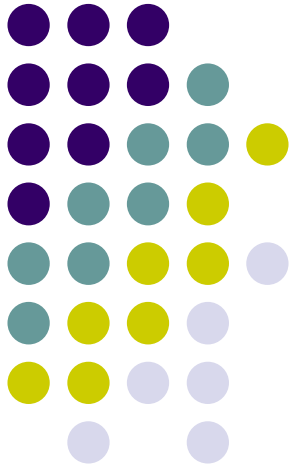
第二十三章
認識Swing



Swing概述

認識JFrame類別

學習Swing的基本物件

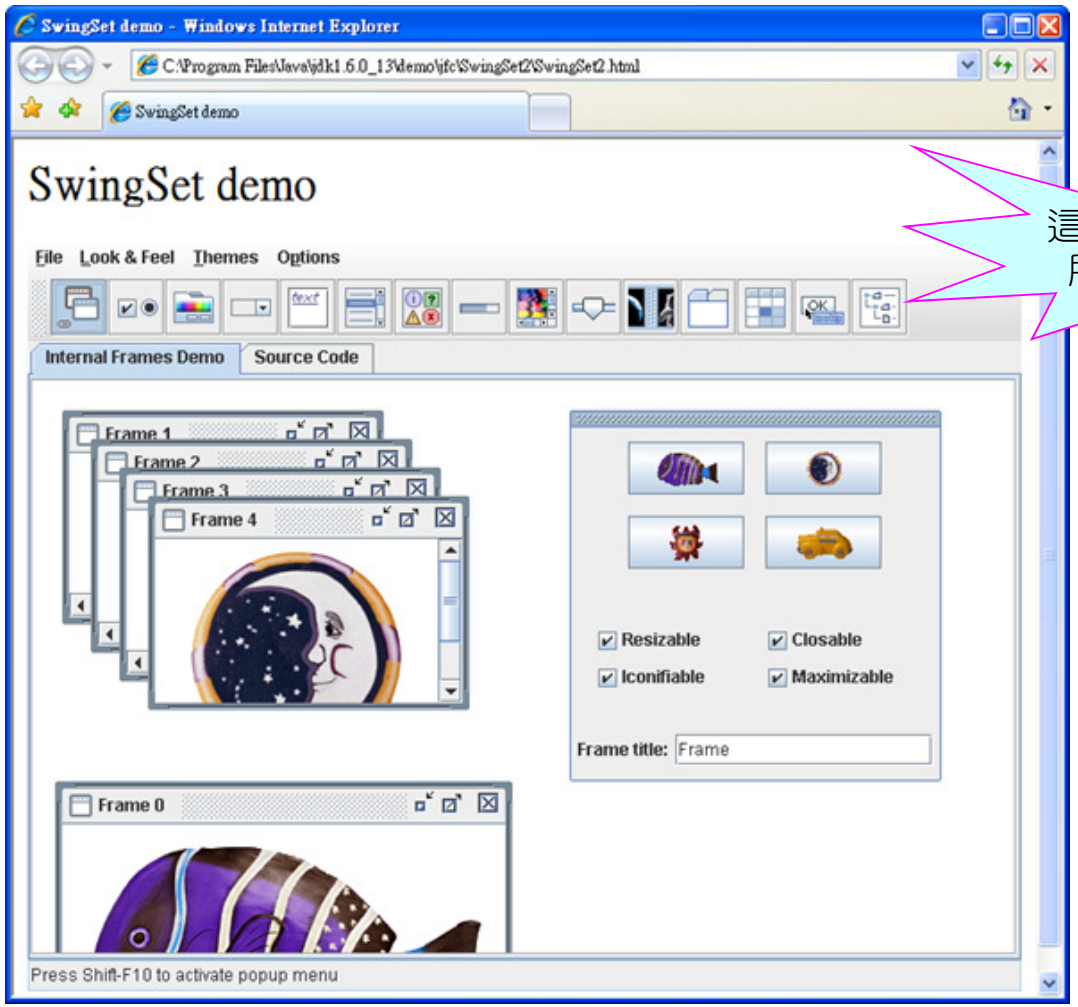
學習Swing物件之間的互動

取代AWT的Swing

- Swing物件
  - javax.swing類別庫可建立Swing物件
  - Swing提供豐富的物件、美觀的圖形介面，及更高的執行效率
  - 每一個AWT 物件都有一個相對應的Swing介面取代
  - Swing提供進度列（process bar）、內部視窗（internal frame）
- 在下面的目錄內可以找到檔案SwingSet2.html：

```
C:\Program Files\Java\jdk1.6.0_13\demo\jfc\SwingSet2\
```

Swing物件的展示圖



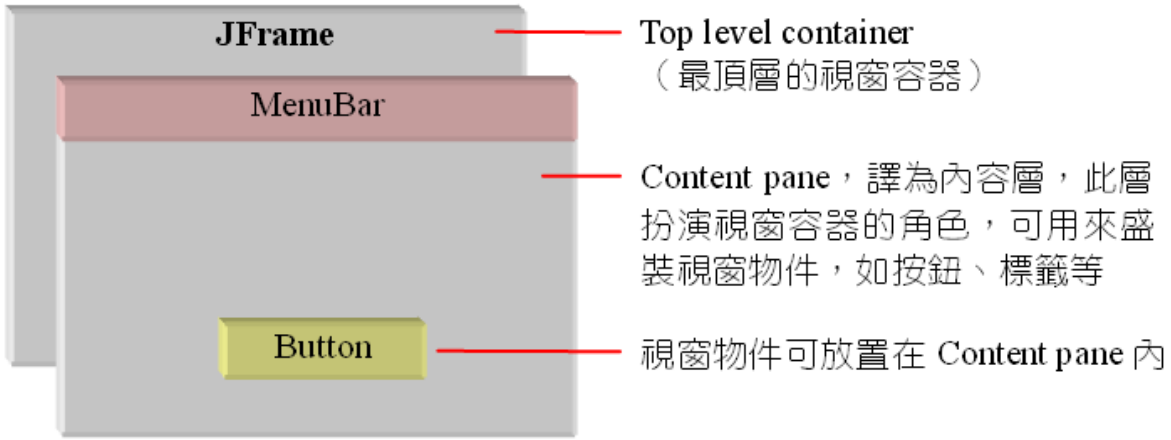
這是由Swing類別
所寫成的applet

Swing物件的繼承關係

Swing物件的
繼承關係圖

JFrame視窗類別

- Swing的視窗包含好幾個層級（layer）
- 以「Content pane」層級較為常用
  - Content pane譯為內容層
  - 下圖為Swing的視窗層級說明圖：



JFrame類別的建構元與method

- 下表列出JFrame類別的建構元與常用的method

表 23.2.1 JFrame 的建構元與常用的 method

建構元	主要功能
JFrame()	建立 JFrame 視窗物件
JFrame(String title)	建立 JFrame 視窗物件，視窗標題為 title

method	主要功能
Container getContentPane()	取得 content pane
void setLayout(LayoutManager manager)	設定版面配置為 manager
void update(Graphics g)	跳過清除背景的步驟，直接呼叫 paint()

- 物件並非擺在JFrame層，而是置於Content pane

JFrame視窗的練習

本範例是JFrame視窗類別的練習

```
01  // app23_1, JFrame 類別的練習
02  import java.awt.*;
03  import java.awt.event.*;
04  import javax.swing.*;          // 載入 javax.swing 類別庫裡的所有類別
05
06  public class app23_1 extends JFrame implements ActionListener
07  {
08     static app23_1 frm=new app23_1();
09     static Button btn=new Button("Click Me");
10     static Container cp=frm.getContentPane();       // 取得視窗容器
11
12     public static void main(String args[])
13     {
14        cp.add(btn);                                 // 將按鈕 btn 加入內容層中
15        cp.setLayout(new FlowLayout());              // 設定內容層的版面配置
16        cp.setBackground(Color.pink);                // 設定內容層的顏色
17        btn.addActionListener(frm);
18        frm.setTitle("JFrame 視窗");
19        frm.setSize(200,150);
20        frm.setVisible(true);
21     }
22     // 按下 btn 按鈕的事件處理
23     public void actionPerformed(ActionEvent e)
24     {
25        if(cp.getBackground()==Color.pink)
26           cp.setBackground(Color.yellow);
27        else
28           cp.setBackground(Color.pink);
29     }
30  }
```

JInternalFrame類別的認識

- Swing可以在視窗內建立子視窗（internal frame）
  - 子視窗的建立是用JInternalFrame類別達成
  - JInternalFrame可以最大化、最小化、關閉視窗與加入視窗物件等
  - 要建立JInternalFrame視窗，必須先建立JFrame視窗

建構元與method

- 下表列出JInternalFrame類別的建構元與method：

表 23.2.2 JInternalFrame 類別的建構元與 method

建構元	主要功能
JInternalFrame()	建立一個子視窗物件
JInternalFrame(String title)	建立一個子視窗物件，視窗標題為 title
JInternalFrame(String title, boolean resizable, boolean closable, boolean maximizable, boolean iconifiable)	建立一個子視窗物件，視窗標題為 title，並且可設定大小是否可調整、是否可關閉、是否可最大化，以及是否可縮小成一個圖示

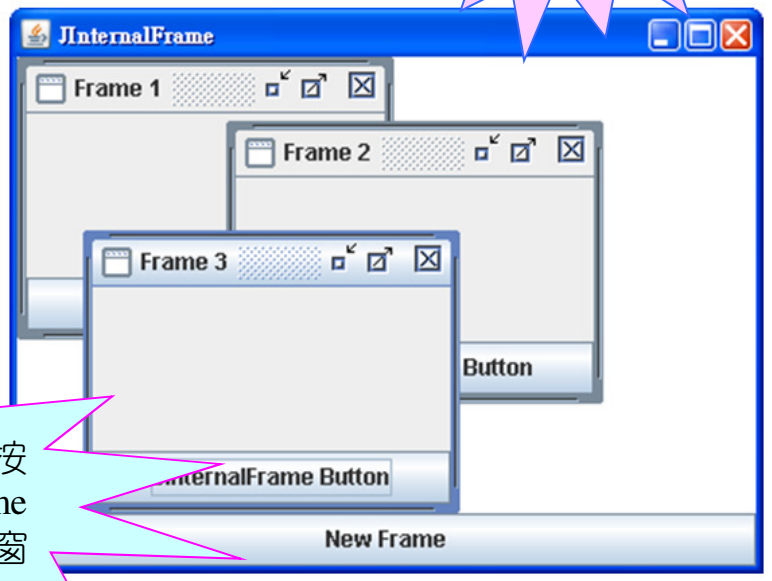
method	主要功能
void dispose()	將子視窗關閉，並釋放資源
Container getContentPane()	取得子視窗的內容層
String getTitle()	取得子視窗的標題
String setTitle(String title)	設定子視窗的標題
void show()	顯示子視窗

建立內部視窗 (1/2)

app23_2 是 JInternalFrame 類別的練習

```
01  // app23_2, JInternalFrame 類別的練習
02  import java.awt.*;
03  import java.awt.event.*;
04  import javax.swing.*;
05
06  public class app23_2
07  {
08     static JFrame frm=new JFrame("JInternalFrame");
09     static JButton btn=new JButton("New Frame"); // 建立 JButton 物件
10
11     static Container cp=frm.getContentPane();   // 取得內容層
12     static JDesktopPane jdp=new JDesktopPane(); // 建立桌面層物件
13
14     public static void main(String args[])
15     {
16         cp.setLayout(new BorderLayout());
17         cp.add(btn,BorderLayout.SOUTH);
18         cp.add(jdp);              // 將桌面層加到內容層中
19
20         btn.addActionListener(new ActLis());
21         frm.setSize(400,300);
22         frm.setVisible(true);
23     }
```

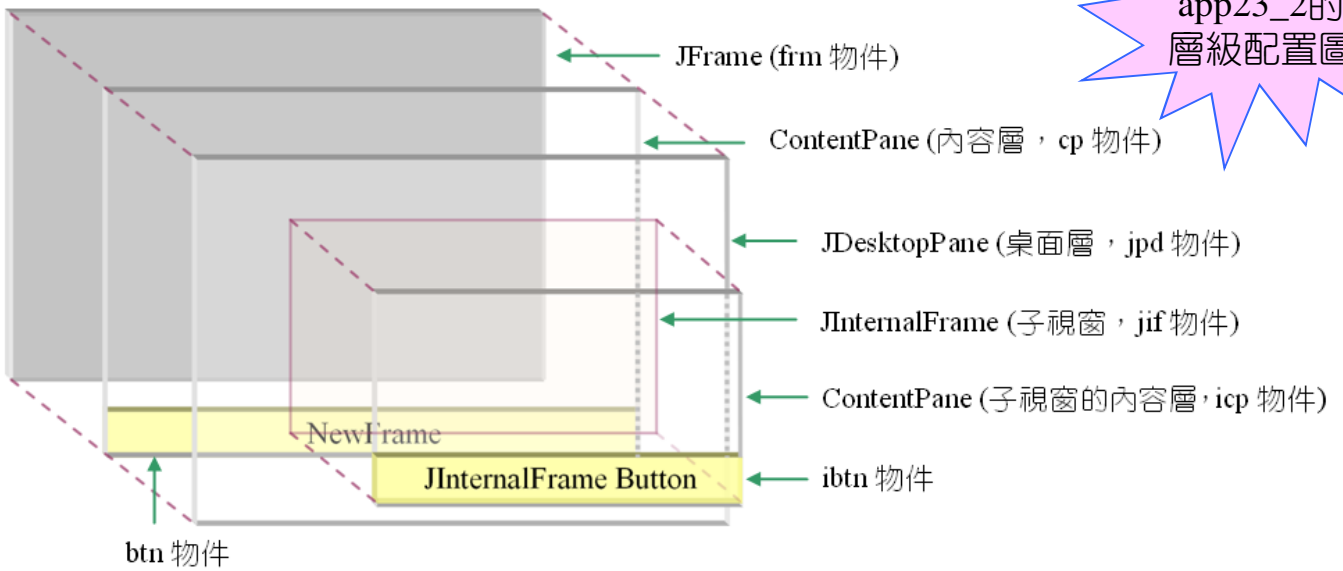
app23_2的執行結果



在JFrame視窗內配置一個按鈕，只要按下按鈕，JFrame視窗內便會建立一個子視窗

建立內部視窗 (2/2)


```
24
25      static class ActLis implements ActionListener
26      {
27         static int count=1;       // 宣告 count 變數，用來記錄子視窗的總數
28         public void actionPerformed(ActionEvent e)
29         {
30            JInternalFrame jif;    // 建立子視窗物件 jif
31            jif=new JInternalFrame("Frame "+(count++),true,true,true,true);
32            Container icp=jif.getContentPane();  // 取得 jif 的內容層
33            JButton ibtn=new JButton("JInternalFrame Button");
34            icp.add(ibtn,BorderLayout.SOUTH);     // 將 ibtn 按鈕加入 icp 中
35            jdp.add(jif);          // 將子視窗物件 jif 加到桌面層中
36            jif.setSize(200,150);
37            jif.setVisible(true);
38         }
39      }
40   }
```



app23_2的層級配置圖

Swing的按鈕和標籤

- Swing的按鈕與標籤
  - 可加上圖片影像
  - 還可以設定按鈕被按下，或是滑鼠指標停在按鈕上時所顯示的影像圖形（image icon）
- 把影像加到按鈕（或標籤）的步驟：

(1) 利用ImageIcon() 建構元讀入圖檔，即會建立ImageIcon類別的物件

(2) 把物件當成引數傳遞給按鈕（或標籤）類別的建構元或method

JButton類別的建構元

- Swing的按鈕
  - 以JButtom類別處理
  - Swing按鈕常用的method多半定義在JButton的父類別AbstractButton中
- 下表列出JButton類別的建構元：

表 23.3.1　JButton 的建構元

建構元	主要功能
JButton()	建立 JButton 物件
JButton(Icon icon)	建立 JButton 物件，並使用 icon 為圖示
JButton(String text)	建立 JButton 物件，標題為 text
JButton(String text, Icon icon)	建立 JButton 物件，標題為 text，圖示為 icon

JButton類別常用的method

- 使用JButton類別時常用到的method：

表 23.3.2 JButton 常用的 method（這些 method 定義在 JButton 的父類別 AbstractButton 中）

method	主要功能
Icon getIcon()	傳回按鈕的圖示
void setIcon(Icon icon)	設定按鈕的圖示為 icon
Icon getPressedIcon()	傳回按鈕被按下時的圖示
void setPressedIcon(Icon icon)	設定按鈕被按下時的圖示為 icon
Icon getRolloverIcon()	傳回滑鼠從上面經過時，按鈕的圖示
void setRolloverIcon(Icon icon)	設定滑鼠從上面經過時，按鈕的圖示為 icon
String getText()	傳回按鈕的標題
void setText(String str)	設定按鈕的標題為 str
void setHorizontalTextPosition(int pos)	設定按鈕的標題在圖示的左邊或右邊，pos 的值可為 JButton.LEFT 或 JButton.RIGHT
void setVerticalTextPosition(int pos)	設定按鈕標題的垂直位置，pos 的值可為 JButton.TOP、JButton.CENTER 或 JButton.BOTTOM
void setEnabled(boolean b)	設定按鈕是否可用
void setRolloverEnabled(boolean b)	設定滑鼠指標與按鈕是否有互動效果

使用JButton的範例

app23_3是JButtom使用的範例

JButton按鈕的圖示會隨著滑鼠指標位置的不同而有所變化

```
01  // app23_3, JButton 影像圖示的變化
02  import java.awt.*;
03  import javax.swing.*;
04
05  public class app23_3
06  {
07     static JFrame frm=new JFrame("JButton 測試");
08     static Container cp=frm.getContentPane();
09
10     static ImageIcon general=new ImageIcon("c:\\Java\\img1.gif");
11     static ImageIcon rollover=new ImageIcon("c:\\Java\\img2.gif");
12     static ImageIcon pressed=new ImageIcon("c:\\Java\\img3.gif");
13     static JButton btn=new JButton("JButton");  // 建立 JButton 物件
14
15     public static void main(String args[])
16     {
17        cp.setLayout(new FlowLayout());
18        cp.add(btn);                          // 將按鈕加入內容層中
19
20        btn.setRolloverEnabled(true);  // 設定滑鼠指標與按鈕有互動效果
21        btn.setIcon(general);          // 設定在一般情況下，按鈕的圖示
22        btn.setRolloverIcon(rollover); // 設定指標在按鈕上方時的圖示
23        btn.setPressedIcon(pressed);   // 設定滑鼠按鍵按下時的圖示
24
25        frm.setSize(200,120);
26        frm.setVisible(true);
27     }
28  }
```

滑鼠沒有停在按鈕上

滑鼠停在按鈕上

按下滑鼠按鈕時

JLabel類別的建構元

- 在標籤內加入影像
  - 使用JLabel類別
  - 下表列出JLabel的建構元：

表 23.3.3　JLabel 的建構元

建構元	主要功能
JLabel()	建立 JLabel 物件
JLabel(Icon icon)	建立 JLabel 物件，並使用 icon 為圖示
JLabel(String text)	建立 JLabel 物件，標題為 text
JLabel(String text, Icon icon, int align)	建立 JLabel 物件，標題為 text，圖示為 icon，水平的對齊方式為 align（可為 CENTER、LEFT 或 RIGHT）

JLabel類別常用的method

- 下表列出JLabel常用的method：

表 23.3.4　JLabel 的 method

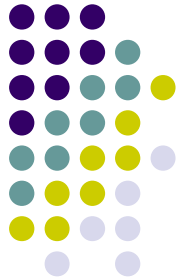
method	主要功能
Icon getIcon()	傳回標籤的圖示
void setIcon(Icon icon)	設定標籤的圖示為 icon
Icon getDisabledIcon()	傳回標籤無作用時的圖示
void setDisabledIcon(Icon icon)	設定標籤無作用時的圖示為 icon
int getIconTextGap()	取得標籤和文字間的距離
void setIconTextGap(int gap)	設定標籤和文字間的距離為 gap
void setHorizontalTextPosition(int pos)	設定標籤的名稱在圖示的左邊或右邊，pos 可為 JLabel.LEFT 或 JLabel.RIGHT
void setVerticalTextPosition(int pos)	設定標籤名稱的垂直位置，pos 可為 JLabel.TOP、JLabel.CENTER 或 JLabel.BOTTOM
String getText()	傳回標籤的名稱
void setText(String str)	設定標籤的名稱為 str

加入JLabel標籤 (1/2)

本範例配置JButton按鈕與JLabel標籤

```
01  // app23_4, JButton 與 JLabel 的綜合應用
02  import java.awt.*;
03  import java.awt.event.*;
04  import javax.swing.*;
05
06  public class app23_4
07  {
08     static JFrame frm=new JFrame("JButton & JLabel");
09     static Container cp=frm.getContentPane();
10
11     static ImageIcon pic[]=new ImageIcon[4];  // 建立 ImageIcon 陣列
12
13     static ImageIcon left=new ImageIcon("c:\\Java\\left.gif");
14     static ImageIcon right=new ImageIcon("c:\\Java\\right.gif");
15
16     static JButton btn1=new JButton(" 前一張 ",left);
17     static JButton btn2=new JButton(" 後一張 ",right);
18     static JLabel lab=new JLabel();
19
20     static int index=0;        // index 變數，用來記錄哪一張影像正被顯示
21
22     public static void main(String args[])
23     {
24        pic[0]=new ImageIcon("c:\\Java\\pic0.jpg");    // 載入影像
25        pic[1]=new ImageIcon("c:\\Java\\pic1.jpg");
26        pic[2]=new ImageIcon("c:\\Java\\pic2.jpg");
27        pic[3]=new ImageIcon("c:\\Java\\pic3.jpg");
28
29        cp.setLayout(new FlowLayout());
30        btn2.setHorizontalTextPosition(JButton.LEFT);// 設定文字水平位置
```

加入JLabel標籤 (2/2)



```
31          cp.add(btn1);
32          cp.add(btn2);
33          cp.add(lab);
34          lab.setIcon(pic[0]);
35          lab.setText("pic0.jpg");
36          lab.setHorizontalTextPosition(JLabel.CENTER); // 設定文字水平位置
37          lab.setVerticalTextPosition(JLabel.BOTTOM);   // 設定文字垂直位置
38
39          btn1.addActionListener(new ActLis());
40          btn2.addActionListener(new ActLis());
41
42          frm.setSize(400,350);
43          frm.setVisible(true);
44      }
45
46      static class ActLis implements ActionListener
47      {
48         public void actionPerformed(ActionEvent e)
49         {
50            JButton btn=(JButton) e.getSource();     // 取得被按下的按鈕
51            int num=pic.length;
52
53            if(btn==btn1 && index>0)    // 若 btn1 被按下，且 index>0
54               index--;
55            if(btn==btn2 && index<num-1) // 若 btn2 被按下，且 index<num-1
56               index++;
57
58            lab.setText("pic"+ index%num +".jpg");  // 設定標題名稱
59            lab.setIcon(pic[index%num]);
60         }
61      }
62  }
```

核取方塊--JCheckBox類別

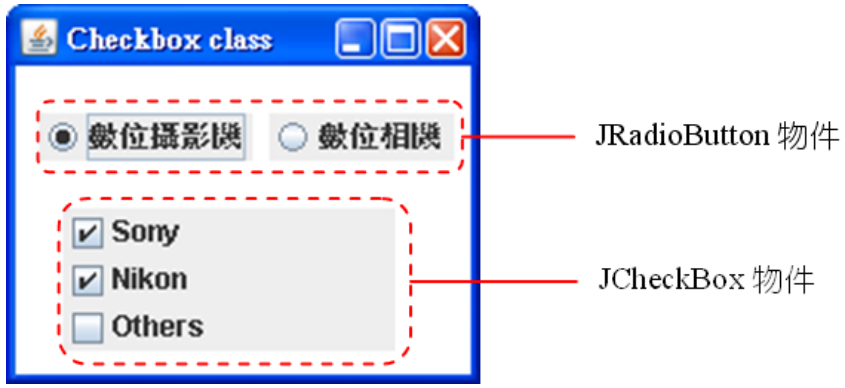
- 核取方塊（check box）
  - 可讓使用者選取一個或數個選項
  - Swing以JCheckBox與JRadioButton類別處理
  - JCheckBox類別可用來處理選項的複選
  - JCheckBox使用的是方形的選擇圖形
  - 下表列出JCheckBox類別常用的建構元：

表 23.4.1　javax.swing.JCheckBox 的建構元

建構元	主要功能
JCheckBox()	建立核取方塊
JCheckBox(String label)	建立標題為 label 的核取方塊
JCheckBox(Icon icon)	建立圖示為 icon 的核取方塊
JCheckBox(String label, boolean state)	建立標題為 label 的核取方塊，並設定 state 狀態，若 state 為 true，則核取方塊呈被選取狀態

選項方塊--JRadioButton類別

- JRadioButton類別
  - 常用的method多是繼承AbstractButton而來
  - JRadioButton使用圓形的選擇圖形
  - JRadioButton可以設計成複選，但使用者多半習慣單選物件的圖形為圓形，因此建議依循慣例
  - 下圖為JRadioButton與JCheckBox物件的示意圖



JRadioButton類別的建構元

- 下表列出JRadioButton類別常用的建構元

表 23.4.2　javax.swing.JRadioButton 的建構元

建構元	主要功能
JRadioButton()	建立選項方塊
JRadioButton (String label)	建立標籤為 label 的選項方塊
JRadioButton (Icon icon)	建立圖示為 icon 的選項方塊
JRadioButton (String label, boolean st)	建立標籤為 label 的選項方塊，並設定 st 狀態，若 st 為 true，則選項方塊呈被選取狀態

- 要把JRadioButton設為單選，須配合ButtonGroup類別
  - ButtonGroup可限制只有一個物件的狀態為true

核取方塊與選項方塊的應用 (1/2)

- 下面的範例說明JCheckBox與JRadioButton的應用

```
01  // app23_5, 核取方塊與選項方塊的應用
02  import java.awt.*;
03  import javax.swing.*;
04
05  public class app23_5
06  {
07     static JFrame frm=new JFrame("Checkbox class");
08     static Container cp=frm.getContentPane();
09     static JRadioButton rb1=new JRadioButton("數位攝影機");
10     static JRadioButton rb2=new JRadioButton("數位相機");
11
12     static JCheckBox ckb1=new JCheckBox("Sony",true);
13     static JCheckBox ckb2=new JCheckBox("Nikon",true);
14     static JCheckBox ckb3=new JCheckBox("Others");
15
16     public static void main(String args[])
17     {
```

核取方塊與選項方塊的應用 (2/2)

```
18          rb1.setBounds(10,20,90,20);
19          rb2.setBounds(107,20,78,20);
20          ckb1.setBounds(20,60,140,20);
21          ckb2.setBounds(20,80,140,20);
22          ckb3.setBounds(20,100,140,20);
23
24          ButtonGroup bgroup=new ButtonGroup();   // 建立 ButtonGroup 物件
25          bgroup.add(rb1);                        // 將 rb1 設定為單選
26          bgroup.add(rb2);                        // 將 rb2 設定為單選
27          rb1.setSelected(true);                  // 設定 rb1 被選擇
28
29          cp.add(rb1);
30          cp.add(rb2);
31          cp.add(ckb1);
32          cp.add(ckb2);
33          cp.add(ckb3);
34          cp.setLayout(null);
35          cp.setBackground(Color.white);
36          frm.setSize(200,160);
37          frm.setVisible(true);
38      }
39  }
```

JList類別的建構元

- 選擇選單裡特定的項目
  - 可使用JList類別
  - JList直接繼承自JComponent類別
  - 下表列出JList常用的建構元：

表 23.5.1　javax.swing.JList 的建構元與 method

建構元	主要功能
JList()	建立一個 JList 物件
JList (Object listData[])	利用 Object 陣列建立 JList 物件
JList(Vector listData)	利用 Vector 類別的物件來建立 JList 物件

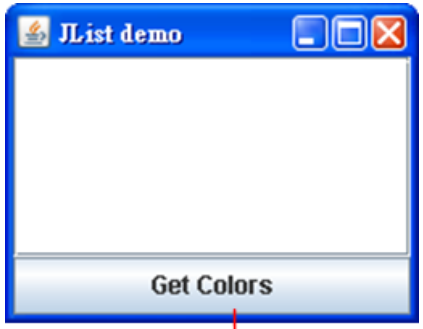
JList類別的method

- 下表列出JList常用的method：

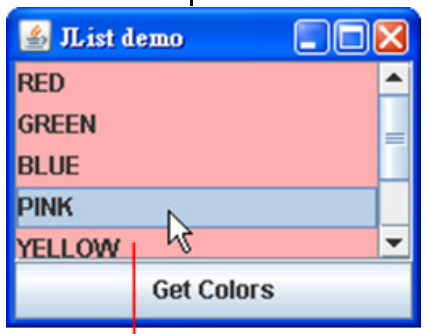
method	主要功能
void addListSelectionListener(ListSelectionListener listener)	設定 JList 物件的傾聽者
void clearSelection()	取消所選取的項目
Color getSelectionForeground()	取得被選取選項的前景顏色（即文字的顏色）
void setSelectionForeground(Color selectionForeground)	設定被選取選項的前景顏色（即文字的顏色）為 selectionForeground
Color getSelectionBackground()	取得被選取選項的背景顏色
void setSelectionBackground(Color selectionBackground)	設定被選取選項的背景顏色為 selectionBackground
int locationToIndex(Point location)	將 JList 物件上的任一點位置 location 轉換成 JList 選項的註標
void setListData(Object[] listData)	以 Object 陣列設定 JList 物件的選單
void setListData(Vector listData)	以 Vector 類別的物件設定 JList 物件內的選項
void setSelectedIndex(int index)	設定在註標為 index 的選項被選取
int getSelectedIndex()	取得被選取選項的註標值
Object getSelectedValue()	取得被選取選項的值

JList的練習 (1/2)

- app23_6的程式碼如下所示：



(1) 按下顏色按鈕，可以取得顏色選單



(2) 選擇選單內的選項，即可將 JList 物件的底色改為選項所指定的顏色

```
01  // app23_6, JList 的練習 (一)
02  import java.awt.*;
03  import java.awt.event.*;
04  import javax.swing.*;
05  import javax.swing.event.*;
06
07  public class app23_6
08  {
09     static JFrame frm=new JFrame("JList demo");
10     static Container cp=frm.getContentPane();
11     static JButton btn=new JButton("Get Colors");
12     static JList lst=new JList();          // 建立 JList 物件
13
14     public static void main(String args[])
15     {
16        cp.setLayout(new BorderLayout());
17        cp.add(btn,BorderLayout.SOUTH);
18        cp.add(new JScrollPane(lst));       // 將 lst 加入 JScrollPane 中
19        btn.addActionListener(new ActLis());      // 設定 btn 的傾聽者
20        lst.addListSelectionListener(new LSLis()); // 設定 lst 的傾聽者
21        frm.setSize(200,155);
22        frm.setVisible(true);
23     }
```

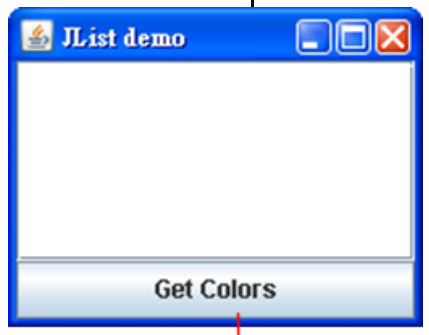
如果直接把lst物件放進JFrame的內容層中，也就是修改成

```
cp.add(lst);
```

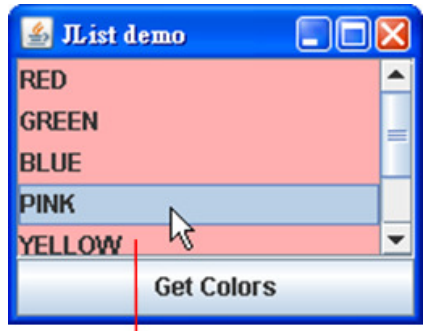
則lst物件沒有捲軸

JList的練習 (2/2)

```
24      static class ActLis implements ActionListener
25      {
26         public void actionPerformed(ActionEvent e)
27         {
28            String s[]={"RED","GREEN","BLUE","PINK","YELLOW","CYAN","GRAY"};
29            lst.setListData(s); // 將陣列 s 的內容加入 lst 中，做為 lst 的選項
30         }
31      }
32      static class LSLis implements ListSelectionListener
33      {
34         public void valueChanged(ListSelectionEvent e)
35         {
36            int color=lst.getSelectedIndex();   // 取得被選取選項的註標
37            switch(color)
38            {
39               case 0: lst.setBackground(Color.RED);      break;
40               case 1: lst.setBackground(Color.GREEN);    break;
41               case 2: lst.setBackground(Color.BLUE);     break;
42               case 3: lst.setBackground(Color.PINK);     break;
43               case 4: lst.setBackground(Color.YELLOW);   break;
44               case 5: lst.setBackground(Color.CYAN);     break;
45               case 6: lst.setBackground(Color.GRAY);     break;
46            }
47         }
48      }
49   }
```



(1) 按下顏色按鈕，可以取得顏色選單

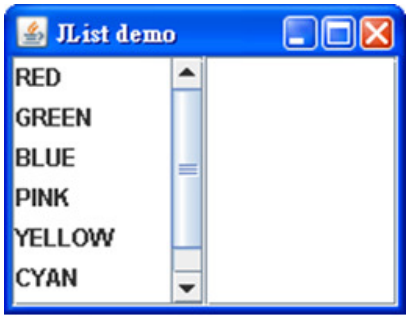


(2) 選擇選單內的選項，即可將 JList 物件的底色改為選項所指定的顏色

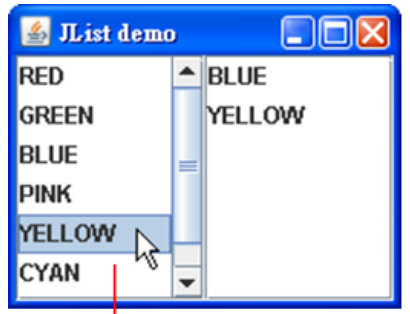
連按兩下選項的事件處理 (1/2)

- 下面是在JList物件中，連按兩下滑鼠左鍵的事件處理

```
01  // app23_7, JList 的練習 (二)
02  import java.awt.*;
03  import java.awt.event.*;
04  import javax.swing.*;
05  import java.util.Vector;            // 載入 util 類別庫裡的 Vector 類別
06
07  public class app23_7
08  {
09     static JFrame frm=new JFrame("JList demo");
10     static Container cp=frm.getContentPane();
11     static JList lst1=new JList();       // 建立 lst1 物件
12     static JList lst2=new JList();       // 建立 lst2 物件
13     static String s[]={"RED","GREEN","BLUE","PINK","YELLOW","CYAN","GRAY"};
14     static Vector<String> v=new Vector<String>(); // 建立 Vector 類別的物件 v
15
16     public static void main(String args[])
17     {
18        cp.setLayout(new GridLayout(1,2));
19        cp.add(new JScrollPane(lst1));    // 將 JScrollPane 加入 cp 中
20        cp.add(new JScrollPane(lst2));    // 將 JScrollPane 加入 cp 中
```



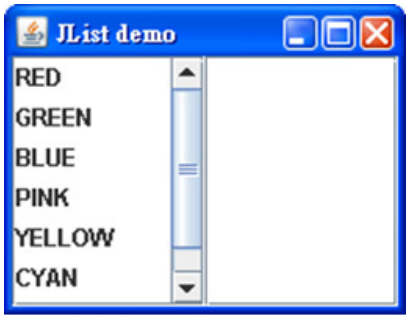
(1) 程式執行時最初的狀態



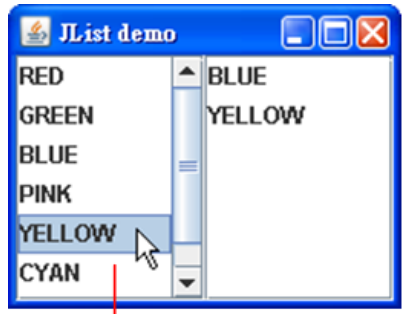
(2) 連按兩下選擇選單內的選項，即可將選項送到右邊的 JList 物件中

連按兩下選項的事件處理 (2/2)

```
21          lst1.setListData(s);                        // 設定 lst1 物件的選單
22          lst1.addMouseListener(new MouseLis());      // 設定 lst1 物件的傾聽者
23          frm.setSize(200,155);
24          frm.setVisible(true);
25       }
26       static class MouseLis extends MouseAdapter
27       {
28          public void mouseClicked(MouseEvent e)
29          {
30             if(e.getSource()==lst1)                  // 若是 lst1 物件被按下
31                if(e.getClickCount()==2)              // 如果連續被按了兩下
32                {
33                   int index=lst1.getSelectedIndex();
34                   String str=s[index];
35                   v.add(str);                        // 將字串 str 加入向量 v
36                   lst2.setListData(v);          // 設定向量 v 為 lst2 物件的選單
37                }
38          }
39       }
40    }
```



(1) 程式執行時最初的狀態



(2) 連按兩下選擇選單內的選項，即可將選項送到右邊的 JList 物件中

JColorChooser類別

- 顏色的選擇
  - 利用JColorChooser類別選取所要的顏色
  - showDialog() method可顯示「顏色選擇對話方塊」
  - 下表列出JColorChooser類別常用的建構元與method：

表 23.6.1　javax.swing.JColorChooser 的建構元與 method

建構元	主要功能
JColorChooser()	建立 JColorChooser 物件，預設顏色為白色
JColorChooser(Color iColor)	建立 JColorChooser 物件，預設顏色為 iColor

method	主要功能
Color getColor()	取得顏色選擇對話方塊中所選取的顏色
void setColor(Color color)	設定顏色選擇對話方塊中的顏色
void setColor(int r, int g, int b)	以 r,g,b 三個顏色設定對話方塊中的顏色
Color showDialog(Component c, String title, Color iColor)	顯示顏色選擇對話方塊，其父類別的物件為 c，標題為 title，預設顏色為 iColor

使用JColorChooser類別 (1/2)

- 下面是使用JColorChooser類別來設定視窗顏色的範例，執行結果如下：

使用JColorChooser類別 (2/2)

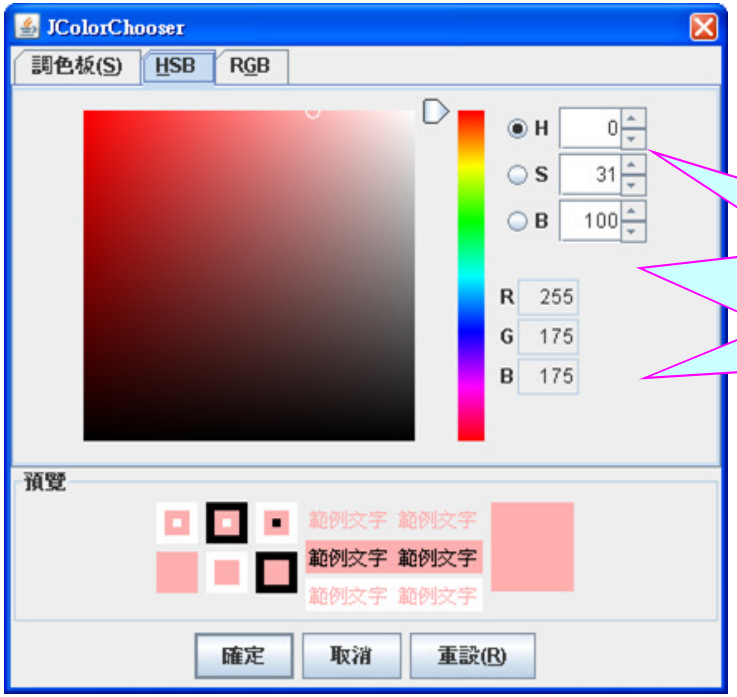
使用JColorChooser類別來設定視窗顏色的範例

```
01  // app23_8, JColorChooser 示範練習
02  import java.awt.*;
03  import java.awt.event.*;
04  import javax.swing.*;
05
06  public class app23_8
07  {
08     static JFrame frm=new JFrame("JColorChooser");
09     static Container cp=frm.getContentPane();
10     static JButton btn=new JButton("Get Color");
11     static JColorChooser JCC=new JColorChooser();  // 建立 JCC 物件
12     static Color color;                 // 宣告 Color 型態的變數 color
13
14     public static void main(String args[])
15     {
16         cp.setLayout(new BorderLayout());
17         cp.add(btn,BorderLayout.SOUTH);
18         btn.addActionListener(new ActLis());
19         cp.setBackground(Color.YELLOW);
20         frm.setSize(200,150);
21         frm.setVisible(true);
22     }
23     static class ActLis implements ActionListener
24     {
25        public void actionPerformed(ActionEvent e)
26        {
27           color=JCC.showDialog(frm,"JColorChooser",Color.pink);
28           cp.setBackground(color);              // 將視窗背景設為 color
29        }
30     }
31  }
```

顏色選擇對話方塊的標籤 (1/2)

- 顏色選擇對話方塊有三個標籤
  - 調色板
  - HSB
  - RGB
- HSB標籤的內容如下圖所示：

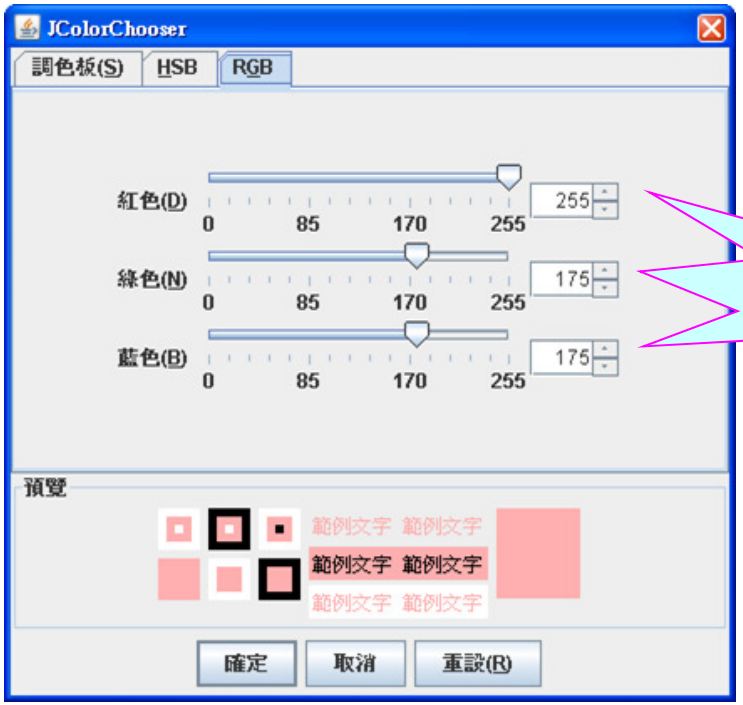
請參考app23_8的執行結果



HSB是Hue、Saturation與Brightness的縮寫，分別代表色調、色濃度與亮度

顏色選擇對話方塊的標籤 (2/2)

- RGB標籤的內容可參考下圖：



RGB代表紅、綠、藍三個顏色