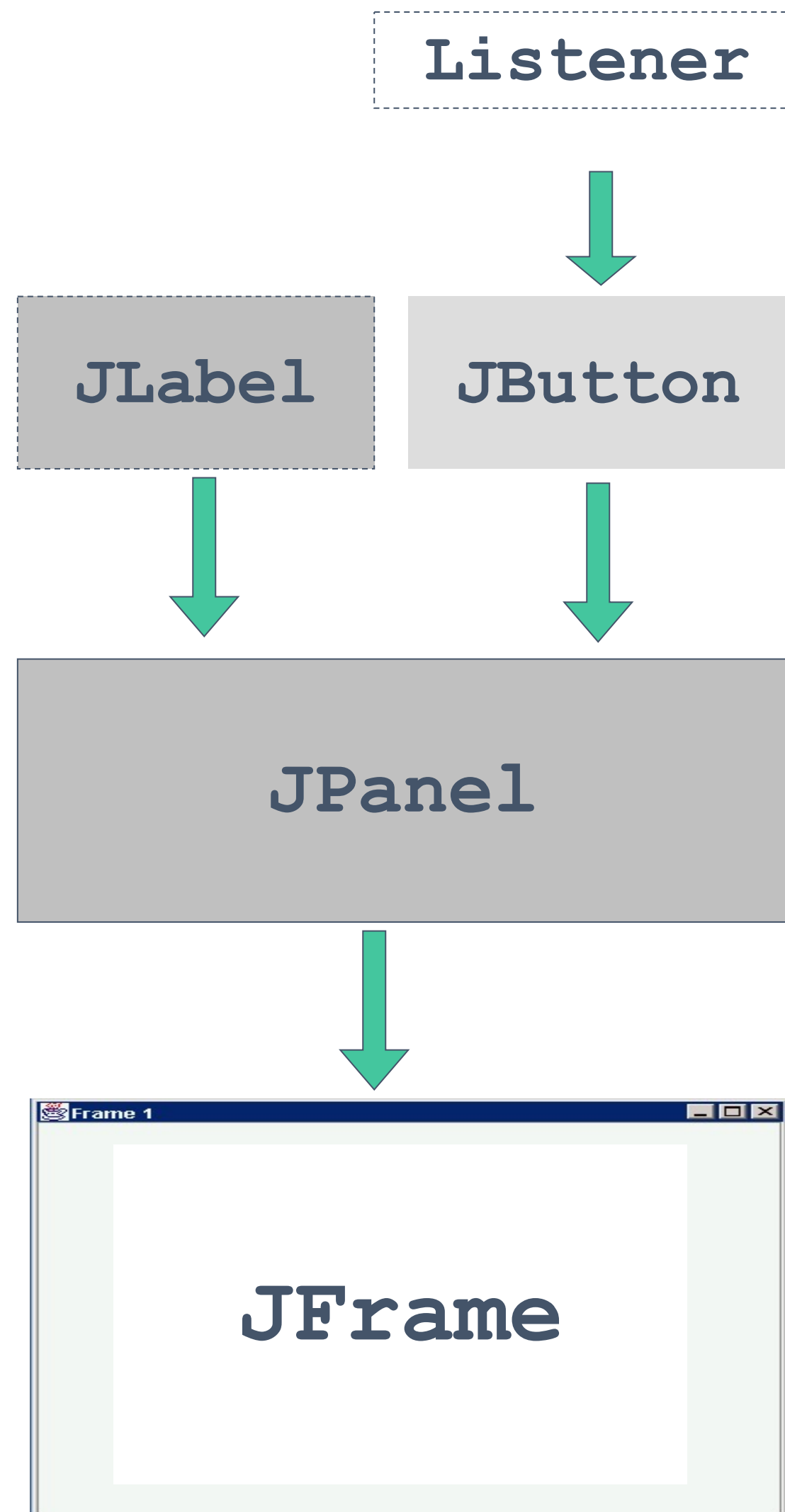


(Containers)
★ บาง component สามารถใส่บาง
component ขึ้นมาข้างในได้

ลำดับโครงสร้างภายใน



ขั้นตอนการสร้าง GUI



- ขั้นตอนที่ 1: ประกาศและสร้างวัตถุต่าง ๆ อาทิเช่น Frame, Panel, Components และ Listeners *ประกาศและสร้าง component*

```
 JButton b = new JButton("press me") ;
```

- ขั้นตอนที่ 2: กำหนดค่าพารามิเตอร์ต่าง ๆ ของวัตถุที่สร้างในขั้นตอนแรก (optional) *กำหนดค่าและตาแหน่ง*

```
 b.text = "press me" ; // avoided
```

```
 b.setText("press me") ;
```

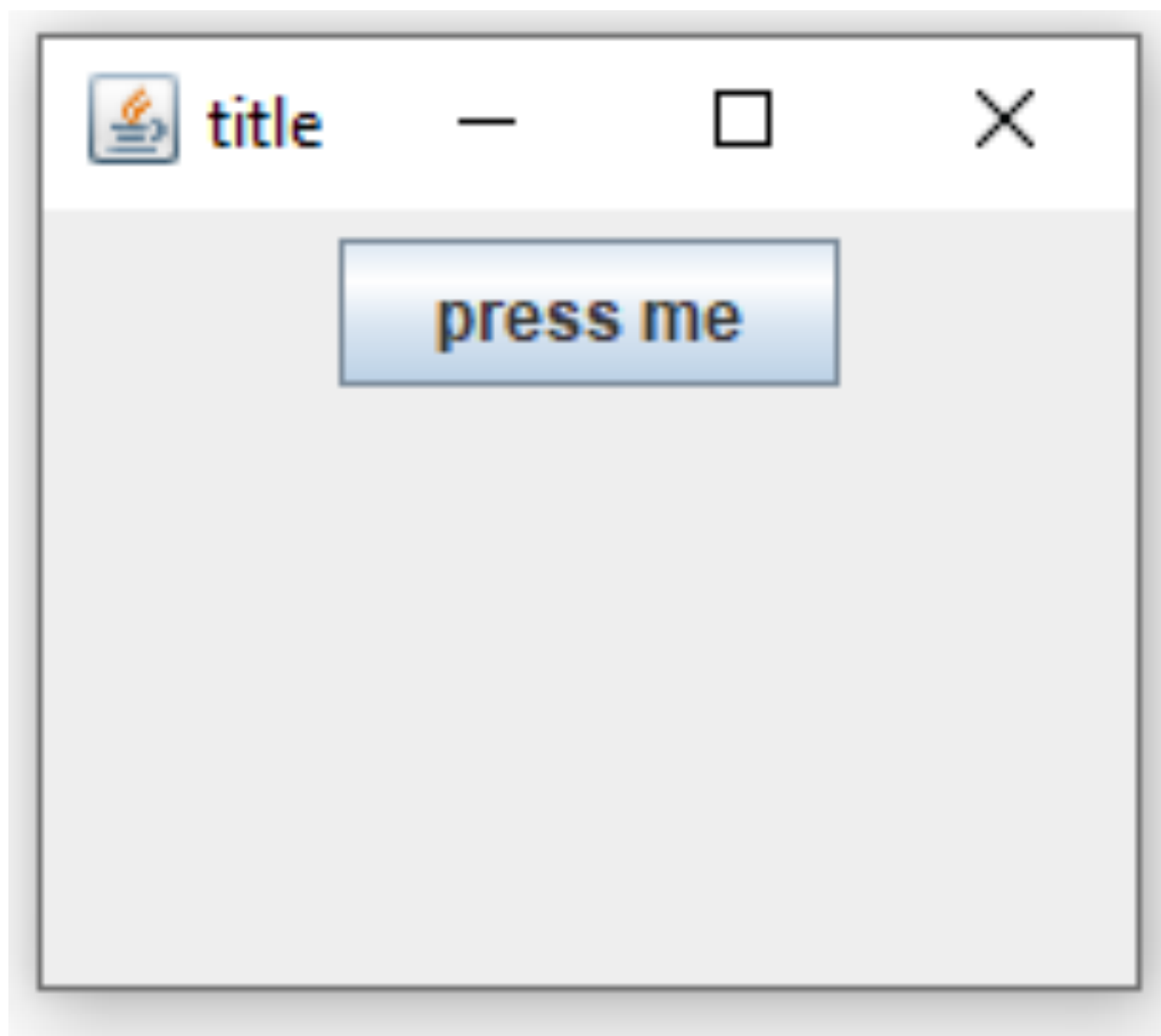
- ขั้นตอนที่ 3: เพิ่มวัตถุข้างต้นลงในวัตถุอื่น ๆ ตามลำดับ (bottom up) *จัด layout*

```
 panel.add(b) ;
```

- ขั้นตอนที่ 4: ตั้งค่า listeners ให้แต่ละวัตถุ เพื่อดักฟังเหตุการณ์ที่อาจจะเกิดขึ้น *จัด event*

```
 // Listeners Code
```


ตัวอย่างที่ 1



```
import javax.swing.*;
public class GUI01 {
    public static void main(String[] args) {
        // step 1: create
        JFrame f = new JFrame("title");
        JPanel p = new JPanel( );
        JButton b = new JButton("press me");
        // step 2: config
        f.setSize(200, 200);

        // step 3: add
        p.add(b); ★
        f.add(p); ★

        // step 4: listener
    }
}
```

★ *คำสั่งเรียงจากเล็กไปใหญ่*

Diagram illustrating the hierarchy of GUI components:

- A box labeled 'f' (JFrame) contains a box labeled 'p' (JPanel).
- The box labeled 'p' contains a box labeled 'b' (JButton).

Comments for step 3:

- `p.add(b); ★` // add button to panel
- `f.add(p); ★` // add panel to frame

ตัวอย่างที่ 1

Containers and
Components

Layouts

Events

```
import javax.swing.*;
public class GUI01 {
    public static void main(String[] args) {
        // step 1: create
        JFrame f = new JFrame("title");
        JPanel p = new JPanel( );
        JButton b = new JButton("press me");
        // step 2: config
        f.setSize(200, 200);

        // step 3: add
        p.add(b); // add button to panel
        f.setContentPane(p); // add panel to frame

        // step 4: listener
    }
}
```

คอนเทนเนอร์ (Container)

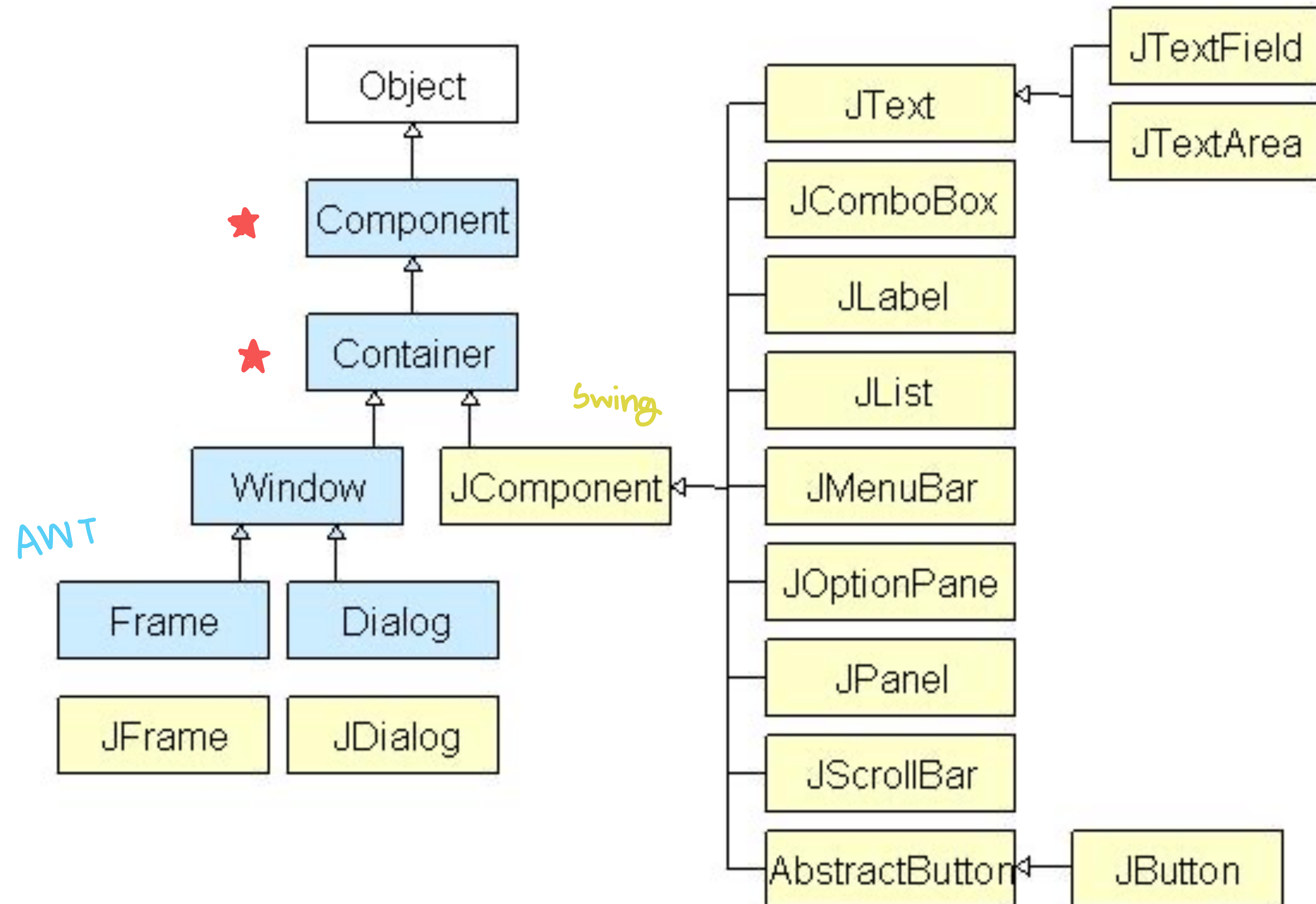
คลาสประเภท Container

คือ *วัตถุที่ใช้จัดเก็บวัตถุของส่วนประกอบกราฟิกหลาย ๆ วัตถุไว้* เพื่อแสดงผล ซึ่งส่วนต่อประสานกราฟิกกับผู้ใช้งานจะต้องมีการสร้างวัตถุของคลาสประเภท **Container** อย่างน้อยหนึ่งวัตถุมาก่อน ได้แก่ **Frame**, **JFrame**, **Panel**, **JPanel**, **JDialog**, **Dialog** และ **Applet**

คลาสประเภท **Container** เป็นคลาสที่สืบทอดมาจากคลาสที่ชื่อ **Component** ซึ่งเป็นคลาสแบบ **abstract** ซึ่งเราไม่สามารถที่จะสร้างออบเจ็คของคลาสดังกล่าวได้ แต่จะต้องสร้างออบเจ็คของคลาสอื่นๆที่สืบทอดมาจากคลาสที่ชื่อ **Container** แทน

```
// คลาสที่ชื่อ Container จะมีเมธอดที่ชื่อ add() ที่ใช้ในการใส่ส่วนประกอบกราฟิกอื่น ๆ  
เมธอดนี้จะมีรูปแบบที่สำคัญดังนี้  
public void add (Component c)  
public void add (Component c,int position)
```

Swing Package

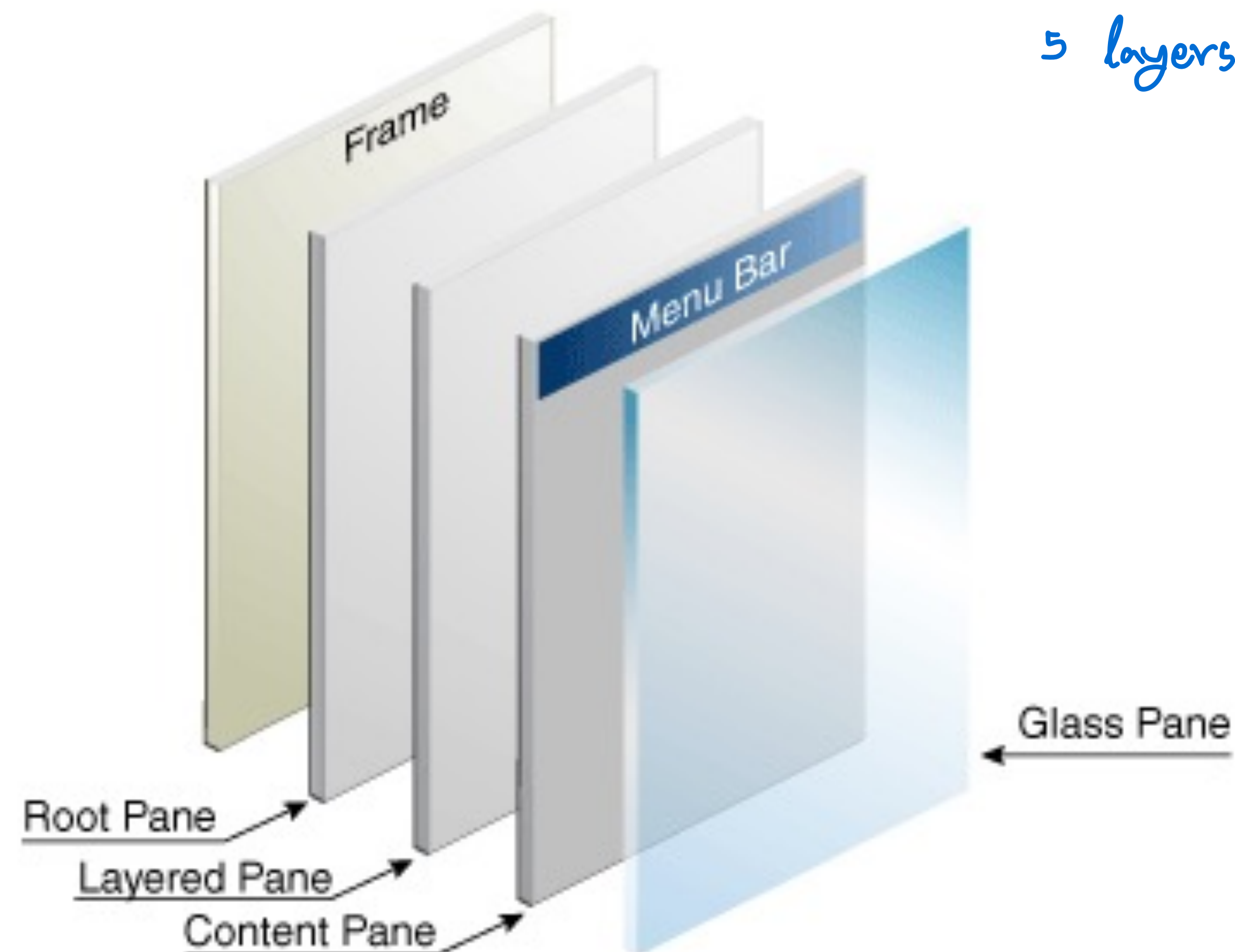


JFrame



ออปเจ็คของคลาส **JFrame** แตกต่างกับ **Frame** ตรงที่มีหน้าต่าง (pane) อยู่ 4 หน้าต่างดังนี้

5 layers (4 pane in front of frame)



หลักการสร้าง JFrame



ออปเจ็คของคลาส **JFrame** สามารถสร้างได้ 2 รูปแบบ

- สร้างออปเจ็คของคลาส **JFrame** โดยตรง
- สร้างคลาสที่สืบทอดมาจากคลาส **JFrame** เพื่อสร้างหรือเปลี่ยนแปลงเป็น **JFrame** ของเราเอง

JFrame



```

import javax.swing.*;
public class GUI02 {
    public static void main(String[] args) {
        //JFrame frame = new JFrame();
        JFrame frame = new JFrame("Tilte GUI 02");
        JLabel label = new JLabel("Faculty of Information Technology");
  
```

ตัวนี้ทำงานต่าง
 แตกต่างจากตัวอื่น *
 จำแนกกันไม่ได้

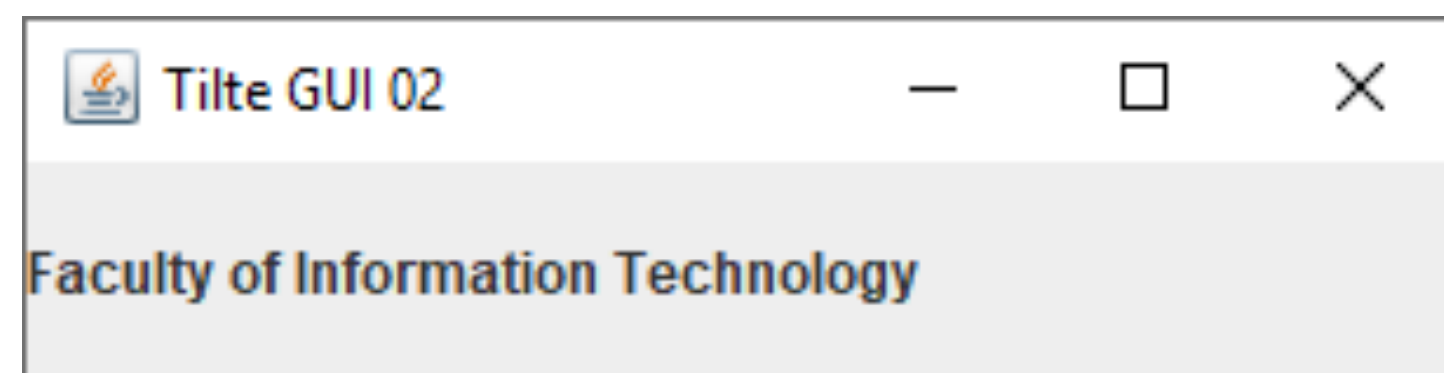
```
frame.add(label);
```

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

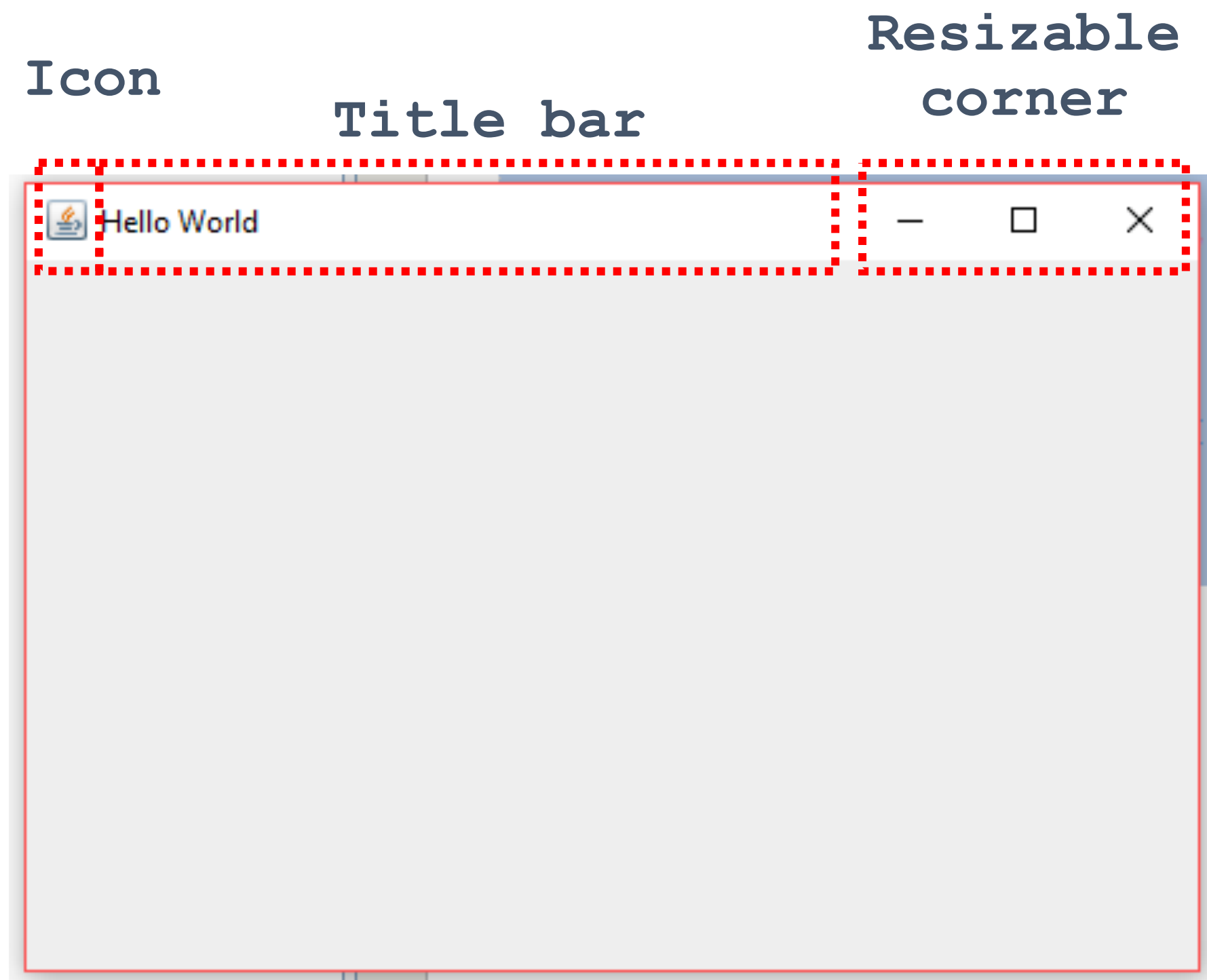
```
frame.pack();
```

```
frame.setVisible(true);
```

* setVisible() จะทำงานก่อน
 (.pack() หรือ setSize())



JFrame



```
import javax.swing.*;

public class MyJFrame extends JFrame {
    public MyJFrame() {
        this.setTitle("Hello World");
        this.setSize(480, 320);
        this.setVisible(true);
        this.setDefaultCloseOperation(...
            this.EXIT_ON_CLOSE);
    }
    public static void main(String[] args) {
        MyJFrame m = new MyJFrame();
    }
}
```


JFrame



- `public void setTitle(String str)`
กำหนดข้อความ Title ของ JFrame
- `public void setSize(int width, int height)`
กำหนดความกว้างและความสูงของ JFrame *หน่วยเป็น pixel*
- `public void setLocation(int x, int y)`
กำหนดตำแหน่งของ JFrame บนหน้าต่าง
- `public void setBounds(int x, int y, int width, int height)`
กำหนดความกว้าง ความสูง และตำแหน่งของ JFrame
- `public void setVisible(boolean v)`
กำหนดการมองเห็น (true) หรือไม่เห็น (false) ของ JFrame

JFrame

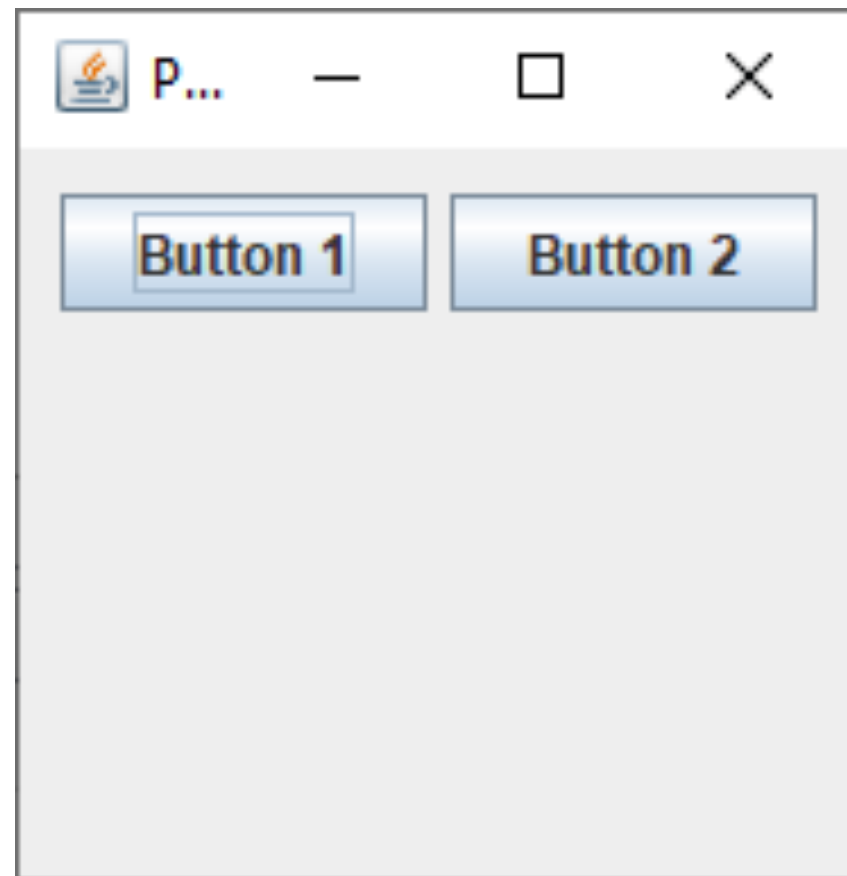
- `public void setResizable (boolean r)`
 กำหนดให้ปรับขนาดได้ (true) หรือไม่อนุญาตให้ปรับขนาด (false) ของ **JFrame**
- `public void setDefaultCloseOperation(int op)`
 กำหนดการกระทำเมื่อกดเครื่องหมายกากบาทบริเวณด้านขวาบนของ **Jframe**
 - **DO_NOTHING_ON_CLOSE** จะไม่ทำอะไร โดยให้เรียกใช้เมธอด `windowClosing` ของอ็อบเจกต์ `WindowListener` ที่มีการลงทะเบียนไว้แล้วแทน
 - **HIDE_ON_CLOSE** จะทำการซ่อน `Jframe` หลังจากเรียกใช้อ็อบเจกต์ของ `WindowListener` ที่มีการลงทะเบียนไว้แล้วแทน *กดแล้วจะซ่อนหน้าต่างเอาไว้แต่ยังมีคำสั่ง process อยู่*
 - **DISPOSE_ON_CLOSE** จะทำการซ่อนและทำลาย `Jframe` หลังเรียกใช้งานอ็อบเจกต์ของ `WindowListener` ที่มีการลงทะเบียนไว้แล้วแทน *กดแล้วหน้าต่างจะหายไปนั้น ๆ ของโปรแกรม*
 - **EXIT_ON_CLOSE** จะมีการเรียกใช้เมธอด `exit` ของคลาส `System` อ็อบเจกต์ของ *กดแล้วมีทุกหน้าต่างของโปรแกรม*

JPanel



JPanel (*`javax.swing.JPanel`*) จัดอยู่ในกลุ่มของ **Container** ไว้สำหรับจัดกลุ่มของ **Component Controls** ให้อยู่ในกลุ่มเดียวกัน เพื่อใช้ง่ายต่อการแสดงผล และการนำไปใช้งาน โดยเราสามารถประกาศ **Panel** ได้ และ ทำการสร้าง **Controls** หลาย ๆ ตัวเข้ามาภายใน **Panel** นั้น ๆ

JPanel



```
import java.awt.*;
import javax.swing.*;
public class PanelExample {
    public PanelExample() {
        JFrame f = new JFrame("Panel Example");
        JPanel panel = new JPanel();
        JButton b1 = new JButton("Button 1");
        JButton b2 = new JButton("Button 2");
        panel.add(b1);
        panel.add(b2);
        f.add(panel);
        f.setSize(200,200);
        f.setLayout(new FlowLayout());
        f.setVisible(true);
    }
    public static void main(String args[]) {
        new PanelExample();
    }
}
```


JPanel



- **public JPanel ()**
สร้าง panel ใหม่
- **public JPanel (LayoutManager layout)**
สร้าง panel ใหม่พร้อมระบุรูปแบบการจัดวาง (FlowLayout เป็นค่าเริ่มต้น)
- **public void add (Component c)**
เพิ่ม component ใหม่ลงใน panel
- **public void remove (Component c)**
ลบ component ออกจาก panel
- **public void setLayout (LayoutManager layout)**
กำหนดรูปแบบการจัดวาง (FlowLayout เป็นค่าเริ่มต้น)
- **public void setLocation (int x, int y)**
กำหนดตำแหน่งที่ตั้งของ JPanel บน JFrame โดยที่ตำแหน่งซ้ายบน คือ พิกัด (0,0)

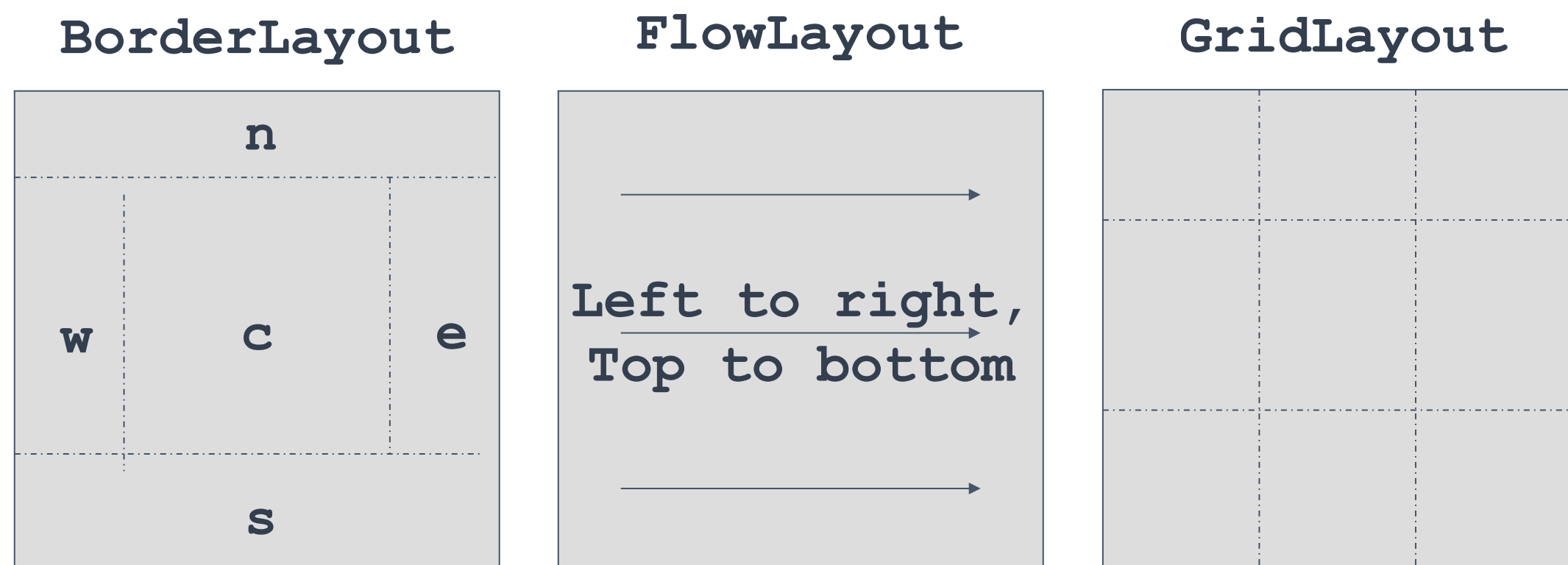
การจัดตำแหน่ง (Layout)

การจัดวางผังของส่วนประกอบกราฟิก



โปรแกรม GUI ของภาษาจาวาจะ*จัดวางส่วนประกอบกราฟิกต่าง ๆ ลงในออบเจ็คของคลาสประเภท*
Container โดยอัตโนมัติ ซึ่ง `LayoutManager` เป็นอินเตอร์เฟสที่ใช้ในการกำหนดวิธีการจัดวาง
 ผังส่วนประกอบกราฟิก คลาสที่ `implements` อินเตอร์เฟสที่ชื่อ `LayoutManager` เพื่อใช้เป็นตัว
 จัดวางผังของส่วนประกอบกราฟิกมีทั้งหมด 3 คลาส คือ

- ✓ `BorderLayout`
- ✓ `FlowLayout`
- ✓ `GridLayout`



การจัดวางผังของส่วนประกอบกราฟิก



- เมธอดที่ชื่อ *method ของ container* **setLayout()** ใช้ในการกำหนดการวางผัง
- เมธอด **add()** สำหรับใส่ส่วนประกอบกราฟิก ซึ่งต้องมีการระบุตำแหน่งที่วางส่วนประกอบ เช่น

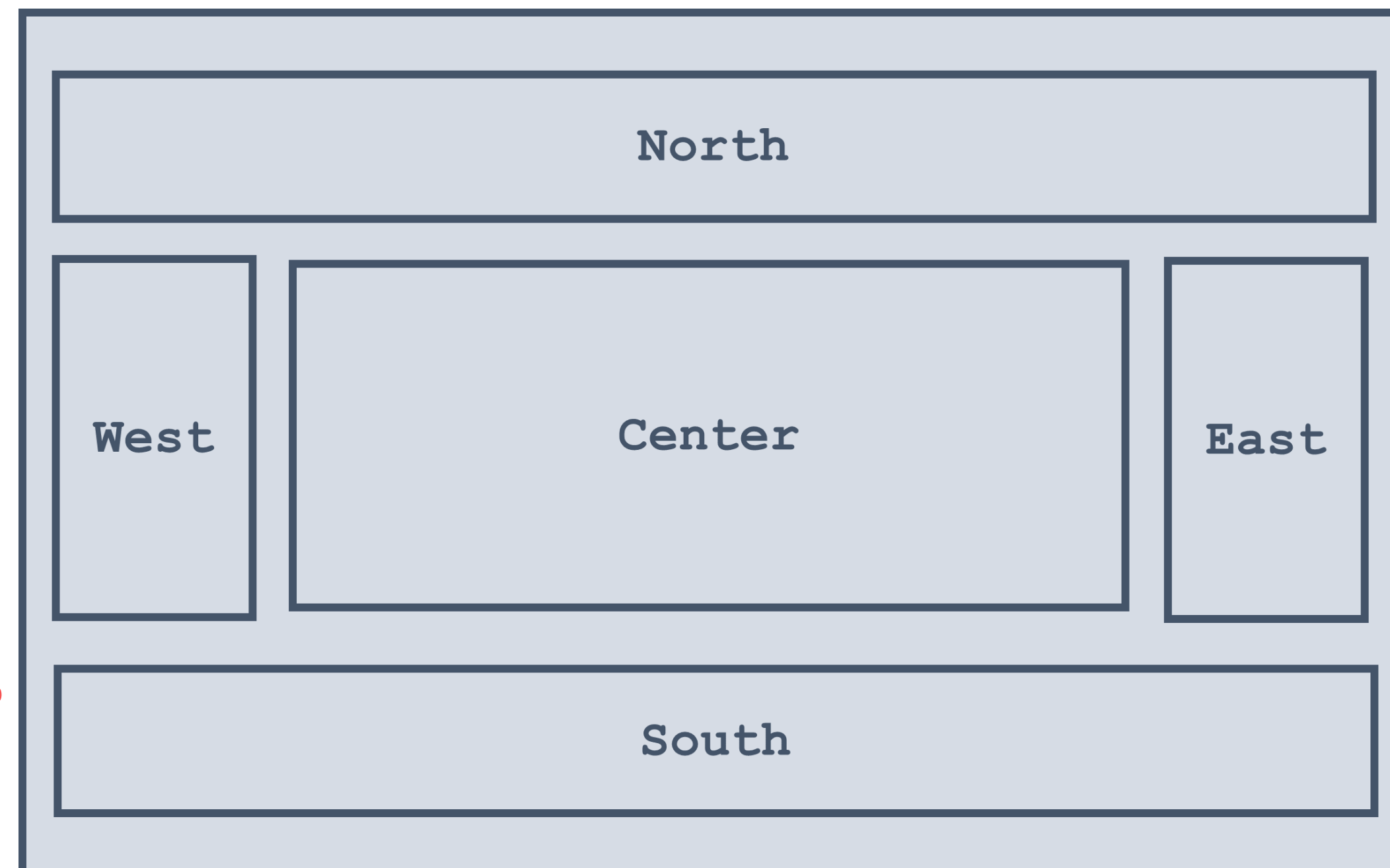
`fr.add(btn1, BorderLayout.NORTH)`

หากไม่ระบุตำแหน่งทิศ ส่วนประกอบจะถูกใส่ไว้ตรงกลาง

การจัดรูปแบบ BorderLayout

- จะสามารถกำหนดตำแหน่งที่วางส่วนประกอบกราฟิกไว้ได้เพียง 5 ตำแหน่ง ได้แก่ North, South, East, West, และ Center *ขนาดไม่เท่ากัน*
- BorderLayout เป็นการจัดรูปแบบการจัดวางเริ่มต้นของ JWindow, JFrame, และ JDialog

- ★ 1. ใส่อะไรก็ได้ในส่วนที่แบ่งไว้
มีได้แค่ 1 component เท่านั้น
- 2. ใส่ component ใดๆก็ได้
ใน 5 ส่วนที่แบ่งไว้
- 3. ไม่จำเป็นต้องใส่ในส่วน 5 ส่วน



- ★ BorderLayout เป็น
default layout ของ JFrame

การจัดรูปแบบ BorderLayout

```

import java.awt.*; → import มาเพื่อเรียกใช้ LayoutManager
import javax.swing.*;
  
```

```

public class BorderLayoutSample {
  
```

① private JFrame fr;

★ { private JButton bn1, bn2, bn3, bn4, bn5;

public BorderLayoutSample() { Constructor

```

    fr = new JFrame("Button Sample");
  
```

```

    fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  
```

```

    bn1 = new JButton("B1");
  
```

```

    bn2 = new JButton("B2");
  
```

```

    bn4 = new JButton("B4");
  
```

```

    bn3 = new JButton("B3");
  
```

```

    bn5 = new JButton("B5");
  
```

```

    fr.setLayout(new BorderLayout());
  
```

```

    fr.add(bn1, BorderLayout.NORTH);
  
```

```

    fr.add(bn2, BorderLayout.SOUTH);
  
```

```

    fr.add(bn3, BorderLayout.EAST);
  
```

```

    fr.add(bn4, BorderLayout.WEST);
  
```

```

    fr.add(bn5); ★ ถ้าไม่กำหนดตำแหน่ง default = CENTER
  
```

```

    fr.setSize(200, 150);    fr.setVisible(true);
  
```

```

}public static void main(String args[]) {
  
```

```

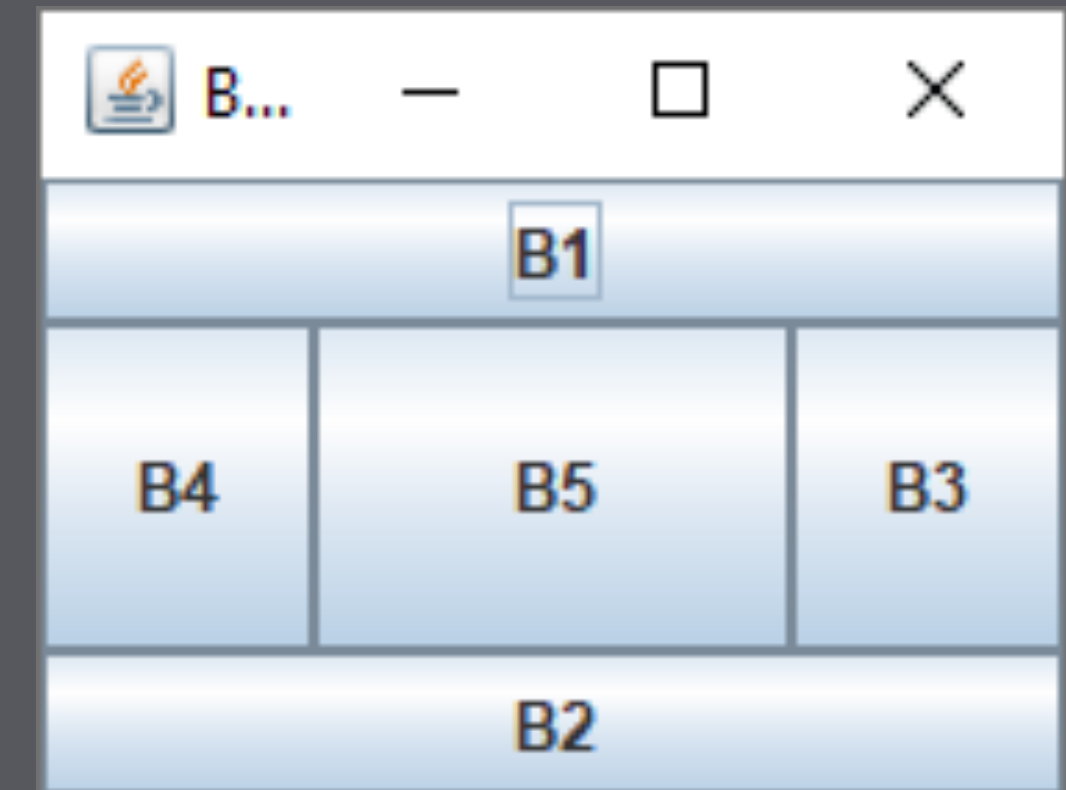
    BorderLayoutSample b = new BorderLayoutSample();
  
```

```

}
  
```

```

}
  
```



Object ที่ส่งมาจาก LayoutManager

[BorderLayout b = new BorderLayout();]

การจัดรูปแบบ **FlowLayout**

★ FlowLayout เป็น
default layout ของ
JPanel

- ตัวจัดวางผังแบบ FlowLayout จะจัดวางผังส่วนประกอบกราฟิกจากซ้ายไปขวา
- ถ้าความกว้างของ Container ไม่พอ ตัวจัดวางผังจะนำส่วนประกอบกราฟิกที่เหลือลงไปในตำแหน่งถัดไปด้านล่าง
- JFrame สามารถใช้ตัวจัดวางผังแบบนี้ได้ โดยเรียกใช้เมธอด `setLayout()` แล้วสร้างตัวจัดวางผังโดยใช้คำสั่ง `new FlowLayout()` ดังนี้

```
fr.setLayout(new FlowLayout());
```

- FlowLayout เป็นการจัดรูปแบบการจัดวางเริ่มต้นของ JApplet และ JPanel

ไม่แบ่งพื้นที่

★ 1. ใส่ component ให้อยู่กันได้

Left to right,
Top to bottom

FlowLayout

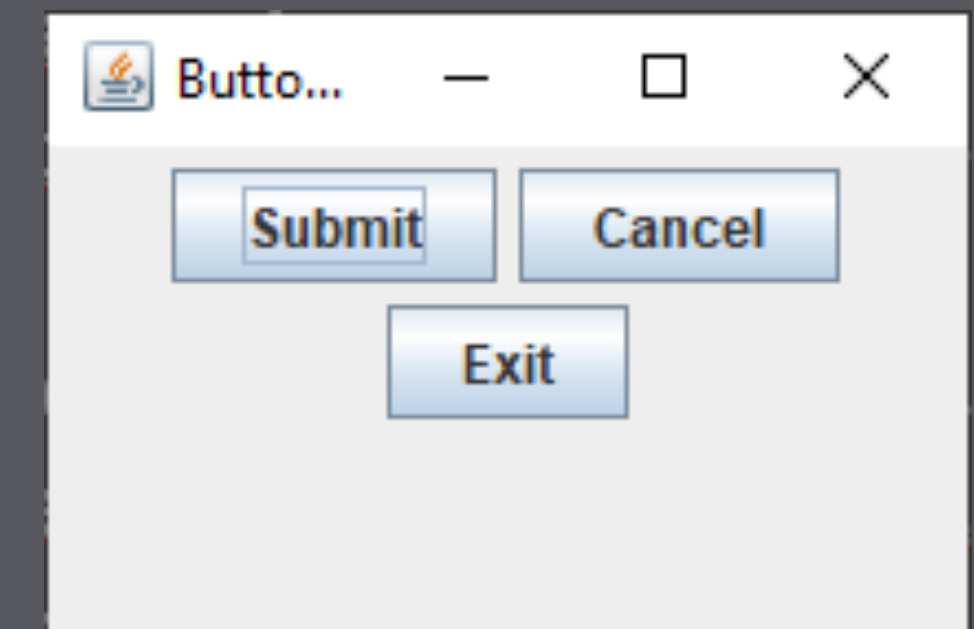
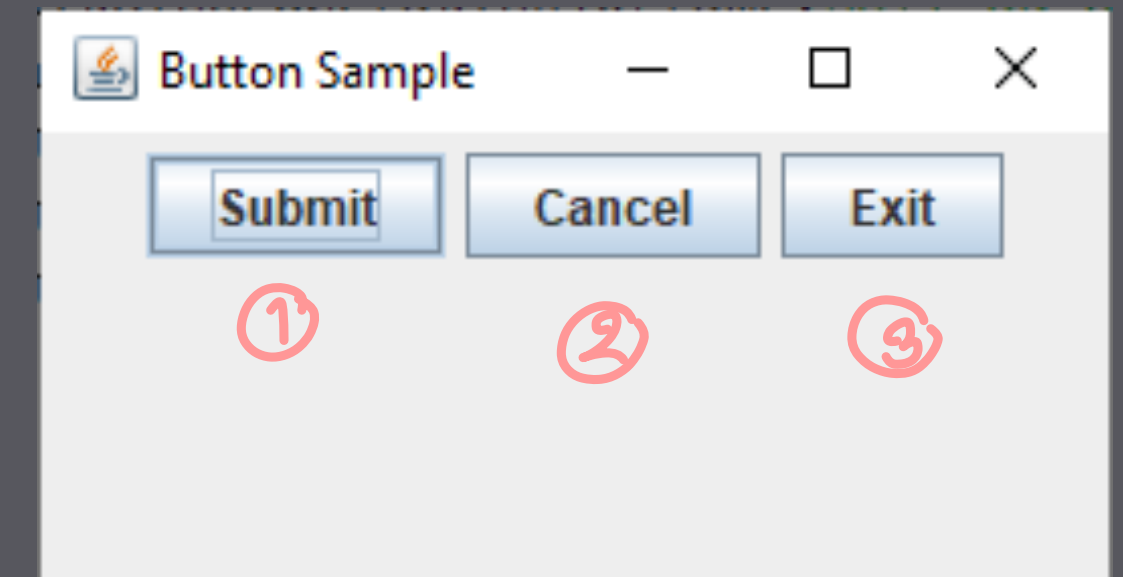
การจัดรูปแบบ **FlowLayout**

```

import java.awt.*;
import javax.swing.*;
public class FlowLayoutSample {
    private JFrame fr;
    private JButton bn1, bn2, bn3;
    public FlowLayoutSample() {
        fr = new JFrame("Button Sample");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr.setLayout(new FlowLayout());
        bn1 = new JButton("Submit");
        bn2 = new JButton("Cancel");
        bn3 = new JButton("Exit");
        ① fr.add(bn1);
        ② fr.add(bn2);
        ③ fr.add(bn3);
        fr.setSize(200, 150);
        fr.setVisible(true);
    }
    public static void main(String args[]) {
        FlowLayoutSample f = new FlowLayoutSample();
    }
}

```

★ ตำแหน่งการใส่ของปุ่มก็ขึ้นกับการ add



★ FlowLayout sensitive ต่อ
ขนาดหน้าต่าง

การจัดรูปแบบ **GridLayout**

- จะแบ่ง **Container** เป็นช่องย่อย ๆ ขนาดเท่ากัน

★ 1. ใส่ component ได้เพียง 1 ต่อจำนวนช่องที่กำหนดไว้
 2. ในแต่ละ component ใส่ได้จะถูกใส่ให้เต็มพื้นที่

- **JFrame** สามารถใช้ตัวจัดวางผังแบบนี้ได้ โดยเรียกใช้เมธอด **setLayout** แล้วสร้างตัวจัดวางผังโดยใช้คำสั่ง **new GridLayout(row, col)** ซึ่งจะต้องระบุจำนวนแถวและคอลัมน์ที่ต้องการแบ่งช่องย่อยด้วย ดังนี้

★ ขนาดช่องย่อย User กำหนด

```
fr.setLayout(new GridLayout(3, 2));
```

1	2
3	4
5	6

GridLayout

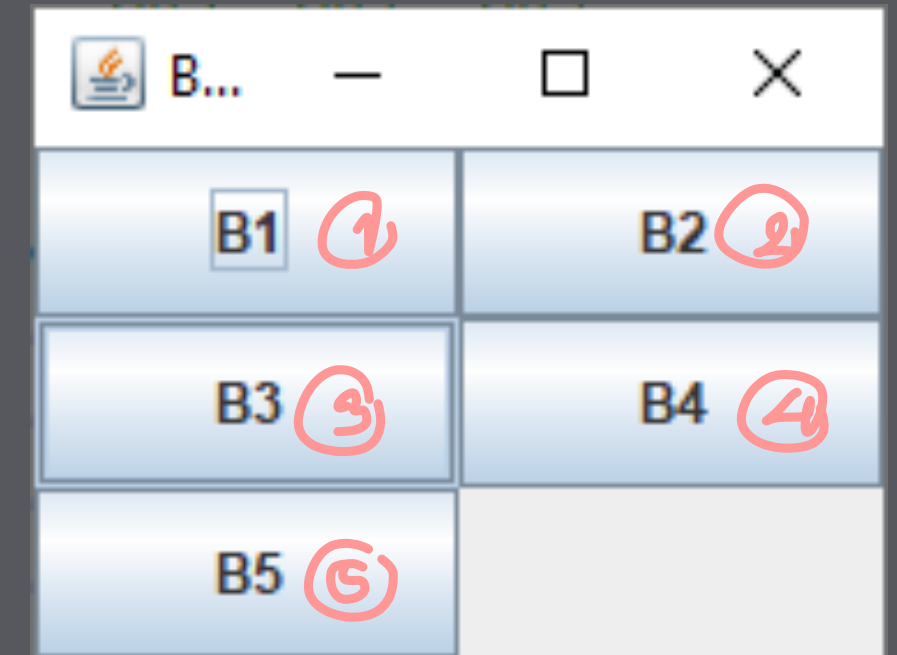
1	2	3
4	5	6
7	8	9

- จะแบ่ง **Container** เป็น 6 ช่องย่อย ที่มี 3 แถว ๆ ละ 2 คอลัมน์
- ใช้เมธอด **add()** ในการใส่ส่วนประกอบกราฟิกลงในแต่ละช่องย่อย โดยจะใส่เรียงจากซ้ายไปขวาและบนลงล่าง

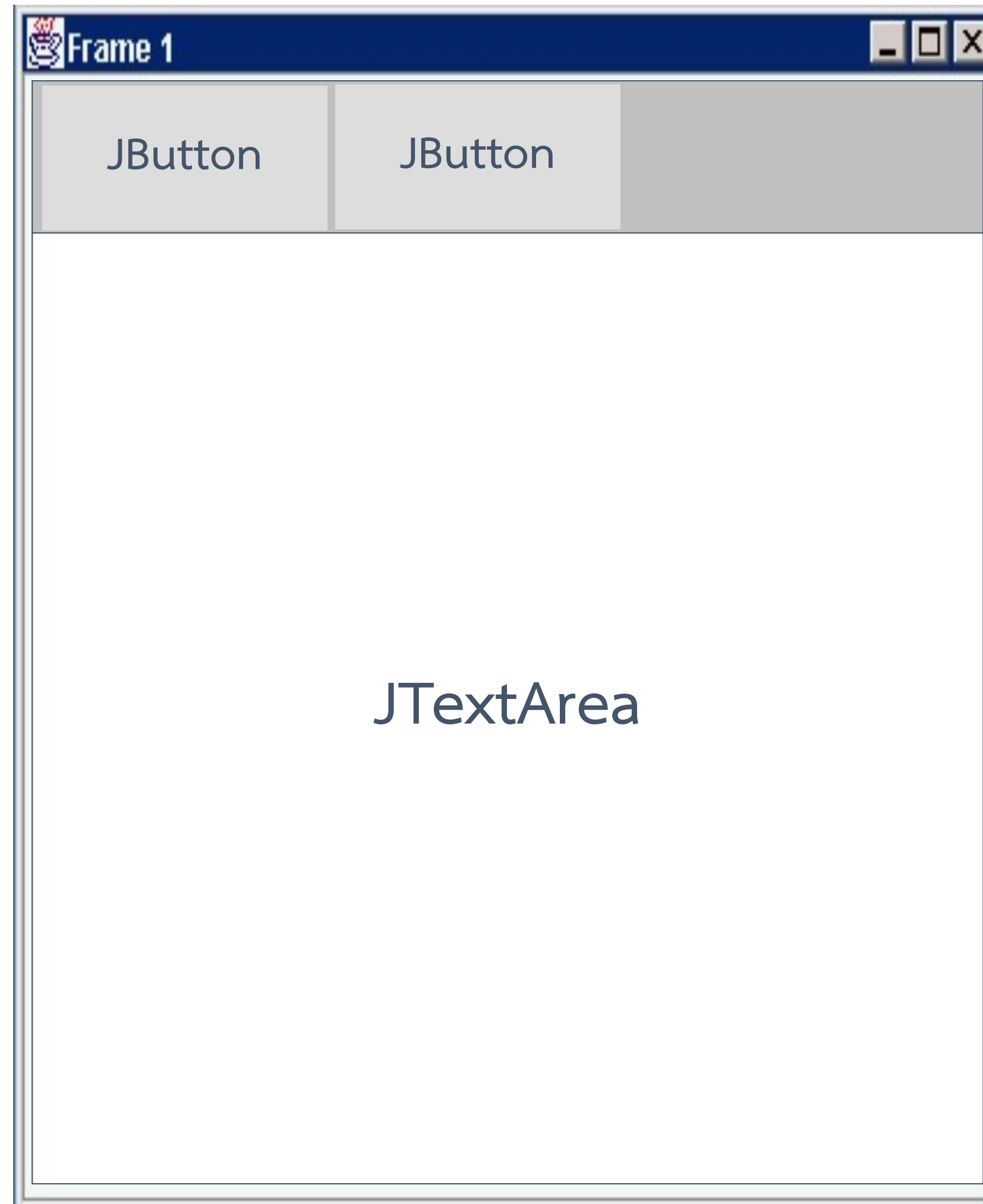
การจัดรูปแบบ GridLayout

```

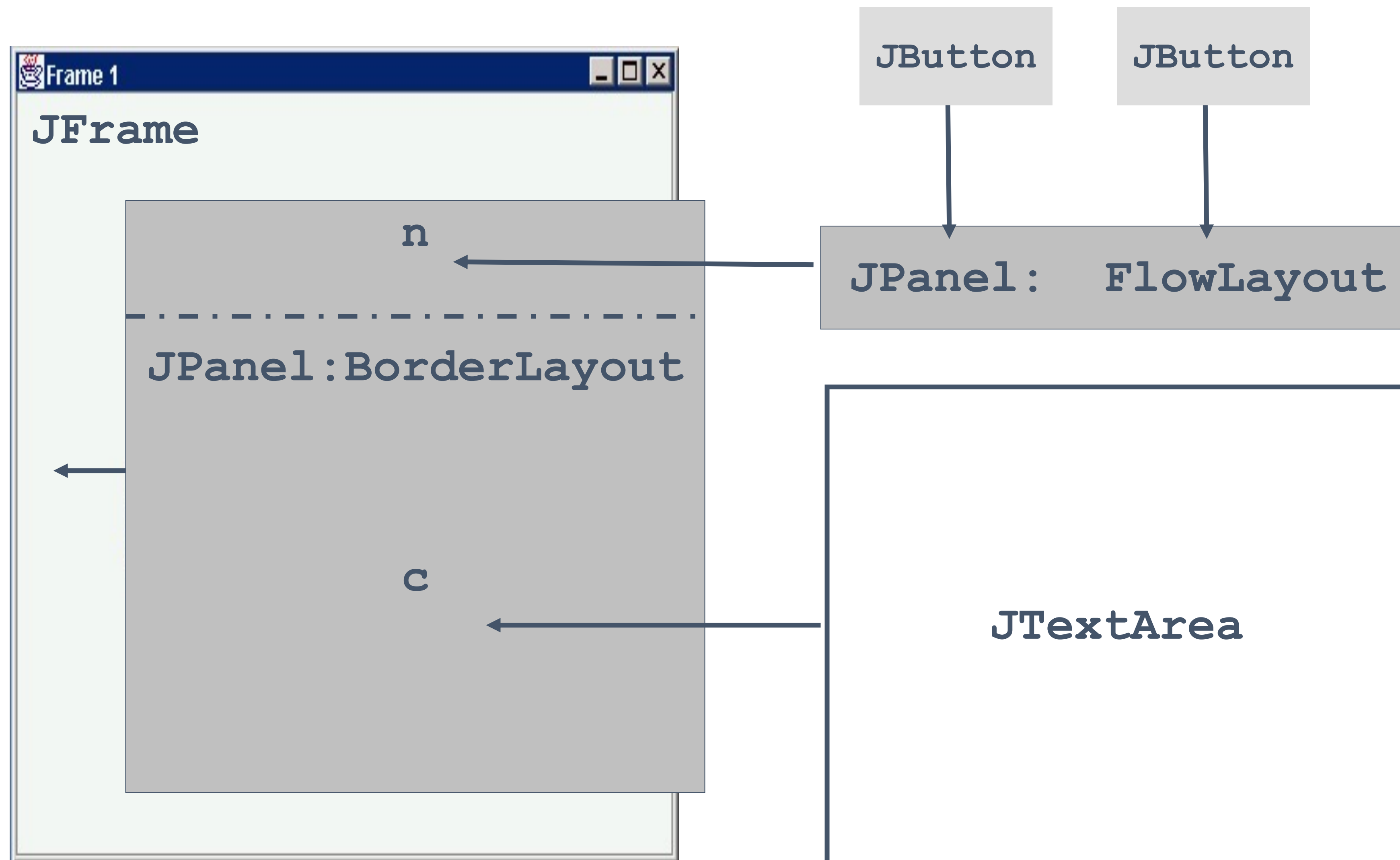
import java.awt.*;
import javax.swing.*;
public class GridLayoutSample {
    private JFrame fr;
    private JButton bn1, bn2, bn3, bn4, bn5;
    public GridLayoutSample() {
        fr = new JFrame("Button Sample");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        bn1 = new JButton("B1");
        bn2 = new JButton("B2");
        bn3 = new JButton("B3");
        bn4 = new JButton("B4");
        bn5 = new JButton("B5");
        fr.setLayout(new GridLayout(3, 2));
        fr.add(bn1); ❶ fr.add(bn2); ❷
        fr.add(bn3); ❸ fr.add(bn4); ❹
        fr.add(bn5); ❺
        fr.setSize(200, 150);
        fr.setVisible(true);
    } public static void main(String args[]) { new GridLayoutSample(); }
}
  
```



การจัดรูปแบบผสม

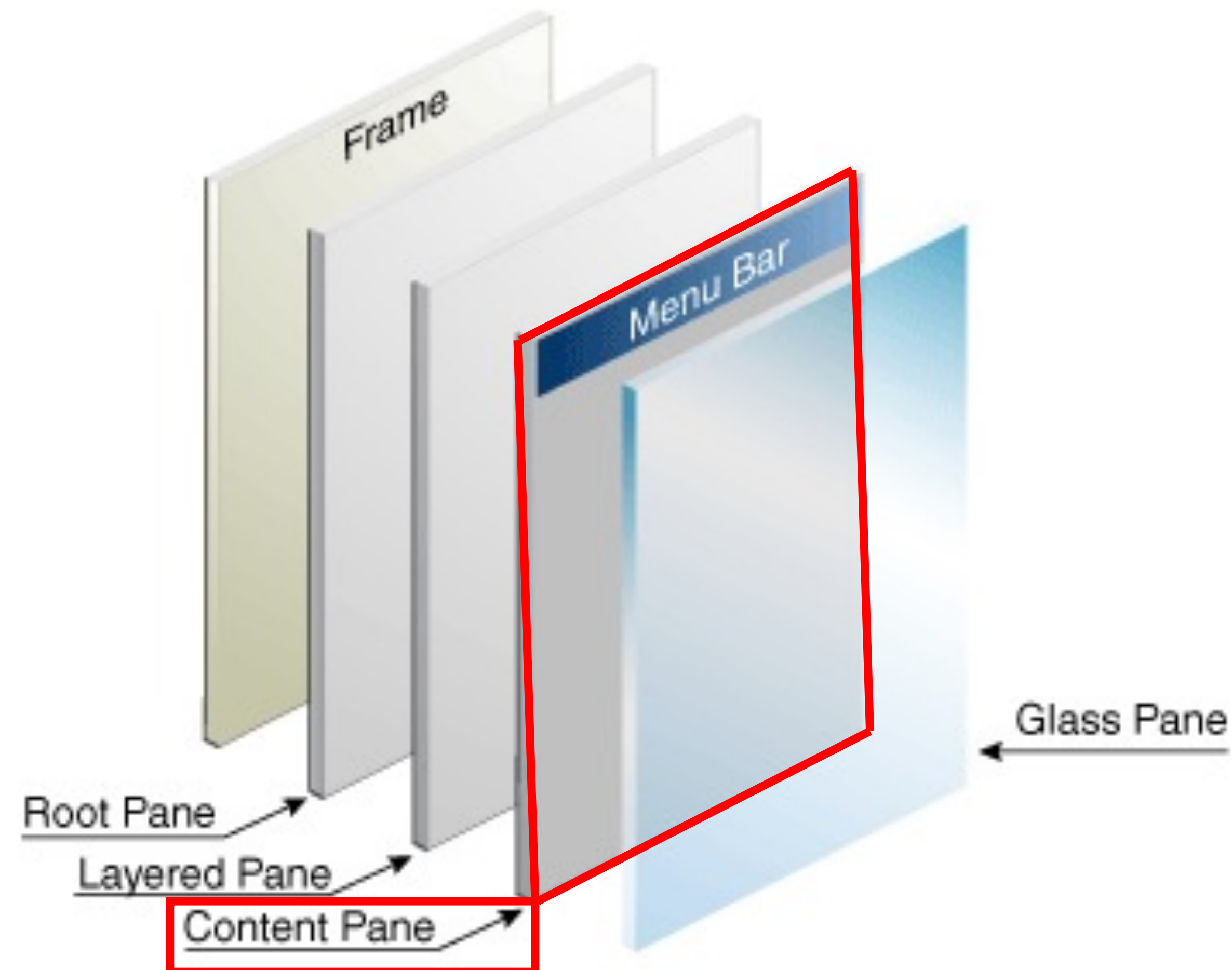


การจัดรูปแบบผสม



JFrame

ออปเจ็คของคลาส **JFrame** แตกต่างกับ **Frame** ตรงที่มีหน้าต่าง (pane) อยู่ 4 หน้าต่างดังนี้



เมธอด `getContentPane()`

- เราไม่สามารถที่จะใส่ส่วนประกอบกราฟิกลงใน `JFrame` ได้โดยตรง แต่จะต้องใส่ลงในหน้าต่างที่เป็น `content pane` แทน
- เราสามารถที่จะเรียกออกเจ็คของคลาสประเภท `Container` ดังกล่าวมาได้ โดยใช้เมธอดที่ชื่อ `getContentPane()` และสามารถที่จะใส่ส่วนประกอบกราฟิกลงในออกเจ็คดังกล่าวได้โดยใช้เมธอด `add()`

```
// ตัวอย่างเช่น
JFrame fr = new JFrame();
JButton bn1 = new JButton("Submit");
fr.add(bn1);
```

การเพิ่ม Component ลงบน JFrame

มีด้วยกัน 3 วิธี ได้แก่

- ① • การเพิ่ม component ลงบน JFrame โดยตรงผ่านเมธอด `add()` // ตั้งแต่ **JDK 1.7** ขึ้นไป

`[object ของคลาส JFrame].add([object ของ Component])`

★ วิธีเก่า : ไม่ใช้แล้ว

- ② • การเพิ่ม component ลงบน Content Pane ใน JFrame ผ่านเมธอด `getContentPane()`

`[object ของคลาส JFrame].getContentPane().add([object ของ Component])`

- ③ • การแทนที่ Content Pane ด้วย JPanel

`[object ของคลาส JFrame].setContentPane([object ของคลาส JPanel])`

```
import java.awt.*;
import javax.swing.*;
public class Test01 {
    private JFrame fr;
    private JButton bn1
    public Test01() {
        fr = new JFrame("Button Sample");
        bn1 = new JButton("B1");
        fr.setLayout(new BorderLayout());

        // แบบที่ 1
        // fr.add(bn1);
        // แบบที่ 2
        fr.getContentPane().add(bn1);

        fr.pack();
        fr.setVisible(true);
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    } public static void main(String args[]) {
        new Test01();
    }
}
```



```
import java.awt.*;
import javax.swing.*;
public class Test01 {
    private JFrame fr;
    private JButton bn1
    private JPanel p1;
    public Test01() {
        fr = new JFrame("Button Sample");
        p1 = new JPanel();
        bn1 = new JButton("B1");

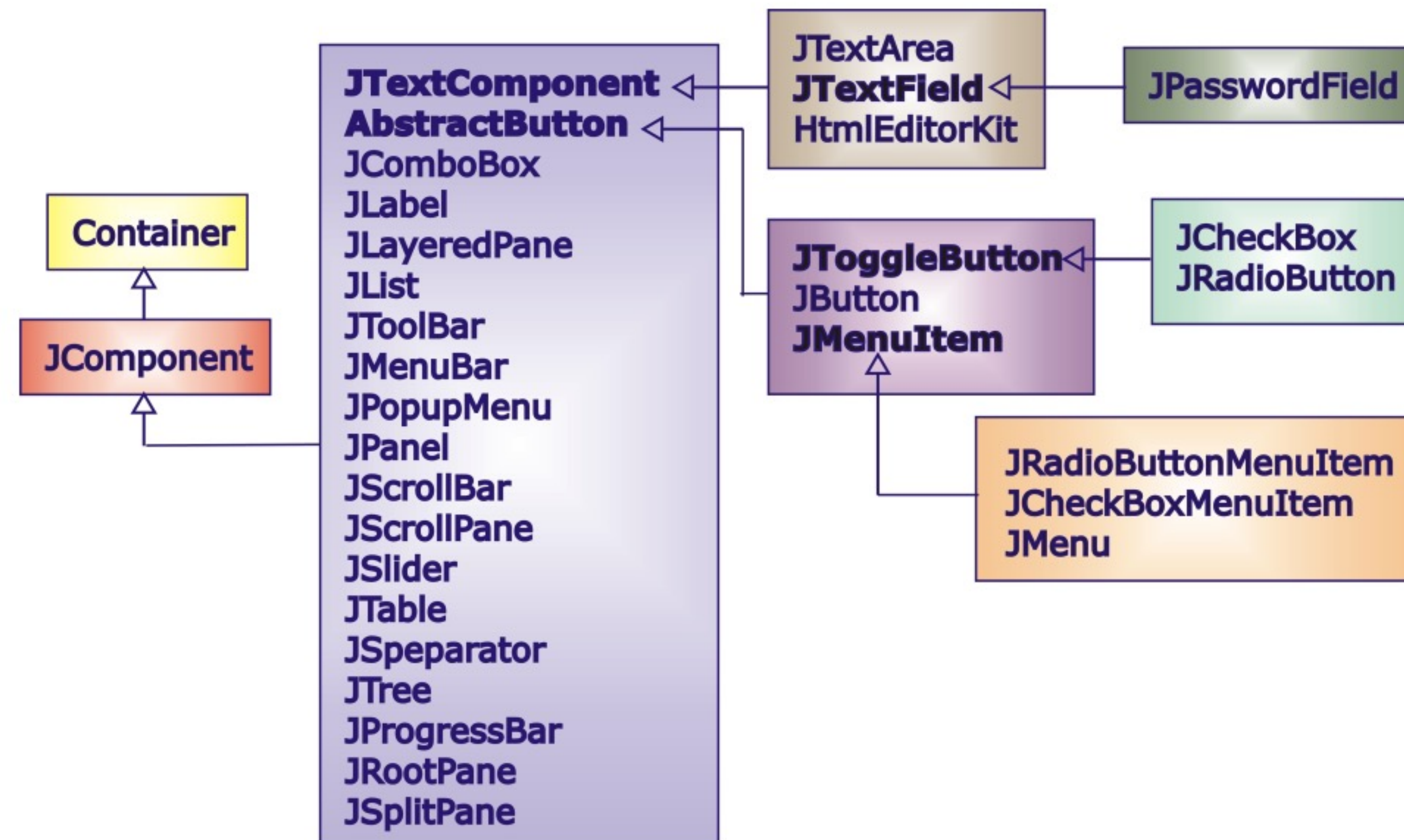
        fr.setLayout(new BorderLayout());

        // แบบที่ 3
        p1.setLayout(new FlowLayout());
        p1.add(bn1);
        fr.setContentPane(p1);

        fr.pack();
        fr.setVisible(true);
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    } public static void main(String args[]) {
        new Test01();
    }
}
```

คอมโพเนนต์ (Component)

Swing Components



Swing Components




แพ็คเกจ Swing จะมีคลาสที่เป็นส่วนประกอบกราฟิกที่สอดคล้องกับคลาสในแพ็คเกจ AWT โดยคลาสเหล่านี้จะมีชื่อขึ้นต้นด้วยตัวอักษร 'J' โดยมีคลาสที่สำคัญดังนี้

- JButton เป็นคลาสที่ทำหน้าที่เป็นปุ่มในแพ็คเกจ Swing
- JLabel เป็นคลาสที่ใช้ในการสร้างอ็อบเจกต์ที่ใช้ในการแสดงข้อความในแพ็คเกจ Swing
- JTextField เป็นคลาสที่ใช้ในการป้อนข้อความหนึ่งบรรทัดในแพ็คเกจ Swing
- JTextArea เป็นคลาสที่ใช้ในการป้อนข้อความหลายบรรทัดในแพ็คเกจ Swing
- JScrollBar เป็นคลาสที่ทำหน้าที่เป็นแถบควบคุมเพื่อให้ผู้ใช้เลื่อนไปยังตำแหน่งที่ต้องการได้ในแพ็คเกจ Swing
- JCheckBox เป็นคลาสที่ทำหน้าที่คล้ายปุ่มในแพ็คเกจ Swing
- JChoice เป็นคลาสที่ผู้ใช้สามารถเลือกรายการได้ในแพ็คเกจ Swing


Swing Components

Basic Controls
Simple components that are used primarily to get input from the user; they may also show simple state.



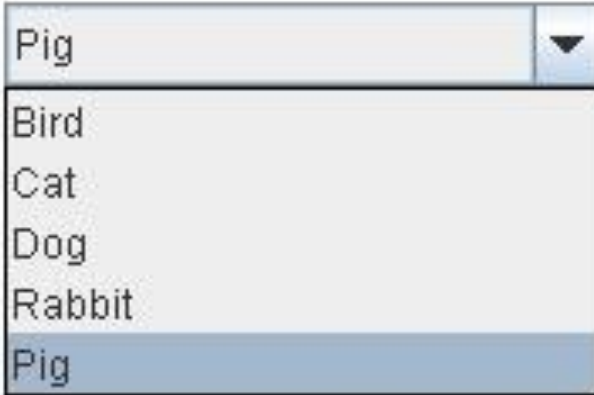
JButton

☒ Chin
☒ Glasses
☒ Hair
☒ Teeth



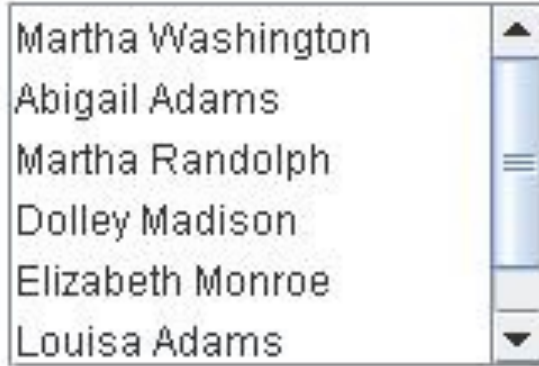
JCheckBox

Pig
Bird
Cat
Dog
Rabbit
Pig





JComboBox


Martha Washington
Abigail Adams
Martha Randolph
Dolley Madison
Elizabeth Monroe
Louisa Adams



JList


A Menu Another Menu

A text-only menu item Alt-1
 Both text and icon

☒ A radio button menu item
☐ Another one
☐ A check box menu item
☐ Another one
A submenu



JMenu


☐ Bird
☐ Cat
☐ Dog
☐ Rabbit
☒ Pig



JRadioButton

Frames Per Second

0 10 20 30



JSlider

Date: 07/2006

JSpinner

City: Santa Rosa

JTextField

Enter the password:

JPasswordField

Swing Components



Interactive Displays of Highly Formatted Information

Color chooser: A window with tabs for Swatches, HSB, and RGB, displaying a grid of color swatches.

File chooser: A window titled 'Open' showing a file list with columns for Look in, C:\, and a list of files (emacslib, host-news, java, mbin).

Table: A table with columns First Name, Last Name, and Favorite Food.

First Name	Last Name	Favorite Food
Jeff	Dinkins	
Ewan	Dinkins	
Amy	Fowler	
Hania	Gajewska	
David	Geary	

Text: A text area showing various text styles: red, blue, green, small, large, italic, and bold.

Tree: A tree view showing a hierarchy of folders: Music, Classical, Beethoven, Brahms, Mozart, Jazz, and Rock.

Uneditable Information Displays

Label: A window titled 'LabelDemo' showing a label with a sun icon and the text 'Image and Text' and 'Text-Only Label'.

Progress bar: A progress bar showing 18% completion.

Tool tip: A tooltip displaying the text 'Moooooooo' over a cow image.

JButton

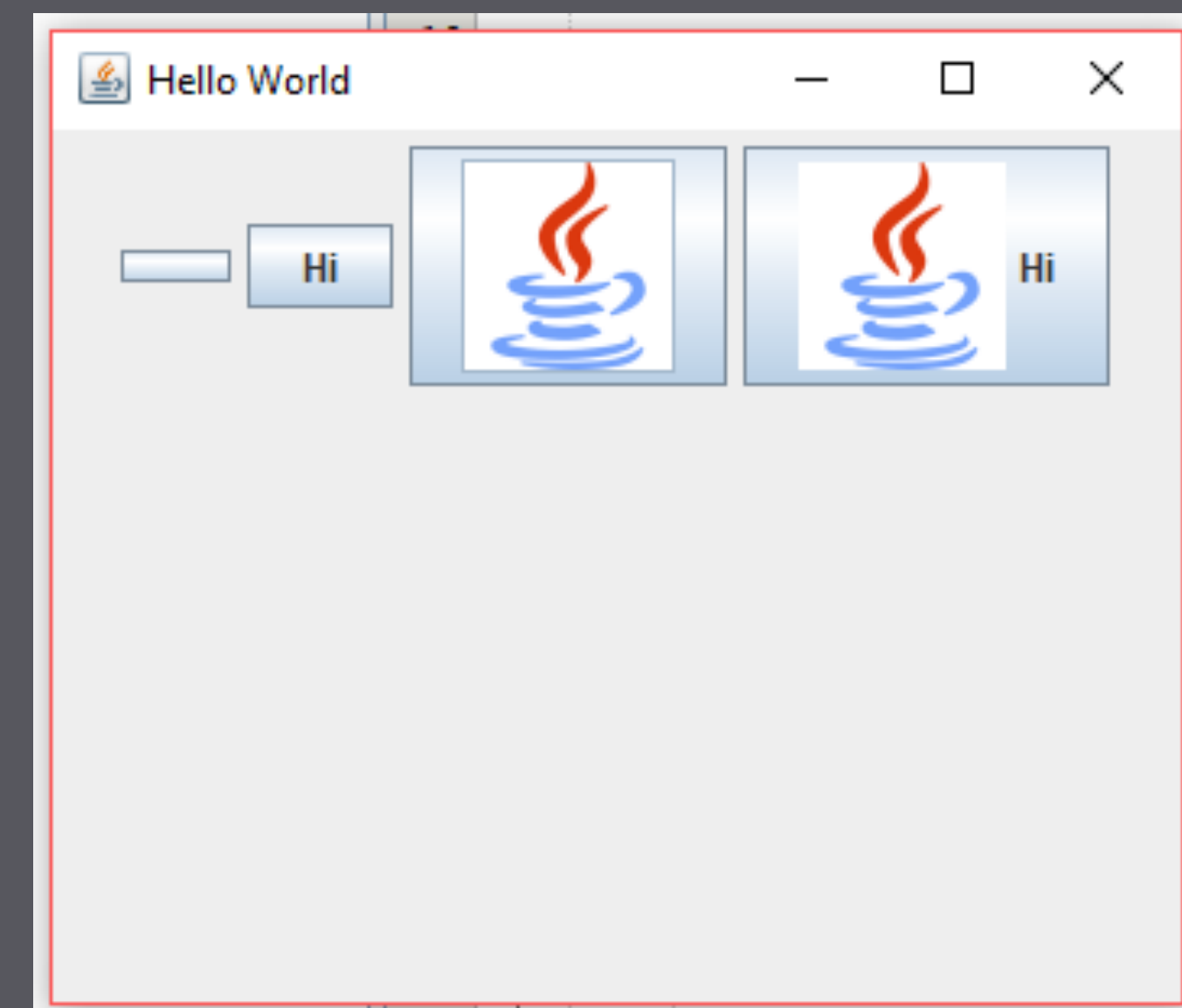
JButton เป็นคลาสที่ใช้ในการสร้างออปเจ็คที่แสดงเป็นปุ่ม โดยจะมีข้อความ (text) ปรากฏอยู่บนปุ่ม โดยสืบทอดมาจากคลาสที่ชื่อ Component

//Constructor ของคลาส JButton มีรูปแบบดังนี้

```
public JButton()  
public JButton(String text)  
public JButton(Icon icon)  
public JButton(String text, Icon icon)
```

//เมธอดที่สำคัญในการจัดการกับ JButton มีดังนี้

```
public void setText(String text)  
public String getText()  
public void setMnemonic(char c)  
public void setIcon(Icon c)  
public void setToolTipText(String text)
```



JButton



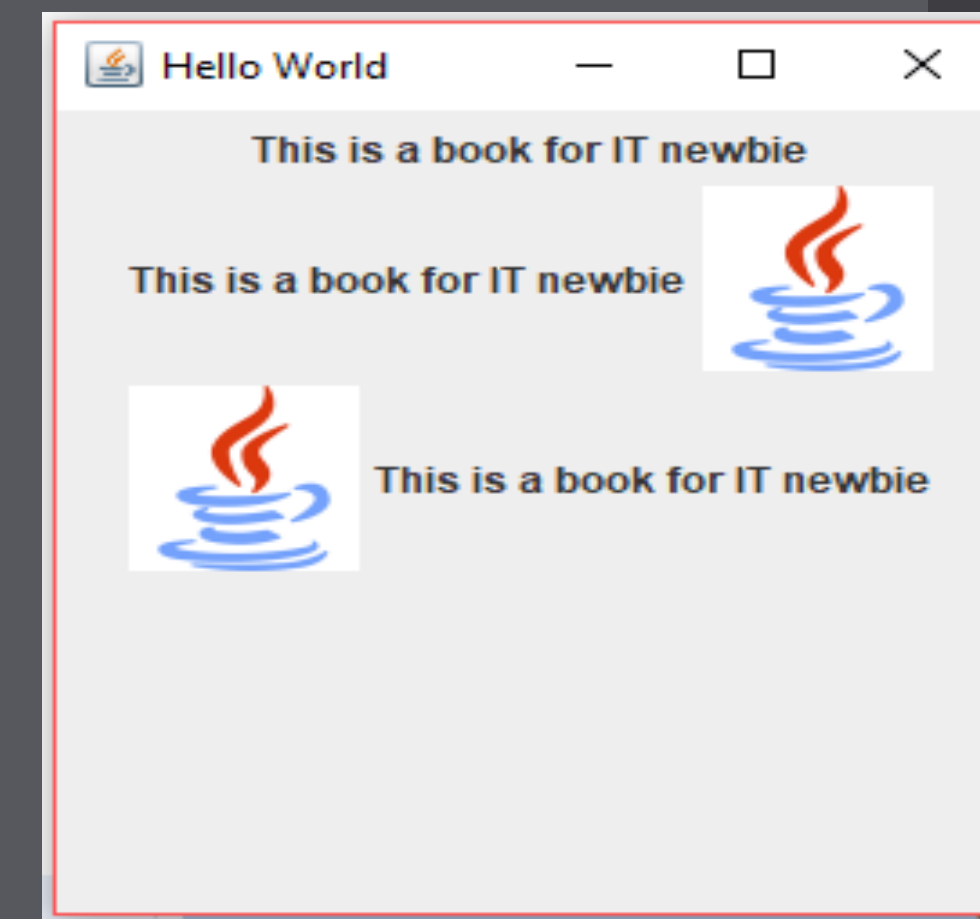
ตัวอย่างการสร้างอ็อบเจกต์ของคลาส `JButton` ที่แสดงข้อความ (`ToolTip`) และกำหนด `shortcut` โดยอาศัยคำสั่งต่อไปนี้

```
JButton b1 = new JButton("Demo button");  
b1.setMnemonic(KeyEvent.VK_D);           // Short Key Alt + D  
b1.setToolTipText("Click Here");
```

JLabel

คือ component ไว้**แสดงข้อความ** ส่วนมากจะใช้เพื่ออธิบายการทำงานบางอย่าง อาทิเช่น บ่งบอกว่า
Textbox นี้ต้องการให้กรอกค่าอะไร

```
//Constructor ของคลาส Button มีรูปแบบดังนี้  
public JLabel(String str);  
public JLabel(String str, int alignment);  
public JLabel(ImageIcon img);  
public JLabel(String str, ImageIcon img, int alignment);  
  
//เมธอดที่สำคัญในการจัดการกับข้อความมีดังนี้  
public void setText(String text)
```



JLabel



```
import java.awt.*;
import java.net.URL;
import javax.swing.*;

public class JLabelDemo {
    private JFrame fr;
    private JLabel label1, label2, label3;

    public JLabelDemo() {
        ImageIcon icon = null;
        fr = new JFrame("JLabel Sample");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        URL imageURL = JLabelDemo.class.getResource("images/testImg.jpg");
        if (imageURL != null) {
            icon = new ImageIcon(imageURL);
        }

        fr.setLayout(new GridLayout(3, 1));
        //Create the first label.
        label1 = new JLabel("Image and Text", icon, JLabel.CENTER);
```

มีต่อ



JLabel



```
//Set the position of its text, relative to its icon:
label1.setVerticalTextPosition(JLabel.BOTTOM);
label1.setHorizontalTextPosition(JLabel.CENTER);

//Create the other labels.
label2 = new JLabel("Text-Only Label");
label3 = new JLabel(icon);

//Create tool tips, for the heck of it.
label1.setToolTipText("A label containing both image and text");
label2.setToolTipText("A label containing only text");
label3.setToolTipText("A label containing only an image");

//Add the labels.
fr.add(label1);
fr.add(label2);
fr.add(label3)

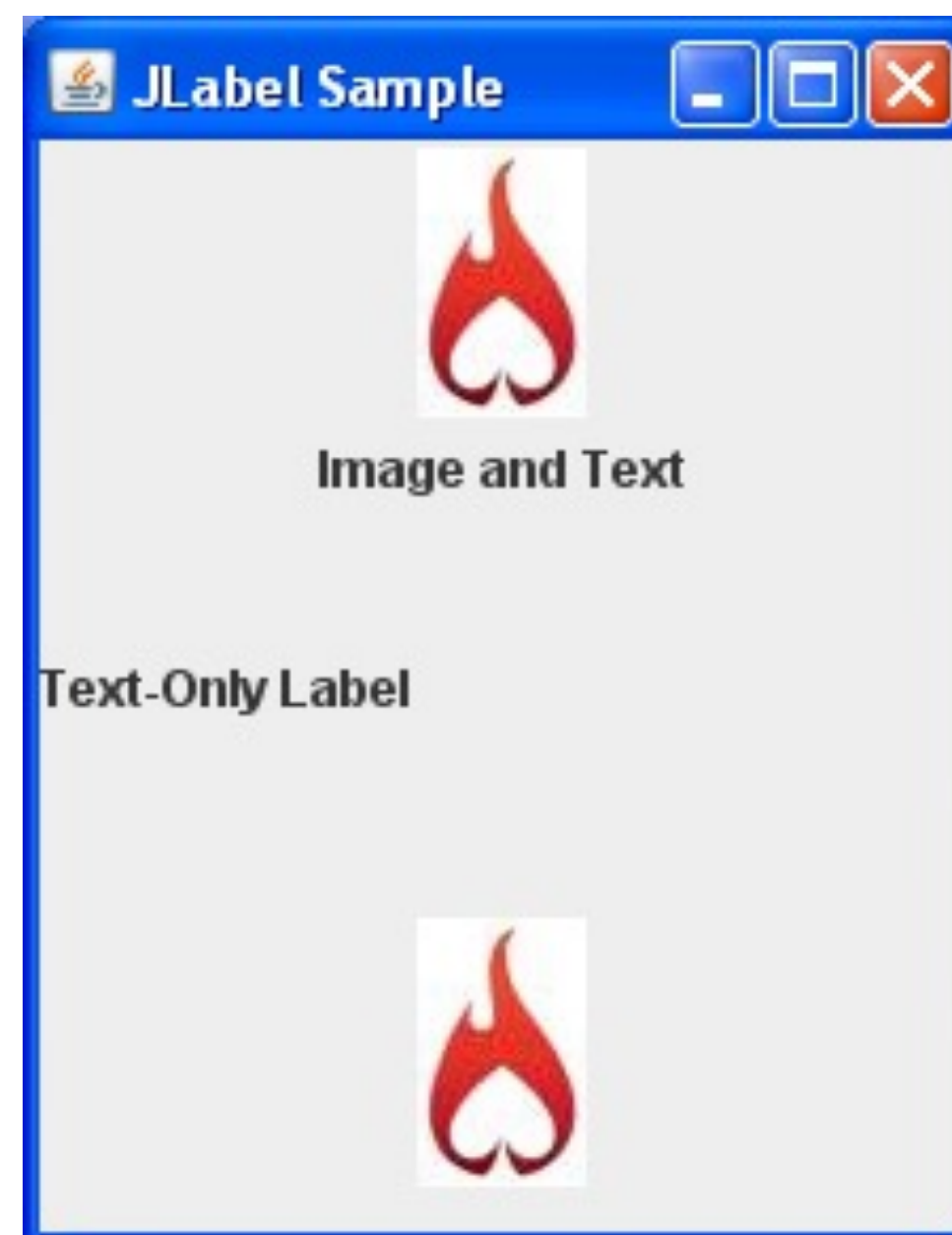
fr.pack();
fr.setVisible(true);
}
```

มีต่อ



JLabel

```
public static void main(String args[]) {  
    JLabelDemo obj = new JLabelDemo();  
}  
}
```

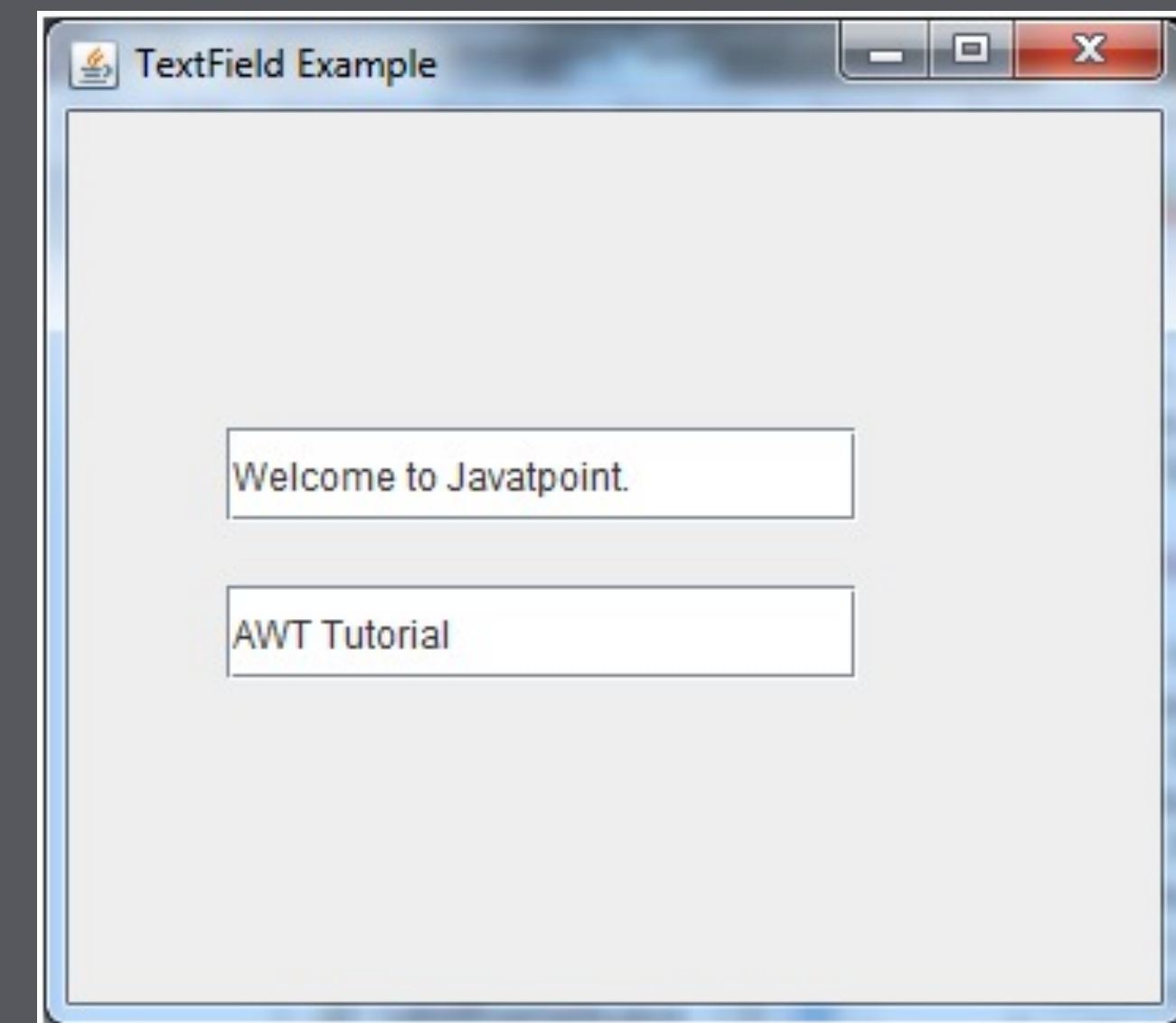


JTextField or Textbox

คือ ส่วนประกอบกราฟิกเพื่อให้ผู้ใช้ป้อนข้อความยาวหนึ่งบรรทัดได้

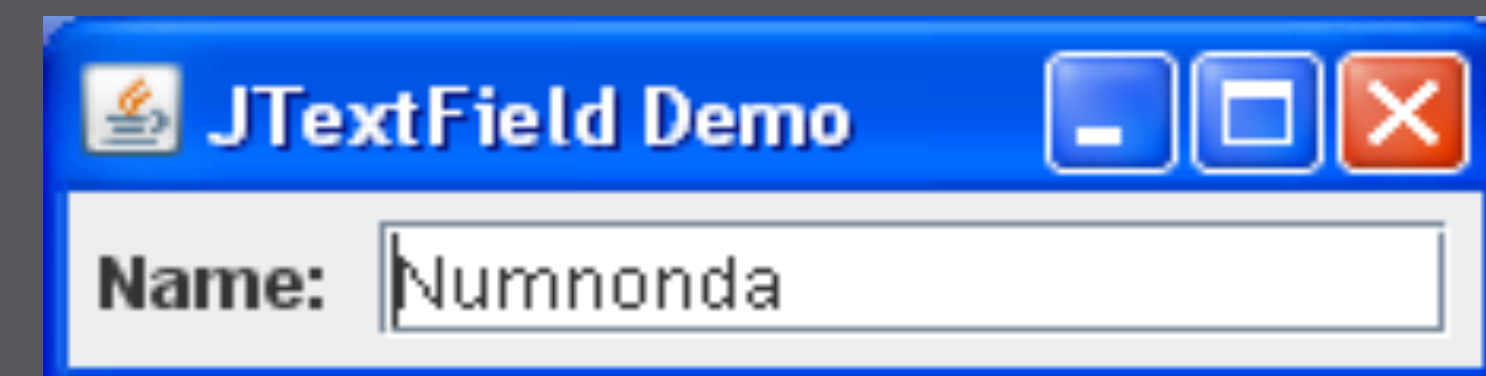
```
// Constructor ของคลาส JTextField ที่สำคัญมีดังนี้
public JTextField()
public JTextField(int col)
public JTextField(String text)
public JTextField(String text, int col)

//เมธอดที่สำคัญในการจัดการกับ JTextField มีดังนี้
public void setText(String text)
public String getText()
public void setEditable(boolean b)
```



JTextField

```
import java.awt.*;
import javax.swing.*;
public class JTextFieldDemo {
    private JFrame fr;
    private JLabel l;
    private JTextField tf;
    public JTextFieldDemo() {
        fr = new JFrame("JTextField Demo");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        l = new JLabel("Name: ");
        tf = new JTextField("Numnonda", 15);
        fr.setLayout(new FlowLayout());
        fr.add(l);
        fr.add(tf);
        fr.pack();
        fr.setVisible(true);
    }
    public static void main(String args[]) {
        new JTextFieldDemo();
    }
}
```



JTextArea

คือ ส่วนประกอบกราฟิกเพื่อให้ผู้ใช้ป้อนข้อความได้จำนวนหลายบรรทัดตามที่ระบุ

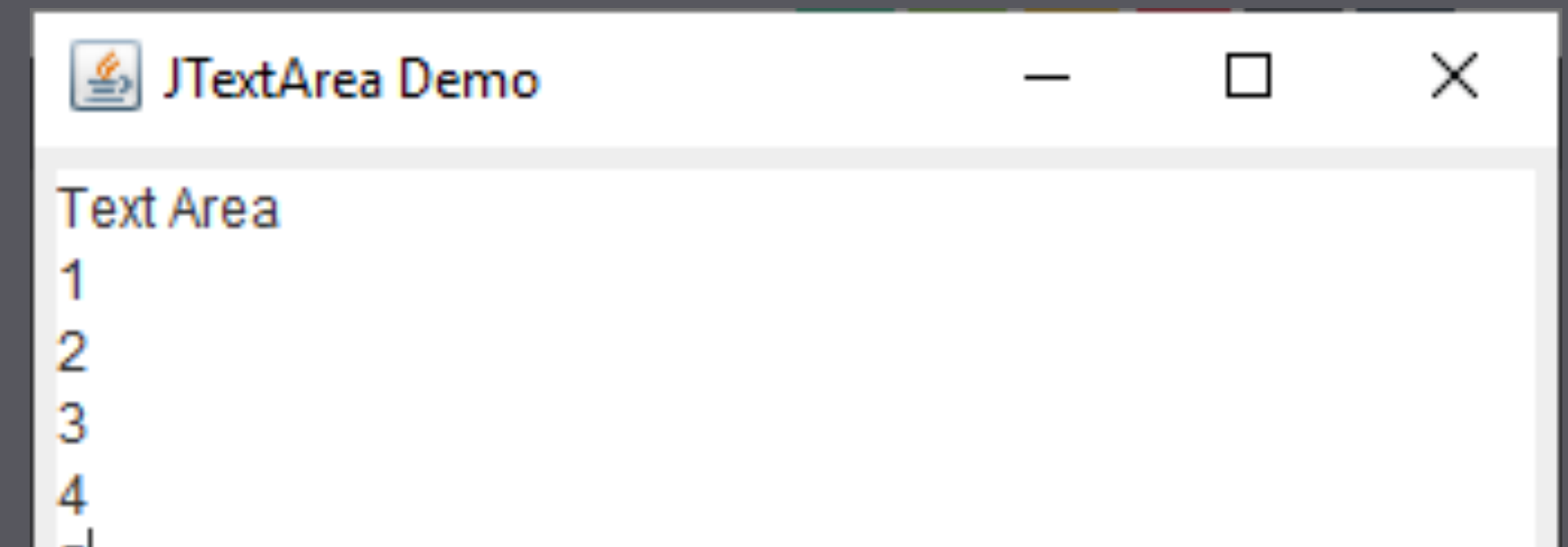
```
// Constructor ของคลาส JTextArea ที่สำคัญมีดังนี้
public JTextArea()
public JTextArea(String Text)
public JTextArea(String Text, int row, int col)
public JTextArea(int row, int col)

//เมธอดที่สำคัญในการจัดการกับ JTextArea มีดังนี้
public void setText(String text)
public String getText()
```


JTextArea

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

public class JTextAreaDemo {
    private JFrame fr;
    private JTextArea ta;
    public JTextAreaDemo() {
        fr = new JFrame("JTextArea Demo");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ta = new JTextArea("Text Area", 5, 30);
        fr.setLayout(new FlowLayout());
        fr.add(ta);
        fr.pack();
        fr.setVisible(true);
    }
    public static void main(String args[]) {
        JTextAreaDemo obj = new JTextAreaDemo();
    }
}
```



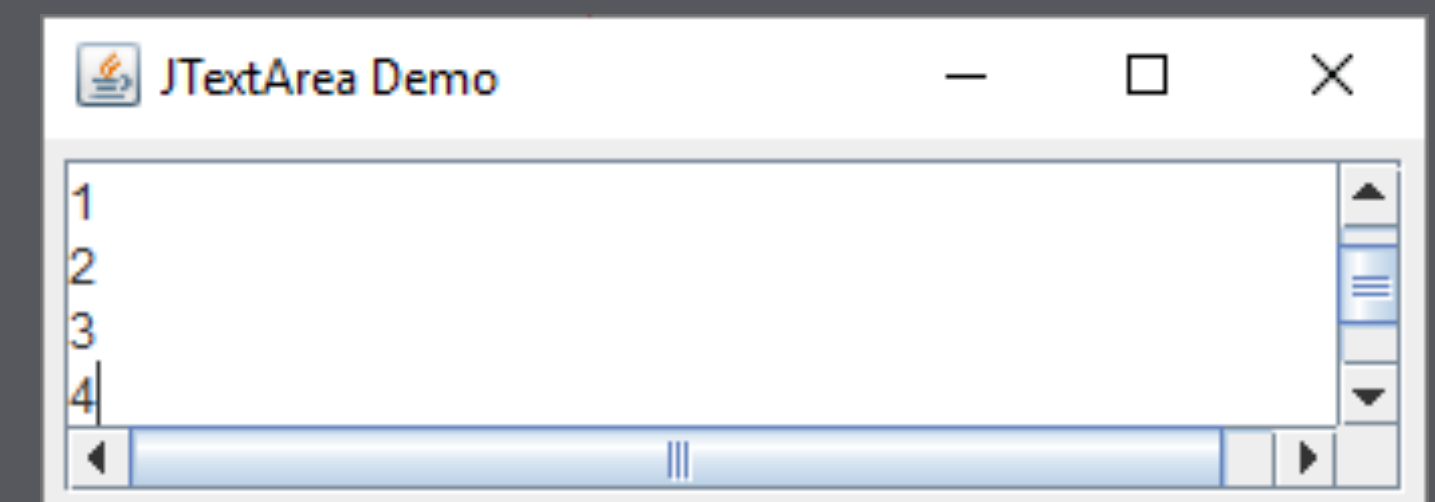
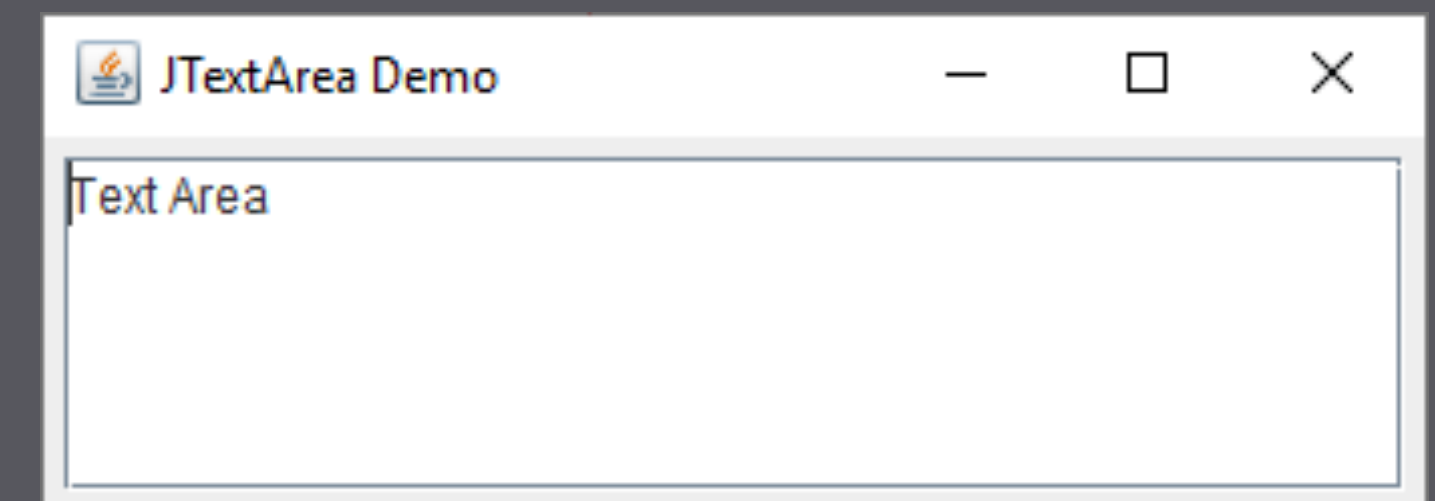
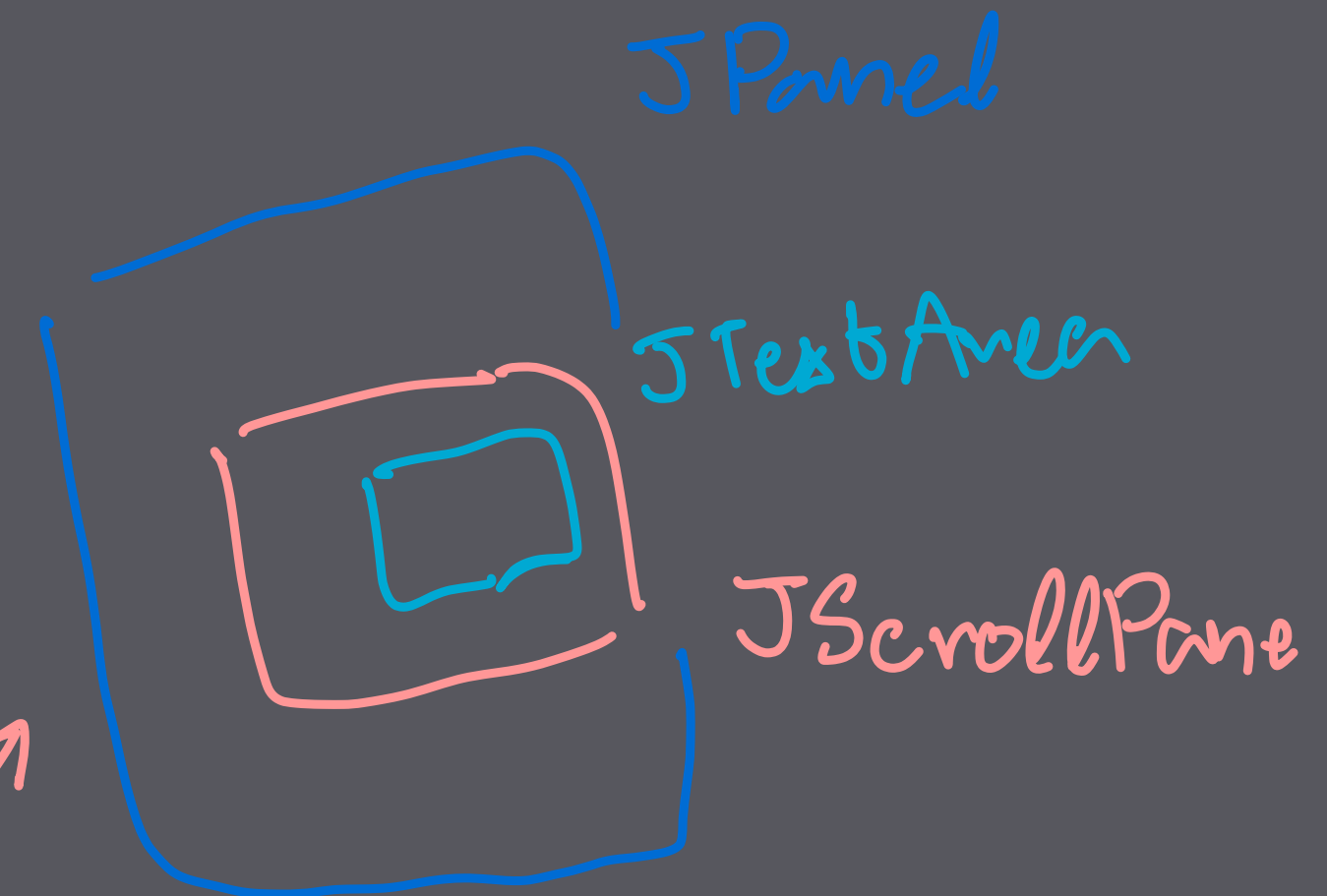
JTextArea

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

public class JTextAreaDemo {
    private JFrame fr;
    private JTextArea ta;
    private JScrollPane jScrollPane;

    public JTextAreaDemo() {
        fr = new JFrame("JTextArea Demo");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ta = new JTextArea("Text Area", 5, 30);
        jScrollPane = new JScrollPane(ta);
        fr.setLayout(new FlowLayout());
        fr.add(jScrollPane);
        fr.pack();
        fr.setVisible(true);
    }

    public static void main(String args[]) {
        JTextAreaDemo obj = new JTextAreaDemo();
    }
}
```



JCheckBox

คือ คือส่วนประกอบกราฟิกที่ให้ผู้ใช้งานเลือกหรือไม่เลือกช่องต่าง ๆ ได้ โดยสามารถเลือกได้หลายช่องพร้อม ๆ กัน และมีข้อความ (Label) อยู่ข้างๆ

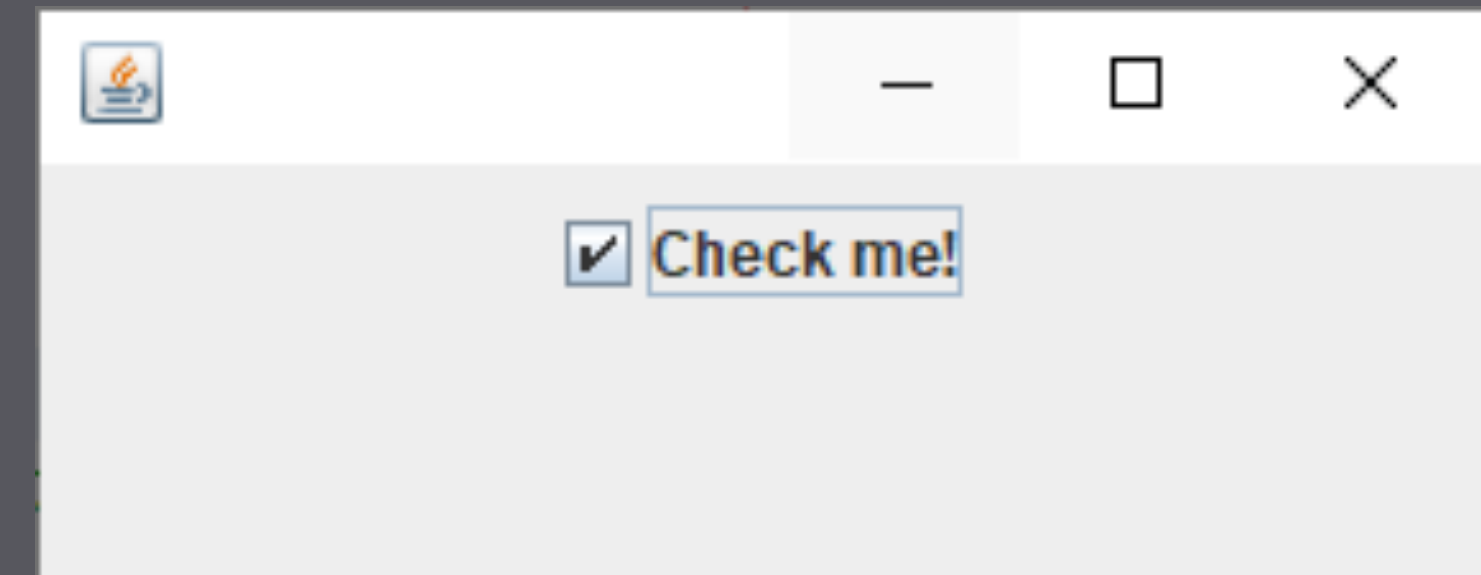
```
// Constructor ของคลาส JCheckbox ที่สำคัญมีดังนี้
public JCheckBox(String label)
public JCheckBox(String label, boolean state)
public JCheckBox(Icon icon)
public JCheckBox(Icon icon, boolean state)
public JCheckBox(String label, Icon icon)
public JCheckBox(String label, Icon icon, boolean state)

//เมธอดที่สำคัญในการจัดการกับ JCheckBox มีดังนี้
public void setSelected(boolean b) ;
public boolean isSelected()
```

JCheckBox

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

public class TestJCheckBox {
    private JFrame fr;
    private JCheckBox ck;
    private TestJCheckBox() {
        fr = new JFrame();
        ck = new JCheckBox("Check me!");
        ck.setSelected(true);
        fr.setSize(300, 120);
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr.setLayout(new FlowLayout());
        fr.getContentPane().add(ck);
        fr.setVisible(true);
        boolean selected = ck.isSelected();
        if (selected)
            System.out.println("Check box state is selected.");
        else
            System.out.println("Check box state is not selected.");
    }
    public static void main(String[] args) {
        new TestJCheckBox();
    }
}
```



JRadioButton



คือ เราสามารถที่จะสร้างตัวเลือกที่เป็นแบบ Radio Button โดยใช้ JRadioButton

- ในกรณีนี้จะกำหนดให้อ็อบเจกต์ JRadioButton หลายตัวอยู่ในกลุ่มของอ็อบเจกต์ของคลาส ButtonGroup ซึ่งจะต้องใช้เมธอด add() มีข้อความ (Label) อยู่ข้างๆ
- คลาส ButtonGroup ไม่ใช่คลาสที่เป็นส่วนประกอบกราฟิก แต่จะใช้ในการสร้างอ็อบเจกต์เพื่อกำหนดกลุ่มของ AbstractButton

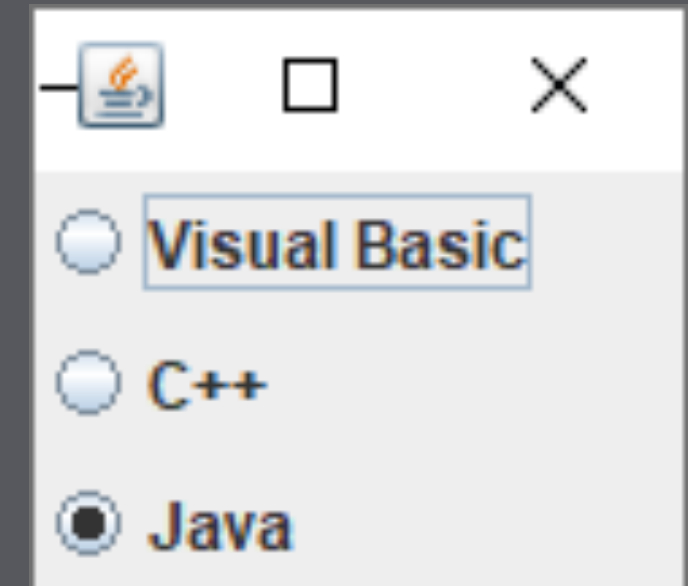
JRadioButton

```
// Constructor ของคลาส JRadioButton ที่สำคัญมีดังนี้
public JRadioButton()
public JRadioButton(Icon icon)
public JRadioButton(Icon icon, boolean selected)
public JRadioButton(String text)
public JRadioButton(String text, boolean selected)
public JRadioButton(String text, Icon icon)
public JRadioButton(String text, Icon icon, boolean selected)

//เมธอดที่สำคัญในการจัดการกับ JRadioButton มีดังนี้
public boolean isSelected()
Public void setSelected(boolean b)
```

JRadioButton

```
import java.awt.GridLayout;
import javax.swing.*;
public class JRBDemo {
    private JFrame fr;
    private JRadioButton c1, c2, c3;
    private ButtonGroup chg;
    public JRBDemo() {
        fr = new JFrame("JRadioButton Demo");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        c1 = new JRadioButton("Visual Basic");
        c2 = new JRadioButton("C++", false);
        c3 = new JRadioButton("Java", true);
        [ chg = new ButtonGroup();
          chg.add(c1);      chg.add(c2);      chg.add(c3); ]
        fr.setLayout(new GridLayout(3,1));
        fr.add(c1);      fr.add(c2);      fr.add(c3);
        fr.pack();
        fr.setVisible(true);
    }
    public static void main(String args[]) { new JRBDemo(); }
}
```



JComboBox

เป็นคลาสสำหรับการสร้างอปเจ็คที่เป็นรายการให้ผู้ใช้สามารถเลือกได้ ซึ่งจะแสดงรายการปรากฏให้เห็นเฉพาะรายการที่เลือกเพียงรายการเดียว แต่จะแสดงรายการทั้งหมดหากมีการคลิกเมาส์

```
// Constructor ของคลาส JComboBox ที่สำคัญมีดังนี้
public JComboBox()
public JComboBox([]Object objs)

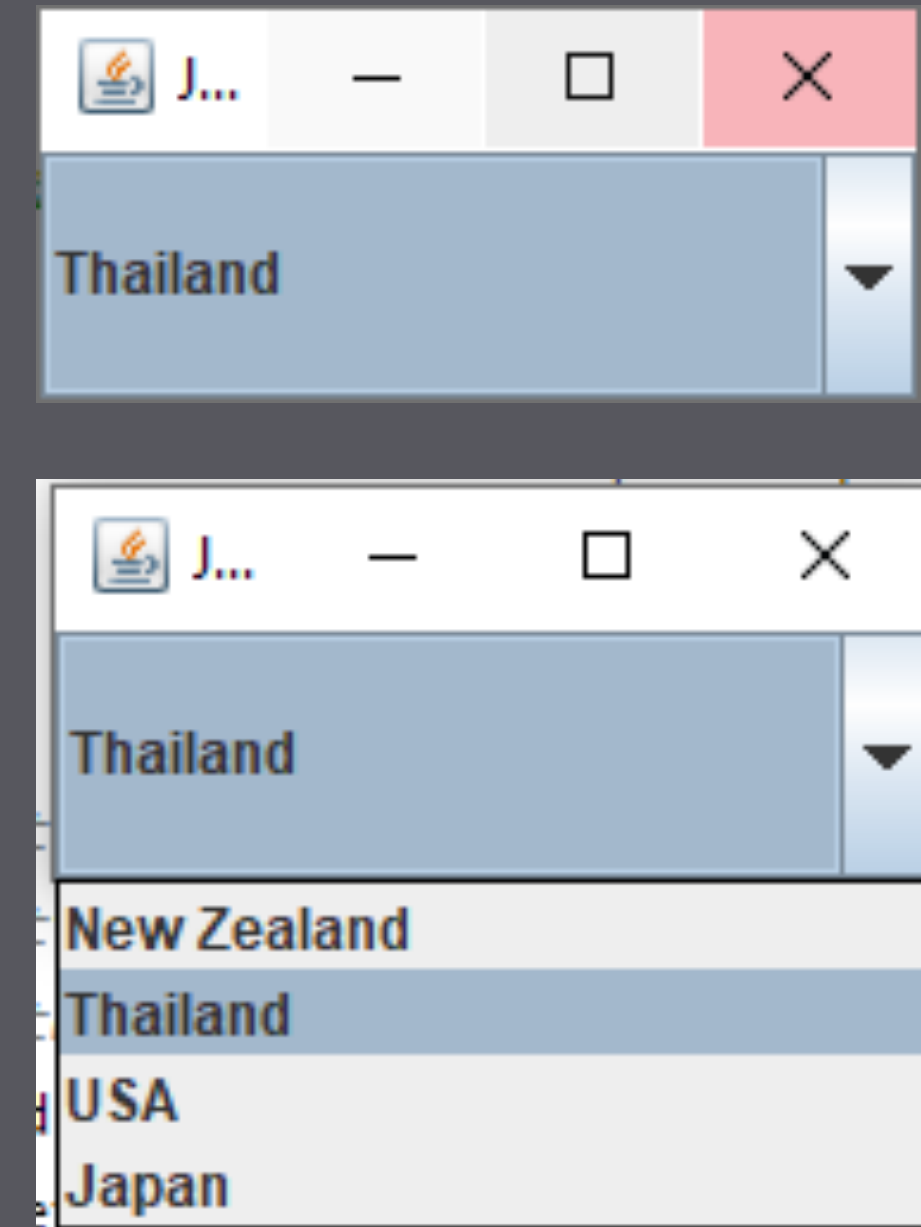
// เมธอดที่สำคัญในการจัดการกับ JComboBox มีดังนี้
// เป็นเมธอดที่ใช้สำหรับใส่รายการลงในรายการ
public void addItem(Object item)

// เป็นเมธอดที่ใช้ในการเลือกให้แสดงรายการที่ตำแหน่งหรือข้อความใดข้อความหนึ่งได้
public void setSelectedIndex(int pos)
public void setSelectedItem(Object item)

public Object getSelectedItem()
```

JComboBox

```
import javax.swing.*;
public class JComboBoxDemo {
    private JFrame fr;
    private JComboBox cb;
    public JComboBoxDemo() {
        fr = new JFrame("JRadioButton Demo");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        cb = new JComboBox();
        cb.addItem("New Zealand");
        cb.addItem("Thailand");
        cb.addItem("USA");
        cb.addItem("Japan");
        cb.setSelectedItem("Thailand");
        fr.add(cb);
        fr.pack();
        fr.setVisible(true);
    }
    public static void main(String args[]) {    new JComboBoxDemo(); }
}
```



JList

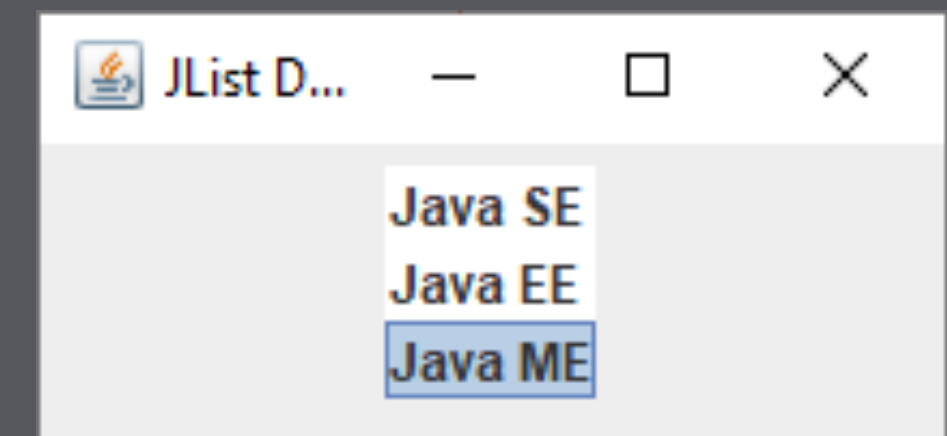
เป็นคลาสที่ใช้สร้างอ็อบเจกต์ที่เป็นส่วนประกอบกราฟิกเพื่อให้ผู้ใช้สามารถเลือกรายการคล้ายกับ
JComboBox แตกต่างกันตรงที่ JList จะแสดงรายการหลายรายการ

```
// Constructor ของคลาส JList ที่สำคัญมีดังนี้
public JList()
public JList([]Object objs)

//เมธอดที่สำคัญในการจัดการกับ JList มีดังนี้
public void addItem(Object item) // เป็นเมธอดที่ใช้สำหรับใส่รายการลงในรายการ
public void setSelectedIndex(int pos)
public void setSelectedIndices(int[] pos) ★
public Object getSelectedValue()
public Object[] getSelectedValues() ★
```


JList

```
import java.awt.*;
import javax.swing.*;
public class JListDemo {
    private JFrame fr;
    private JList list;
    private String[] choices = {"Java SE", "Java EE", "Java ME"};
    public JListDemo() {
        fr = new JFrame("JList Demo");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        list = new JList(choices);
        list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        fr.setLayout(new FlowLayout());
        fr.add(list);
        fr.pack();
        fr.setVisible(true);
    }
    public static void main(String args[]) {    new JListDemo();    }
}
```



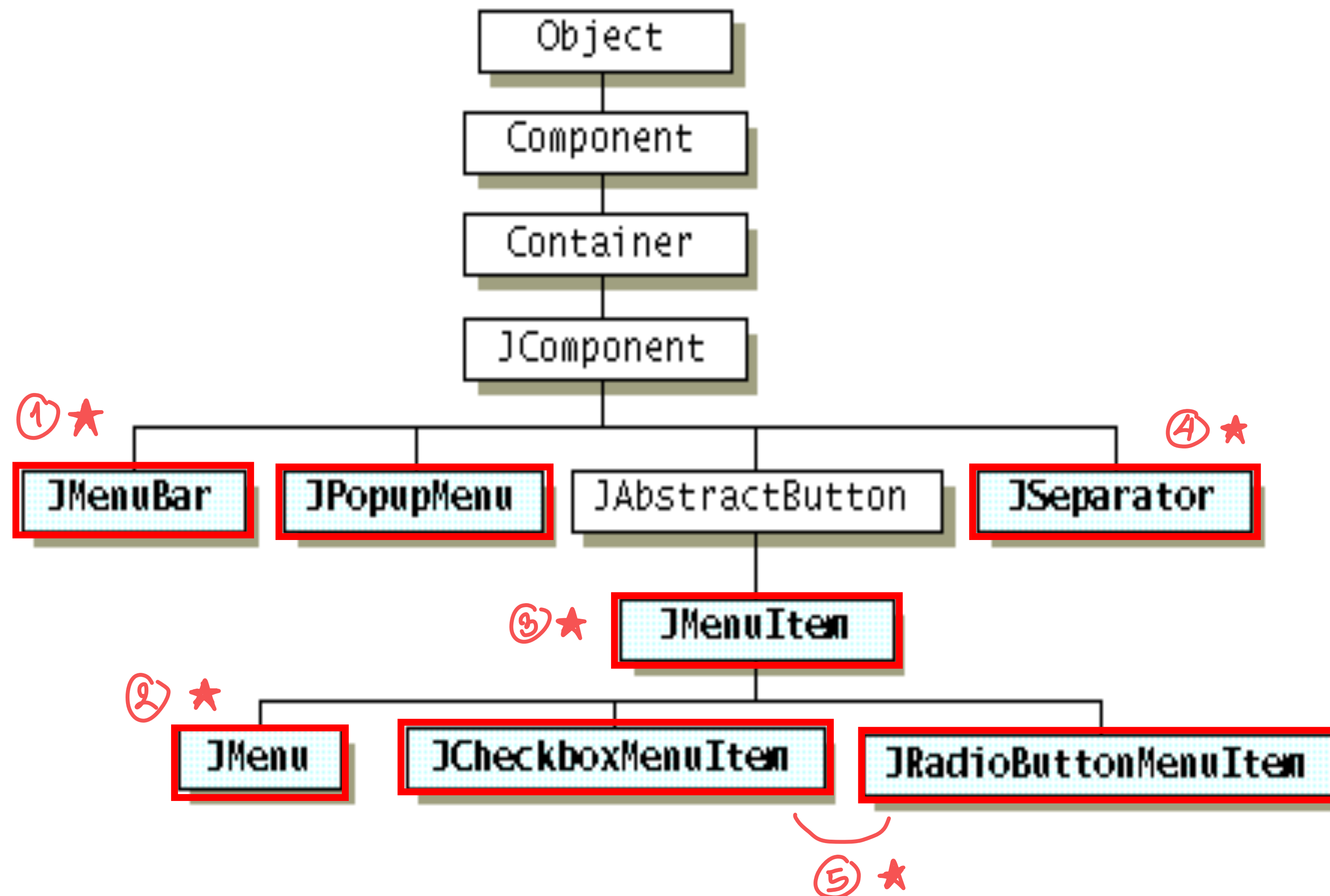
การสร้างเมนู



เมนูเป็นออปเจ็คของคลาสที่สืบทอดมาจากคลาสที่ชื่อ **JComponent** ซึ่งมีคลาสที่เกี่ยวข้องกับเมนู ที่สำคัญมีดังนี้

- **JMenuBar** เป็นคลาสที่ใช้ในการสร้างอ็อบเจกต์ที่เก็บกลุ่มของอ็อบเจกต์ของคลาส **JMenu** ซึ่งจะปรากฏเป็นแถบเมนูโดยอ็อบเจกต์ของคลาสนี้จะต้องมีอ็อบเจกต์ของคลาส **JFrame** ที่คู่กัน
- **JMenu** เป็นคลาสที่ใช้ในการสร้างอ็อบเจกต์ที่เก็บกลุ่มของอ็อบเจกต์ของคลาส **JMenuItem** และตัวแยกรายการ (**JSeparator**)
- **JMenuItem** เป็นคลาสที่ใช้ในการสร้างอ็อบเจกต์ที่เป็นรายการ *ไม่มีเมนูผสมลงไม่ลึก*
- **JCheckboxMenuItem** เป็นคลาสที่ใช้ในการสร้างอ็อบเจกต์ที่เป็นรายการโดยจะมีเครื่องหมายถูกที่จะแสดงขึ้นเมื่อรายการนี้ถูกเลือก
- **JRadioButtonMenuItem** เป็นคลาสที่ใช้ในการสร้างอ็อบเจกต์ที่เป็นรายการให้เลือกเพียงตัวเดียวแบบ
Radio Button

การสร้างเมนู



JMenuBar



เป็นคลาสที่จะแสดงเป็นแถบเมนูที่ปรากฏอยู่บน **JFrame** (ระบบปฏิบัติการบางประเภทอาจจะไม่แสดงแถบของเมนู หากไม่มีรายการในเมนู)

```
// Constructor ของคลาส JMenuBar ที่สำคัญมีดังนี้
public JMenuBar()

//เราสามารถที่จะใส่ชื่อของคลาส JMenuBar ลงใน Container ได้โดยใช้เมธอด
public void setJMenuBar() // JFrame เป็นคนเรียกใช้งาน
```

JMenu



เป็นคลาสที่ใช้ในการสร้างรายการที่จะแสดงอยู่ข้างในอ็อบเจกต์ของคลาส **JMenuBar**

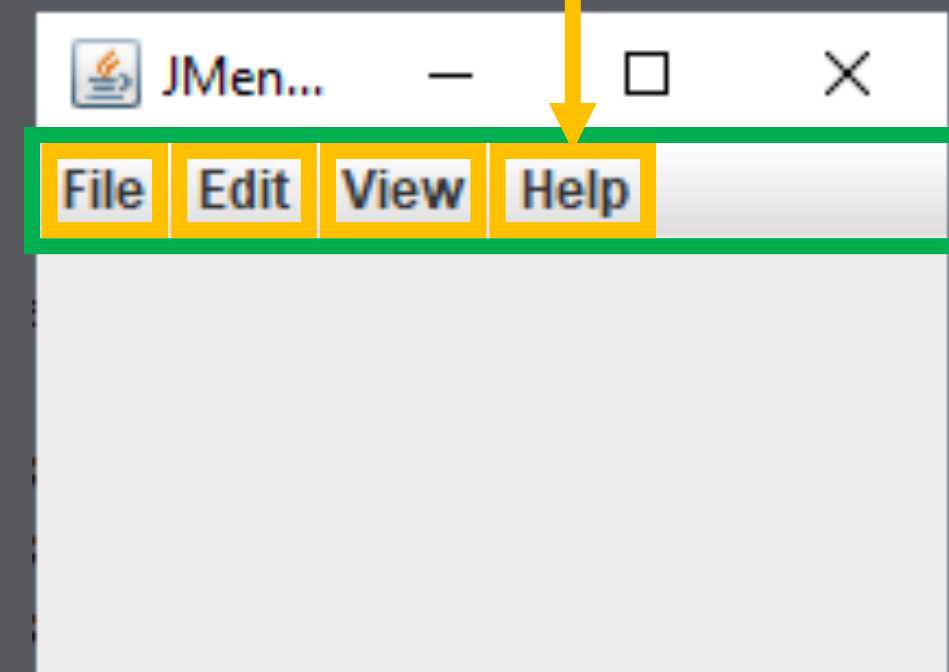
```
// Constructor ของคลาส JMenu ที่สำคัญมีดังนี้
public JMenu()
public JMenu(String label)

//เมธอดที่สำคัญในการจัดการกับ JMenu มีดังนี้
public void setLabel(String label) // กำหนดหรือเปลี่ยนชื่อรายการ
public void add (JMenu m)          // JMenuBar เป็นคนเรียกใช้งาน
```


ตัวอย่าง JMenuBar และ JMenu

```

import javax.swing.*;
public class JMenuDemo {
    private JFrame fr;
    private JMenuBar mb;
    private JMenu m1,m2,m3,m4;
    public JMenuDemo() {
        fr = new JFrame("JMenu Demo");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mb = new JMenuBar();
        m1 = new JMenu("File");
        m2 = new JMenu("Edit");
        m3 = new JMenu("View");
        m4 = new JMenu("Help");
        fr.setJMenuBar(mb); ★ Set Menu Bar for JFrame
        mb.add(m1);    mb.add(m2);    mb.add(m3);    mb.add(m4);
        fr.setSize(200,150);
        fr.setVisible(true);
    }
    public static void main(String args[]) {    new JMenuDemo();    }
}
  
```



JMenuItem



คือ รายการย่อยที่อยู่ในออปเจ็คของคลาส **JMenu**

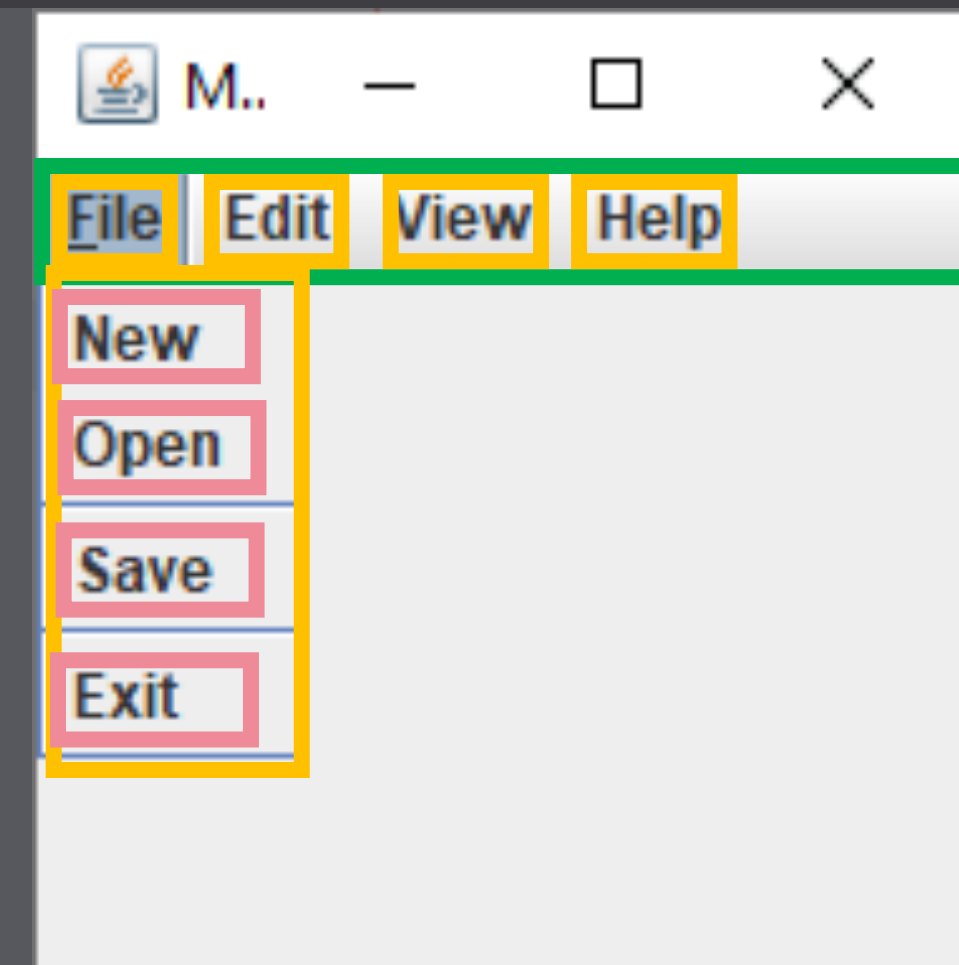
```
// Constructor ของคลาส JMenuItem ที่สำคัญมีดังนี้
public JMenuItem()
public JMenuItem(String label)
public JMenuItem(String label, int mnemonic)

//เมธอดที่สำคัญในการจัดการกับ JMenuItem มีดังนี้
public void setMnemonic(String label)
public void add (JMenuItem m)           // JMenu เป็นคนเรียกใช้งาน
```

ตัวอย่าง JMenuBar, JMenu, JMenuItem

```
import javax.swing.*;
public class JMenuItemDemo {
    private JFrame fr;
    private JMenuBar mb;
    private JMenu m1, m2, m3, m4;
    private JMenuItem mi1, mi2, mi3, mi4;

    public JMenuItemDemo() {
        fr = new JFrame("MenuItem Demo");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mb = new JMenuBar();
        m1 = new JMenu("File");
        m1.setMnemonic('F');
        m2 = new JMenu("Edit");
        m3 = new JMenu("View");
        m4 = new JMenu("Help");
        fr.setJMenuBar(mb);
        mb.add(m1); mb.add(m2);
        mb.add(m3); mb.add(m4);
    }
}
```



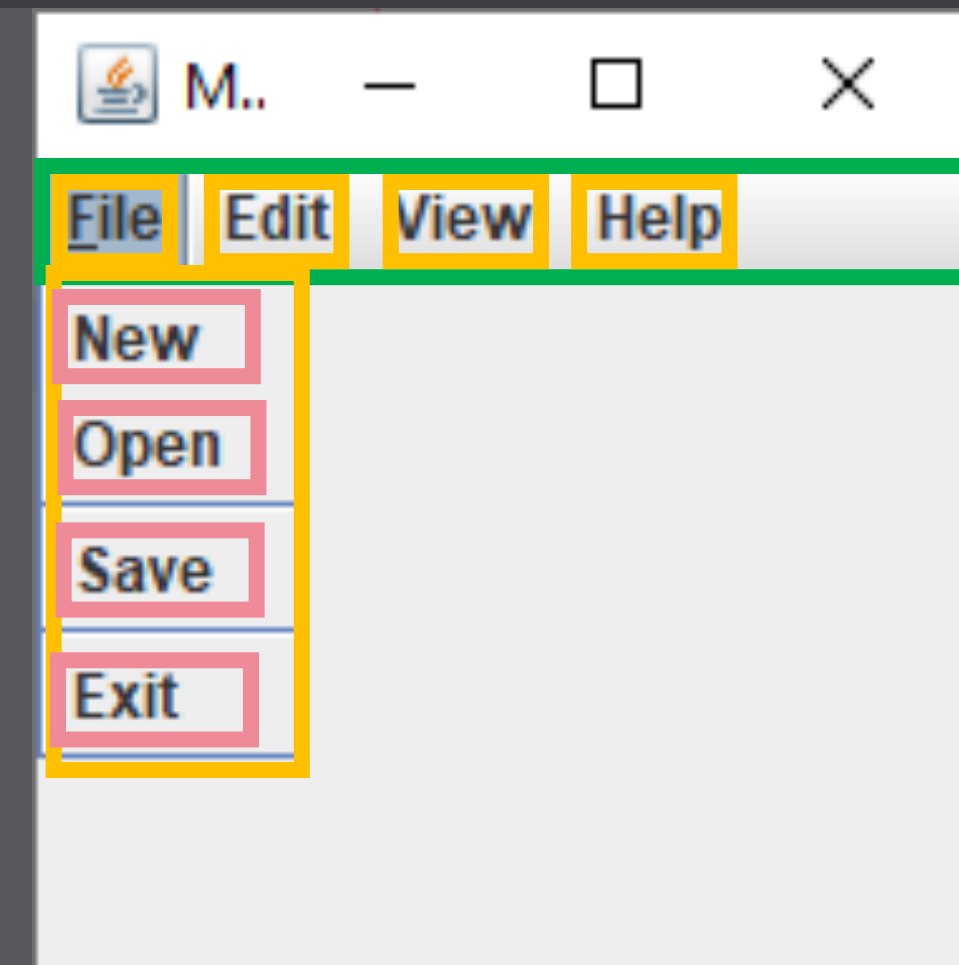
มีต่อ



ตัวอย่าง JMenuBar, JMenu, JMenuItem

```
mi1 = new JMenuItem("New");
mi2 = new JMenuItem("Open");
mi3 = new JMenuItem("Save");
mi4 = new JMenuItem("Exit");
m1.add(mi1);
m1.add(mi2);
m1.addSeparator();
m1.add(mi3);
m1.addSeparator();
m1.add(mi4);

fr.setSize(200, 200);
fr.setVisible(true);
}
public static void main(String args[]) {
    JMenuItemDemo mid = new JMenuItemDemo();
}
}
```



JCheckBoxMenuItem



คือ รายการเมนูที่มีเครื่องหมายระบุว่ารายการนี้ถูกเลือก

```
// Constructor ของคลาส JCheckBoxMenuItem ที่สำคัญมีดังนี้
public JCheckBoxMenuItem()
public JCheckBoxMenuItem(String label)
public JCheckBoxMenuItem(String label, boolean state)

//เราสามารถที่จะเปลี่ยนสถานะของออปเจ็คชนิด JCheckBoxMenuItem ได้โดยใช้เมธอด
public void setState(boolean b)
```


ตัวอย่าง JCheckBoxMenuItem

```
import javax.swing.*;

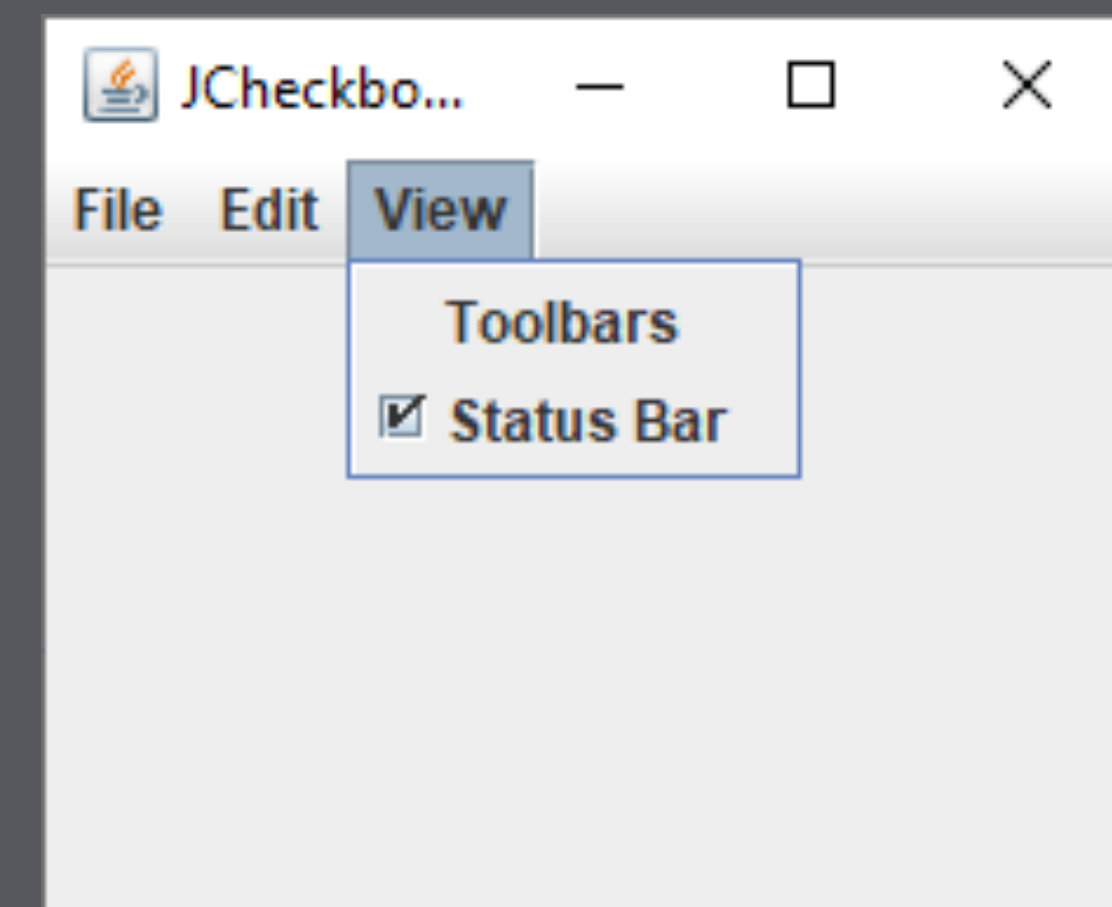
public class JCheckBoxMenuDemo {
    private JFrame fr;
    private JMenuBar mb;
    private JMenu m1,m2,m3;
    private JMenuItem mi;
    private JCheckBoxMenuItem cbm;
    public JCheckBoxMenuDemo() {
        fr = new JFrame("JCheckboxMenuItem Demo");
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mb = new JMenuBar();
        m1 = new JMenu("File");
        m2 = new JMenu("Edit");
        m3 = new JMenu("View");
        fr.setJMenuBar(mb);
        mb.add(m1);
        mb.add(m2);
        mb.add(m3);
    }
}
```

มีต่อ



ตัวอย่าง JCheckBoxMenuItem

```
mi = new JMenuItem("Toolbars");  
cbm = new JCheckBoxMenuItem("Status Bar", true);  
m3.add(mi);  
m3.add(cbm);  
fr.setSize(200,200);  
fr.setVisible(true);  
}  
public static void main(String args[]) {  
    JCheckBoxMenuDemo obj= new JCheckBoxMenuDemo();  
}  
}
```



การสร้างเมนูย่อย (เมนูซ้อนกัน)

- ▶ สร้างออปเจ็ทของคลาส JMenuBar แล้วใส่ลงไปในออปเจ็ทของคลาสประเภท Container เช่น JFrame
- ▶ สร้างออปเจ็ทของคลาส JMenu แล้วใส่ลงไปในออปเจ็ทของคลาส JMenuBar
- ▶ สร้างออปเจ็ทของคลาส **JMenu** สำหรับเป็นเมนูย่อย แล้วใส่ลงไปในออปเจ็ทของ **คลาส JMenu** ที่เป็นเมนูหลัก
- ▶ สร้างออปเจ็ทของคลาส JMenuItem แล้วใส่ลงไปในออปเจ็ทของคลาส JMenu ที่เป็นเมนูย่อย

การสร้างเมนูย่อย (เมนูซ้อนกัน)

```

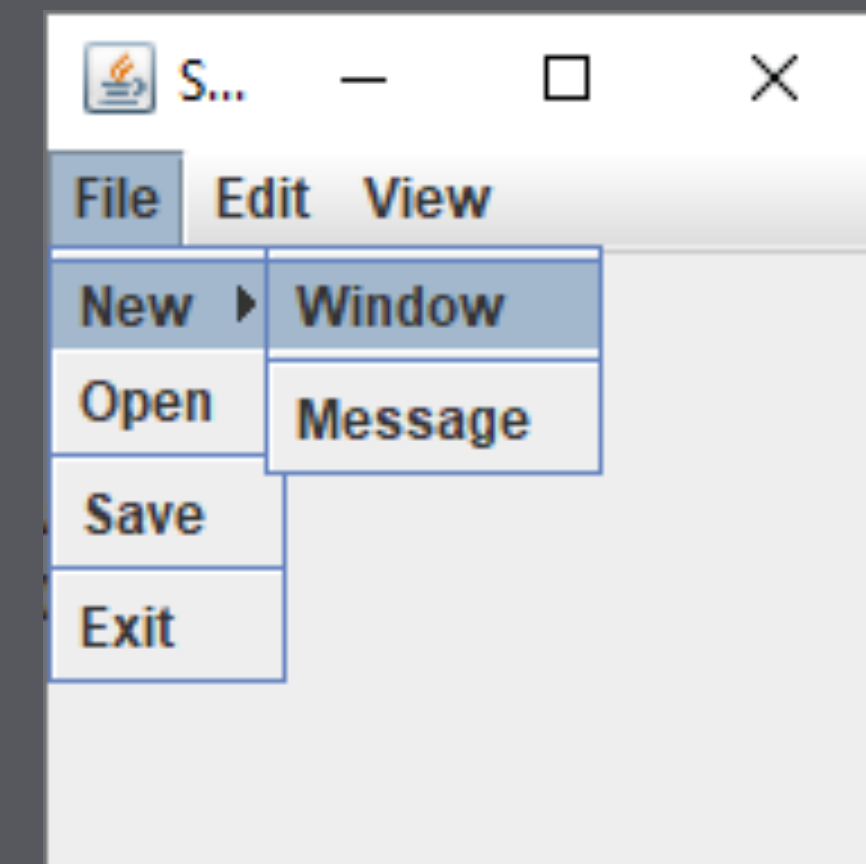
import javax.swing.*;
public class SubmenuDemo {
    private JFrame fr;
    private JMenuBar mb;
    private JMenu m1, m2, m3, ms1;
    private JMenuItem mi2, mi3, mi4, ms1, ms2;
    public SubmenuDemo() {
        ① [ fr = new JFrame("SubMenuItem Demo");
           fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ② mb = new JMenuBar();
        ③ [ m1 = new JMenu("File");
           m2 = new JMenu("Edit");
           m3 = new JMenu("View");
        ④ [ fr.setJMenuBar(mb);
           mb.add(m1);      mb.add(m2);      mb.add(m3);
           ms1 = new JMenu("New");
           mi2 = new JMenuItem("Open");
           mi3 = new JMenuItem("Save");
           mi4 = new JMenuItem("Exit");
        ⑤ ]
    }
  
```

มีต่อ



ตัวอย่าง เมนูซ้อนกัน

```
m1.add(ms1);  
m1.add(mi2);  
m1.addSeparator();  
m1.add(mi3);  
m1.addSeparator();  
m1.add(mi4);  
msi1 = new JMenuItem("Window");  
msi2 = new JMenuItem("Message");  
ms1.add(msi1);  
ms1.addSeparator();  
ms1.add(msi2);  
fr.setSize(200, 200);  
fr.setVisible(true);  
}  
  
public static void main(String args[]) {  
    SubmenuDemo obj = new SubmenuDemo();  
  
}
```



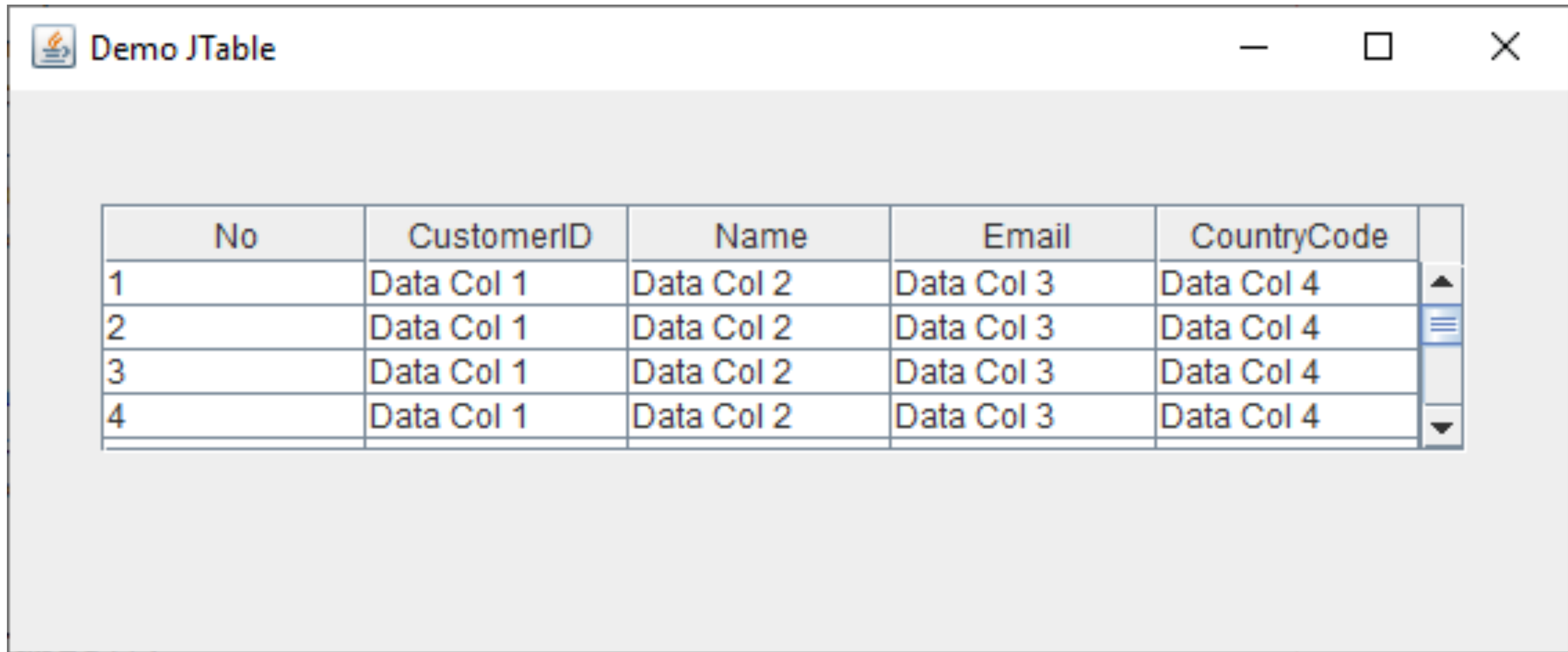
JTable



จัดอยู่ในกลุ่มของ Component ใช้แสดงข้อมูลในรูปแบบของ Table และ Grid ซึ่งประกอบด้วย Header (ส่วนหัว) , Column และ Cell โดยข้อมูลที่จะแสดงใน JTable จะอยู่ในรูปแบบของ Array ซึ่งจะเป็นแบบ Array 2 มิติ และการนำไปใช้งานกับ JTable สามารถนำไปใช้งานได้โดยตรง หรือในกรณีที่ข้อมูลมีความซับซ้อนสามารถนำข้อมูลเข้ากับ Model ของ Table ก่อนที่จะนำไปแสดงผลใน JTable ส่วนประกอบของ Table ซึ่งหลัก ๆ แล้วประกอบด้วย Header , Column และ Cell

นอกจากนี้ JTable ยังสามารถใช้งานกับข้อมูลที่ถูกอ่านมาจาก Database ซึ่งก็ใช้หลักการเช่นเดียวกับ Array คือจะต้องทำการ Query ข้อมูลให้อยู่ในรูปแบบของ ResultSet แล้วค่อยนำข้อมูลที่ได้มา Loop เพื่อแสดงผลใน JTable

JTable



The screenshot shows a Java Swing window titled "Demo JTable". Inside the window is a JTable with the following structure:

No	CustomerID	Name	Email	CountryCode	
1	Data Col 1	Data Col 2	Data Col 3	Data Col 4	▲
2	Data Col 1	Data Col 2	Data Col 3	Data Col 4	☰
3	Data Col 1	Data Col 2	Data Col 3	Data Col 4	
4	Data Col 1	Data Col 2	Data Col 3	Data Col 4	▼

JTable

```
import java.awt.*;
import javax.swing.*;
import javax.swing.table.*;
public class JTableDemo {
    private JFrame fr;
    private JScrollPane scrollPane;
    private JTable table;
    public JTableDemo() {
        fr = new JFrame();
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr.setBounds(100, 100, 580, 242);
        fr.setTitle("Demo JTable");
        fr.getContentPane().setLayout(null);
        // ScrollPane for Table
        scrollPane = new JScrollPane();
        scrollPane.setBounds(33, 41, 494, 90);
        fr.getContentPane().add(scrollPane);
        // Table
        table = new JTable();
        scrollPane.setViewportView(table);
    }
}
```

มีต่อ



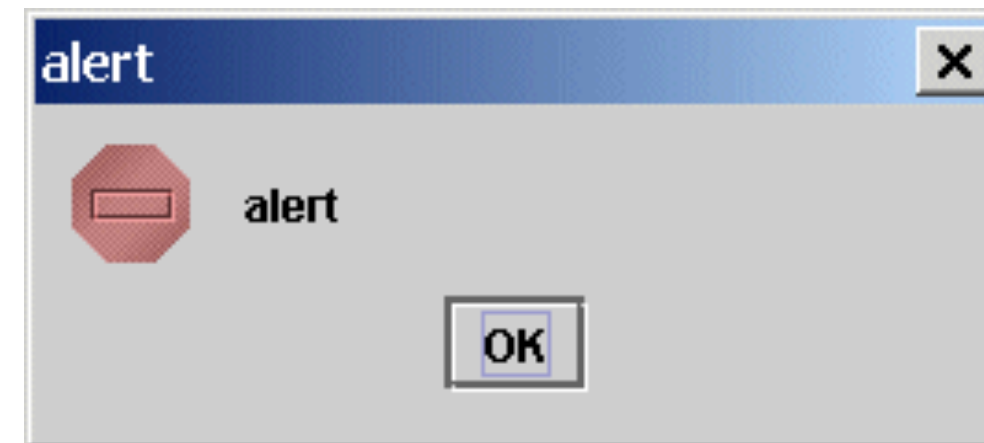
JTable

```
// Model for Table
DefaultTableModel model = (DefaultTableModel) table.getModel();
model.addColumn("No");
model.addColumn("CustomerID");
model.addColumn("Name");
model.addColumn("Email");
model.addColumn("CountryCode");
// Data Row
for(int i=0;i <= 10; i++) {
    model.addRow(new Object[0]);
    model.setValueAt(i+1, i, 0);
    model.setValueAt("Data Col 1", i, 1);
    model.setValueAt("Data Col 2", i, 2);
    model.setValueAt("Data Col 3", i, 3);
    model.setValueAt("Data Col 4", i, 4);
}
fr.setVisible(true);
}
public static void main(String[] args) { new JTableDemo(); }
}
```

JOptionPane

// **show an error dialog**

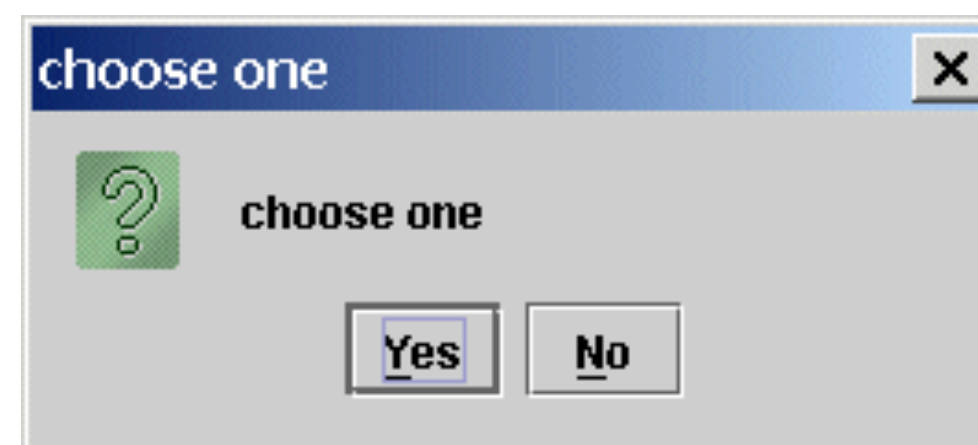
```
JOptionPane.showMessageDialog(null, "alert", "alert", JOptionPane.ERROR_MESSAGE);
```



- `JOptionPane.ERROR_MESSAGE`
- `JOptionPane.PLAIN_MESSAGE`
- `JOptionPane.QUESTION_MESSAGE`
- `JOptionPane.INFORMATION_MESSAGE`
- `JOptionPane.WARNING_MESSAGE`

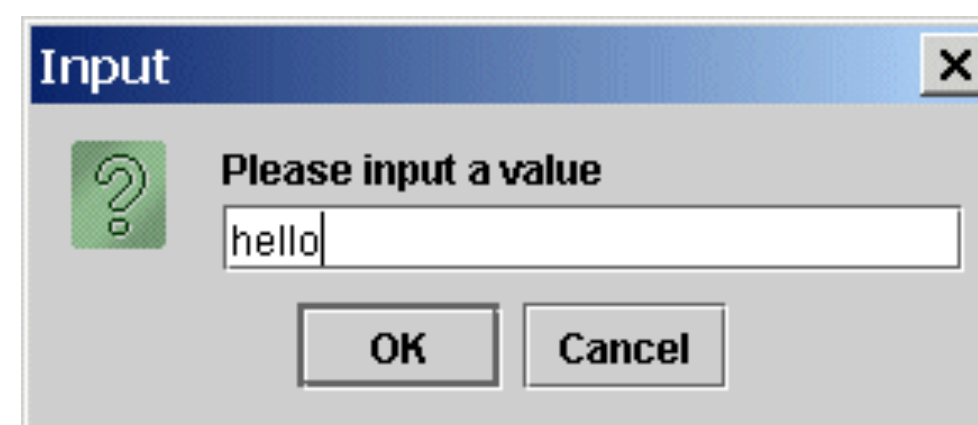
// **show Yes/No dialog**

```
int x = JOptionPane.showConfirmDialog(null, "choose one", "choose one", JOptionPane.QUESTION_MESSAGE);  
System.out.println("User clicked button " + x);
```



// **show input dialog**

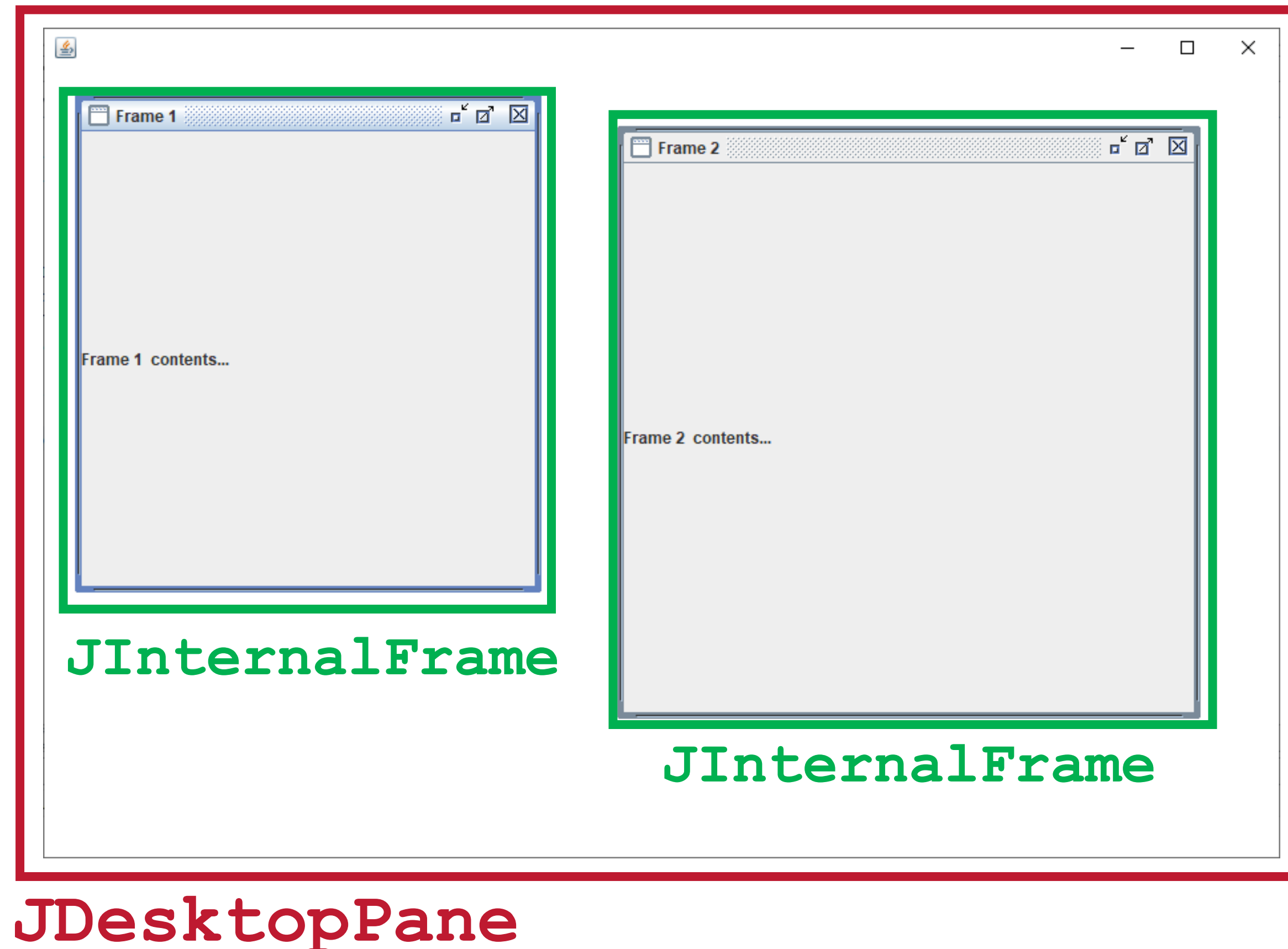
```
String inputValue = JOptionPane.showInputDialog(null, "Please input a value", JOptionPane.QUESTION_MESSAGE);  
System.out.println("User entered " + inputValue);
```



Multiple Document Interface (MDI)



คือ การที่โปรแกรมของเราสามารถมีหน้าต่างหลัก (**main window**) หนึ่งหน้าต่างและภายในนั้นมีหน้าต่างลูก (**child windows**) ย่อย ๆ อีกมากมาย ซึ่งในภาษาจาวานั้น หน้าต่างหลักจะอาศัยการสร้างออบเจ็คของคลาส **JDesktopPane** และหน้าต่างย่อย ๆ จะอาศัยการสร้างออบเจ็คของคลาส **JInternalFrame**



JDesktopPane



```
// Creates a new JDesktopPane.  
public JDesktopPane()
```

JInternalFrame



```
// Constructor ของคลาส JInternalFrame ที่สำคัญมีดังนี้
// Creates a non-resizable, non-closable, non-maximizable, non-iconifiable JInternalFrame with no title.
public JInternalFrame()

// Creates a non-resizable, non-closable, non-maximizable, non-iconifiable JInternalFrame with the
// specified title.
public JInternalFrame(String title)

// Creates a non-closable, non-maximizable, non-iconifiable JInternalFrame with the specified title and
// resizability.
public JInternalFrame(String title, boolean resizable)

// Creates a non-maximizable, non-iconifiable JInternalFrame with the specified title, resizability, and
// closability.
public JInternalFrame(String title, boolean resizable, boolean closable)

// Creates a non-iconifiable JInternalFrame with the specified title, resizability, closability, and
// maximizability.
public JInternalFrame(String title, boolean resizable, boolean closable, boolean maximizable)

// Creates a JInternalFrame with the specified title, resizability, closability, maximizability, and
// iconifiability.
public JInternalFrame(String title, boolean resizable, boolean closable, boolean maximizable, boolean iconifiable)
```

JDesktopPane และ JInternalFrame

```
import java.awt.*;
import javax.swing.*;
public class MDISample extends JFrame {
    private JDesktopPane desktopPane;
    private JInternalFrame frame1, frame2;
    public MDISample() {
        desktopPane = new JDesktopPane();
        frame1 = new JInternalFrame("Frame 1", true, true, true, true);
        frame2 = new JInternalFrame("Frame 2", true, true, true, true);

        frame1.getContentPane().add(new JLabel("Frame 1 contents..."));
        frame1.pack();
        frame1.setVisible(true);

        frame2.getContentPane().add(new JLabel("Frame 2 contents..."));
        frame2.pack();
        frame2.setVisible(true);

        int x2 = frame1.getX() + frame1.getWidth() + 10;
        int y2 = frame1.getY();
        frame2.setLocation(x2, y2);
    }
}
```

มีต่อ



JDesktopPane และ JInternalFrame

```
desktopPane.add(frame1);
desktopPane.add(frame2);

this.add(desktopPane, BorderLayout.CENTER);
this.setMinimumSize(new Dimension(300, 300));
this.pack();
this.setVisible(true);
this.setExtendedState(this.MAXIMIZED_BOTH);
}
public static void main(String[] args) {
    MDISample frame = new MDISample();
}
}
```


Java Look And Feel

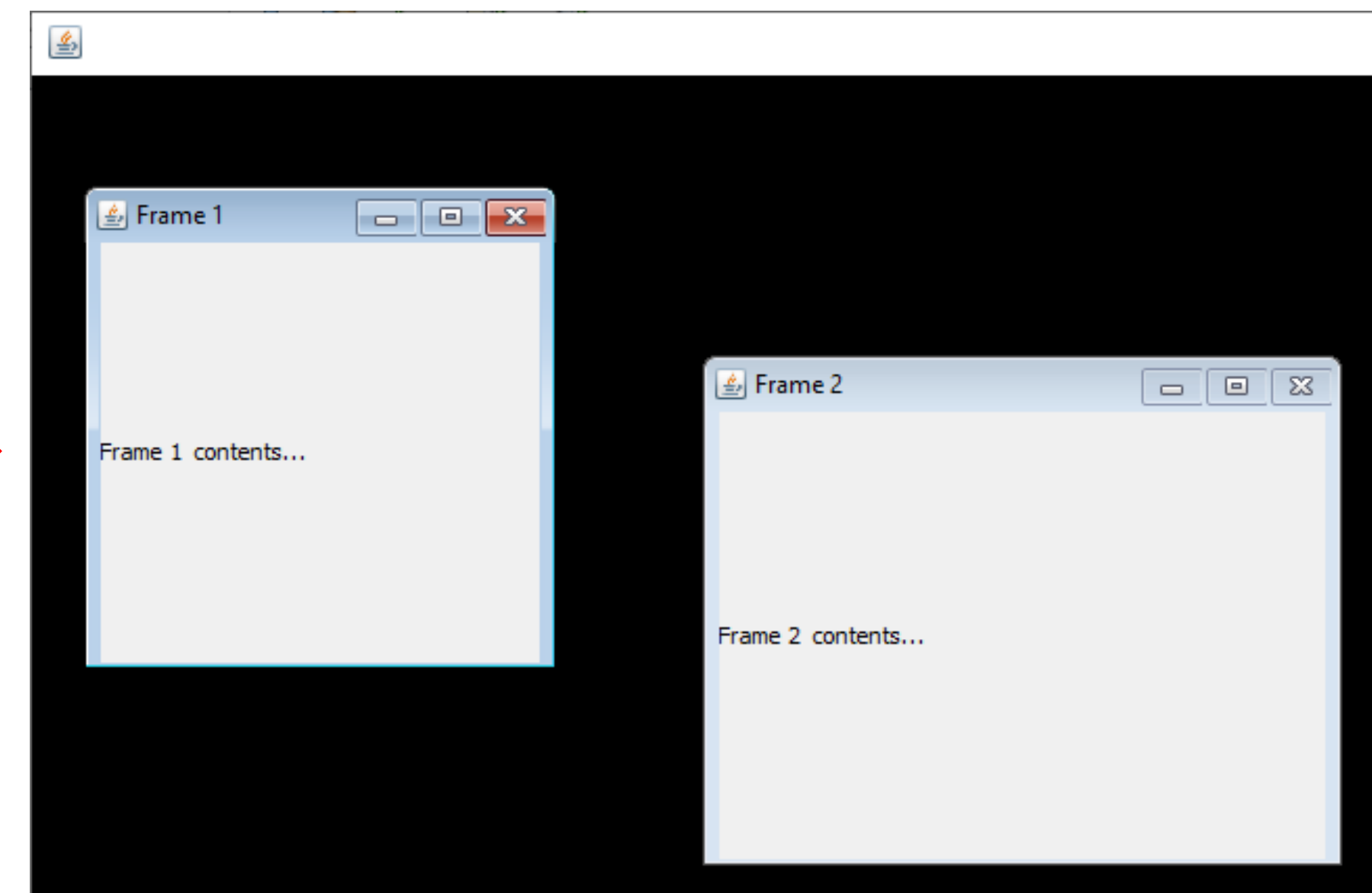
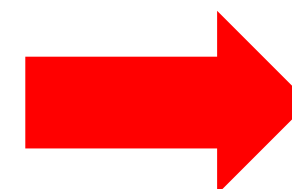
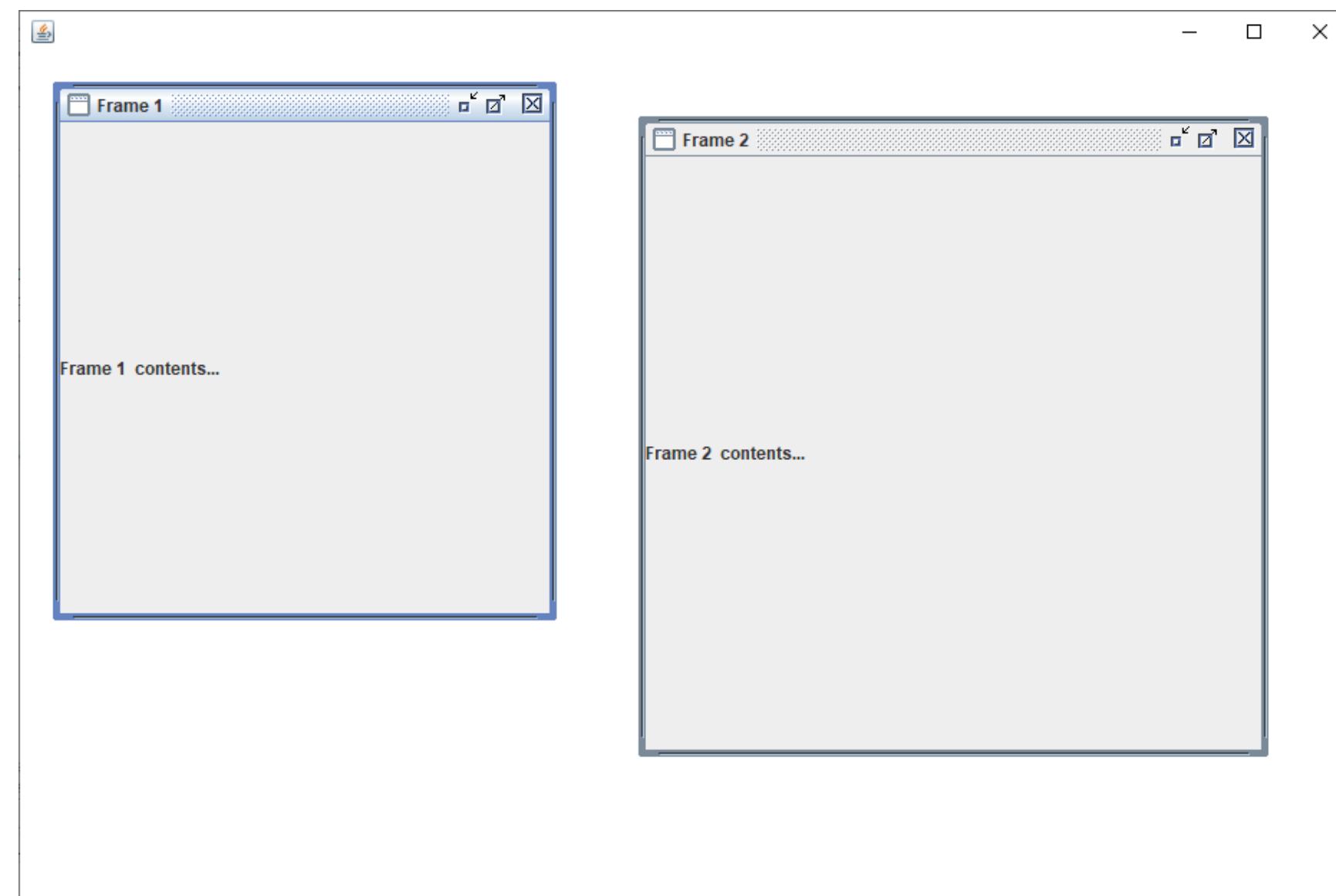


คือ การปรับแต่งหน้าต่างของ GUI ใน java โดยปกติแล้ว GUI ของมัน java จะเรียกใช้มาจากตัว **Library** ของ java ซึ่งจะไม่เหมือนกับหน้าต่างโปรแกรมที่ประมวลผลบนระบบปฏิบัติการนั้น ๆ

```
try {  
    //เรียกใช้จาก Library ของ java โดย Default  
    //UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());  
  
    //เรียกใช้จาก Library ของ java เช่นกัน  
    //UIManager.setLookAndFeel("com.sun.java.swing.plaf.motif.MotifLookAndFeel");  
  
    //เรียกใช้จากรูปแบบของระบบปฏิบัติการ  
    UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

Java Look And Feel

```
public static void main(String[] args) {  
    try {  
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    SwingUtilities.invokeLater(() -> { MDISample frame = new MDISample(); });  
}
```



SwingUtilities.invokeLater()



การโปรแกรมโดยอาศัยชุดคำสั่งของ Swing นั้นต้องเข้าใจว่าหาก component ใด ๆ ที่ถูก paint ไปแล้ว (หรือ หลังจากการเรียก pack(), setVisible(true)) component นั้นหากต้องการอัปเดตหน้าตาการแสดงผลในภายหลังจะต้องทำผ่าน

(1) **Thread AWT-EventQueue-0** โดยที่ Thread AWT-EventQueue-0 คือ Thread ที่ทำหน้าที่ประมวลผล Event Handling ของโปรแกรมที่เราพัฒนาขึ้น รวมถึงจัดการอัปเดตการแสดงผลของ component ต่าง ๆ ในหน้า GUI

เนื่องจากบางทีการอัปเดต component ไม่ได้เกิดขึ้นจาก event ที่ผู้ใช้กระทำอย่างเดียวนะ อ่าทึเช่น การอัปเดตการแสดงผลตามเวลาที่กำหนด ดังนั้น

(2) การทำผ่าน **SwingUtilities.invokeLater()** คือ Runnable Thread ที่รับมาจะเอาไปเข้าคิว EventQueue ไว้และเมื่อถึงเวลาของ Thread EventQueue ก็จะประมวลผลเองโดยอัตโนมัติ

คุณลักษณะของคลาส Component

ส่วนประกอบกราฟิกต่างๆจะมีคุณลักษณะอื่นอาทิเช่น รูปแบบของฟอนต์ สีของพื้นหลังหรือสีของพื้นหน้า (**Foreground**) เราสามารถที่จะกำหนดคุณลักษณะของส่วนประกอบกราฟิกได้ โดยปกติส่วนประกอบกราฟิกจะใช้คุณลักษณะแบบเดียวกับออปเจ็คประเภท **Container** ที่บรรจุอยู่เว้นแต่จะมีการกำหนดคุณลักษณะเฉพาะของส่วนประกอบกราฟิกนั้นๆ

```
// เมธอดที่ใช้ในการกำหนดคุณลักษณะของส่วนประกอบกราฟิก จะอยู่ในคลาส Component  
โดยมีเมธอดที่สำคัญคือ  
setFont(Font f)  
setForeground(Color c)  
setBackground(Color c)
```

คลาส Font



เราสามารถสร้างออบเจ็คของคลาส **Font** เพื่อใช้ในการกำหนดฟอนต์ได้

```
// Constructor ของคลาส Font ที่สำคัญมีดังนี้
public Font(String name,int style,int size)
    name คือ ชื่อฟอนต์
    style คือ รูปแบบของฟอนต์ เช่น Font.PLAIN, Font.BOLD, Font.ITALIC
    size คือ ขนาดของฟอนต์

// เราสามารถกำหนดฟอนต์ให้กับออบเจ็คของคลาส Component โดยใช้
public void setFont(Font f)
```


คลาส Color



เราสามารถสร้างออบเจ็คของคลาส **Color** เพื่อใช้ในการกำหนดสีได้

```
// Constructor ของคลาส Color ที่สำคัญมีดังนี้  
public Color(int r,int g,int b)
```

โดยที่ **r,g,b** คือ ค่าความเข้มของแสงสีแดง เขียว และน้ำเงิน ตามลำดับ
ตัวอย่างเช่น

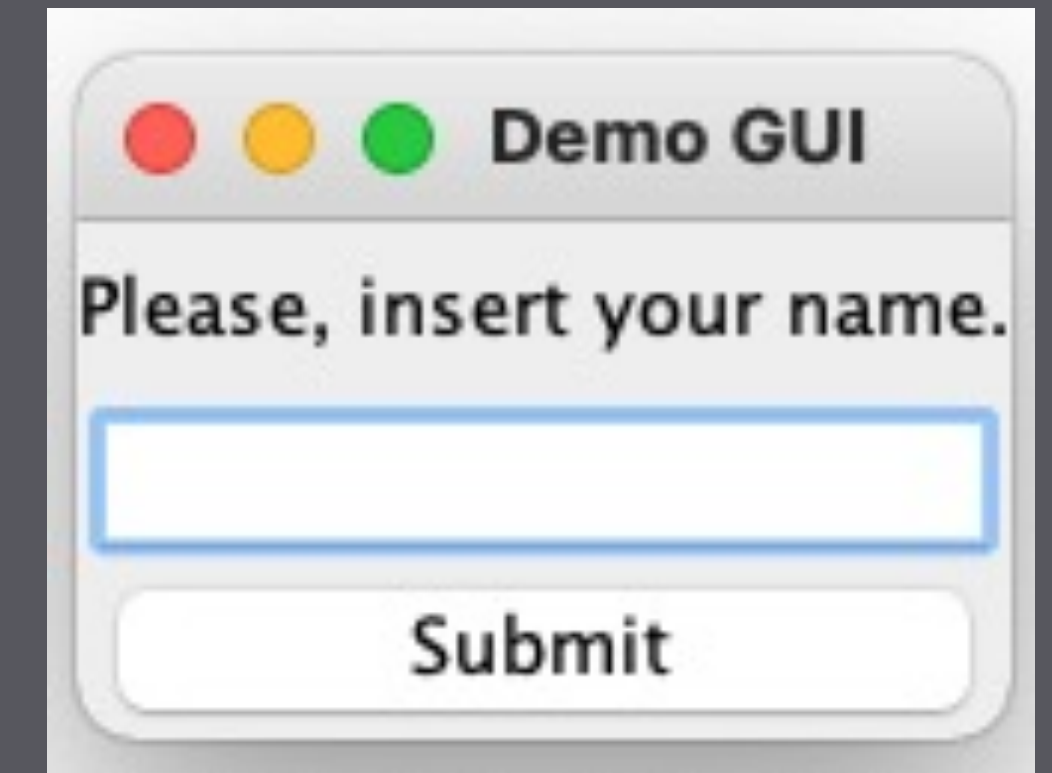
```
yellow = new Color(255,255,0) ;  
black = new Color(0,0,0) ;
```

สรุปการสร้างส่วนติดต่อผู้ใช้งานแบบที่ 1

```
import java.awt.*;
import javax.swing.*;
public class G1 {
    public static void main(String[] args) {
        JFrame fr = new JFrame();
        JPanel p = new JPanel();
        JLabel lbl = new JLabel("Please, insert your name.");
        JTextField txt = new JTextField();
        JButton btn = new JButton("Submit");

        p.setLayout(new GridLayout(3,1));
        p.add(lbl);
        p.add(txt);
        p.add(btn);

        fr.add(p);
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr.pack();
        fr.setVisible(true);
        //new G1();
    }
}
```



สรุปการสร้างส่วนติดต่อผู้ใช้งานแบบที่ 2

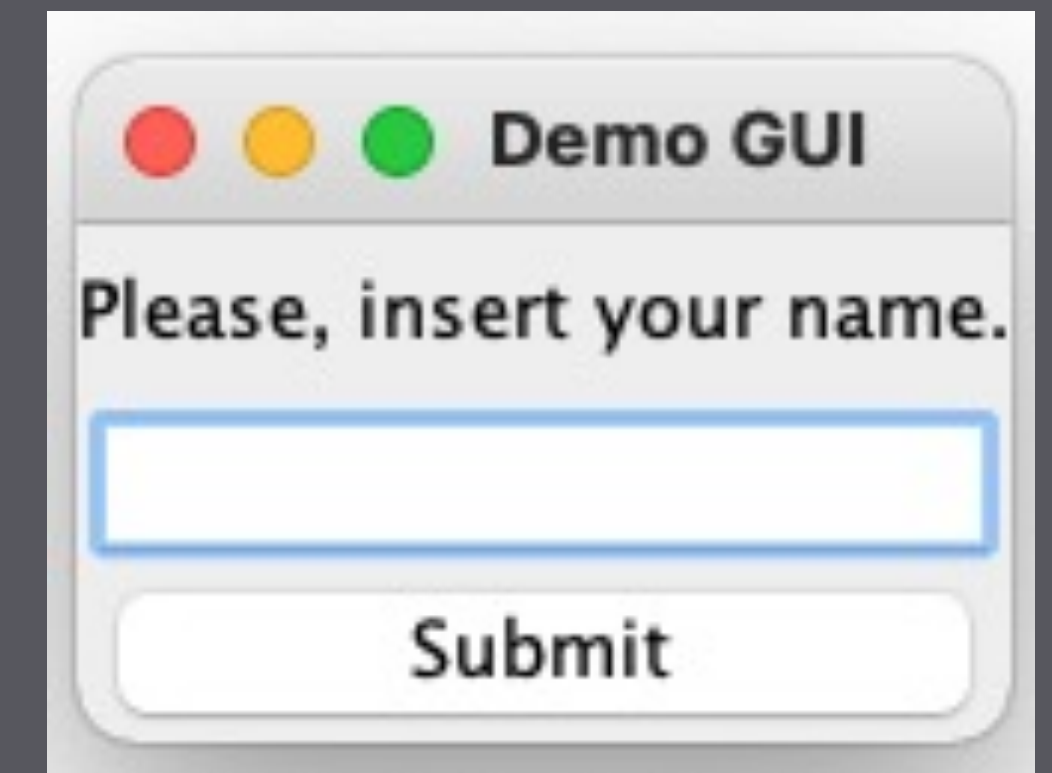
```
import java.awt.*;
import javax.swing.*;
public class G2 extends JFrame {
    private JPanel p;
    private JLabel lbl;
    private JTextField txt;
    private JButton btn;

    public G2() {
        p = new JPanel();
        lbl = new JLabel("Please, insert your name.");
        txt = new JTextField();
        btn = new JButton("Submit");

        p.setLayout(new GridLayout(3,1));
        p.add(lbl);          p.add(txt);          p.add(btn);
        this.add(p);

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

    public static void main(String[] args) {        new G2();    }
```



สรุปการสร้างส่วนติดต่อผู้ใช้งานแบบที่ 3

```
import java.awt.*;
import javax.swing.*;
public class G3 {
    private JFrame fr;
    private JPanel p;
    private JLabel lbl;
    private JTextField txt;
    private JButton btn;

    public G3() {
        fr = new JFrame("Demo GUI");
        p = new JPanel();
        lbl = new JLabel("Please, insert your name.");
        txt = new JTextField();
        btn = new JButton("Submit");

        p.setLayout(new GridLayout(3,1));
        p.add(lbl);      p.add(txt);      p.add(btn);      fr.add(p);

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

    public static void main(String[] args) {    new G3();    }
}
```

