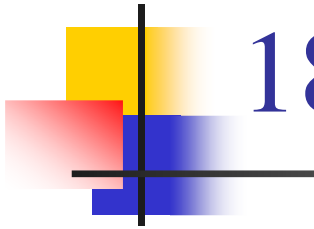
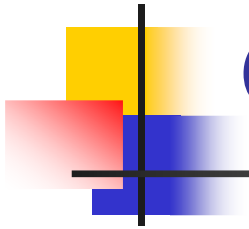


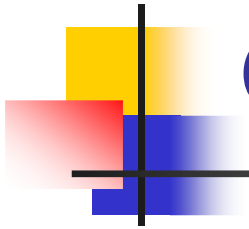
18-4 版面配置管理員

- 如果想多加幾個按鈕, 例如再加個按鈕讓程式顯示的按鈕次數遞減, 會發現程式仍只顯示一個按鈕
- 因為 JFrame 有預設的版面配置 (layout) 方式, 如果不依其規則指定按鈕的位置, 則按鈕都會放到同一個地方, 使得視窗上只看得到一個按鈕
- 除了指定按鈕位置外, 您也可改變預設的配置方式



GUI 元件配置的基本觀念

- 不管使用哪一種方法, 都需用到 AWT 的版面配置管理員 (layout manager)
- 在 Java 的 GUI 介面設計中, 是透過 AWT 的版面配置管理員來控制 GUI 元件在容器中的排列方式和位置
- 使用版面配置管理員的**好處**
  - 程式設計人員不需擔心 GUI 元件在視窗中的位置, 只要選擇合適的版面配置管理員, 不論使用者如何調整視窗的大小, 版面配置管理員都會依其內建的排列規則, 將 GUI 元件陳列在視窗之中

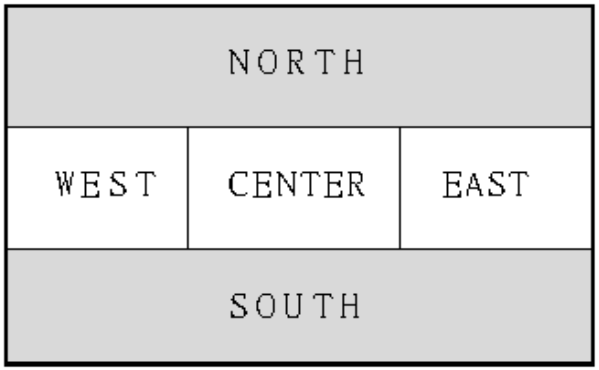


GUI 元件配置的基本觀念

- AWT 中定義的配置管理員雖然有很多種, 但一般我們比較會用到的不外以下幾種 (每個配置管理員都有一個對應的同名類別)：
  - BorderLayout
  - FlowLayout
  - GridLayout

BorderLayout 配置管理員

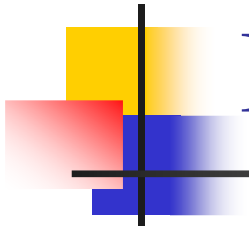
- Swing 中的每個容器類別都有其預設的版面配置管理員, 以下我們就先從 JFrame 預設使用的 BorderLayout 談起
- JFrame 預設採用的 BorderLayout 配置管理員, 是比較固定的的一種配置方式。
- BorderLayout 將可用的空間劃分為 5 個部份



BorderLayout 配置管理員

- 圖中的 NORTH 、WEST 等就是在加入元件時, 需指定的位置參數 (稱為 Constraint), 表示要將元件加到哪一個位置上：

```
add(new Button("North"), BorderLayout.NORTH);   // 在上面放置一個 "North" 按鈕
add(new Button("South"), BorderLayout.SOUTH);   // 在下面放置一個 "South" 按鈕
add(new Button("West"), BorderLayout.WEST);     // 在左邊放置一個 "West" 按鈕
add(new Button("East"), BorderLayout.EAST);     // 在右邊放置一個 "East" 按鈕
add(new Button("Center"), BorderLayout.CENTER); // 在中間放置一個 "Center" 按鈕
```



BorderLayout 配置管理員

- BorderLayout 除了位置固定外, 還有幾個**特點**：
  - 加入某位置的元件將會**填滿**整個位置；
  - 放大視窗時, 將以放大中間的部份為主, 上、下位置則是依視窗放大的情況動態調整。
- 以下就是一個應用 BorderLayout 的簡例, 這個程式會用到 JLabel、JText Field 這 2 個新的 GUI 元件

BorderLayout 配置管理員

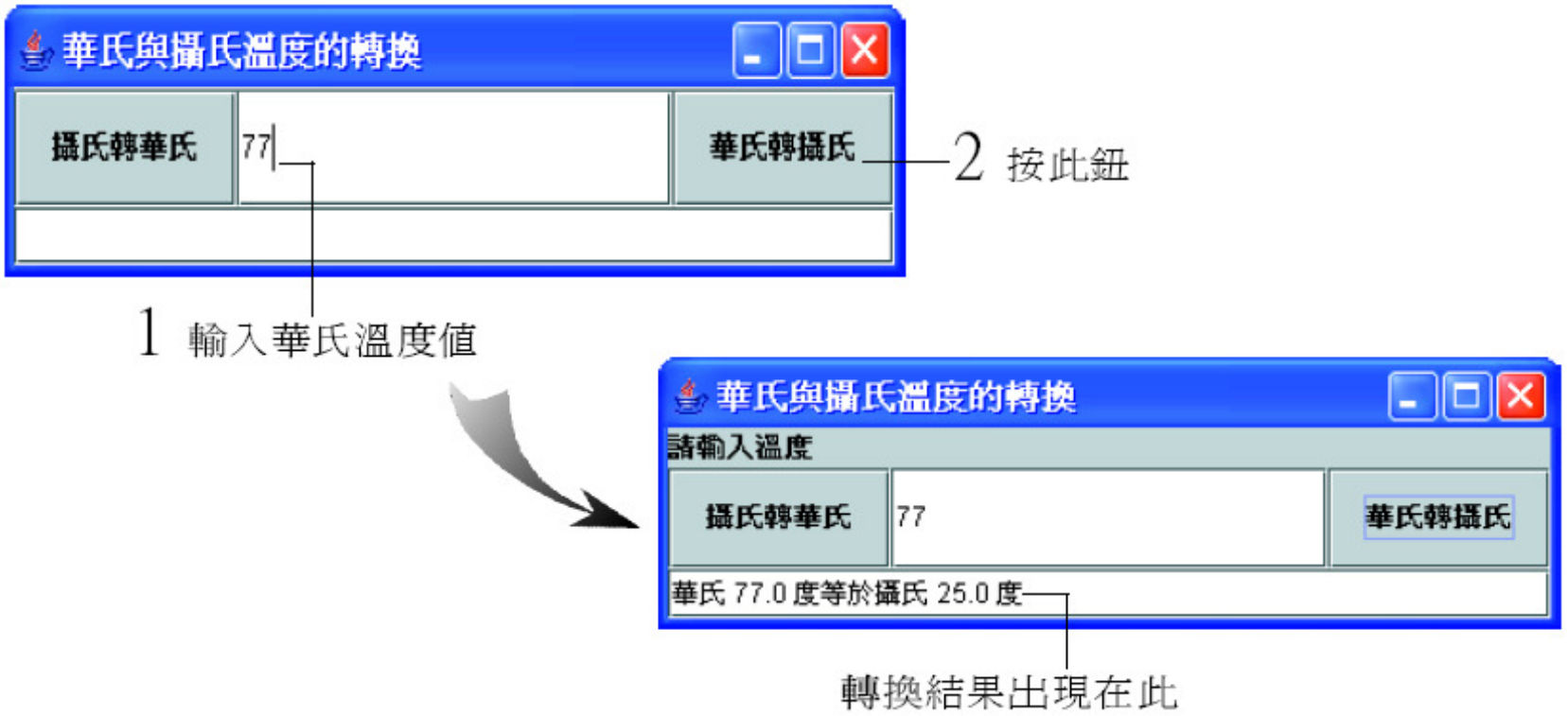
```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.*;
4  public class TempConverter{
5      JFrame myFrame = new JFrame("華氏、攝氏溫度轉換");
6      JTextField result = new JTextField();
7      JTextField degree = new JTextField();
8      JButton f2c = new JButton("華氏 -> 攝氏");
9      JButton c2f = new JButton("攝氏 -> 華氏");
10     public static void main(String[] argv){
11         TempConverter test = new TempConverter();
12     }
13     public TempConverter(){
14         Container contentPane = myFrame.getContentPane();
15         contentPane.add(new JLabel("請輸入溫度"), BorderLayout.NORTH);
16         contentPane.add(f2c, BorderLayout.EAST);
17         contentPane.add(c2f, BorderLayout.WEST);
18         contentPane.add(degree, BorderLayout.CENTER);
19         contentPane.add(result, BorderLayout.SOUTH);
```

BorderLayout 配置管理員

```
20          f2c.addActionListener(new ActionListener(){
21              public void actionPerformed(ActionEvent e){
22                  try{
23                      double f = Double.parseDouble(degree.getText());
24                      result.setText("華氏 "+f+" 度等於攝氏 "+ ((f-32)*5/9));
25                  }catch(NumberFormatException ne){
26                      degree.setText("");
27                  }
28              }
29          });
30          c2f.addActionListener(new ActionListener(){
31              public void actionPerformed(ActionEvent e){
32                  try{
33                      double c = Double.parseDouble(degree.getText());
34                      result.setText("攝氏 "+c+" 度等於華氏 "+ (c/5*9+32));
35                  }catch(NumberFormatException ne){
36                      degree.setText("");
37                  }
38              }
39          });
40          myFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
41          myFrame.setSize(400, 120);
42          myFrame.setVisible(true);
43      }
44  }
```

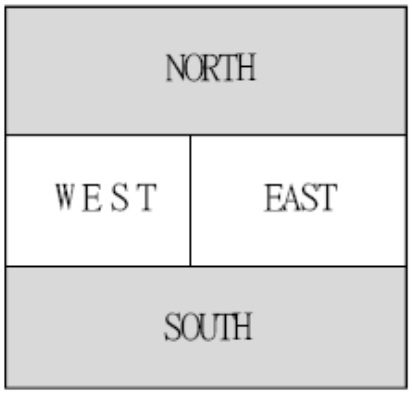
BorderLayout 配置管理員

執行結果

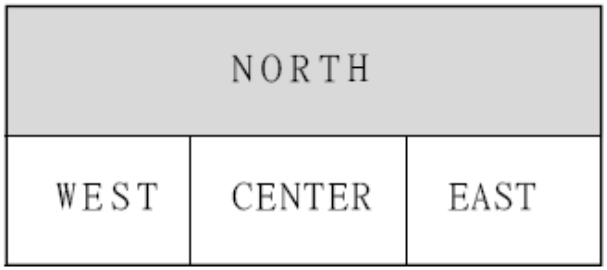


BorderLayout 配置管理員

- BorderLayout 的**特點**就是配置的方式/位置都預先設好了, **無法做變動**。但如果要放的元件不到 5 個, 則仍是有些彈性, 因為未用到的區域, 在視窗上就不會顯示出來, 所以就能產生不同的配置效果, 例如：

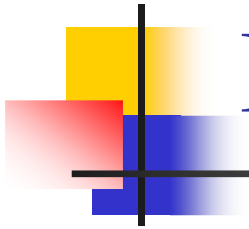


沒有用到 CENTER 時



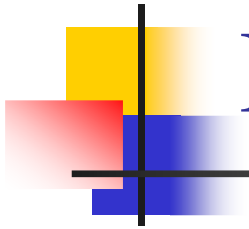
沒有用到 SOUTH 時的配置

只用到 NORTH 和 CENTER 時



BorderLayout 配置管理員

- BorderLayout 的主要**缺點**
  - 只有五個區域, 最多只能放五個元件
  - 若要放置更多的元件, 就需放入額外的容器物件
  - 若想讓元件以不同的方式排列, 則需改用其它的配置管理員



FlowLayout 配置管理員

- FlowLayout 配置管理員
  - 將加入的物件依序由容器左上角向右排列, 到視窗右邊界時會自動換行繼續排列
  - 各元件間的相對位置可能會隨視窗大小不同而變動
- Swing 中預設採用 FlowLayout 配置管理員的容器類別是 JPanel

FlowLayout 配置管理員

- 一般在設計 GUI 程式時, 並不會將元件直接加到 JFrame, 而是將 JPanel 加到 JFrame 中, 然後再將所需的元件加入 JPanel 容器, 如此會比較有彈性
- 但如果您要設計的 GUI 並不複雜, 而想直接在 JFrame 使用 FlowLayout 配置管理員, 則可用 Content Pane 物件的 setLayout() 方法來設定：

```
JFrame myframe...
...
myframe.getContentPane().setLayout(new FlowLayout());
```

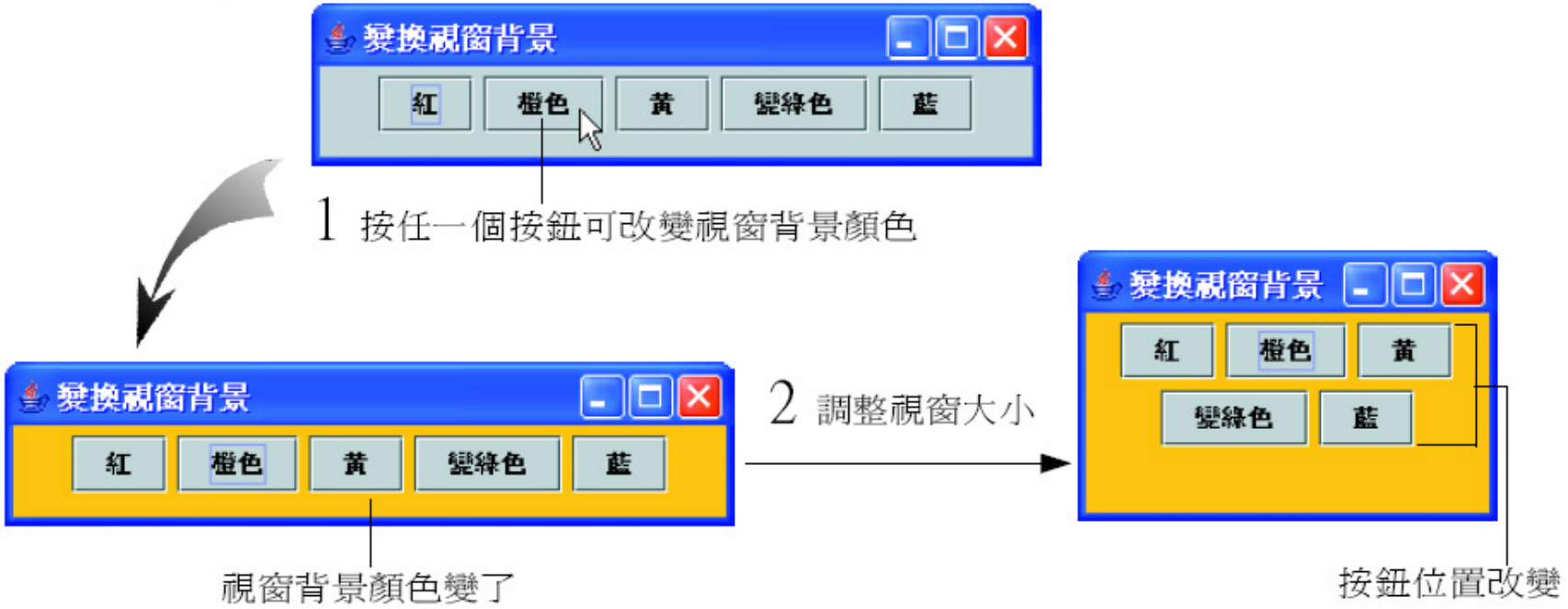
FlowLayout 配置管理員

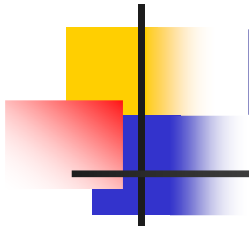
```
5  import java.awt.*;
6  import java.awt.event.*;
7  import javax.swing.*;
8
9  public class ChangeColor extends JPanel implements ActionListener{
10     JButton red = new JButton("紅色");
11     JButton orange = new JButton("橙色");
12     JButton yellow = new JButton("黃");
13     JButton green = new JButton("綠");
14     JButton blue = new JButton("藍");
15     public static void main(String[] argv){
16         ChangeColor cc = new ChangeColor();
17         JFrame f = new JFrame("變換視窗底色");
18         f.getContentPane().add(cc);
19         f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20         f.setSize(360, 80);
21         f.setVisible(true);
22     }
23     public ChangeColor(){
24         add(red);
25         add(orange);
26         add(yellow);
27         add(green);
28         add(blue);
29
30         red.addActionListener(this);
31         orange.addActionListener(this);
32         yellow.addActionListener(this);
33         green.addActionListener(this);
34         blue.addActionListener(this);
35     }
```

```
36     public void actionPerformed(ActionEvent e){
37         JButton s = (JButton) e.getSource();
38         if(s == red){
39             setBackground(Color.red);
40         }else if( s == orange){
41             setBackground(Color.orange);
42         }else if(s == yellow){
43             setBackground(Color.yellow);
44         }else if(s == green){
45             setBackground(Color.green);
46         }else{
47             setBackground(Color.blue);
48         }
49     }
50 }
```

FlowLayout 配置管理員

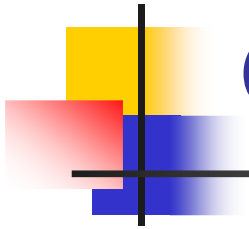
執行結果





FlowLayout 配置管理員

- 調整視窗大小時, 就能察覺 FlowLayout 的動態配置效果：當視窗夠寬時, 按鈕都會放在同一列；當視窗變窄時, 按鈕就會被擠到下一行顯示。



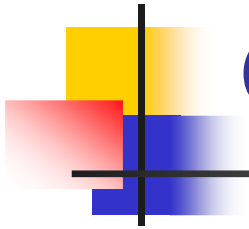
GridLayout 配置管理員

- GridLayout 配置管理員和 BorderLayout 有個相似之處, 就是它們都提供**制式**的元件配置方式
- 但 GridLayout 配置管理員的**制式排列**並不是由 Java 預先定死的, 而是由我們自行指定將視窗切割成幾個部份, 接著就能將 GUI 元件擺到指定的位置上

GridLayout 配置管理員

- GridLayout 配置管理員是將容器內部切割成整齊的格子 (就像試算表的格子一樣), 放置元件時, 就是放到每個固定住的格子上面。但格子的多寡則是由我們自行指定, 在建立 GridLayout 配置管理員物件時, 需在建構方法中指定格子的行數與列數。

```
GridLayout(int rows, int cols)   // 縱向分成 rows 格
                                 // 橫向分成 cols 格
GridLayout(int rows, int cols, int hgap, int vgap)
                                  // 縱向格子間距爲 hgap 像素 (pixel)
                                 // 橫向格子間距爲 vgap 像素
```



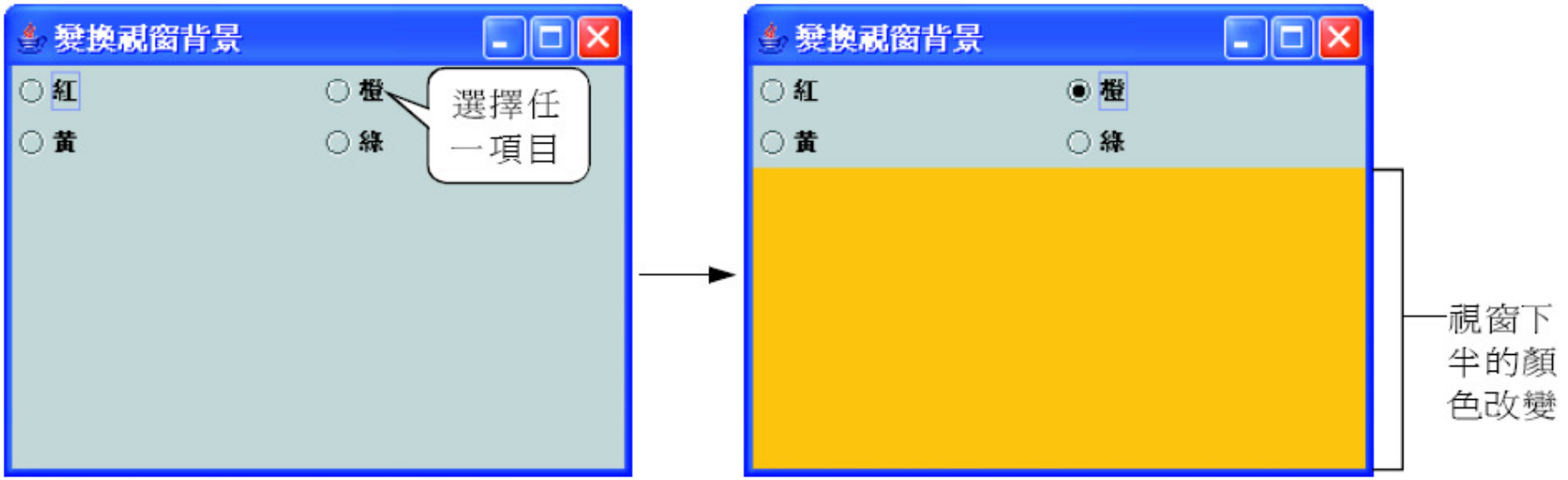
GridLayout 配置管理員

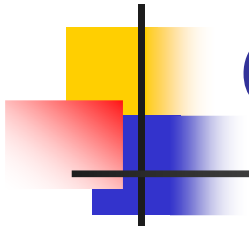
- 沿續前一個改變視窗背景顏色的範例, 將程式改成在 JFrame 中放入使用 GridLayout 配置管理員的 JPanel 容器, 然後再將數個元件放入 JPanel 中依序排列

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

public class GridChangeColor implements ItemListener{
    JFrame mainFrame = new JFrame("變色龍");
    JPanel mainPanel = new JPanel();
    String[] colorName = {"紅", "橙", "黃", "綠"};
    JRadioButton[] colorButton = new JRadioButton[4];
    public static void main(String[] argv){
        GridChangeColor gb = new GridChangeColor();
        gb.init();
    }
    public void init(){
        Container contentPane = mainFrame.getContentPane();
        contentPane.add(mainPanel, "Center");
        JPanel up = new JPanel();
        contentPane.add(up, "North");
        up.setLayout(new GridLayout(2,2));
        ButtonGroup g = new ButtonGroup();
        for(int i=0;i<colorName.length;i++){
            colorButton[i] = new JRadioButton(colorName[i]);
            up.add(colorButton[i]);
            g.add(colorButton[i]);
            colorButton[i].addItemListener(this);
        }
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainFrame.setSize(320, 240);
        mainFrame.setVisible(true);
    }

    public void itemStateChanged(ItemEvent e){
        if(e.getStateChange() ==  ItemEvent.SELECTED){
            JRadioButton s = (JRadioButton) e.getSource();
            if(s == colorButton[0]){
                mainPanel.setBackground(Color.RED);
            }else if(s == colorButton[1]){
                mainPanel.setBackground(Color.ORANGE);
            }else if(s == colorButton[2]){
                mainPanel.setBackground(Color.YELLOW);
            }else{
                mainPanel.setBackground(Color.GREEN);
            }
        }
    }
}
```

GridLayout 配置管理員

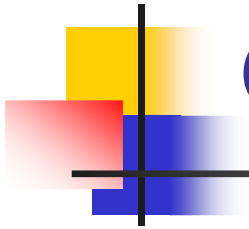
執行結果





GridLayout 配置管理員

- 試著調整視窗大小, 您會發現不管如何調整, GridLayout 配置管理員固定會將其區域分成指定的格子數 (本例為 2 x 2), 所以置入該格的元件也會隨之放大或縮小
- 在上個範例中還用到另一類型的事件處理介面 ItemListener, 凡是可以被選取的元件, 都會實作 ItemSelectable 介面, 並可呼叫其 addItemListener() 方法來加入被選取/ 取消選取事件的傾聽者。



GridLayout 配置管理員

- ItemListener 只有一個事件處理方法 itemStateChanged(), 在此方法中, 可透過 ItemEvent 事件物件呼叫 getStateChange() 方法取得元件是被選取 (ItemEvent.SELECTED) 或被取消選取 (ItemEvent.DESELECTED)。