Java Foundation Class (JFC)

Vittayasak Rujivorakul

Lecture 12

Java Foundation Class (JFC)

Vittayasak Rujivorakul

Lecture 12

JFC และ Swing คืออะไร

JFC คือชื่อย่อสำหรับ **JAVA Foundation Class** ซึ่งรวบรวมกลุ่มของความสามารถที่จะช่วยให้เราสร้าง GUI ได้ง่าย

JFC เกิดครั้งแรกเมื่อปี 1997 โดยผู้ พัฒนา JAVA One ซึ่งในตอนแรกๆ Swing เป็น codename ของโครงการที่ พัฒนาองค์ประกอบใหม่ๆ และ ตระกูลของ API โดยจะจัดอยู่ใน package ที่ขึ้นต้นด้วย **“javax.swing”**

สิ่งที่ JFC สนับสนุน

- **Swing Component :** รวบรวมตั้งแต่ buttons จนถึง split pane และ table
- **Pluggable Look & Feel :** สามารถเลือกใช้ Java Look & Feel หรือ Windows Look & Feel สำหรับจัดการการแสดงผลให้เหมือนกับสิ่งที่ user คุ้นเคย
- **Accessibility API :** สนับสนุนเทคโนโลยีการ**อ่านข้อมูลจากจอภาพ** และการแสดง**ตัวนูน** สำหรับคนตาบอด
- **Java 2D API :** รวมความสามารถในการจัดการกับภาพ 2D ทั้ง ข้อความและรูปภาพ ในการทำงานกับ Application หรือ Applet
- **Drag and Drop Support:** รองรับการทำงานที่ทำให้เขียนติดต่อกับผู้ใช้ง่านขึ้น

Swing Package

- javax.accessibility
- **javax.swing**
- javax.swing.border
- javax.swing.colorchooser
- **javax.swing.event**
- javax.swing.filechooser
- javax.swing.plaf
- javax.swing.plaf.basic
- javax.swing.plaf.metal
- javax.swing.plaf.multi
- javax.swing.table
- javax.swing.text
- javax.swing.text.html
- javax.swing.tree
- javax.swing.undo

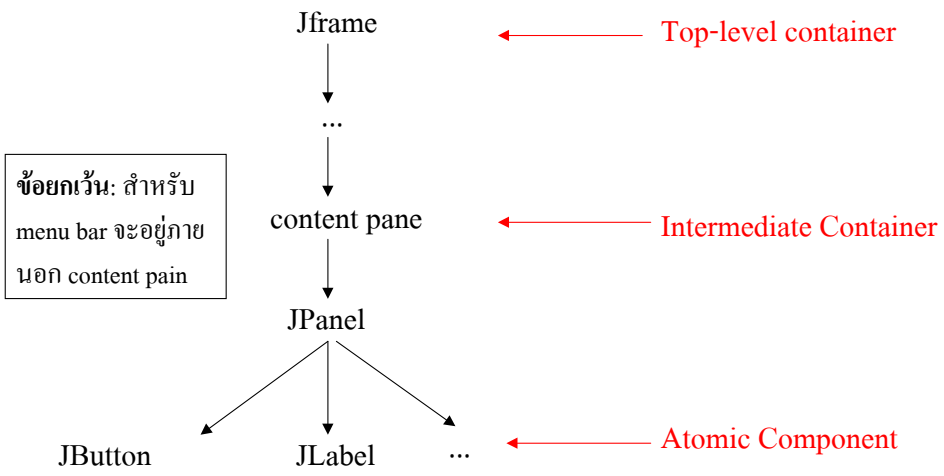
Swing vs AWT Component

- **package** ของ swing จะอยู่ใน “javax.swing” และชื่อขององค์ประกอบต่างๆ จะขึ้นต้นด้วย “J” เช่น “**J**Button” ส่วนของ AWT จะอยู่ใน package ชื่อ “java.awt”
- Swing จะไม่สนับสนุน **native code** ซึ่งทำให้สามารถทำงานได้มากกว่า AWT และไม่ขึ้นกับ platform อีกทั้งมีความสามารถสูงกว่า AWT ดังนี้
  - button และ label ของ Swing สามารถแสดงรูปภาพแทนได้
  - ง่ายในการเพิ่มหรือเปลี่ยนขอบเขตหรือกรอบรอบๆ Swing component
  - ง่ายที่จะเปลี่ยนคุณสมบัติหรือรูปแบบการแสดง Swing Component
  - Swing Component ไม่ได้มีเพียงรูปสี่เหลี่ยมเท่านั้น เช่นปุ่มสามารถเป็นวงกลมได้

องค์ประกอบหลักของ Swing Component

- **Top-level Container:** เป็นองค์ประกอบหลักที่มีพื้นที่สำหรับให้ Swing Component อื่นๆ วาดตัวเอง โดยมีอยู่ 3 ชนิดคือ
  - frame (Jframe)
  - dialog (Jdialog)
  - applet (Japplet)
- **Intermediate Container:** เป็นองค์ประกอบที่ใช้สำหรับจัดวาง components ในรูปแบบต่างๆ กันหรือใช้แสดงหน้าที่พิเศษทางด้าน UI
- **Atomic Component:** เป็นองค์ประกอบที่เกิดขึ้นโดยมีความสามารถในการทำงานอยู่ในตัวเองและ ไม่สามารถที่จะให้พื้นที่สำหรับวาง Component อื่นได้

โครงสร้าง Swing Component



ข้อยกเว้น: สำหรับ menu bar จะอยู่ภายนอก content pain

การวาง component บน container

- เราจะใช้ **method add** ในการวาง component ซึ่งจะมีอย่างน้อย 1 argument คือ component ที่จะ add และบางครั้งอาจระบุข้อมูลเกี่ยวกับ layout หรือตำแหน่งที่ต้องการจะวาง

```
frame = new JFrame(…);
pane = new JPanel(…);
frame.getContentPane( ).add(pane,BorderLayout.(CENTER));
```

Layout Management

- Layout management คือ กระบวนการกำ**หนดขนาด**และ **ตำแหน่ง**ของ component โดยปกติทุกๆ container จะมี layout manager ซึ่งเป็น object ที่จัดการ layout manager สำหรับ components ภายใน container นั้นๆ
- **Java มี layout manager** สำหรับการใช้งานทั่วๆ ไปอยู่ 5 แบบคือ
  - Border Layout
  - Box Layout
  - Flow Layout
  - Grid Bag Layout
  - Grid Layout

นอกจากนั้นยังมี special-purpose layout manager เช่น **Card Layout** ช่วยในการจัดการนำ layout manager อื่นๆ หลายๆ แบบมาเชื่อมต่อใช้งานร่วมกัน

การติดตั้ง Layout Manager

- เราสามารถเปลี่ยน layout manager ซึ่ง container ใช้งานอยู่ ได้โดยการใช้ method setLayout เช่น

```
JPanel  pane = new JPanel( );
pain.setLayout(new BorderLayout( ));
```

- เราสามารถกำหนดให้ container ไม่มี layout ได้โดยการ set ค่า leyout ให้เป็น null ซึ่งเรียกว่า **absolute positioning** ทำให้เราจะต้องกำหนดขนาดและตำแหน่งของ component ทุกตัวใน container

* **ข้อแนะนำ:** ให้ใช้ layout manager เพราะเมื่อมีการปรับขนาด container ตำแหน่งและขนาด absolute จะมีการเปลี่ยนแปลงที่ไม่เหมาะสม

การกำหนดขนาดของ Component

- เราสามารถกำหนดขนาดที่เหมาะสมของ component อย่างคร่าวๆ โดยการกำหนด parameter minimum, preferred และ maximum ซึ่งสามารถอ้างถึงขนาดของ component โดยการใช้ set method คือ **setMinimumSize, setPreferedSize และ setMaximumSize**
- นอกจากนั้นเรายังสามารถกำหนดตำแหน่งอย่างคร่าวๆ ได้โดยการ ใช้ set method ในการจัดการตำแหน่งของแนวต่างๆ คือ **setAlignmentX** และ **setAlignmentY** ซึ่งใช้ได้กับ BoxLayout เท่านั้น

การกำหนดช่องว่างระหว่าง Component

มีปัจจัย 3 อย่าง ที่สามารถควบคุมช่องว่างระหว่าง component ได้คือ

- **layout manager :** บาง layout manager จะใส่ช่องว่างระหว่าง component โดยอัตโนมัติ และมีบาง layout manager ที่จะอนุญาตให้เรากำหนดช่องว่าง
- **invisible components :** เราสามารถสร้าง lightweight component ที่ไม่มีการวาดแต่ทำให้เกิดช่องว่างใน GUI ปกติเราใช้ invisible component ใน container ที่ควบคุมโดย Boxlayout
- **Empty Border :** เราสามารถกำหนดช่องว่างระหว่าง component โดยใช้ empty border ซึ่ง Empty border จะเหมาะสำหรับการใช้กับ component ที่ไม่มีขอบเขตที่ชัดเจน เช่น panel และ label

การทำงานของ Layout Management

- หลังจาก GUI ถูกสร้าง pack method จะถูกเรียกใช้เป็น JFrame โดยกำหนด preferred size
- layout manager ของ frame จะเพิ่มขนาดของ frame โดยใช้ ผลบวกของ preferred size ของ content pane ของ frame รวมกับขนาดของ menu bar ของ frame
- layout manager ของ content pane จะคำนวณจาก preferred size ของแต่ละ component และเลือกใช้ preferred size ที่ใหญ่ที่สุดเป็นหลัก
- หาก user กำหนด preferred size ไว้ มันจะใช้ขนาดที่ user เป็นหลัก หากไม่ได้กำหนด ก็จะใช้การกำหนดโดย Look and Feel

การ Handle Event

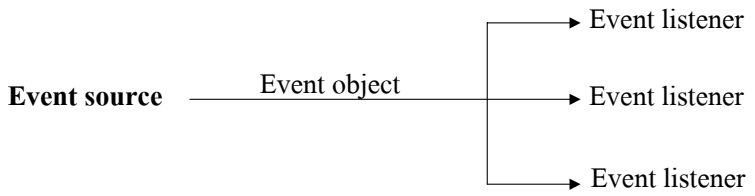
ทุกครั้งที่ user พิมพ์ตัวหนังสือ, กดปุ่มบน mouse จะเกิด event ขึ้น บาง object จะถูกเรียก event ซึ่ง object เหล่านี้จะต้อง **implement interface** ที่เหมาะสมและต้องถูกบันทึกว่าเป็น event listener บน event source ซึ่ง swing component สามารถที่จะสร้าง event ชนิดต่างๆ ได้อย่างมากมาย

Listener Type

Act that results in the event	Listener Type
User clicks a button, press Return while typing in a text field, or chooses a menu item	ActionListener
User close a frame (main window)	WindowListener
User presses a mouse button while cursor's over A component	MouseListener
User move the mouse over a component	MouseMotionListener
Component becomes visible	ComponentListener
Component get the keyboard focus	FocusListener
Table of List selection changes	ListSelectionListener

event object

- แต่ละ event จะถูกแทนโดย object ซึ่งให้ข้อมูลเกี่ยวกับ event และบ่งบอกถึงที่มาของ event (event source) ซึ่ง event source ก็คือ ชนิดของ component และ event source ยังสามารถที่จะมี listener ได้หลายๆ ตัวที่บันทึกไว้บนตัวมัน เช่นเดียวกับ listener ก็สามารถที่จะบันทึกไปยังหลายๆ event source ได้



ขั้นตอนการจับ event

- class ใดที่เราต้องการกำหนดให้เป็นตัวตรวจจับ event code ใน class นั้นๆ จะต้อง implements listener interface หรือ extends class ที่ implements listener interface นั้นๆ เช่น

  public class MyClass **implement ActionListener** { …. }

- code ที่เป็นตัวตรวจจับ event ต้องบันทึกเป็น listener ของ component อย่างน้อย 1 component

  SomeComponent.**addActionListene**r(instanceOfMyClass);

- ทำการ implement method ใน listener interface ดังตัวอย่าง

```
public void actionPerformed(ActionEvent e){
        …// code that reacts to the action ….
}
```


button event

- เมื่อผู้ใช้กด button (JButton) บนจอภาพ (หรือ keyboard) โปรแกรมจะต้องมี object ซึ่ง implement ActionListener interface และโปรแกรมต้องบันทึก object นี้เป็น action listener ของ button ที่เป็น event source โดยใช้ **addActionListener method** เมื่อ user กด button บนจอภาพ button จะส่ง action event ออกมา ซึ่งจะถูกส่งไปเป็นค่าให้กับ actionPerformed method ตัวแปรที่ส่งให้กับ method นี้คือ **ActionEvent object** ซึ่งมีข้อมูลเกี่ยวกับ event และ source

Thread และ Event Handling

- **Event Handling code จะทำงานภายใน thread เดียวคือ event-dispatching thread เพื่อให้แน่ใจว่าแต่ละเหตุการที่ตรวจจับ จะทำงานให้เสร็จก่อนเหตุการต่อไปจะเกิดขึ้น เช่น** actionPerformed method กำลังทำงานอยู่ภายใน event-dispatching thread เนื่องจาก code ของการวาดก็ทำงานอยู่ภายใน event-dispatching thread นั่นหมายความว่า ในขณะที่ actionPerformed method กำลังทำงาน GUI จะถูกแช่แข็ง นั่นคือจะไม่สามารถ repaint หรือตอบสนองต่อการกด mouse เป็นต้น