

圖形使用者介面

- 什麼是圖形使用者介面？
 - 圖形使用者介面 (Graphics User Interface, GUI)
 - 以圖像方式與使用者互動：
 - 程式顯示訊息
 - 程式顯示資訊
 - 操作介面
- 例子：
 - 電視遊樂器
 - 電腦遊戲



圖形使用者介面

- 要讓 Java 在 MS Windows 作業系統下或 Linux/Unix 下的 X windows 環境下以圖形方式呈現必須靠 Java 的 AWT 或 Swing 套件
- 藉由 AWT 及 Swing 提供的 GUI 元件便可畫出程式的使用者介面

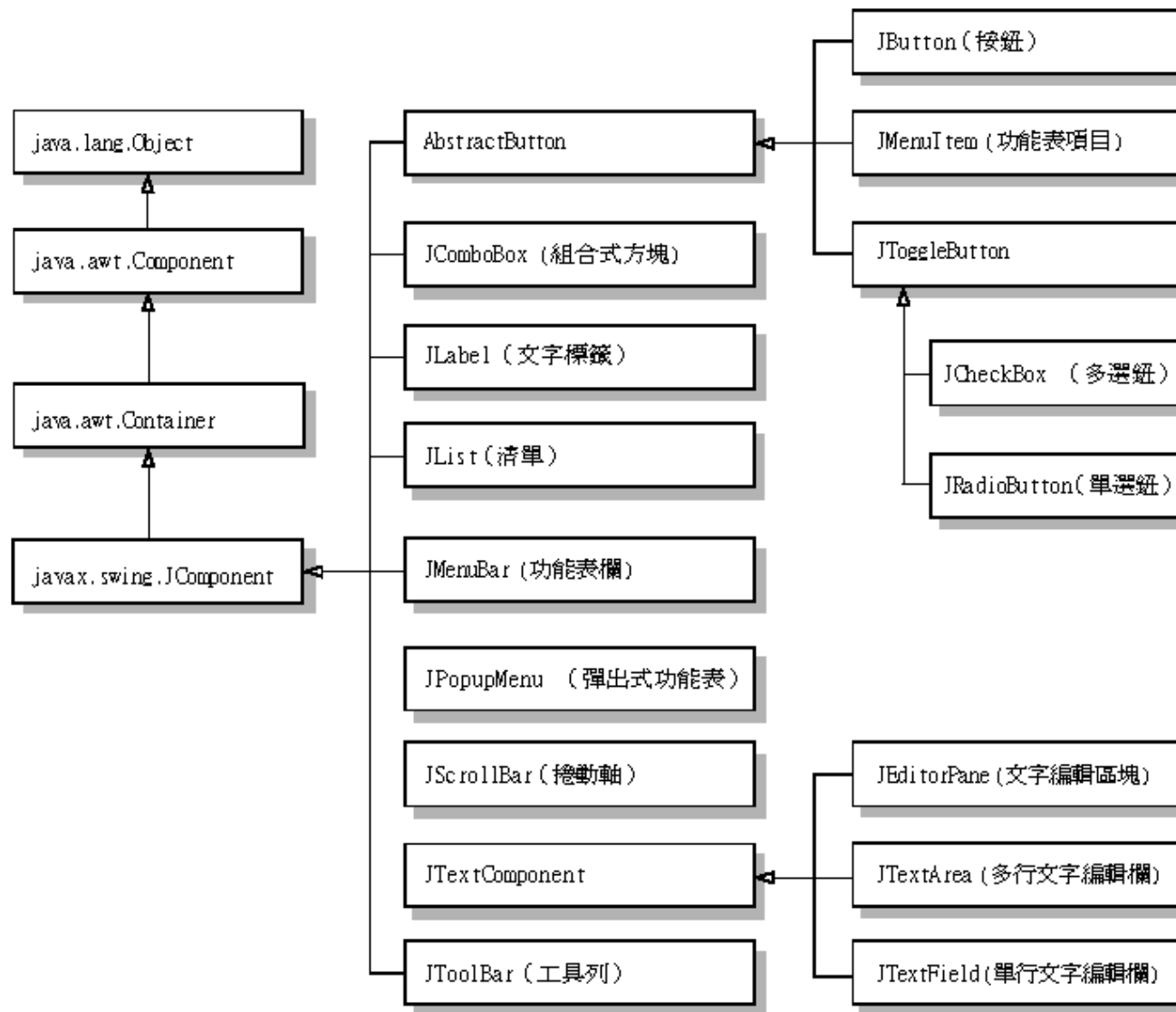
JAVA AWT vs. Swing

- Java 推出時僅提供 AWT(Abstract Windows Toolkit) 這套 GUI 元件
- Java 1.1 時又推出 Swing 強化的 GUI 類別庫
- AWT 所提供的功能比較陽春
 - 提供基本的 GUI 元件，如視窗、按鈕
- Swing 剛推出時算是外加的擴充模組到 Java 1.2 時被納入 Java 核心之中
- Swing 與 AWT 最大不同：Swing 都是由 JAVA 自行繪製，而不是靠作業系統，因此 Swing 畫出來的介面在不同作業系統下會比較一致

JAVA Swing

- Swing 套件
 - 匯入 Swing 套件：套件名稱 `javax.swing.*`;
 - 匯入 Swing 套件後便可使用其所提供的 GUI 元件，產生 GUI 物件
- Component 類別
 - Swing 中最常用的類別，包含按鈕、功能表、工具列、文字方塊、下拉選單 ... 等

JAVA Swing (Cont.)



JAVA Swing (Cont.)

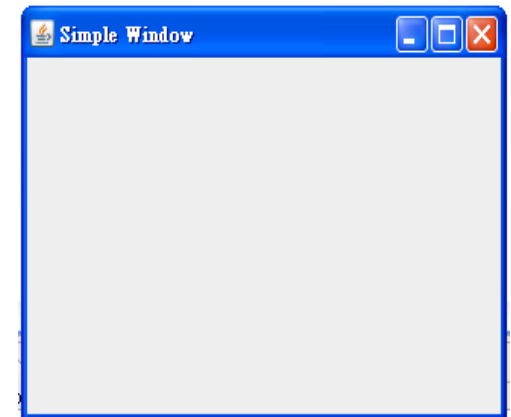
- 容器類別
 - 要顯示各種 GUI 元件，必須要先有視窗或其它類型的**容器物件** (Container) 來包含這個元件
 - 所以圖形介面程式一開始必須要先建立一個容器物件，才能將所需元件加到容器物件中，並顯示出來
- Swing 的容器物件種類
 - JFrame：典型視窗，建立一般視窗時使用
 - JDialog：交談窗類別的視窗
 - JWindow: 不含視窗標題等基本視窗要件的陽春型視窗

JAVA Swing (Cont.)

- JFrame 與 JPanel：兩者均不能用來建立獨立的視窗，必須在前述三種容器物件中使用
 - 建立子視窗可用 JFrame
 - 建立面板則必須靠 JPanel
- 建立陽春型的視窗

設定當使用者關閉視窗時要做什麼動作EXIT_ON_CLOSE 這個常數的意思, 就是指在關閉視窗時即結束程式

```
5 import javax.swing.*;  
6 public class GUITest(  
7     public static void main(String[] argv){  
8         JFrame mainFrame = new JFrame("Simple Window");  
9         //設定關閉視窗時同時結束程式  
10        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
11        mainFrame.setSize(320, 240);  
12        mainFrame.setVisible(true);  
13    }  
14 }
```



JAVA Swing (Cont.)

- 建立視窗後，將元件加入『容器』裡
 - 不能將元件直接加在 JFrame 、 JDialog 裡，必須放在 Content Pane 裡才行
 - 將 JFrame 想像成視窗的外框
 - Content Pane 想像成視窗中實際可用的區域



JAVA Swing (Cont.)

- 要取得 JFrame 的 Content Pane 必須呼叫
 - `public Container getContentPane()` 方法
- 接著呼叫
 - `Component add(Component comp)` 方法
 - ps. 此方法是繼承自 `java.awt.Container`

JAVA Swing (Cont.)

- JFrame 、 JDialog 、 JWindow 都是 java.awt.Window 的衍生類別， JWindow 定義了一組視窗容器共用的操作方法

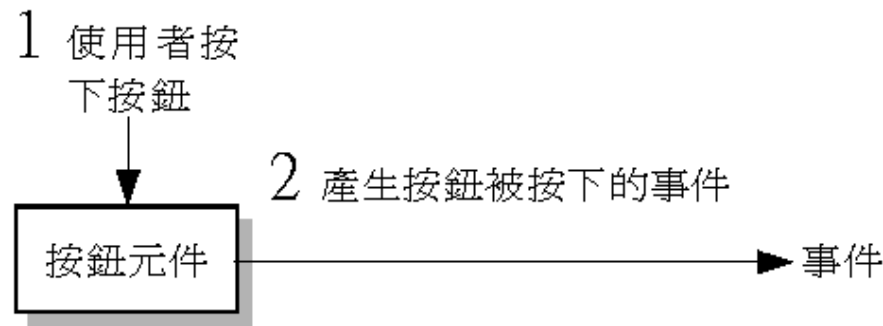
```
boolean isActive()           // 檢查視窗是否在使用中
boolean isAlwaysOnTop()      // 檢查視窗是否永遠在最上層
boolean isShowing()         // 檢查視窗是否有被顯示

void setAlwaysOnTop(boolean alwaysOnTop)
                           // 設定視窗是否永遠在最上層
void setBounds(int x, int y, int width, int height)
                           // 設定視窗位置 (x,y) 和寬高 (width, height)
void setSize(int width, int height) // 設定視窗大小 (寬與高)
void setVisible(boolean b)         // 設定是否顯示視窗

void toBack()                // 將視窗移到背景
void toFront()               // 將視窗移到前景 (畫面最前面)
```

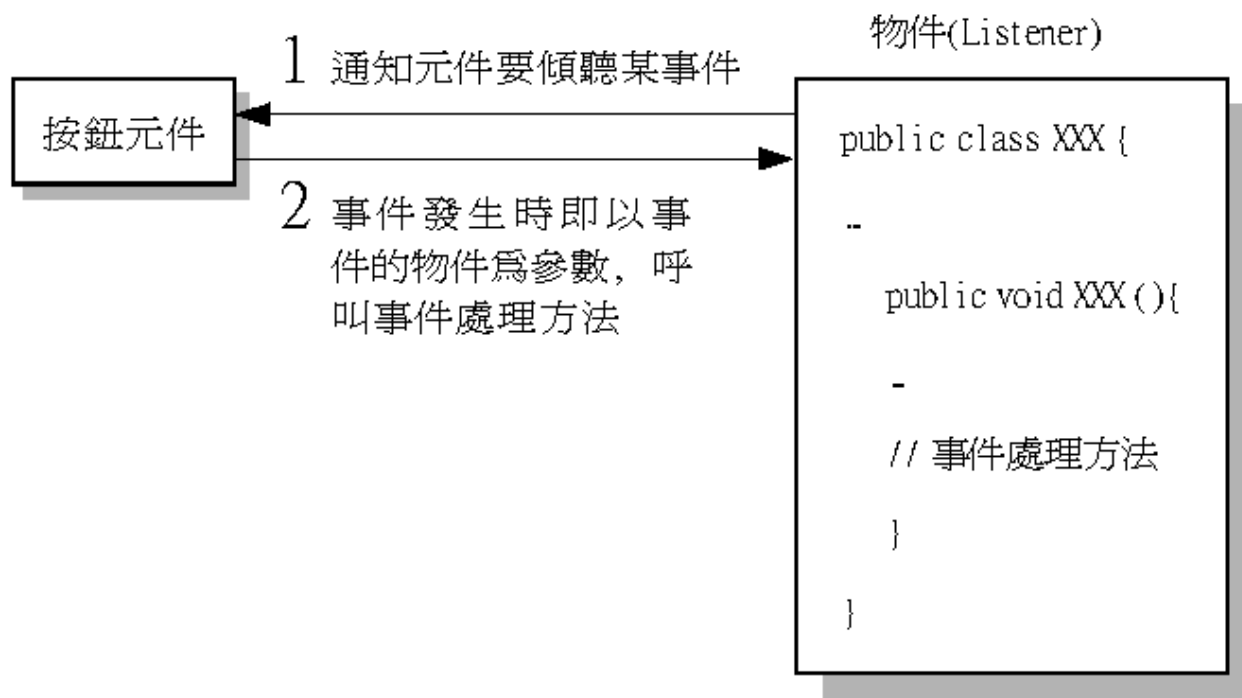
GUI 的事件處理

- AWT 的委派事件處理 → 當使用者在 GUI 元件中做動作時都會產生事件 (Event)，例如：按下按鈕



- 如何達到按下按鈕會有動作？
 - 撰寫事件處理方法來處理相對應的事件
 - 通知會產生該事件的元件：我的物件要當事件的傾聽者 (Listener)

GUI 的事件處理 (Cont.)



GUI 的事件處理 (Cont.)

- 實作 XXXListener 介面：
 - 負責事件處理的類別必須宣告實作 XXXListener 介面
 - 不同類型的事件需要實作不同的介面
 - 常見的介面如下

動作種類 (事件種類)	對應的 Listener 介面
按下按鈕	ActionListener
視窗開啓或關閉	WindowListener
按滑鼠按鈕	MouseListener
滑鼠移動	MouseMotionListener

GUI 的事件處理 (Cont.)

- 撰寫事件處理方法：
 - 即上述的 Listener 介面中所宣告的方法
 - 每種介面所宣告方法數量都不一
 - 例如：
 - ActionListener 介面只有 actionPerformed() 方法
 - MouseMotionListener 介面有 mouseDragged() 及 mouseMoved() 兩個方法
 - 需要回應事件的動作寫在對應的方法中
 - 例：使用者按鈕時希望改變視窗的背景顏色，則改變顏色的指令要寫在 actionPerformed() 裡

GUI 的事件處理 (Cont.)

- 告知元件我們要當傾聽者
 - 在程式中呼叫按鈕元件的 `addActionListener()` 方法，目的是通知該元件，我們的物件是個傾聽者
 - 按下按鈕時會呼叫我們寫好的 `actionPerformed()` 方法，執行我們指定的動作
 - 呼叫 `addActionListener()` 方法時，必須實作 `actionListener` 介面的物件為參數，意即『我這個物件是個傾聽者』
 - 程式中並不會呼叫 `actionPerformed()` 事件處理方法，因為這個方法是給元件呼叫的，是事件處理架構是一種**被動**的處理方法

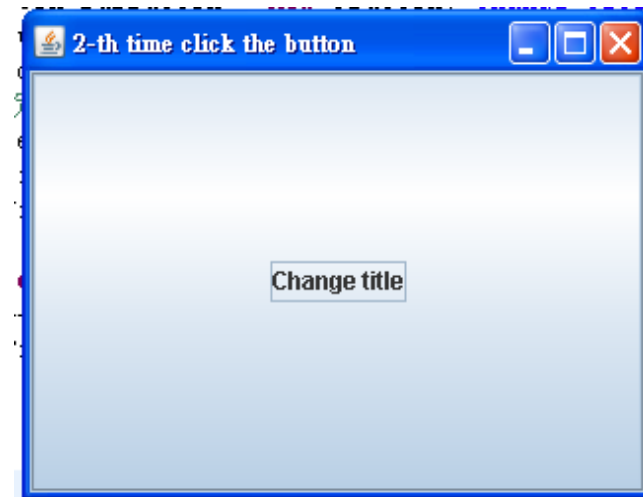
實作 Listener 介面

- 如何寫一個會處理按鈕事件的傾聽者？
 - 此部份功能源自 AWT → 必須匯入 `awt.event.*`; 套件
 - 實作 Listener 介面可在主程式外另建一個類別，將其宣告實作 Listener 介面，同時實作事件處理方法
 - 利用主程式外的類別來實作 Listener 介面將使得程式撰寫變得複雜，因為我們通常會在事件處理方法中使用到視窗容器或其他元件物件
 - 建議：在主程式中實作 Listener 介面

實作 Listener 介面

- 以處理按鈕事件為例，必須完成三個動件
 - 宣告實作 Listener 介面
 - 撰寫事件處理方法
 - 告知元件我們要當傾聽者

```
5 import javax.swing.*;
6 import java.awt.event.*;
7 public class GUITest extends JFrame implements ActionListener{
8     int act = 0;
9     public static void main(String[] argv){
10         GUITest gt = new GUITest();
11     }
12     public GUITest(){
13         setTitle("Listener demo");
14         JButton surButton = new JButton("Change title");
15         surButton.addActionListener(this);
16         getContentPane().add(surButton);
17         //設定關閉視窗時同時結束程式
18         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19         setSize(320, 240);
20         setVisible(true);
21     }
22     public void actionPerformed(ActionEvent e){
23         act++;
24         setTitle(act+"-th time click the button");
25     }
26 }
27 }
```



實作 Listener 介面

- 有趣的問題 - 程式並未呼叫 actionPerformed() ，按鈕卻會有動作
 - 事件處理方法是被元件 (or 系統) 呼叫
 - 不同於以往呼叫元件 (or 系統) 提供的方法
 - 事件處理方法也稱 call-back 方法
- 如何處理多個按鈕的事件？
 - 在視窗中多加幾個元件或按鈕，以按鈕為例，不同按鈕做不同事，但類別中只有一個 actionPerformed() 方法，如何達到要求？

實作 Listener 介面

- 簡單作法 - 在 actionPerformed() 方法中用 if 或 switch 等方法檢查產生事件的是哪一個元件，以決定要做哪些動作
- 此種作法僅適用於性質 / 功能相似的方法

```
public void actionPerformed(ActionEvent e) {  
    if (e.getSource() == XXX) // getSource() 方法可取得產生事件的物件  
        // XXX 按鈕的處理動作  
    } else {  
        // 其它按鈕的處理動作  
    }  
    ...  
}
```

實作 Listener 介面

- 要處理完全不同性質的按鈕元件
 - 使用 JAVA 提供的內部類別程匿名類別功能
 - 此種寫法較符合物件導向
 - 具有較佳的閱讀性及方便程式撰寫
- 內部類別
 - 定義在另一個類別內部，且非 static 的類別
 - 巢狀類別 (Nested Class): 定義在另一個類別內部的類別
 - 內部類別 (Inner Class): 未被明確或隱含宣告為 static 的巢狀類別

實作 Listener 介面

- 相對於內部類別而言，包含住它的類別稱之為外部 (Outer) 類別，或稱之為外層類別

```
Class A {          // 外部類別
    ...

    Class B {      // 內部類別
        ...
    }
}
```

- 回想一下：類別內部的方法可存取類別內部的成員

實作 Listener 介面

- 內部類別最大的特點，內部類別可存取外部類別的成員
- ★ 注意 ★ 宣告為 static 的巢狀類別則無法存取外部類別的成員

```
5 import javax.swing.*;
6 import java.awt.event.*;
7 public class MultiListener extends JFrame{
8     int act = 0;
9     public static void main(String[] argv){
10         MultiListener ml = new MultiListener();
11     }
12     public MultiListener(){
13         setTitle("Listener Demo");
14         JButton sureButton = new JButton("換標題");
15         sureButton.addActionListener(new InnerListener());
16         getContentPane().add(sureButton);
17         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
18         setSize(420, 140);
19         setVisible(true);
20     }
21     class InnerListener implements ActionListener{
22         public void actionPerformed(ActionEvent e){
23             act++;
24             setTitle("發生 "+act+" 次按鈕事件");
25         }
26     }
27 }
```

匿名類別 (Anonymous Class)

- 只有類別的本體，但沒有類別的名稱，進一步說，連物件的參照變數都不用宣告
- 在使用物件時才同時定義類別和產生物件的類別
- 物件導向程式設計的特性之一，就是軟體元件（類別）的重複使用性
- 一般情況下，定義一個類別即可在程式中用它來建立多個物件，

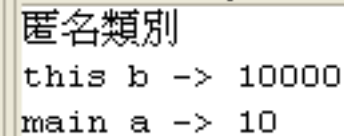
匿名類別 (Anonymous Class) (Cont.)

- 有時在程式中所設計的類別在程式中只用到一次，不會重複使用到，此時適合利用匿名類別
- 使用匿名類別也可以使程式碼較簡化
- 只要需要在需要匿名類別物件的敘述上，直接將類別物件放在要產生物件的地方
- 匿名類別必須衍生自一既有的類別或介面下

```
new 類別或介面名稱() {  
    // 匿名類別的定義內容  
}
```


匿名類別 (Anonymous Class) (Cont.)

```
6 public class AnonymousDemo{
7     public static void main(String[] argv){
8         final int a = 10;
9
10        (new Object(){//匿名類別
11            int b = 10000;
12            public void show(){ //匿名類別的方法
13                System.out.println("匿名類別");
14                System.out.println("this b -> "+b);
15                System.out.println("main a -> "+a);
16            }
17        }).show();
18
19    }
20 }
```



```
匿名類別
this b -> 10000
main a -> 10
```

- 以匿名類別實作 ActionListener
 - 當視窗含有多個按鈕元件，又不適合統一用一個類別來實作所有的元件傾聽者，可以用匿名類別實作

匿名類別 (Anonymous Class) (Cont.)

```
5 import javax.swing.*;
6 import java.awt.event.*;
7 public class MultiListener extends JFrame{
8     int act = 0;
9     public static void main(String[] argv){
10         MultiListener ml = new MultiListener();
11     }
12     public MultiListener(){
13         setTitle("Listener Demo");
14         JButton sureButton = new JButton("換標題");
15         sureButton.addActionListener(
16             new ActionListener(){
17                 public void actionPerformed(ActionEvent e){
18                     act++;
19                     setTitle("發生 "+act+" 次按鈕事件");
20                 }
21             }
22         );
23         getContentPane().add(sureButton);
24         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25         setSize(420, 140);
26         setVisible(true);
27     }
28 }
```

```
5 import javax.swing.*;
6 import java.awt.event.*;
7 public class GUITest extends JFrame implements ActionListener{
8     int act = 0;
9     public static void main(String[] argv){
10         GUITest gt = new GUITest();
11     }
12     public GUITest(){
13         setTitle("Listener demo");
14         JButton surButton = new JButton("Change title");
15         surButton.addActionListener(this);
16         getContentPane().add(surButton);
17         //設定關閉視窗時同時結束程式
18         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19         setSize(320, 240);
20         setVisible(true);
21     }
22     public void actionPerformed(ActionEvent e){
23         act++;
24         setTitle(act+"-th time click the button");
25     }
26 }
27 }
```