

Swing (四)

人工智慧與無線感應設備開發專班

湜憶電腦知訊顧問股份有限公司

馬傳義

版面配置

- Java在視窗程式設計上提供「版面配置管理者」，讓程式設計師可選擇各種不同配置方式呈現物件的位置。
- 前面的視窗程式範例中，有一個setLayout(null)敘述，其實那是宣告視窗不套用任何的「版面配置」。
 - ◆ 當設定不使用任何的「版面配置」時，元件（物件）的位置與大小就必須利用setLocation()、setSize()、setBounds()等方法來決定。
- 當程式較為複雜時，要一一為這些元件（物件）規劃擺放位置，在佈局上可預見的會是一件很吃力的工作。

版面配置

例如：

一個視窗（JFrame）內可能會有好幾個面板（JPanel）或Swing元件。


■ 若希望元件（物件）的位置與大小能隨視窗大小改變而自動彈性改變，就要套用「版面配置」來解決。

■ 要套用「版面配置」時，程式一開始必須先匯入 `java.awt.*` 套件：

```
import java.awt.*;
```

■ 再用 `setLayout()` 來使用版面配置。

版面配置

 語法：

```
setLayout(new LayoutManager)
```

 說明：

- ◆ 指定視窗的版面配置方式。
- ◆ 如果要使用系統提供的版面配置，則需要使用關鍵字「new」，再使用系統提供的版面配置方式。
 - BorderLayout（邊緣式版面配置）
 - FlowLayout（流程式版面配置）
 - GridLayout（格線式版面配置）

BorderLayout

- 邊緣式版面配置方式會將視窗分割為東、西、南、北和中五個方向來配置物件，而且最多只能擺放5個物件。
- 若視窗的大小被調整，視窗中的物件會自動調整大小。
- BorderLayout的建構子：
 - ◆ BorderLayout()
 - 建立BorderLayout的版面配置。
 - 物件之間預設的間距為0像素。
 - ◆ BorderLayout(int hgap, int vgap)
 - 建立BorderLayout版面配置的同時，指定物件間的水平與垂直間距。

BorderLayout

BorderLayout常用方法：

◆ pack()

- 調整視窗大小，以配合視窗中，物件的大小與版面配置。

◆ add(Component comp, Object constraints)

- Component comp是指已建立的Swing元件。

例如：

按鈕（JButton）。

- Object constraints是指物件放置的位置，設定值有：
 - ◆ BorderLayout.EAST（東）、
 - ◆ BorderLayout.SOUTH（南）、
 - ◆ BorderLayout.WEST（西）、
 - ◆ BorderLayout.NORTH（北）、
 - ◆ BorderLayout.CENTER（中）。

BorderLayout

◆ 程式：

```
package CH09_20;
```

```
import java.awt.*;
```

```
import javax.swing.*;
```

```
class MyJFrame extends JFrame
```

```
{
```

```
    MyJFrame()
```

```
{
```

```
    JFrame frame = new JFrame();
```

```
    setTitle("BorderLayout");
```

```
    setLayout(new BorderLayout());
```

```
    setLocation(100,100);
```

```
    setVisible(true);
```

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

BorderLayout

```
JButton btnNorth = new JButton("北");  
add(btnNorth, BorderLayout.NORTH);
```

```
JButton btnWest = new JButton("西");  
add(btnWest, BorderLayout.WEST);
```

```
JButton btnEast = new JButton("東");  
add(btnEast, BorderLayout.EAST);
```

```
JButton btnCenter = new JButton("中");  
add(btnCenter, BorderLayout.CENTER);
```

```
JButton btnSouth = new JButton("南");  
add(btnSouth, BorderLayout.SOUTH);
```


BorderLayout

```
        pack();
    }
}

public class CH09_20
{
    public static void main(String[] args)
    {
        new MyJFrame();
    }
}
```

BorderLayout

```
1 package CH09_20;
2
3 import java.awt.*;
4 import javax.swing.*;
5
6 class MyJFrame extends JFrame
7 {
8     MyJFrame()
9     {
10         JFrame frame = new JFrame();
11         setTitle("BorderLayout");
12         setLayout(new BorderLayout());
13         setLocation(100,100);
14         setVisible(true);
15         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16
17         JButton btnNorth = new JButton("北");
18         add(btnNorth, BorderLayout.NORTH);
19
20         JButton btnWest = new JButton("西");
21         add(btnWest, BorderLayout.WEST);
22
23         JButton btnEast = new JButton("東");
24         add(btnEast, BorderLayout.EAST);
25
26         JButton btnCenter = new JButton("中");
```

BorderLayout

```
27     add(btnCenter, BorderLayout.CENTER);
28
29     JButton btnSouth = new JButton("南");
30     add(btnSouth, BorderLayout.SOUTH);
31
32     pack();
33 }
34 }
35
36 public class CH09_20
37 {
38     public static void main(String[] args)
39     {
40         new MyJFrame();
41     }
42 }
```

◆ 執行結果：



BorderLayout

◆ 說明：

- 行01：
 - ◆ 定義「套件（package）」。
- 行03：
 - ◆ 載入「java.awt.*」套件。
- 行04：
 - ◆ 載入「javax.swing.*」套件。
- 行06~行34：
 - ◆ 建立繼承自「JFrame」父類別的「MyJFrame」子類別。
 - ◆ 行08~行33：
 - ◆ 宣告類別「建構子」`MyJFrame()`。
 - 行10：
 - ◆ 利用「JFrame()」的「建構子」，建立frame物件。

BorderLayout

行11：

- ◆ 設定視窗標題。

行12：

- ◆ 使用「`BorderLayout()`」建構子來建立「邊緣式版面配置」。

注意：

- ◆ 若改用「`BorderLayout(int hgap, int vgap)`」建構子來建立「邊緣式版面配置」時，可調整版面中，各物件的水平及垂直的距離。

例如：

```
BorderLayout(30, 40);
```

行13：

- ◆ 設定視窗的位置。

行14：

- ◆ 設定視窗可以顯示在螢幕上。

BorderLayout

行15：

- ◆ 設定當點按視窗右上角關閉鈕時，一併結束應用程式。

行17：

- ◆ 利用「JButton(String title)」的「建構子」，建立 btnNorth 物件。

行18：

- ◆ 將 btnNorth 物件，增加到 frame 物件中。

行20：

- ◆ 利用「JButton(String title)」的「建構子」，建立 btnWest 物件。

行21：

- ◆ 將 btnWest 物件，增加到 frame 物件中。

BorderLayout

行23：

- ◆ 利用「JButton(String title)」的「建構子」，建立 btnEast物件。

行24：

- ◆ 將btnEast物件，增加到frame物件中。

行26：

- ◆ 利用「JButton(String title)」的「建構子」，建立 btnCenter物件。

行27：

- ◆ 將btnCenter物件，增加到frame物件中。

行29：

- ◆ 利用「JButton(String title)」的「建構子」，建立 btnSouth物件。

行30：

- ◆ 將btnSouth物件，增加到frame物件中。

BorderLayout

行32：

- ◆ 調整視窗大小，以配合視窗中的按鈕及版面配置。

• 行40：

- ◆ 建立MyJFrame物件。

注意：

此為匿名物件，故在main()（即主程式）中，無法修改視窗大小、位置、...、等相關屬性。

FlowLayout

- 流程式版面配置是內定的「版面配置」。
- 它對物件擺放方式是「由左而右，由上而下」。
- 若視窗的寬度一行排不下時，其餘的物件會自動排到下一行。
- 每個物件間不論垂直或水平都有一點小間距。
- FlowLayout的建構子：

- ◆ FlowLayout()

- 建立FlowLayout的版面配置。
- 物件排列時的對齊方式預設為置中對齊。
- 物件之間的水平間距及垂直間距皆預設為5像素。

FlowLayout

◆ FlowLayout(int align)

- 建立FlowLayout版面配置的同時，指定物件對齊的方式，有：
 - ◆ FlowLayout.LEFT（靠左對齊）、
 - ◆ FlowLayout.CENTER（置中對齊，預設值）、
 - ◆ FlowLayout.RIGHT（靠右對齊）。

◆ FlowLayout(int align, int hgap, int vgap)

- 建立FlowLayout版面配置的同時，指定物件的對齊方式、物件間的水平間距與垂直間距。

■ FlowLayout常用方法：

◆ pack()

- 調整視窗大小，以配合視窗中，物件的大小與版面配置。

FlowLayout

◆ add(Component comp)

- Component comp是指已建立的Swing元件。

例如：

按鈕（JButton）。

FlowLayout

◆ 程式：

```
package CH09_21;
```

```
import java.awt.*;
```

```
import javax.swing.*;
```

```
class MyJFrame extends JFrame
```

```
{
```

```
    MyJFrame()
```

```
{
```

```
    JFrame frame = new JFrame();
```

```
    setTitle("FlowLayout");
```

```
    setLayout(new FlowLayout());
```

```
    setLocation(100,100);
```

```
    setVisible(true);
```

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

FlowLayout

```
JButton btn1 = new JButton("按鈕1");  
add(btn1);
```

```
JButton btn2 = new JButton("按鈕2");  
add(btn2);
```

```
JButton btn3 = new JButton("按鈕3");  
add(btn3);
```

```
JButton btn4 = new JButton("按鈕4");  
add(btn4);
```

```
JButton btn5 = new JButton("按鈕5");  
add(btn5);
```

FlowLayout

```
        pack();
    }
}

public class CH09_21
{
    public static void main(String[] args)
    {
        new MyJFrame();
    }
}
```

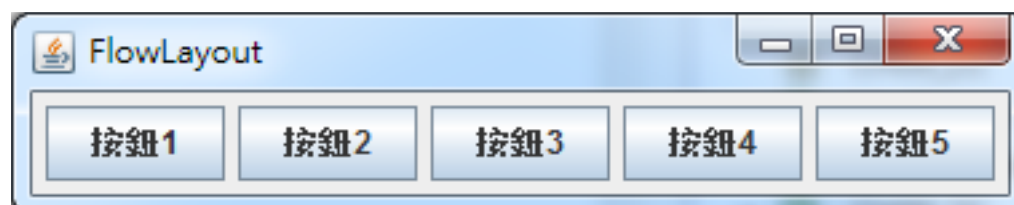
FlowLayout

```
1 package CH09_21;
2
3 import java.awt.*;
4 import javax.swing.*;
5
6 class MyJFrame extends JFrame
7 {
8     MyJFrame()
9     {
10         JFrame frame = new JFrame();
11         setTitle("FlowLayout");
12         setLayout(new FlowLayout());
13         setLocation(100,100);
14         setVisible(true);
15         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16
17         JButton btn1 = new JButton("按鈕1");
18         add(btn1);
19
20         JButton btn2 = new JButton("按鈕2");
21         add(btn2);
22
23         JButton btn3 = new JButton("按鈕3");
24         add(btn3);
25
26         JButton btn4 = new JButton("按鈕4");
```

FlowLayout

```
27     add(btn4);
28
29     JButton btn5 = new JButton("按鈕5");
30     add(btn5);
31
32     pack();
33 }
34 }
35
36 public class CH09_21
37 {
38     public static void main(String[] args)
39     {
40         new MyJFrame();
41     }
42 }
43
```

◆ 執行結果：



FlowLayout

◆ 說明：

- 行01：
 - ◆ 定義「套件（package）」。
- 行03：
 - ◆ 載入「java.awt.*」套件。
- 行04：
 - ◆ 載入「javax.swing.*」套件。
- 行06~行34：
 - ◆ 建立繼承自「JFrame」父類別的「MyJFrame」子類別。
 - ◆ 行08~行33：
 - ◆ 宣告類別「建構子」`MyJFrame()`。
 - 行10：
 - ◆ 利用「JFrame()」的「建構子」，建立frame物件。

FlowLayout

行11：

- ◆ 設定視窗標題。

行12：

- ◆ 使用「FlowLayout ()」建構子來建立「流程式版面配置」。

注意：

- ◆ 若改用「FlowLayout(int align)」建構子來建立「流程式版面配置」時，可調整物件的對齊方式。

例如：

```
FlowLayout(FlowLayout.LEFT);
```

- ◆ 若改用「FlowLayout(int align, int hgap, int vgap)」建構子來建立「流程式版面配置」時，可調整物件的對齊方式及各物件的水平及垂直的距離。

例如：

```
FlowLayout(FlowLayout.LEFT, 30, 40);
```

FlowLayout

行13：

- ◆ 設定視窗的位置。

行14：

- ◆ 設定視窗可以顯示在螢幕上。

行15：

- ◆ 設定當點按視窗右上角關閉鈕時，一併結束應用程式。

行17：

- ◆ 利用「JButton(String title)」的「建構子」，建立btn1物件。

行18：

- ◆ 將btn1物件，增加到frame物件中。

行20：

- ◆ 利用「JButton(String title)」的「建構子」，建立btn2物件。

FlowLayout

行21：

- ◆ 將btn2物件，增加到frame物件中。

行23：

- ◆ 利用「JButton(String title)」的「建構子」，建立btn3物件。

行24：

- ◆ 將btn3物件，增加到frame物件中。

行26：

- ◆ 利用「JButton(String title)」的「建構子」，建立btn4物件。

行27：

- ◆ 將btn4物件，增加到frame物件中。

行29：

- ◆ 利用「JButton(String title)」的「建構子」，建立btn5物件。

FlowLayout

行30：

- ◆ 將btn5物件，增加到frame物件中。

行32：

- ◆ 調整視窗大小，以配合視窗中的按鈕及版面配置。

- 行40：

- ◆ 建立MyJFrame物件。

注意：

此為匿名物件，故在main()（即主程式）中，無法修改視窗大小、位置、...、等相關屬性。

GridLayout

- 格線式版面是依據設定的列數與欄數，將視窗等分為 列x欄 的格子數。
- 一個格子放置一個物件。
- 放置的順序是「由左而右」、「由上而下」。
- GridLayout的建構子：
 - ◆ GridLayout()
 - 建立GridLayout的版面配置。
 - 物件之間預設的間距為0像素。
 - ◆ GridLayout(int row, int col)
 - 建立GridLayout版面配置的同時，指定列數與欄數。

GridLayout

◆ GridLayout(int row, int col, int hgap, int vgap)

- 建立GridLayout版面配置的同時，指定列數、欄數、物件間的水平間距與垂直間距。

■ GridLayout常用方法：

◆ pack()

- 調整視窗大小，以配合視窗中，物件的大小與版面配置。

◆ add(Component comp)

- Component comp是指已建立的Swing元件。

例如：

按鈕（JButton）。

GridLayout

◆ 程式：

```
package CH09_22;
```

```
import java.awt.*;
```

```
import javax.swing.*;
```

```
class MyJFrame extends JFrame
```

```
{
```

```
    MyJFrame()
```

```
{
```

```
    JFrame frame = new JFrame();
```

```
    setTitle("GridLayout");
```

```
    setLayout(new GridLayout());
```

```
    setLocation(100,100);
```

```
    setVisible(true);
```

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


GridLayout

```
JButton btn1 = new JButton("按鈕1");  
add(btn1);
```

```
JButton btn2 = new JButton("按鈕2");  
add(btn2);
```

```
JButton btn3 = new JButton("按鈕3");  
add(btn3);
```

```
JButton btn4 = new JButton("按鈕4");  
add(btn4);
```

```
JButton btn5 = new JButton("按鈕5");  
add(btn5);
```

GridLayout

```
        pack();  
    }  
}  
  
public class CH09_22  
{  
    public static void main(String[] args)  
    {  
        new MyJFrame();  
    }  
}
```

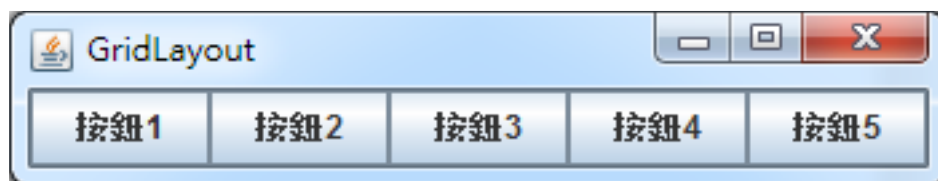
GridLayout

```
1 package CH09_22;
2
3 import java.awt.*;
4 import javax.swing.*;
5
6 class MyJFrame extends JFrame
7 {
8     MyJFrame()
9     {
10         JFrame frame = new JFrame();
11         setTitle("GridLayout");
12         setLayout(new GridLayout());
13         setLocation(100, 100);
14         setVisible(true);
15         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16
17         JButton btn1 = new JButton("按鈕1");
18         add(btn1);
19
20         JButton btn2 = new JButton("按鈕2");
21         add(btn2);
22
23         JButton btn3 = new JButton("按鈕3");
24         add(btn3);
25
26         JButton btn4 = new JButton("按鈕4");
```

GridLayout

```
27     add(btn4);
28
29     JButton btn5 = new JButton("按鈕5");
30     add(btn5);
31
32     pack();
33 }
34 }
35
36 public class CH09_22
37 {
38     public static void main(String[] args)
39     {
40         new MyJFrame();
41     }
42 }
```

◆ 執行結果：



GridLayout

◆ 說明：

- 行01：
 - ◆ 定義「套件（package）」。
- 行03：
 - ◆ 載入「java.awt.*」套件。
- 行04：
 - ◆ 載入「javax.swing.*」套件。
- 行06~行34：
 - ◆ 建立繼承自「JFrame」父類別的「MyJFrame」子類別。
 - ◆ 行08~行33：
 - ◆ 宣告類別「建構子」`MyJFrame()`。
 - 行10：
 - ◆ 利用「JFrame()」的「建構子」，建立frame物件。

GridLayout

行11：

- ◆ 設定視窗標題。

行12：

- ◆ 使用「GridLayout ()」建構子來建立「格線式版面配置」。

注意：

- ◆ 若改用「GridLayout(int row, int col)」建構子來建立「格線式版面配置」時，可設定列數及欄數。

例如：

```
GridLayout(2, 3);
```

- ◆ 若改用「GridLayout(int row, int col, int hgap, int vgap)」建構子來建立「格線式版面配置」時，可設定列數、欄數、各物件的水平及垂直的距離。

例如：

```
GridLayout(2, 3, 30, 40);
```

GridLayout

行13：

- ◆ 設定視窗的位置。

行14：

- ◆ 設定視窗可以顯示在螢幕上。

行15：

- ◆ 設定當點按視窗右上角關閉鈕時，一併結束應用程式。

行17：

- ◆ 利用「JButton(String title)」的「建構子」，建立btn1物件。

行18：

- ◆ 將btn1物件，增加到frame物件中。

行20：

- ◆ 利用「JButton(String title)」的「建構子」，建立btn2物件。

GridLayout

行21：

- ◆ 將btn2物件，增加到frame物件中。

行23：

- ◆ 利用「JButton(String title)」的「建構子」，建立btn3物件。

行24：

- ◆ 將btn3物件，增加到frame物件中。

行26：

- ◆ 利用「JButton(String title)」的「建構子」，建立btn4物件。

行27：

- ◆ 將btn4物件，增加到frame物件中。

行29：

- ◆ 利用「JButton(String title)」的「建構子」，建立btn5物件。

GridLayout

行30：

- ◆ 將btn5物件，增加到frame物件中。

行32：

- ◆ 調整視窗大小，以配合視窗中的按鈕及版面配置。

- 行40：

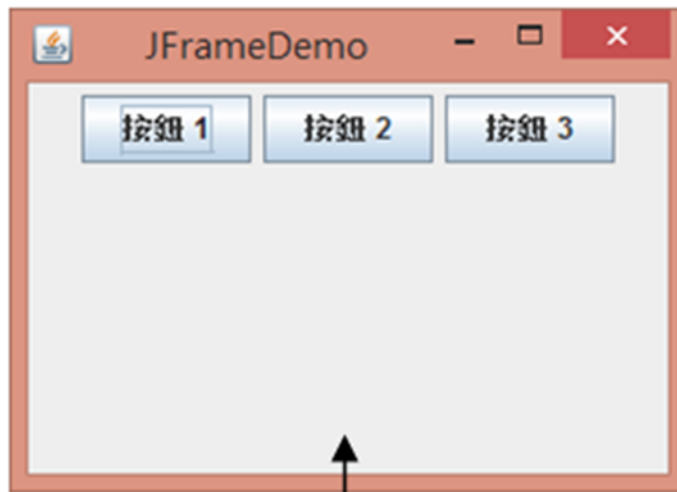
- ◆ 建立MyJFrame物件。

注意：

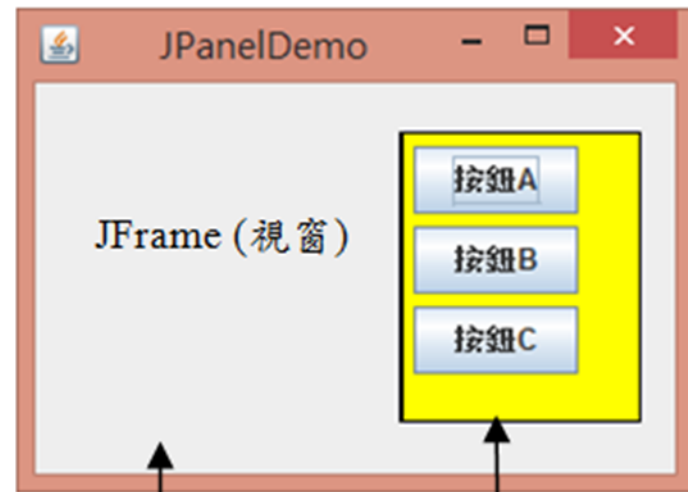
此為匿名物件，故在main()（即主程式）中，無法修改視窗大小、位置、...、等相關屬性。

JPanel

- JPanel（面板）的功能與JFrame（視窗）一樣，都是可以放置Swing元件的容器。
- ◆ JFrame（視窗）是個大容器
- ◆ JPanel（面板）是放在大容器裡面的小容器。



JFrame 視窗



JFrame 視窗

JPanel 面板

JPanel

JPanel的建構子：

◆ JPanel()

- 建立JPanel面板物件。
- 面板物件的版面配置方式另由setLayout方法設定。

◆ JPanel(LayoutManager layout)

- 建立JPanel面板物件。
- 引數用來指定面板的版面配置方式。

常用的方法：

◆ setLayout(LayoutManager layout)

- 設定面板物件的版面配置方式。

JPanel

◆ setBounds(int x, int y, int width, int height)

- 指定面板物件在母件（如視窗）位置大小。
- 此方法要在母件的版面配置設為setLayout(null) 才有效。

◆ setBackground(Color rgb)

- 若沒設定面板物件背景色預設透明（即呈現母件的背景色）。
- 設定面板物件的背景色。

屬性名稱	代表顏色
Color.black	黑色
Color.blue	藍色
Color.cyan	青綠色
Color.darkGray	深灰色
Color.gray	灰色

屬性名稱	代表顏色
Color.green	綠色
Color.lightGray	淺灰色
Color.magenta	洋紅色
Color.orange	橘色
Color.pink	粉紅色

屬性名稱	代表顏色
Color.red	紅色
Color.white	白色
Color.yellow	黃色

JPanel

- ◆ `setBorder(BorderFactory.createLineBorder(Color rgb))`
 - 設定面板物件的邊框線條顏色。
 - 線條厚度預設為1像素。
 - 若沒有設定，則預設為透明。
- ◆ `setBorder(BorderFactory.createLineBorder(Color rgb, int thick))`
 - 設定面板物件的邊框線條顏色與厚度。

JPanel

◆ 程式：

```
package CH09_23;
```

```
import java.awt.*;
```

```
import javax.swing.*;
```

```
class JPanel extends JFrame
```

```
{
```

```
    JPanel()
```

```
    {
```

```
        setTitle("JPanel");
```

```
        setLayout(null);
```

```
        setBounds(100, 100, 300, 200);
```

```
        setVisible(true);
```

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

JPanel

```
JPanel pan = new JPanel();  
pan.setBounds(150, 20, 100, 107);  
pan.setBackground(Color.yellow);  
pan.setBorder(BorderFactory.createLineBorder(Color.black));  
add(pan);
```

```
String st[] = { "按鈕A", "按鈕B", "按鈕C" };  
JButton btn[] = new JButton[st.length];  
for (int i = 0; i < st.length; i++)  
{  
    btn[i] = new JButton(st[i]);  
    pan.add(btn[i]);  
}  
}  
}
```

JPanel

```
public class CH09_23
{
    public static void main(String[] args)
    {
        new MyJPanel();
    }
}
```

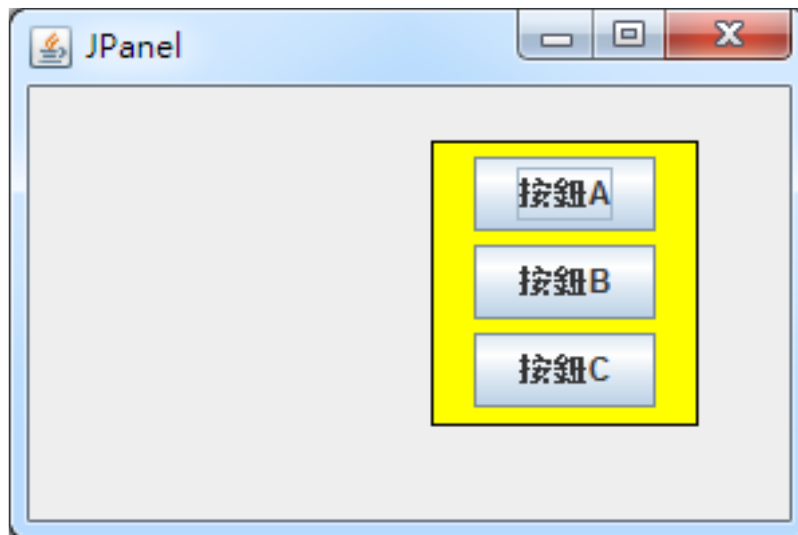

JPanel

```
1 package CH09_23;
2
3 import java.awt.*;
4 import javax.swing.*;
5
6 class MyJPanel extends JFrame
7 {
8     MyJPanel()
9     {
10         setTitle("JPanel");
11         setLayout(null);
12         setBounds(100, 100, 300, 200);
13         setVisible(true);
14         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15
16         JPanel pan = new JPanel();
17         pan.setBounds(150, 20, 100, 107);
18         pan.setBackground(Color.yellow);
19         pan.setBorder(BorderFactory.createLineBorder(Color.black));
20         add(pan);
21
22         String st[] = { "按鈕A", "按鈕B", "按鈕C" };
23         JButton btn[] = new JButton[st.length];
24         for (int i = 0; i < st.length; i++)
25         {
26             btn[i] = new JButton(st[i]);
```

JPanel

```
27     pan.add(btn[i]);
28     }
29 }
30 }
31
32 public class CH09_23
33 {
34     public static void main(String[] args)
35     {
36         new JPanel();
37     }
38 }
```

◆ 執行結果：



JPanel

◆ 說明：

- 行01：
 - ◆ 定義「套件（package）」。
- 行03：
 - ◆ 載入「java.awt.*」套件。
- 行04：
 - ◆ 載入「javax.swing.*」套件。
- 行06~行30：
 - ◆ 建立繼承自「JFrame」父類別的「MyJPanel」子類別。
 - ◆ 行08~行29：
 - ◆ 宣告類別「建構子」 「MyJPanel()」。
 - 行10：
 - ◆ 設定視窗標題。

JPanel

行11：

- ◆ 設定視窗不使用「版面配置」。

行12：

- ◆ 設定視窗的位置及大小。

行13：

- ◆ 設定視窗可以顯示在螢幕上。

行14：

- ◆ 設定當點按視窗右上角關閉鈕時，一併結束應用程式。

行16：

- ◆ 利用「JPanel()」的「建構子」，建立面板物件。

行17：

- ◆ 設定面板的位置及大小。

行18：

- ◆ 設定面板的背景色。

JPanel

行19：

- ◆ 設定面板的邊框顏色。

行20：

- ◆ 將面板物件，增加到視窗中。

行22：

- ◆ 建立字串陣列st[]，並指定陣列內容。

行23：

- ◆ 利用「JButton(String title)」的「建構子」，建立陣列btn[]物件。

行24～行 28：

- ◆ 利用for迴圈，設定btn[]文字，並將btn[]物件，增加到JPanel中。

行24：

設定for迴圈。

JPanel

行26：

設定btn[]文字。

行27：

將btn[]物件，增加到JPanel中。

- 行36：

- ◆ 建立MyJPanel物件。

注意：

此為匿名物件，故在main()（即主程式）中，無法修改視窗大小、位置、...、等相關屬性。

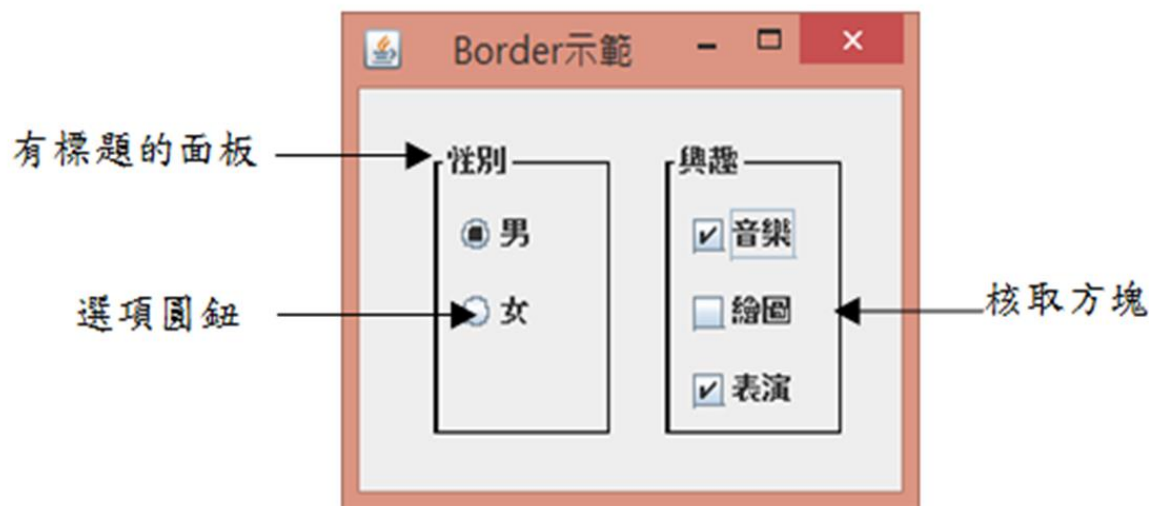
JPanel

「面板」可分門別類放同性質元件。

例如：

如把「同組」的「選項圓鈕」或「核取方塊」放在一起。

為能彰顯集中放置元件歸屬，在「面板」左上角，可以有標題文字。



JPanel

■ 若要設定為「面板」的「標題」，有三個步驟：

◆ 匯入javax.swing.border.*套件：

```
import javax.swing.border.*;
```

◆ 建立框線物件：

```
Border 物件名稱 = BorderFactory.createLineBorder(  
Color rgb);
```

◆ 將已建立的框線物件，套用到面板物件中，並設定面板物件的標題。

```
面板物件名稱.setBorder(BorderFactory.  
createTitledBorder(Border物件名稱, String title));
```


JPanel

◆ 程式：

```
package CH09_24;
```

```
import java.awt.*;
```

```
import javax.swing.*;
```

```
import javax.swing.border.*;
```

```
class MyJPanel extends JFrame
```

```
{
```

```
    MyJPanel()
```

```
{
```

```
    setTitle("BorderDemo");
```

```
    setLayout(null);
```

```
    setBounds(100, 100, 300, 200);
```

```
    setVisible(true);
```

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

JPanel

```
JPanel pan = new JPanel();  
pan.setBounds(150, 20, 100, 128);
```

```
Border lineB = BorderFactory.createLineBorder(Color.black);  
pan.setBorder(BorderFactory.createTitledBorder(lineB, "框架"));
```

```
    add(pan);  
}  
}
```

```
public class CH09_24  
{  
    public static void main(String[] args)  
    {  
        new MyJPanel();  
    }  
}
```

JPanel

```
}  
}
```

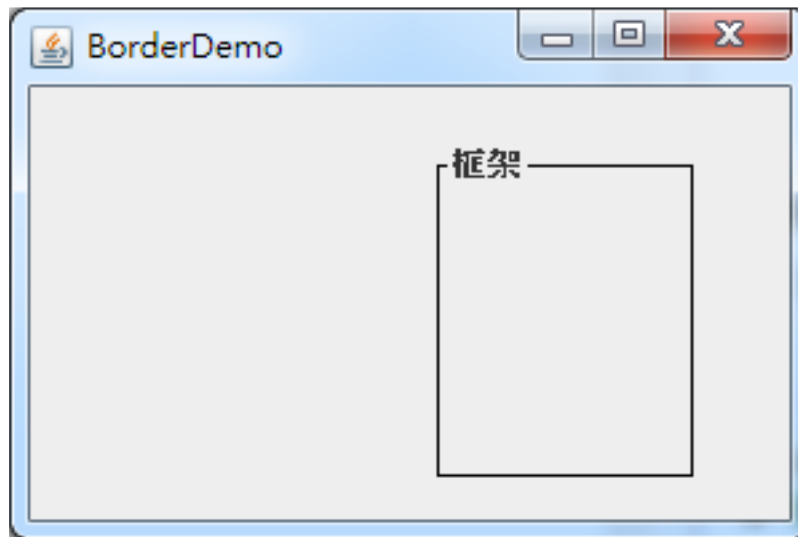
JPanel

```
1 package CH09_24;
2
3 import java.awt.*;
4 import javax.swing.*;
5 import javax.swing.border.*;
6
7 class MyJPanel extends JFrame
8 {
9     MyJPanel()
10    {
11        setTitle("BorderDemo");
12        setLayout(null);
13        setBounds(100, 100, 300, 200);
14        setVisible(true);
15        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16
17        JPanel pan = new JPanel();
18        pan.setBounds(150, 20, 100, 128);
19
20        Border lineB = BorderFactory.createLineBorder(Color.black);
21        pan.setBorder(BorderFactory.createTitledBorder(lineB, "框架"));
22
23        add(pan);
24    }
25 }
26
```

JPanel

```
27 public class CH09_24
28 {
29     public static void main(String[] args)
30     {
31         new MyJPanel();
32     }
33 }
```

◆ 執行結果：



JPanel

◆ 說明：

- 行01：
 - ◆ 定義「套件（package）」。
- 行03：
 - ◆ 載入「java.awt.*」套件。
- 行04：
 - ◆ 載入「javax.swing.*」套件。
- 行05：
 - ◆ 載入「javax.swing.border.*」套件。
- 行07~行25：
 - ◆ 建立繼承自「JFrame」父類別的「MyJPanel」子類別。
 - ◆ 行09~行24：
 - ◆ 宣告類別「建構子」`MyJPanel()`。

行11：

- ◆ 設定視窗標題。

行12：

- ◆ 設定視窗不使用「版面配置」。

行13：

- ◆ 設定視窗的位置及大小。

行14：

- ◆ 設定視窗可以顯示在螢幕上。

行15：

- ◆ 設定當點按視窗右上角關閉鈕時，一併結束應用程式。

行17：

- ◆ 利用「JPanel()」的「建構子」，建立面板物件。

行18：

- ◆ 設定面板的位置及大小。

JPanel

行20：

- ◆ 建立框線物件，並指定邊框顏色。

行21：

- ◆ 面板物件套用已建立的框線物件，並設定面板物件的標題文字。

行23：

- ◆ 將面板物件，增加到視窗中。

- 行31：

- ◆ 建立MyJPanel物件。

注意：

此為匿名物件，故在main()（即主程式）中，無法修改視窗大小、位置、...、等相關屬性。


對話框

 對話框是一種顯示訊息或詢問問題的小視窗，在JOptionPane類別中共四種對話框，分別是：

- ◆ 訊息對話框、
- ◆ 確認對話框、
- ◆ 輸入對話框、
- ◆ 選項對話框。

訊息對話框

■ 用來顯示資訊的對話框。

■ 當使用者看到訊息後，點按唯一按鈕 ，即可關閉對話框。

■ 其多載方法如下：

◆ `showMessageDialog(Component parent, Object msg)`



- `Component parent`：

- ◆ 用來指定顯示對話框的容器。
- ◆ 若設定值為`null`或`this`，則對話框會被顯示在螢幕的中央。

- `Object msg`：

- ◆ 用來設定在對話框中所要顯示的訊息，它可以是文字字串，也可以是圖示或其它物件。

訊息對話框

- ◆ `showMessageDialog(Component parent, Object msg, String title, int msgType)`
 - String title :
 - ◆ 用來指定對話框的標題列文字。
 - msgType :
 - ◆ 用來設定代表對話框內容的警訊圖示，有5種設定值：
 - ◆ `JOptionPane.QUESTION_MESSAGE` :
顯示問題圖示  。
 - ◆ `JOptionPane.WARNING_MESSAGE` :
顯示警告圖示  。
 - ◆ `JOptionPane.ERROR_MESSAGE` :
顯示錯誤圖示  。
 - ◆ `JOptionPane.INFORMATION_MESSAGE` (預設值) :
顯示資訊圖示  。

訊息對話框

- ◆ JOptionPane.PLAIN_MESSAGE :

- 不顯示警訊圖示。

- ◆ showMessageDialog(Component parent, Object msg, String title, int msgType, ImageIcon icon)

- ImageIcon icon :

- ◆ 不使用引數msgType指定的圖示，而要顯示自訂的圖示。
則 msgType引數值要設為0。

- ◆ 若不顯示自訂按鈕，則本引數值要設為null。

訊息對話框

◆ 程式：

```
package CH09_25;
```

```
import javax.swing.*;
```

```
class MyMessageDialog extends JFrame
```

```
{
```

```
    MyMessageDialog()
```

```
{
```

```
    JOptionPane.showMessageDialog(null, "該休息了", "提醒",
```

```
        JOptionPane.INFORMATION_MESSAGE);
```

```
}
```

```
}
```

```
public class CH09_25
```

```
{
```

訊息對話框

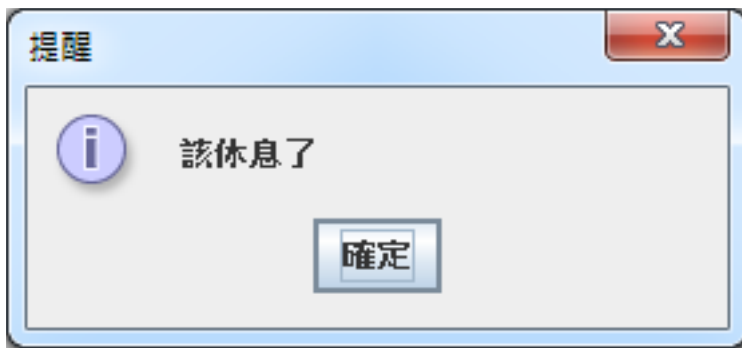
```
public static void main(String[] args)
{
    new MyMessageDialog();
}
```

訊息對話框

```
1 package CH09_25;
2
3 import javax.swing.*;
4
5 class MyMessageDialog extends JFrame
6 {
7     MyMessageDialog()
8     {
9         JOptionPane.showMessageDialog(null, "該休息了", "提醒",
10             JOptionPane.INFORMATION_MESSAGE);
11     }
12 }
13
14 public class CH09_25
15 {
16     public static void main(String[] args)
17     {
18         new MyMessageDialog();
19     }
20 }
```

訊息對話框


◆ 執行結果：



◆ 說明：

- 行01：
 - ◆ 定義「套件（package）」。
- 行03：
 - ◆ 載入「javax.swing.*」套件。
- 行05~行12：
 - ◆ 建立繼承自「JFrame」父類別的「MyMessageDialog」子類別。

訊息對話框

- ◆ 行07~行11：
 - ◆ 宣告類別「建構子」 「MyMessageDialog()」。
 - 行09：
 - ◆ 設定訊息對話框。
 - ◆ 沒有對話框的容器；設定提示訊息；設定對話框標題；使用  圖示。
 - 行18：
 - ◆ 建立MyMessageDialog物件。
- 注意：
- 此為匿名物件，故在main()（即主程式）中，無法修改視窗大小、位置、...、等相關屬性。

確認對話框

■ 用來提出問題，並讓使用者回覆的對話框。

■ 其多載方法如下：

- ◆ `int showConfirmDialog(Component parent, Object msg)`

- ◆ `int showConfirmDialog(Component parent, Object msg, String title, int optionType)`

- `optionType`：

- ◆ 用來設定放置在對話框中等待使用者答覆的按鈕，常用的設定值有：

- ◆ `JOptionPane.YES_NO_CANCEL_OPTION`（預設值）：

- ◆ 顯示  、  、  按鈕。

- ◆ `JOptionPane.YES_NO_OPTION`：

- ◆ 顯示  、  按鈕。




確認對話框

- ◆ `int showConfirmDialog(Component parent, Object msg, String title, int optionType, int msgType)`
- ◆ `int showConfirmDialog(Component parent, Object msg, String title, int optionType, int msgType, ImageIcon icon)`
 - `ImageIcon icon` :
 - ◆ 不使用引數 `msgType` 指定的圖示，而要顯示自訂的圖示。則 `msgType` 引數值要設為 0。
 - ◆ 若不顯示自訂按鈕，則本引數值要設為 `null`。

註：

- 除了增加 `optionType` 引數，且將 `msgType` 的預設值改為 `JOptionPane.QUESTION_MESSAGE` 外，其餘皆與 `showMessageDialog()` 的引數用法相同。

確認對話框

- 當確認對話框showConfirmDialog() 被使用時，會提供給使用者二個或三個按鈕做為點選答覆的工具。
- 其使用者所點選的按鈕與所對應的傳回值如下：
 - ◆ 當使用者點按  鈕時，傳回整數常數 JOptionPane.YES_OPTION。
 - ◆ 當使用者點按  鈕時，傳回整數常數 JOptionPane.NO_OPTION。
 - ◆ 當使用者點按  鈕時，傳回整數常數 JOptionPane.CANCEL_OPTION。

確認對話框

◆ 程式：

```
package CH09_26;
```

```
import java.awt.*;
```

```
import javax.swing.*;
```

```
class MyConfirmDialog extends JFrame
```

```
{
```

```
    MyConfirmDialog()
```

```
{
```

```
    int ans = JOptionPane.showConfirmDialog(null, "現在下雨嗎?", "調查",  
        JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE);
```

```
    JLabel lblAns;
```

```
    if (ans == JOptionPane.YES_OPTION)
```

```
        lblAns = new JLabel("出門請帶雨具。");
```

確認對話框

```
else
    lblAns = new JLabel("出門請擦防曬。");

add(lblAns);

setTitle("確認對話框");
setLayout(new FlowLayout());
setBounds(100, 100, 220, 70);
setVisible(true);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}

public class CH09_26
{
```

確認對話框

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

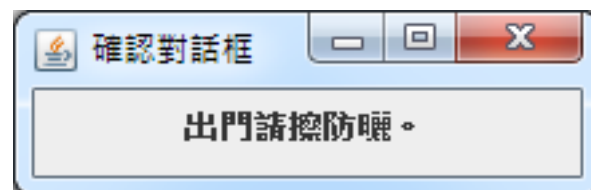
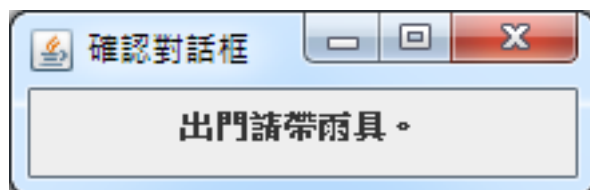
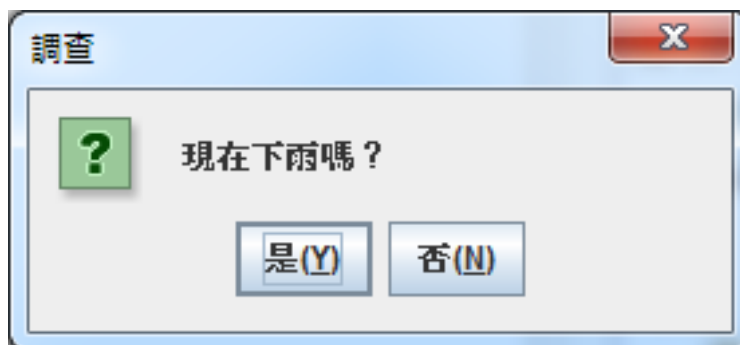
確認對話框

```
1 package CH09_26;
2
3 import java.awt.*;
4 import javax.swing.*;
5
6 class MyConfirmDialog extends JFrame
7 {
8     MyConfirmDialog()
9     {
10         int ans = JOptionPane.showConfirmDialog(null, "現在下雨嗎?", "調查",
11             JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);
12
13         JLabel lblAns;
14         if (ans == JOptionPane.YES_OPTION)
15             lblAns = new JLabel("出門請帶雨具。");
16         else
17             lblAns = new JLabel("出門請擦防曬。");
18
19         add(lblAns);
20
21         setTitle("確認對話框");
22         setLayout(new FlowLayout());
23         setBounds(100, 100, 220, 70);
24         setVisible(true);
25         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
26     }
27 }
```


確認對話框

```
27 }  
28  
29 public class CH09_26  
30 {  
31     public static void main(String[] args)  
32     {  
33         new MyConfirmDialog();  
34     }  
35 }
```

◆ 執行結果：



確認對話框




◆ 說明：

- 行01：
 - ◆ 定義「套件（package）」。
- 行03：
 - ◆ 載入「java.awt.*」套件。
- 行04：
 - ◆ 載入「javax.swing.*」套件。
- 行06~行27：
 - ◆ 建立繼承自「JFrame」父類別的「MyConfirmDialog」子類別。
 - ◆ 行08~行26：
 - ◆ 宣告類別「建構子」 「MyConfirmDialog()」。

確認對話框

行10：

設定確認對話框。

用整數變數ans來存放使用者答覆的傳回值；沒有對話框的容器；使用 、 按鈕；使用  圖示。

行13：

宣告標籤物件lblAns。

行14：

條件判斷式若條件成立（即結果為True），則執行行15的敘述；若條件不成立（即結果為False），則執行行17的敘述。

行15：

利用「JLabel(String text)」的「建構子」，建立標籤物件「lblAns」。

行17：

利用「JLabel(String text)」的「建構子」，建立標籤物件「lblAns」。

確認對話框

行19：

將標籤物件，增加到視窗中。

行21：

設定視窗標題。

行22：

設定視窗使用「FlowLayout()」的版面配置。

行23：

設定視窗的位置及大小。

行24：

設定視窗可以顯示在螢幕上。

行25：

設定當點按視窗右上角關閉鈕時，一併結束應用程式。

- 行33：

- ◆ 建立MyConfirmDialog物件。

確認對話框

注意：

此為匿名物件，故在main()（即主程式）中，無法修改視窗大小、位置、...、等相關屬性。

輸入對話框

■ 用來要求使用者針對問題，輸入對應資料的對話框。

■ 其多載方法如下：

◆ `String showInputDialog(Object msg)`

- 輸入對話框的標題列文字內定為「輸入」。
- 輸入對話框會被顯示在螢幕的中央。

◆ `String showInputDialog(Component parent, Object msg)`

◆ `String showInputDialog(Object msg, Object initValue)`

- `initValue`：
 - ◆ 用來設定輸入對話框內輸入文字欄位的初始值。

輸入對話框

- ◆ `String showInputDialog(Component parent, Object msg, Object initValue)`
- ◆ `String showInputDialog(Component parent, Object msg, String title, int msgType)`

註：

除了增加`initValue`引數，且將`msgType`的預設值改為`JOptionPane.QUESTION_MESSAGE`外，其餘皆與`showMessageDialog()`的引數用法相同。

輸入對話框

◆ 程式：

```
package CH09_27;
```

```
import javax.swing.*;
```

```
class MyInputDialog extends JFrame
```

```
{
```

```
    MyInputDialog()
```

```
{
```

```
    String ans = JOptionPane.showInputDialog(null, "請輸入數字後按確定",  
                                                "輸入", JOptionPane.QUESTION_MESSAGE);
```

```
    if (ans == null)
```

```
        System.exit(0);
```


輸入對話框

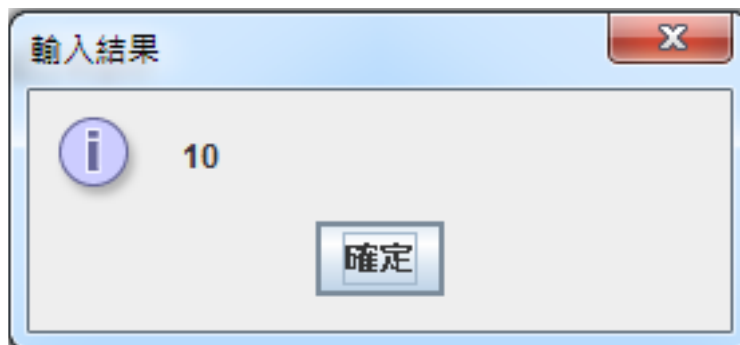
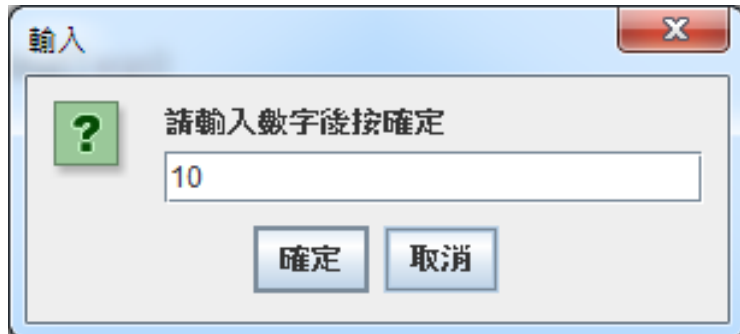
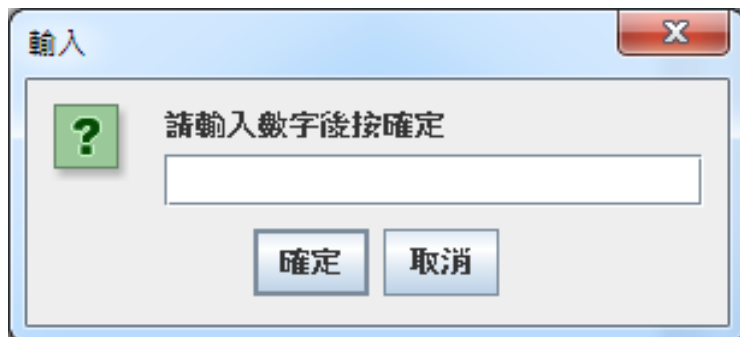
```
        JOptionPane.showMessageDialog(null, ans, "輸入結果",  
        JOptionPane.INFORMATION_MESSAGE);  
    }  
}  
  
public class ch09_27  
{  
    public static void main(String[] args)  
    {  
        new MyInputDialog();  
    }  
}
```

輸入對話框

```
1 package CH09_27;
2
3 import javax.swing.*;
4
5 class MyInputDialog extends JFrame
6 {
7     MyInputDialog()
8     {
9         String ans = JOptionPane.showInputDialog(null, "請輸入數字後按確定",
10             "輸入", JOptionPane.QUESTION_MESSAGE);
11
12         if (ans == null)
13             System.exit(0);
14
15         JOptionPane.showMessageDialog(null, ans, "輸入結果",
16             JOptionPane.INFORMATION_MESSAGE);
17     }
18 }
19
20 public class ch09_27
21 {
22     public static void main(String[] args)
23     {
24         new MyInputDialog();
25     }
26 }
```


輸入對話框

◆ 執行結果：



輸入對話框

◆ 說明：

- 行01：
 - ◆ 定義「套件（package）」。
- 行03：
 - ◆ 載入「javax.swing.*」套件。
- 行05~行18：
 - ◆ 建立繼承自「JFrame」父類別的「MyInputDialog」子類別。
 - ◆ 行07~行17：
 - ◆ 宣告類別「建構子」 「MyInputDialog()」。
 - 行09：
 - ◆ 設定輸入對話框。
 - ◆ 用字串變數ans來存放使用者輸入的值；設定提示訊息；設定對話框標題；使用  圖示。

輸入對話框


行12：

- ◆ 條件判斷式若條件成立（即結果為True），則執行行13的敘述。

行13：

- ◆ 關閉視窗，並結束程式。

行15：

- ◆ 設定訊息對話框。
- ◆ 沒有對話框的容器；顯示使用者輸入的數字；設定對話框標題；使用  圖示。

• 行24：

- ◆ 建立MyInputDialog物件。

注意：

- ◆ 此為匿名物件，故在main()（即主程式）中，無法修改視窗大小、位置、...、等相關屬性。
- ◆ 雖然要求使用者輸入數字，但程式中並沒有設立檢查輸入的內容，是否為數字的機制。

選項對話框

■ 用來提出問題詢問使用者，並可依據問題的需要，自製出相對應的按鈕選項來讓使用者回答。

■ 其方法如下：

◆ `int showOptionDialog(Component parent, Object msg, String title, int optionType, int msgType, ImageIcon icon, Object[] options, Object initValue)`

- 引數parent、msg、title、optionType、msgType、icon的用法與確認對話框相同。
- options：
 - ◆ 不使用引數optionType所設定的按鈕，而要使用自訂的選項按鈕群，而這選項按鈕群的按鈕標題文字常用字串陣列的元素內容來表示，此時，optionType引數值要設為0。若沒有要顯示自訂的按鈕，則本引數值要設為null。

選項對話框

- `initValue` :
 - ◆ 若是使用自訂的選項按鈕群，可用本引數從選項 按鈕群中，指定成預設選取狀的按鈕。
- ◆ 當選項對話框`showOptionDialog()`被使用時，若引數`options`有設定值（該設定值大都使用字串陣列），則選項按鈕群就可做為點選答覆的工具。
- ◆ 使用者所點選的按鈕會以引數`options`所對應的索引值做為傳回值。

選項對話框

◆ 程式：

```
package CH09_28;
```

```
import java.awt.*;
```

```
import javax.swing.*;
```

```
class MyOptionDialog extends JFrame
```

```
{
```

```
    MyOptionDialog()
```

```
{
```

```
    ImageIcon icon = new ImageIcon();
```

```
    String[] options = {"晴天","陰天","雨天"};
```

```
    int ans = JOptionPane.showOptionDialog(null, "請選擇今天的天氣",  
        "天氣", 0,0,icon,options,options[0]);
```


選項對話框

```
JLabel lblAns;  
if(ans==0)  
    lblAns = new JLabel("出門請擦防曬。");  
else if(ans==1)  
    lblAns = new JLabel("出門建議攜帶雨具。");  
else  
    lblAns = new JLabel("出門請帶雨具。");  
  
add(lblAns);  
  
setTitle("選項對話框");  
setLayout(new FlowLayout());  
setBounds(100, 100, 220, 70);  
setVisible(true);  
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
}
```

選項對話框

```
}
```

```
public class CH09_28
```

```
{
```

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

```
    {
```

```
        new MyOptionDialog();
```

```
    }
```

```
}
```

選項對話框

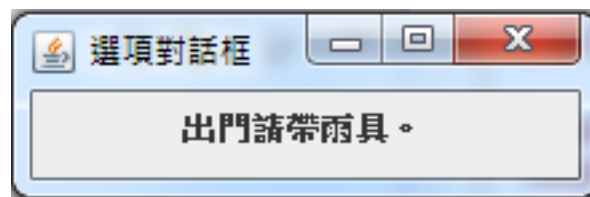
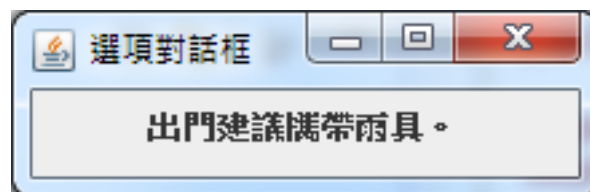
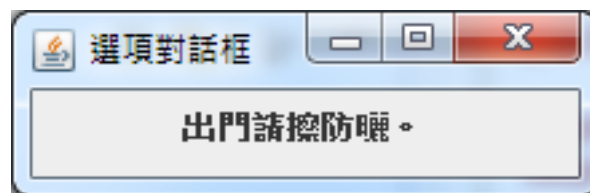
```
1 package CH09_28;
2
3 import java.awt.*;
4 import javax.swing.*;
5
6 class MyOptionDialog extends JFrame
7 {
8     MyOptionDialog()
9     {
10         ImageIcon icon = new ImageIcon();
11
12         String[] options = {"晴天", "陰天", "雨天"};
13         int ans = JOptionPane.showOptionDialog(null, "請選擇今天的天氣",
14             "天氣", 0, 0, icon, options, options[0]);
15
16         JLabel lblAns;
17         if(ans==0)
18             lblAns = new JLabel("出門請擦防曬。");
19         else if(ans==1)
20             lblAns = new JLabel("出門建議攜帶雨具。");
21         else
22             lblAns = new JLabel("出門請帶雨具。");
23
24         add(lblAns);
25
26         setTitle("選項對話框");
```

選項對話框

```
27     setLayout(new FlowLayout());
28     setBounds(100, 100, 220, 70);
29     setVisible(true);
30     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
31 }
32 }
33
34 public class CH09_28
35 {
36     public static void main(String[] args)
37     {
38         new MyOptionDialog();
39     }
40 }
```

選項對話框

◆ 執行結果：



選項對話框

◆ 說明：

- 行01：
 - ◆ 定義「套件（package）」。
- 行03：
 - ◆ 載入「java.awt.*」套件。
- 行04：
 - ◆ 載入「javax.swing.*」套件。
- 行06~行32：
 - ◆ 建立繼承自「JFrame」父類別的「MyOptionDialog」子類別。
 - ◆ 行08~行31：
 - ◆ 宣告類別「建構子」「MyOptionDialog()」。

選項對話框

行10：

- ◆ 建立icon物件。

註：

此處建立空的icon物件。

行12：

- ◆ 建立字串陣列，並指定初值。

行13：

- ◆ 設定選項對話框。
- ◆ 用整數變數ans來存放使用者選擇的結果；設定提示訊息；設定對話框標題；不使用optionType；不使用msgType；使用空的icon；自訂選項按鈕群的按鈕標題文字；指定第一個按鈕為預設選取的按鈕。

行16：

- ◆ 宣告標籤物件lblAns。

選項對話框

行17：

- ◆ 條件判斷式若第一個條件成立（即結果為True），則執行行18的敘述；若第二個條件成立（即結果為True），則執行行20的敘述；若條件都不成立（即結果為False），則執行行22的敘述。

行18：

- ◆ 利用「JLabel(String text)」的「建構子」，建立標籤物件「lblAns」。

行20：

- ◆ 利用「JLabel(String text)」的「建構子」，建立標籤物件「lblAns」。

行22：

- ◆ 利用「JLabel(String text)」的「建構子」，建立標籤物件「lblAns」。

行24：

- ◆ 將標籤物件，增加到視窗中。

選項對話框

行26：

設定視窗標題。

行27：

◆ 設定視窗使用「FlowLayout()」的版面配置。

行28：

◆ 設定視窗的位置及大小。

行29：

◆ 設定視窗可以顯示在螢幕上。

行30：

◆ 設定當點按視窗右上角關閉鈕時，一併結束應用程式。

• 行38：

◆ 建立MyOptionDialog物件。

注意：

此為匿名物件，故在main()（即主程式）中，無法修改視窗大小、位置、...、等相關屬性。

資料來源

- 蔡文龍、何嘉益、張志成、張力元，JAVA SE 10基礎必修課，台北市，碁峰資訊股份有限公司，2018年7月，出版。
- 吳燦銘、胡昭民，圖解資料結構-使用Java(第三版)，新北市，博碩文化股份有限公司，2018年5月，出版。
- Ivor Horton，Java 8 教學手冊，台北市，碁峰資訊股份有限公司，2016年9月，出版。
- 李春雄，程式邏輯訓練入門與運用---使用JAVA SE 8，台北市，上奇科技股份有限公司，2016年6月，初版。
- 位元文化，Java 8視窗程式設計，台北市，松崗資產管理股份有限公司，2015年12月，出版。
- Benjamin J Evans、David Flanagan，Java 技術手冊 第六版，台北市，碁峰資訊股份有限公司，2015年7月，出版。
- 蔡文龍、張志成，JAVA SE 8 基礎必修課，台北市，碁峰資訊股份有限公司，2014年11月，出版。
- 陳德來，Java SE 8程式設計實例，台北市，上奇科技股份有限公司，2014年11月，初版。
- 林信良，Java SE 8 技術手冊，台北市，碁峰資訊股份有限公司，2014年6月，出版。
- 何嘉益、黃世陽、李篤易、張世杰、黃鳳梅，徐政棠譯，JAVA2 程式設計從零開始--適用JDK7，台北市，上奇資訊股份有限公司，2012年5月，出版。