第十二章
AWT 視窗物件

本章學習目標

- 認識 AWT 類別
- 認識並學習如何建立視窗物件
- 學習如何管理與配置版面
- 學習 Panel 類別的使用

17.1 認識 AWT 類別

AWT（Abstract Windowing Toolkit）是用來處理視窗最基本的方式。

17.1.1 簡單的範例

下面是一個簡單的視窗程式設計的範例。

```
01  // app17_1, AWT 簡單的範例 (一)
02  import java.awt.*;   // 載入 java.awt 類別庫裡的所有類別
03  public class app17_1
04  {
05     static Frame frm=new Frame("my first AWT program");
06     static Label lab=new Label("Hello Java!!");
07
08     public static void main(String args[])
09     {
10        frm.setSize(200,150);   // 設定視窗的寬為 200、高為 150 個像素
11        frm.setBackground(Color.yellow);  // 設定黃色的背景
12        frm.setLocation(250,250);         // 設定視窗的位置
13        frm.add(lab);                     // 將標籤物件 lab 加入視窗中
14        frm.setVisible(true);             // 將視窗顯示出來
15     }
16  }
```

第 05、06 行：類別 app17_1 的資料成員

執行後將會看到如下的畫面：

想讓關閉鈕也可以有作用，只要把 java.awt.event.*載入：

```
import java.awt.event.*; // 載入 java.awt.event 裡所有的類別
```

在 main() method 裡的任何位置補上這兩行：

```
frm.addWindowListener(new WindowAdapter(){
    public void windowClosing(WindowEvent e){System.exit(0);}});
```

下面的程式碼是將 frm 與 lab 宣告在 main() method 裡：

```
01  // app17_2, AWT 簡單的範例 (二)
02  import java.awt.*;   // 載入 java.awt 類別庫裡的所有類別
03  public class app17_2
04  {
05     public static void main(String args[])
06     {
07        Frame frm=new Frame("my first AWT program");
08        Label lab=new Label("Hello Java!!");
09        frm.setSize(200,150);
10        frm.setBackground(Color.yellow);
11        frm.setLocation(250,250);
12        frm.add(lab);
13        frm.setVisible(true);
14     }
15  }
```

把 frm 與 lab 宣告在 main() method 內，程式依然可以執行

下面是利用繼承自 Frame 類別的方式來建立視窗：

```
01  // app17_3, AWT 簡單的範例 (三)
02  import java.awt.*;
03  public class app17_3 extends Frame  // 指定 app17_3 繼承自 Frame 類別
04  {
05     public static void main(String args[])
06     {
07        app17_3 frm=new app17_3();   // 用 app17_3 類別產生 frm 物件
08
09        Label lab=new Label("Hello Java!!");
10        frm.setTitle("my first AWT program"); // 在視窗中加入標題
11        frm.setSize(200,150);
12        frm.setBackground(Color.yellow);
13        frm.setLocation(250,250);
14        frm.add(lab);
15        frm.setVisible(true);
16     }
17  }
```

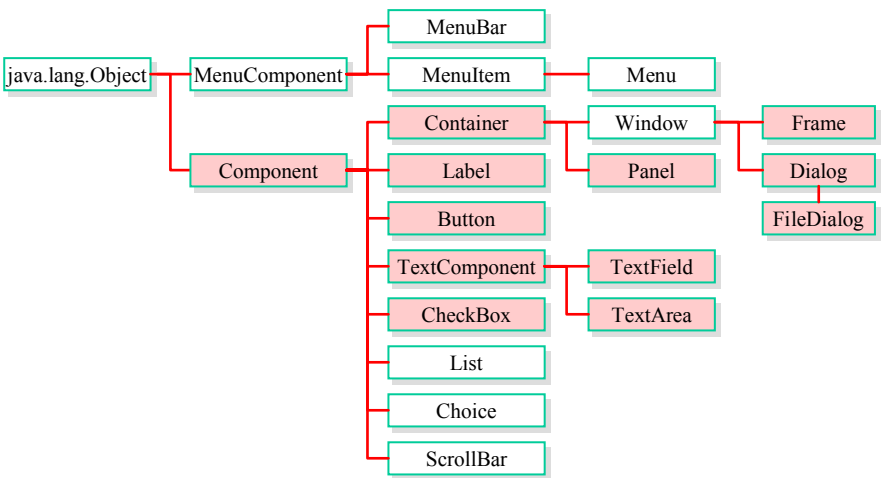
繼承自 Frame 類別

下面是利用建構元來建立視窗物件的程式碼：

```
01  // app17_4, AWT 簡單的範例 (四)
02  import java.awt.*;
03  public class app17_4 extends Frame
04  {
05     Label lab=new Label("Hello Java!!");  // 建立 lab 物件
06
07     public app17_4(String str) // 建構元 app17_4()
08     {
09        super(str);      // 呼叫父類別(Frame)的建構元
10        add(lab);        // 將標籤 lab 物件加入視窗中
11     }
12
13     public static void main(String args[])
14     {
15        app17_4 frm=new app17_4("my first AWT program");// 呼叫app17_4()建構元
16
17        frm.setSize(200,150);
18        frm.setBackground(Color.yellow);
19        frm.setLocation(250,250);
20        frm.setVisible(true);
21     }
22  }
```

17.1.2 視窗物件的類別簡介

下圖為 java.awt 類別庫提供的類別，以及它們之間的繼承關係：



java.awt.Component 類別

下表列出了 Component 類別的建構元與 method：

表 17.1.1 java.awt.Component 的建構元與 method

建構元	主要功能
Component()	建立一個新的視窗物件

method	主要功能
void add(PopupMenu popup)	加入跳出式的功能表
boolean contains(int x, int y)	測試物件的範圍是否包含點(x,y)
float getAlignmentX()	取得物件在 x 軸方向的對齊方式，它會傳回 0~1 之間的數，0 代表物件剛好在原點上，1 則代表物件位於離原點最遠的地方，其餘的位置依 0~1 之間的內插值傳回
float getAlignmentY()	同上，但取得物件在 y 軸方向的對齊方式
Color getBackgrount()	傳回物件的背景顏色
void setBackground(Color color)	設定物件的背景顏色為 color

method	主要功能
Rectangle getBounds()	傳回物件所佔矩形面積的大小
void setBounds(intx, int y,int w, int h)	設定物件的顯示區域，物件的左上角座標為(x,y)，物件的寬度為 w，高度為 h
Component getComponentAt(int x, int y)	傳回包含點(x,y)的物件
Font getFont()	傳回物件字型的樣式
void setFont(Font font)	設定物件字型的樣式為 font
Color getForeground()	傳回物件的前景顏色
Color setForeground(Color color)	設定物件的前景顏色為 color
Point getLocation()	傳回物件左上角之座標位置
void setLocation(int x, int y)	設定物件的顯示位置的左上角座標為(x,y)
void setSize(int w, int h)	設定物件的大小，寬為 w，高為 h
String getName()	傳回物件的名稱
String setName(String str)	設定物件的名稱為 str
int getWidth()	取得物件的寬度
int getHeight()	取得物件的高度
int getX()	傳回物件在 x 軸方向的座標

method	主要功能
int getY()	傳回物件在 y 軸方向的坐標
boolean isVisible()	測試物件的屬性是否可見
void setEnabled(boolean v)	設定物件是否呈可使用狀態
void setVisible(boolean v)	設定物件是否為可見。若 v 為 true 則可見，若為 false 則為不可見

java.awt.Container 類別

下表列出 Container 類別常用的建構元與 method：

表 17.1.2 java.awt.Container 的建構元與 method

建構元	主要功能
Container()	建立一個 Container 物件

method	主要功能
Component add(Component comp)	將物件 comp 加到容器物件內
Component add(Component comp, int index)	將物件 comp 加到容器物件內，並給予編號
Component add(Component comp, Object constraints)	將 comp 加到容器物件內，並依 constraints 指定的方式來配置
Component add(Component comp, Object constraints, int index)	將 comp 加到容器物件內，給予編號並依 constraints 指定的方式來配置
void doLayout()	讓容器物件依版面配置來調整物件的位置
float getAlignmentX()	取得與 x 軸的對齊方式

method	主要功能
float getAlignmentY()	取得與 y 軸的對齊方式
Component getComponent(int n)	取出容器物件內，編號為 n 的物件
Component getComponentAt(int x, int y)	依指定的位置 (x,y)取出容器物件內的物件
Component getComponentAt(Point p)	依指定的點 p 取出容器物件內的物件
int getComponentCount()	取得容器物件內，所有物件的個數
Component[] getComponents()	取得容器物件內的所有物件，以陣列傳回
LayoutManager getLayout()	取得容器物件所使用的版面配置
void paint(Graphics g)	重繪容器物件
void paintComponents(Graphics g)	重繪容器物件裡所有的物件
void remove(Component comp)	移除容器物件裡指定的物件
void remove(int index)	依編號移除容器物件裡的物件
void removeAll()	全部移除容器物件裡的物件
void setFont(Font f)	設定容器物件之字型
void setLayout(LayoutManager mgr)	設定容器物件使用 mgr 版面配置
void update(Graphics g)	更新容器物件

17.2 建立視窗

Frame 可以當成是一個容器，用來容納其它視窗物件。下表列出了 Frame 類別常用的建構元與 method：

表 17.2.1 java.awt.Frame 的建構元與 method

建構元	主要功能
Frame()	建立一個沒有標題的視窗
Frame(String title)	建立視窗，並以 title 為其標題

method	主要功能
Image getIconImage()	傳回視窗最小化時的圖示
void setIconImage(Image img)	設定視窗最小化時的圖示為 img
int getState()	傳回視窗的狀態，Frame.Normal 代表一般狀態，Frame.ICONIFIED 代表視窗為最小化。Normal 與 ICONIFIED 為 Frame 類別裡定義的常數，其值分別定義成 0 與 1
void setState()	設定視窗的狀態，Frame.Normal 代表一般狀

method	主要功能
	態，Frame.ICONIFIED 代表視窗為最小化
MenuBar getMenuBar()	傳回視窗裡的功能表物件
void setMenuBar(Menubar mb)	設定視窗使用的功能表物件為 mb
void remove(MenuCompoent mb)	移除視窗中的功能表物件
String getTitle()	取得視窗的標題
String setTitle(String title)	設定視窗的標題為 title
boolean isResizable()	測試視窗是否可改變大小。若傳回值為 true，則可改變，若為 false，則不能改變
void setResizable(boolean b)	設定視窗是否允許改變大小。若 b 為 true，則可以改變，若 b 為 false，則不能改變

下面舉一個實例來說明如何以 Frame 類別建立視窗物件：

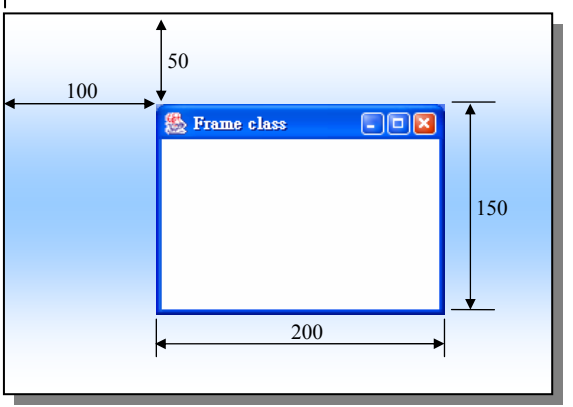
```
01  // app17_5, 建立視窗物件
02  import java.awt.*;
03  public class app17_5
04  {
05     static Frame frm=new Frame("Frame class");
06
07     public static void main(String args[])
08     {
09        frm.setSize(200,150);          // 設定視窗的大小為 200*150
10        frm.setLocation(100,50);       // 設定視窗位置為(100,50)
11        frm.setVisible(true);          // 設定視窗為可見
12
13        System.out.println("state="+frm.getState());
14        System.out.println("title="+frm.getTitle());
15        System.out.println("visible="+frm.isVisible());
16     }
17  }

/* app17_5 OUTPUT--

state=0
title=Frame class
visible=true
--------------------*/
```

參考下圖來解說 setSize() 與 SetLocation() 裡引數的意義：

桌面的左上角



17.3 建立標籤

標籤（label）用來在視窗中顯示文字的文字方塊。下表列出了 Label 類別常用的建構元與 method：

表 17.3.1 java.awt.Label 的建構元與 method

建構元	主要功能
Label()	建立一個沒有文字的標籤
Label(String text)	建立標籤，並以 text 為標籤上的文字
Label(String text, int align)	建立標籤，以 text 為標籤上的文字，並以 align 的方式對齊，其中 align 的值可為 Label.LEFT、Label.RIGHT 與 Label.CENTER，分別代表靠左、靠右與置中對齊

method	主要功能
int getAlignment()	傳回標籤內文字的對齊方式，傳回的值可能為 Label.LEFT、Label.RIGHT 與 Label.CENTER
int setAlignment(int align)	設定標籤內文字的對齊方式，align 的值可為

method	主要功能
	Label.LEFT、Label.RIGHT 與 Label.CENTER
String getText()	傳回標籤內的文字
String setText(String text)	設定標籤內的文字為 text

下面的範例是在 Frame 視窗中加上一個標籤：

```
01  // app17_6, 在視窗中加入標籤物件
02  import java.awt.*;
03  public class app17_6
04  {
05     static Frame frm=new Frame("Label class");
06     static Label lab=new Label();    // 建立標籤物件 lab
07
08     public static void main(String args[])
09     {
10        frm.setSize(200,150);
11        frm.setBackground(Color.pink); // 設定視窗背景為粉紅色
12        lab.setText("Hello Java");              // 在標籤中加入文字
13        lab.setBackground(Color.white);         // 設定標籤背景為白色
14        lab.setAlignment(Label.CENTER);         // 將標籤中的文字置中
15        lab.setForeground(Color.blue);          // 設定標籤文字為藍色
16        Font fnt=new Font("Serief",Font.ITALIC+Font.BOLD,18);
17        lab.setFont(fnt);                       // 設定字型的樣式
18        frm.add(lab);
19        frm.setVisible(true);
20     }
21  }
```

本範例的執行結果如下圖所示：

Color 類別

建立一個顏色物件，可以利用 Color() 建構元，格式為：

```
public Color(int r, int g, int b)   // Color()建構元
```

於 app17_6 中，如果標籤的顏色要改成紫色，把 15 行改成下面的敘述:

```
lab.setForeground(new Color(255,0,255));
```

new Color(255,0,255) 建立顏色物件

Font 類別

要產生 Font 類別的物件，可以使用 Font() 建構元，其格式如下：

```
public Font(String font_name, int style, int size)
```

style 為字型的樣式，可設為 Font.PLAIN、Font.BOLD、與 Font.ITALIC。

要同時設定粗體與斜體，可用下列的語法來表示：

```
Font.BOLD+Font.ITALIC    // 同時設定粗體與斜體
```

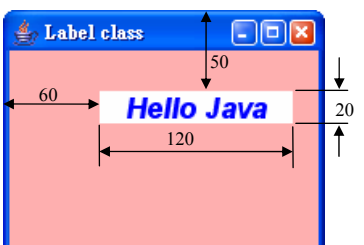
關於版面配置

如果把預設的版面配置取消，則可看到原來視窗的樣子，如下面的範例：

```
01  // app17_7, 指定標籤物件的大小
02  import java.awt.*;
03  public class app17_7
04  {
05     static Frame frm=new Frame("Label class");
06     static Label lab=new Label();
07
08     public static void main(String args[])
09     {
10        frm.setLayout(null);                    // 取消版面配置
11        frm.setSize(200,150);
12        frm.setBackground(Color.pink);
13        lab.setText("Hello Java");
14        lab.setBackground(Color.white);
15        lab.setAlignment(Label.CENTER);
16        lab.setForeground(Color.blue);
17        lab.setLocation(60,50);                 // 設定標籤位置
18        lab.setSize(120,20);                    // 設定標籤大小
19        lab.setFont(new Font("Serief",Font.ITALIC+Font.BOLD,18));
```

```
20          frm.add(lab);
21          frm.setVisible(true);
22      }
23  }
```

app17_7 執行的結果如下圖：



17.4 建立按鈕

下表列出了 Button 類別常用的建構元與 method：

表 17.4.1 java.awt.Button 的建構元與 method

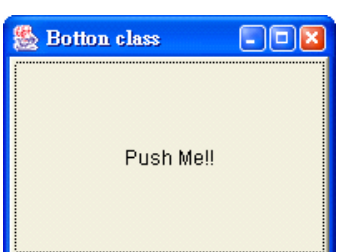
建構元	主要功能
Button()	建立一個沒有標題的按鈕
Button(String title)	建立標題為 title 的按鈕

method	主要功能
String getLabel()	傳回按鈕的標題
String setLabel(String title)	設定按鈕的標題為 title

下面以一個簡單的範例來說明 Button 類別的使用：

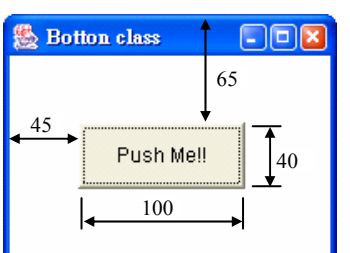
```
01  // app17_8, Button 類別
02  import java.awt.*;
03  public class app17_8
04  {
05     static Frame frm=new Frame("Button class");
06     static Button btn=new Button("Push Me!!");   // 建立按鈕物件
07
08     public static void main(String args[])
09     {
10        frm.setSize(200,150);
11        frm.add(btn);                    // 在視窗中加入按鈕
12        frm.setVisible(true);
13     }
14  }
```

右圖為 app17_8 執行後的畫面：



若要將按鈕(或其它視窗物件)設定大小與位置，可以使用 setBounds():

```
01  // app17_9, 設定按鈕的大小
02  import java.awt.*;
03  public class app17_9
04  {
05     static Frame frm=new Frame("Button class");
06     static Button btn=new Button("Push Me!!");
07
08     public static void main(String args[])
09     {
10        frm.setLayout(null);                  // 不使用版面配置
11        btn.setBounds(45,65,100,40);          // 設定按鈕的大小與位置
12        frm.setSize(200,150);
13        frm.add(btn);
14        frm.setVisible(true);
15     }
16  }
```



17.5 建立核取方塊

下表列出了 Checkbox 類別常用的建構元與 method：

表 17.5.1 java.awt.Checkbox 的建構元與 method

建構元	主要功能
Checkbox()	建立一核取方塊
Checkbox(String label)	建立一標籤為 label 的核取方塊
Checkbox(String label, boolean state)	建立一標籤為 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
Checkbox(String label, boolean state, CheckboxGroup grp)	建立一單選的核取方塊，並將它加入 grp 群組中

method	主要功能
CheckboxGroup getCheckboxGroup()	傳回核取方塊是屬於哪一個群組
void setCheckboxGroup(CheckboxGroup grp)	設定核取方塊屬於 grp 群組
String getLabel()	傳回核取方塊的標籤
boolean getState()	傳回核取方塊是否呈被選取狀態

method	主要功能
void setState(boolean state)	設定核取方塊是否呈被選取狀態

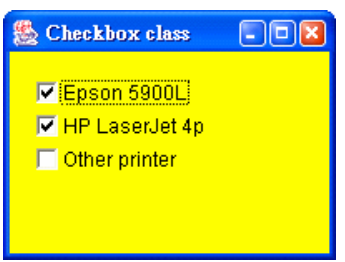
可供複選的核取方塊

下面的範例是在視窗中建立三個可供複選的核取方塊：

```
01  // app17_10, 核取方塊的應用 (一)
02  import java.awt.*;
03  public class app17_10
04  {
05     static Frame frm=new Frame("Checkbox class");
06     static Checkbox ckb1=new Checkbox("Epson 5900L",true);
07     static Checkbox ckb2=new Checkbox("HP LaserJet 4p",true);
08     static Checkbox ckb3=new Checkbox("Other printer");
09
10     public static void main(String args[])
11     {
12        frm.setSize(200,150);
13        frm.setLayout(null);
14        frm.setBackground(Color.yellow);
15        ckb1.setBounds(20,40,140,20);     // 設定核取方塊的位置與大小
```

```
16          ckb2.setBounds(20,60,140,20);
17          ckb3.setBounds(20,80,140,20);
18          frm.add(ckb1);                      // 加入核取方塊到視窗中
19          frm.add(ckb2);
20          frm.add(ckb3);
21          frm.setVisible(true);
22      }
23  }
```

app17_10 的執行結果：



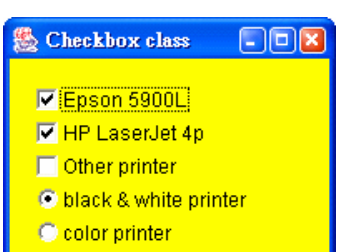
單選鈕的檢取方塊

下面的範例是 app17_10 的延伸：

```
01  // app17_11, 檢取方塊的狀況(二)
02  import java.awt.*;
03  public class app17_11
04  {
05     static Frame frm=new Frame("Checkbox class");
06     static Checkbox ckb1=new Checkbox("Epson 5900L",true);
07     static Checkbox ckb2=new Checkbox("HP LaserJet 4p",true);
08     static Checkbox ckb3=new Checkbox("Other printer");
09     static Checkbox ckb4=new Checkbox("black & white printer");
10     static Checkbox ckb5=new Checkbox("color printer");
11
12     public static void main(String args[])
13     {
14        CheckboxGroup grp=new CheckboxGroup();  // 建立群組物件 grp
15        frm.setSize(200,150);
16        frm.setLayout(null);
17        frm.setBackground(Color.yellow);
18        ckb1.setBounds(20,40,140,20);
19        ckb2.setBounds(20,60,140,20);
```

```
20          ckb3.setBounds(20,80,140,20);
21          ckb4.setBounds(20,100,140,20);
22          ckb5.setBounds(20,120,140,20);
23          ckb4.setCheckboxGroup(grp);          // 將 ckb4 加入 grp 群組中
24          ckb5.setCheckboxGroup(grp);          // 將 ckb5 加入 grp 群組中
25          ckb4.setState(true);                 // 將 ckb4 設為選取狀態
26          frm.add(ckb1);
27          frm.add(ckb2);
28          frm.add(ckb3);
29          frm.add(ckb4);
30          frm.add(ckb5);
31          frm.setVisible(true);
32      }
33  }
```

app17_11 的執行結果如右所示：



17.6 建立文字輸入物件

AWT 中主要用來處理文字輸入物件的類別有兩個，分別為 TextField 與 TextArea。下表為這些 method 的整理：

表 17.6.1 java.awt.TextComponent 的 method

method	主要功能
Color getBackground()	取得背景顏色
String getSelectedText()	取得被選取區域的文字
String getText()	取得文字區塊裡的文字
boolean isEditable()	測試文字區塊裡的文字是否可被編輯
void select(int selStart, int selEnd)	選擇位置為 selStart 與 selEnd 之間的字元
void selectAll()	選擇文字區塊裡的所有文字
void setBackground(Color c)	設定背景顏色
void setEditable(boolean b)	文字區塊設定為可編輯的

17.6.1 TextField 建立文字方塊

Java 用 TextField 類別來建立文字方塊。下表列出 TextField 類別所提供的建構元與常用的 method：

表 17.6.2 java.awt.TextField 的建構元與 method

建構元	主要功能
TextField()	建立文字方塊
TextField(int columns)	建立文字方塊，並設定文字方塊的寬度可容納 columns 個字元
TextField(String text)	建立文字方塊，並以 text 為預設的文字
TextField(String text, int length)	建立文字方塊，以 text 為預設的文字，並設定文字方塊的寬度可容納 columns 個字元

method	主要功能
boolean echoCharIsSet()	測試文字方塊中的文字是否會被顯示成其它字元，true 代表可被顯示成其它字元，false 代表不能被顯示成其它字元

method	主要功能
int getColumns()	取得文字方塊預設的寬度（以字元數為單位）
char getEchoChar()	取得文字方塊的回應字元
void setColumns(int columns)	設定文字方塊的寬度為 columns 個字元
void setEchoChar(char c)	設定文字方塊的回應字元為 c
void setText(String text)	設定文字方塊的文字為 text

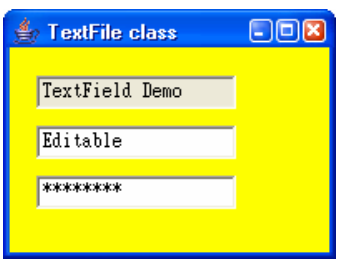
下面的範例是 TextField 的應用：

```
01  // app17_12, TextField的應用
02  import java.awt.*;
03  public class app17_12
04  {
05     static Frame frm=new Frame("TextFile class");
06     static TextField txf1=new TextField("TextField Demo");
07     static TextField txf2=new TextField("Editable");
08     static TextField txf3=new TextField("password");
09
10     public static void main(String args[])
11     {
12        frm.setSize(200,150);
13        frm.setLayout(null);
14        frm.setBackground(Color.yellow);
15        txf1.setBounds(20, 40,120,20);
16        txf2.setBounds(20, 70,120,20);
17        txf3.setBounds(20,100,120,20);
18        txf1.setEditable(false);      // 設定 txf1 為不可編輯
19        txf3.setEchoChar('*');        // 設定 txf3 的回應字元為'*'
20        frm.add(txf1);
21        frm.add(txf2);
```

```
22          frm.add(txf3);
23          System.out.println(txf1.getText());
24          System.out.println(txf2.getText());
25          System.out.println(txf3.getText());
26          frm.setVisible(true);
27      }
28  }
```

```
/* app17_12 OUTPUT--

TextField Demo
Editable
password
----------------------*/
```



17.6.2 TextArea 建構子與 method

文字區（text area）可呈現多行文字，並具有自動換行的功能。下表列出 TextArea 類別常用的建構元與 method：

表 17.6.3 java.awt.TextArea 的建構元與 method

建構元	主要功能
TextArea()	建立文字區
TextArea(int rows, int cols)	建立一個文字區，並指定高與寬分別可供 rows 與 cols 個字元來顯示
TextArea(String text)	建立的文字區，並預設文字為 text
TextArea(String text, int rows, int cols)	建立的文字區，並預設文字及指定大小
TextArea(String text, int rows, int cols, int scrollbars)	建立的文字區，預設文字並指定大小，同時加上捲軸的顯示方式

method	主要功能
void append(String str)	在目前的文字區內的文字之後加上新的文字 str
int getColumns()	取得文字區的寬度（以字元數為單位）

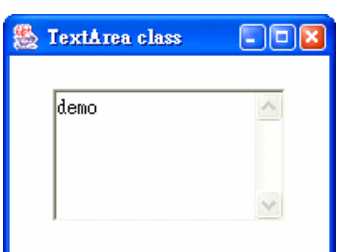
method	主要功能
int getRows()	取得文字區的高度（以字元數為單位）
int getScrollbarVisibility()	取得捲軸的顯示狀態
void insert(String str, int pos)	在文字區的 pos 位置插入 str 字串
void replaceRange(String str, int start, int end)	在文字區內，位置 start 到 end 的文字以字串 str 來取代
void setColumns(int columns)	設定文字區的寬度（以字元數為單位）
void setRows(int rows)	設定文字區可顯示的行數
void setText(String txt)	設定文字區內的文字為 txt

表 17.6.4 java.awt.TextArea 的資料成員 (field)

資料成員 （field）	主要功能
SCROLLBARS_BOTH	設定文字區有垂直與水平捲軸
SCROLLBARS_HORIZONTAL_ONLY	設定文字區只有水平捲軸
SCROLLBARS_NONE	設定文字區沒有捲軸
SCROLLBARS_VERTICAL_ONLY	設定文字區只有垂直捲軸

以一個實例來說明 TextArea 類別的應用：

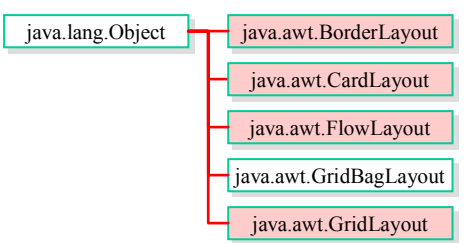
```
01  // app17_13, TextArea 類別的應用
02  import java.awt.*;
03  public class app17_13
04  {
05     static Frame frm=new Frame("TextArea class");
06     static TextArea txa;
07
08     public static void main(String args[])
09     {
10        txa=new TextArea("demo",8,14,TextArea.SCROLLBARS_VERTICAL_ONLY);
11        frm.setLayout(null);                    // 不使用版面配置
12        txa.setBounds(30,45,140,80);            // 設定文字區的大小
13        frm.setSize(200,150);
14        frm.add(txa);
15        frm.setVisible(true);
16     }
17  }
```



17.7 版面配置管理器

版面配置（layout）是指視窗上的物件遵循一定的規則來排列，並會隨著視窗的大小來改變物件大小與位置的一種配置方式。

AWT 提供了 5 個類別來進行版面配置的管理：



17.7.1 使用 BorderLayout 類別

下表列出了 BorderLayout 類別常用的建構元與 method：

表 17.7.1 java.awt.BorderLayout 的建構元與 method

建構元	主要功能
BorderLayout()	建立 BorderLayout 類別的物件
BorderLayout(int hgap, int vgap)	建立 BorderLayout 類別的物件，並設定水平間距為 hgap，垂直間距為 vgap

method	主要功能
int getHgap()	取得 BorderLayout 的水平間距
int getVgap()	取得 BorderLayout 的垂直間距
void removeLayoutComponent(Component comp)	移除 BorderLayout 中的物件 comp
void setHgap(int hgap)	設定 BorderLayout 的水平間距
void setVgap(int vgap)	設定 BorderLayout 的垂直間距

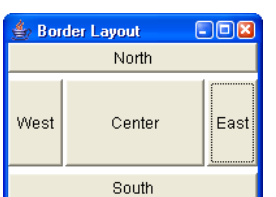
下表列出 BorderLayout 類別常用的成員與其主要功能：

表 17.7.2 java.awt.BorderLayout 類別常用的資料成員

資料成員（field）	主要功能
static String CENTER	將物件放在視窗的中間
static String EAST	將物件放在視窗的右邊
static String NORTH	將物件放在視窗的上方
static String SOUTH	將物件放在視窗的下方
static String WEST	將物件放在視窗的左邊

下面的範例是利用「邊界版面配置」放置按鈕：

```
01  // app17_14, BorderLayout 類別的使用
02  import java.awt.*;
03  public class app17_14
04  {
05     static Frame frm=new Frame("Border Layout");
06     public static void main(String args[])
07     {
08        BorderLayout border=new BorderLayout(2,5);  // 建構元
09        frm.setLayout(border);   // 將版面配置設定為 BorderLayout
10        frm.setSize(200,150);
11        frm.add(new Button("East"),border.EAST);
12        frm.add(new Button("West"),border.WEST);
13        frm.add(new Button("South"),border.SOUTH);
14        frm.add(new Button("North"),border.NORTH);
15        frm.add(new Button("Center"),border.CENTER);
16        frm.setVisible(true);
17     }
18  }
```



17.7.2 使用 CardLayout 類別

多重版面配置（card layout）把每一個物件都視為視窗中的一層，每一個物件都會佔滿整個視窗。

下表列出了常用的建構元與 method：

表 17.7.3 java.awt.CardLayout 的建構元與 method

建構元	主要功能
CardLayout()	建立 CardLayout 類別的物件
CardLayout(int hgap, int vgap)	建立 CardLayout 類別的物件，並設定物件與視窗的水平間距為 hgap，垂直間距為 vgap

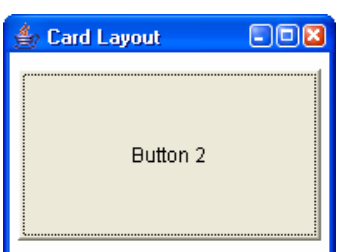
method	主要功能
void first(Container parent)	顯示 Container 中的第一個物件
int getHgap()	取得 CardLayout 的水平間距
int getVgap()	取得 CardLayout 的垂直間距
void last(Container parent)	顯示 Container 中的最後一個物件

method	主要功能
void next(Container parent)	顯示下一個物件
void previous(Container parent)	顯示前一個物件
void removeLayoutComponent (Component comp)	移除 CardLayout 中的物件 comp
void setHgap(int hgap)	設定物件與容器的水平間距
void setVgap(int vgap)	設定物件與容器的垂直 間距
void show(Container parent, String name)	顯示 Container 中名稱為 name 的物件

下面是多重版面配置的簡單範例：

```
01  // app17_15, CardLayout 類別的用法
02  import java.awt.*;
03  public class app17_15
04  {
05     static Frame frm=new Frame("Card Layout");
06     public static void main(String args[])
07     {
08        CardLayout card=new CardLayout(5,10);  // 使用多重版面配置
09        frm.setLayout(card);
10        frm.setSize(200,150);
11        frm.add(new Button("Button 1"),"c1");
12        frm.add(new Button("Button 2"),"c2");
13        frm.add(new Button("Button 3"),"c3");
14        card.show(frm,"c2");
15        frm.setVisible(true);
16     }
17  }
```

將按鈕加入視窗，並賦予名稱



17.7.3 使用 FlowLayout 類別

流動式版面配置（flow layout）可自動依視窗的大小，將物件由左而右、由上而下的次序來排列。

下表列出了 FlowLayout 類別常用的建構元與 method：

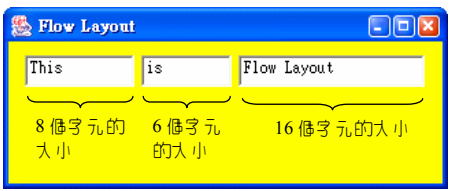
表 17.7.4 java.awt.FlowLayout 的建構元與 method

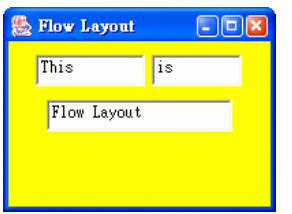
建構元	主要功能
FlowLayout()	建立 FlowLayout 類別的物件，物件置中對齊，物件的垂直與水平間距皆預設為 5 個畫位
FlowLayout(int align)	建立 FlowLayout 類別的物件，物件的垂直與水平間距皆為 5 個畫位，對齊方式可以為 FlowLayout.LEFT、FlowLayout.CENTER 與 FlowLayout.RIGHT，分別代表靠左、置中與靠右對齊
FlowLayout(int align, int hgap, int vgap)	建立 FlowLayout 類別的物件，物件的水平間距為 hgap，垂直間距為 vgap，對齊方式為 align

method	主要功能
int getAlignment()	取得版面配置的對齊方式
int getHgap()	取得物件之間的水平間距
int getVgap()	取得物件之間的垂直間距
void setAlignment(int align)	設定物件的對齊方式為 FlowLayout.LEFT、FlowLayout.CENTER 與 FlowLayout.RIGHT，分別代表靠左、置中與靠右對齊
void setHgap(int hgap)	設定物件的水平間距為 hgap
void setVgap(int vgap)	設定物件的垂直間距為 vgap

下面的範例是在「流動式版面配置」裡建立三個文字方塊：

```
01  // app17_16, FlowLayout 類別的使用
02  import java.awt.*;
03  public class app17_16
04  {
05     static Frame frm=new Frame("Flow Layout");
06     public static void main(String args[])
07     {
08        FlowLayout flow=new FlowLayout(FlowLayout.CENTER,5,10);
09        frm.setLayout(flow);                    // 設定版面配置為流動
式
10        frm.setSize(200,150);
11        frm.setBackground(Color.yellow);
12        frm.add(new TextField("This",8));       // 加入文字方塊
13        frm.add(new TextField("is",6));         // 加入文字方塊
14        frm.add(new TextField("Flow Layout",16)); // 加入文字方塊
15        frm.setVisible(true);
16     }
17  }
```





17.7.4 使用 GridLayout 類別

AWT 利用 GridLayout 類別來處理 Grid Layout 這個類型的版面配置。

下表列出常用的建構元與 method：

表 17.7.5 java.awt.GridLayout 的建構元與 method

建構元	主要功能
GridLayout()	建立 GridLayout 類別的物件，將物件配置在只有一列的數個格子內
GridLayout(int rows, int cols)	建立 GridLayout 類別的物件，將物件配置在 rows 列，cols 行的數個格子內
GridLayout(int rows, int cols, int hgap, int vgap)	建立 GridLayout 類別的物件，將物件配置在 rows 列，cols 行的數個格子內，並指定水平間距為 hgap，垂直間距為 vgap

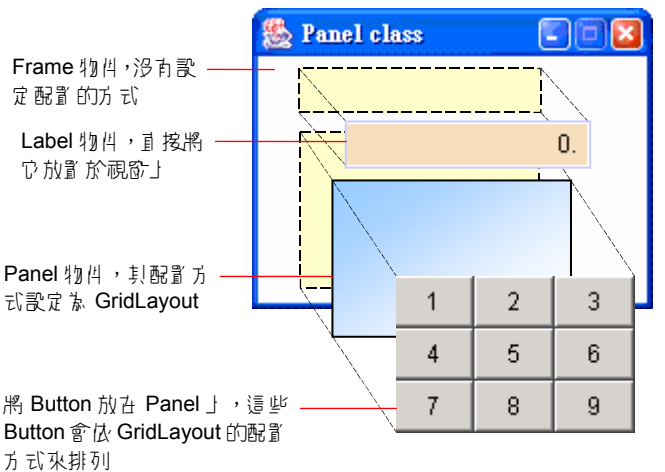
method	主要功能
int getColumns()	傳回物件排列的行數
int getHgap()	傳回物件水平的間距
int getRows()	傳回物件排列的列數
int getVgap()	傳回物件垂直的間距
void setColumns(int cols)	設定物件排列的行數
void setHgap(int hgap)	設定物件水平的間距
void setRows(int rows)	設定物件排列的列數
void setVgap(int vgap)	設定物件垂直的間距

下面的範例是在視窗中，利用 GridLayout 配置 3 列 5 行的按鈕。

```
01  // app17_17, GridLayout 類別的使用
02  import java.awt.*;
03  public class app17_17
04  {
05     static Frame frm=new Frame("Grid Layout");
06     public static void main(String args[])
07     {
08        GridLayout grid=new GridLayout(3,5);  // 3 列 5 行的配置
09        frm.setLayout(grid);
10        frm.setSize(200,150);
11        for(int i=1;i<=15;i++)
12           frm.add(new Button(Integer.toString(i)));  // 加入按鈕
13        frm.setVisible(true);
14     }
15  }
```

17.8 使用 Panel 面板

面板（panel）可以用來封裝物件，如下圖是在一個面板裡配置版面設定 GridLayout，把按鈕放進面板之後，再把面板放進視窗裡：

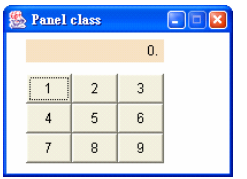


下表列出了 Panel 類別常用的建構元。

表 17.8.1 java.awt.Panel 的建構元

建構元	主要功能
Panel()	建立面板
Panel(layoutManager layout)	建立面板，並指定版面配置方式為 layout

下面的程式示範了如何以 Panel 類別繪出一個小計算機：




```
01  // app17_18, 使用 Panel 類別
02  import java.awt.*;
03  public class app17_18
04  {
05     static Frame frm=new Frame("Panel class");    // 建立視窗 frm
06     static Panel pnl=new Panel(new GridLayout(3,3)); // 建立面板 pnl
07     static Label lab=new Label("0. ",Label.RIGHT); // 建立標籤 lab
08     public static void main(String args[])
09     {
10        frm.setLayout(null);          // 取消視窗的版面設定
11        frm.setSize(200,150);
12
13        frm.setResizable(false);      // 將視窗設定為固定大小
14        lab.setBounds(20,30,120,20);
15        lab.setBackground(new Color(240,220,190)); // 設定標籤的顏色
16        pnl.setBounds(20,60,120,80); // 設定 pnl 置於視窗內的位置
17        for(int i=1;i<=9;i++)
18           pnl.add(new Button(Integer.toString(i)));  // 加入按鈕
19
20        frm.add(lab);   // 將 lab 放進視窗中
21        frm.add(pnl);   // 將面板放進視窗中
22        frm.setVisible(true);
23     }
24  }
```

-The End-