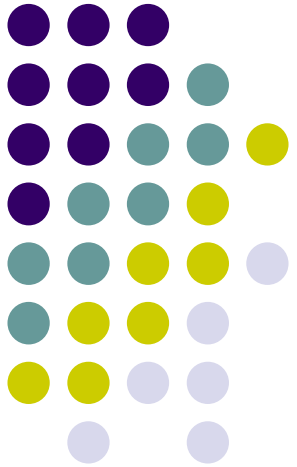
第十七章
AWT視窗物件



認識AWT類別

認識並學習如何建立視窗物件

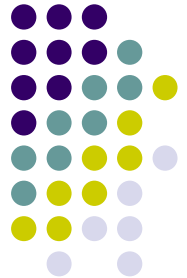
學習如何管理與配置版面

學習Panel類別的使用

簡單的範例 (1/2)

- AWT（Abstract Windowing Toolkit）
  - 處理圖形（包含視窗與繪圖）最基本的方式
  - 下面為AWT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");   } 類別 app17_1 的
06     static Label lab=new Label("Hello Java!!");           } 資料成員
07
08     public static void main(String args[])
09     {
10        frm.setSize(220,150);     // 設定視窗的寬為 220、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  }
```



簡單的範例 (2/2)

- 下圖為app17_1的執行結果：

- 想讓關閉鈕有作用的方法：

這兩行所使用的語法，
就是「匿名內部類別」

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

- 然後在main() method裡的任何位置補上這兩行：

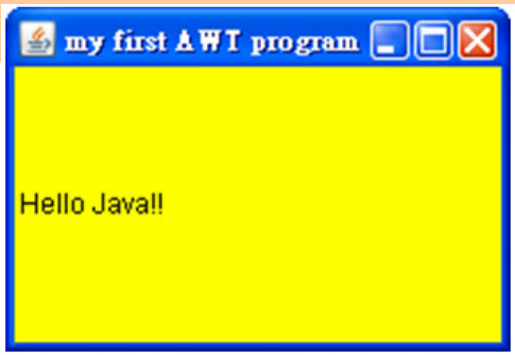
```
frm.addWindowListener(new WindowAdapter(){
    public void windowClosing(WindowEvent e){System.exit(0);}});
```

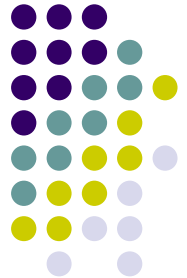
簡單的範例二

- 下面為AWT的另一個範例

```
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(220,150);
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



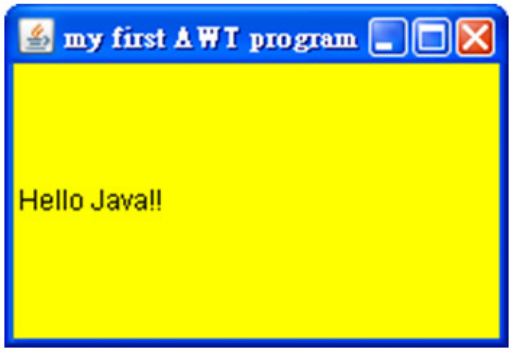


簡單的範例三

- 下面為AWT的第三個範例

```
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(220,150);
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



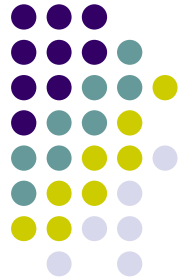
簡單的範例四

- app17_4自行設計一個有引數的建構元，給予視窗標題：

```
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(220,150);
18        frm.setBackground(Color.yellow);
19        frm.setLocation(250,250);
20        frm.setVisible(true);
21     }
22  }
```

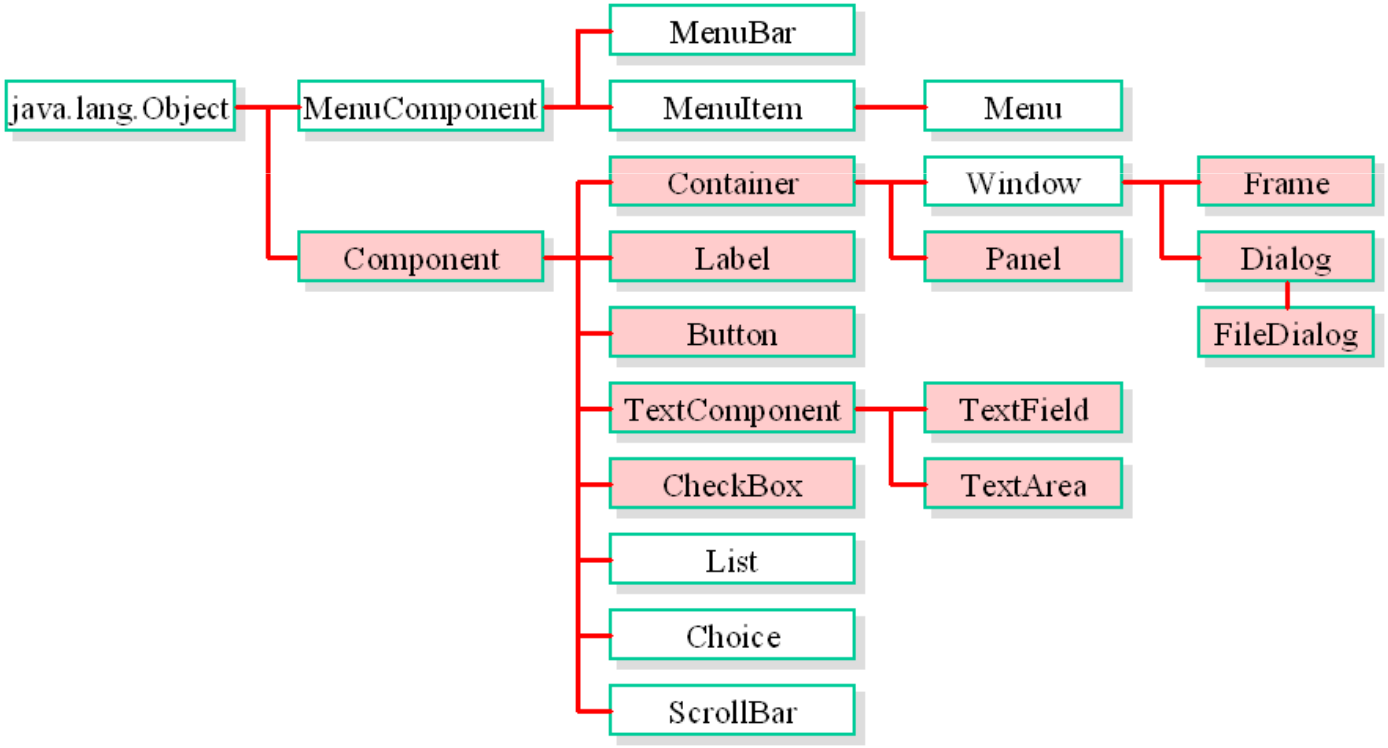
本行相當於呼叫
`Frame("my first AWT program");`

第10行不能寫成`frm.add(lab);`
因為建構元裡不能用物件來呼叫method



java.awt類別庫

- java.awt類別庫的類別，以及它們之間的繼承關係：



java.awt.Component 類別 (1/2)

- 下表列出Component類別較常用的建構元與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 getBackground()	傳回物件的背景顏色
void setBackground(Color color)	設定物件的背景顏色為 color
Rectangle getBounds()	傳回物件所佔的矩形
void setBounds(int x, 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

java.awt.Component 類別 (2/2)

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
void 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
void setName(String name)	設定物件的名稱為 name
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
int getY()	傳回物件在 y 軸方向的座標
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

通常用Component類別的子類別（如Frame、Label、Button等）建立物件

Container類別 (1/2)

- Container類別的物件可用來容納其它的視窗物件
  - 下表列出Container類別常用的建構元與method：

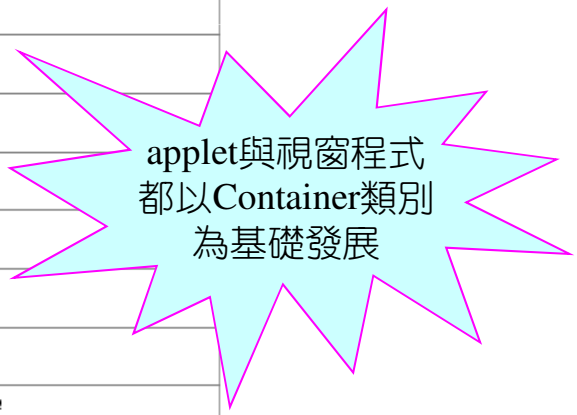
表 17.1.2 java.awt.Container 的建構元與 method

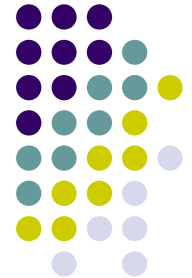
建構元	主要功能
Container()	建立一個 Container 物件

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
void 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
float getAlignmentY()	取得與 y 軸的對齊方式

Container類別 (2/2)

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





Frame類別 (1/2)

- Frame可用來容納其它視窗物件，如按鈕、標籤
  - 下表列出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

Frame類別 (2/2)

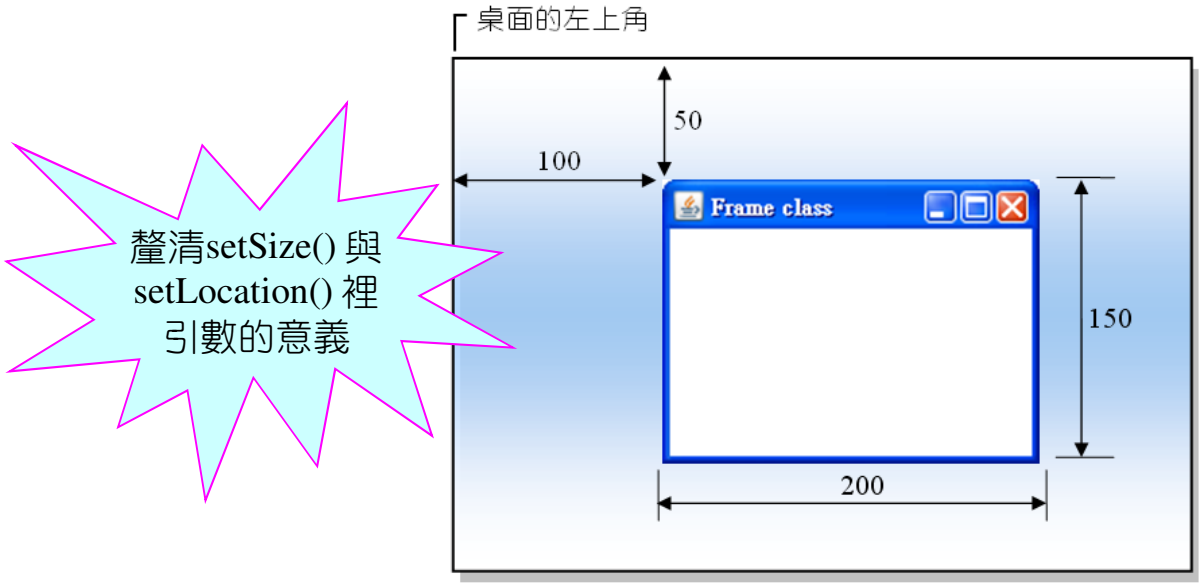
void setState()	設定視窗的狀態，Frame.Normal 代表一般狀態，Frame.ICONIFIED 代表視窗為最小化
MenuBar getMenuBar()	傳回視窗裡的功能表物件
void setMenuBar(Menubar mb)	設定視窗使用的功能表物件為 mb
void remove(MenuCompoent mb)	移除視窗中的功能表物件
String getTitle()	取得視窗的標題
String setTitle(String title)	設定視窗的標題為 title
boolean isResizable()	測試視窗是否可改變大小。若傳回值為 true，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

建立視窗物件

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
--------------------*/
```



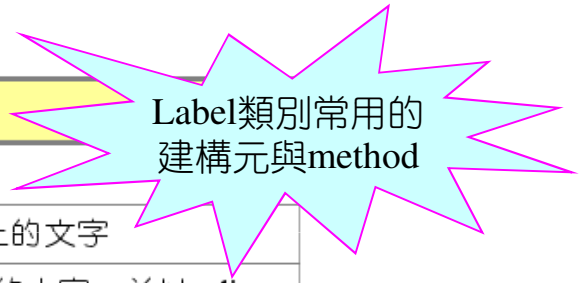
Label類別

- 標籤（label）用來在視窗中顯示文字的文字方塊

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
void setAlignment(int align)	設定標籤內文字的對齊方式，align 的值可為 Label.LEFT、Label.RIGHT 與 Label.CENTER
String getText()	傳回標籤內的文字
void setText(String text)	設定標籤內的文字為 text



加入標籤物件

- 下面的範例是在視窗中加上標籤物件：


```
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);          // 設定視窗底色為粉紅色
12        lab.setText("Hello Java");              // 在標籤內加上文字
13        lab.setBackground(Color.white);         // 設定標籤底色為白色
14        lab.setAlignment(Label.CENTER);         // 將標籤內的文字置中
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  }
```

標籤物件的顏色

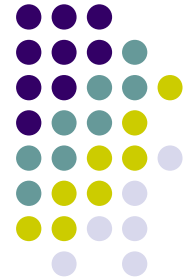
- Color是Java.awt類別庫裡常用的類別
- 要建立一個顏色物件可利用Color() 建構元，格式為：

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

  - r代表紅色（red）
  - g代表綠色（green）
  - b代表藍色（blue）
- 如果標籤的顏色要改成紫色，改成下面的敘述即可：

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

建立顏色物件



標籤物件的字型

- Font類別用來規範物件的字型樣式、大小與字體等
  - 要產生Font類別的物件，可使用Font() 建構元，格式如下：

```
public Font(String font_name, int style, int size)  // Font()建構元
```

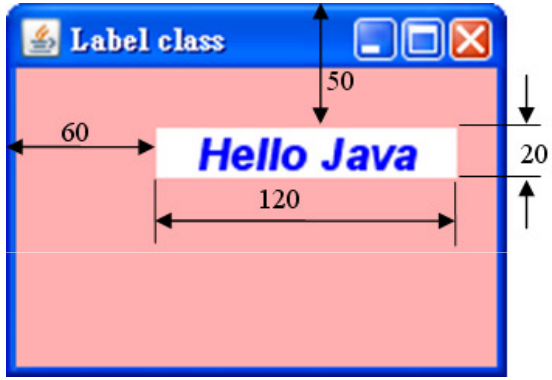
  - font_name為字型名稱
  - style為字型的樣式
  - PLAIN、BOLD與ITALIC都是定義於Font類別裡的類別變數（class variable）
- 想要同時設定粗體與斜體，可用下列的語法來表示：

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

版面配置

- 預設的版面配置為「邊界版面配置

```
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將預設的
版面配置取消

Button類別

- 按鈕（button）可以讓使用者按下它來控制程式執行的流程
- 下表列出Button類別常用的建構元與method：

表 17.4.1　java.awt.Button 的建構元與 method

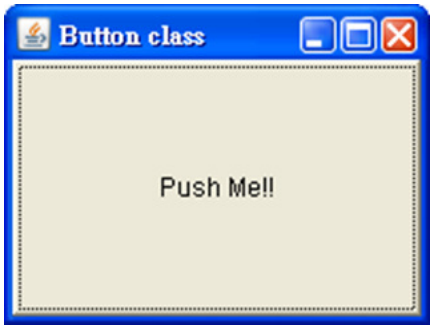
建構元	主要功能
Button()	建立一個沒有標題的按鈕
Button(String title)	建立標題為 title 的按鈕

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

建立按鈕物件 (1/2)

- 以一個簡單的範例來說明Button類別的使用：

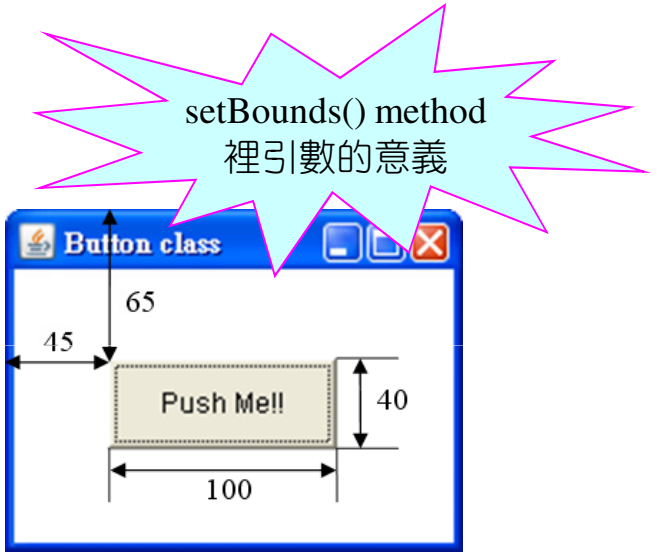
```
01  // app17_8, Button 類別
02  import java.awt.*;
03  public class app17_8
04  {
05     static Frame frm=new Frame("Button class");
06     static Button btn=new Button("Push Me!!");    // 建立按鈕物件
07
08     public static void main(String args[])
09     {
10        frm.setSize(200,150);
11        frm.add(btn);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



建立按鈕物件 (2/2)

- 下面的範例可以設定按鈕的大小：

```
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  }
```



核取方塊（check box）

- 核取方塊可讓使用者選取項目，分為
  - 單選
  - 複選
- 下表列出Checkbox類別常用的建構元與method：

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

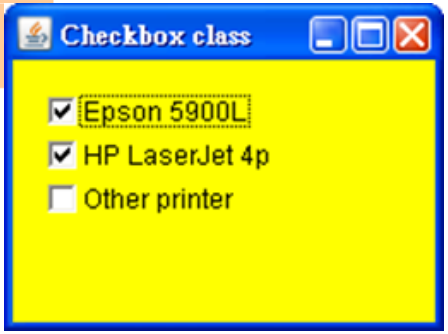
建構元	主要功能
Checkbox()	建立核取方塊
Checkbox(String label)	建立標籤為 label 的核取方塊
Checkbox(String label, boolean state)	建立標籤為 label 的核取方塊，並設定 state 狀態，若 state 為 true，則核取方塊呈被選取狀態
Checkbox(String label, boolean state, CheckboxGroup grp)	建立單選的核取方塊，並將它加入 grp 群組中

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
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：

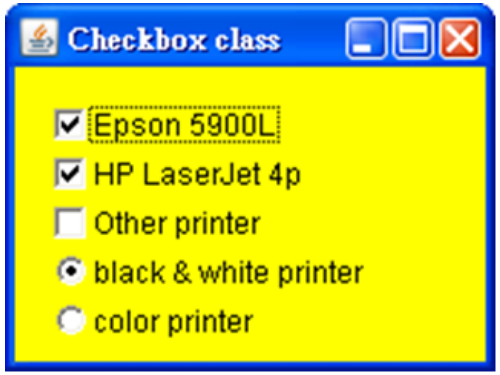
```
01  // app17_10, 核取方塊的應用(一)
02  import java.awt.*;
03  public class app17_10
04  {
05     static Frame frm=new Frame("Checkbox class");
06     static Checkbox ckb1=new Checkbox("Epson 5900L",true);
07     static Checkbox ckb2=new Checkbox("HP LaserJet 4p",true);
08     static Checkbox ckb3=new Checkbox("Other 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  }
```



僅供單選的核取方塊

本範例延伸自app17_10，再加上單選的核取方塊

```
01  // app17_11, 核取方塊的應用(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  }
```



文字輸入的類別

- 處理文字輸入物件的類別
  - TextField類別
  - TextArea類別
- 下表為TextComponent類別裡常用的method：

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

TextField類別的建構元與method

- 下表列出TextField類別的建構元與常用的method：

文字方塊（text field）可以
- ✓ 輸入文字
- ✓ 把輸入的文字轉成特定的符

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

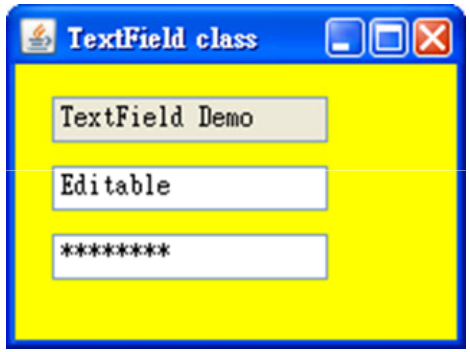
建構元	主要功能
TextField()	建立文字方塊
TextField(int columns)	建立文字方塊，並設定文字方塊的寬度可容納 columns 個字元
TextField(String text)	建立文字方塊，並以 text 為預設的文字
TextField(String text, int columns)	建立文字方塊，以 text 為預設的文字，並設定文字方塊的寬度可容納 columns 個字元

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
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

TextField的應用

本範例是
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("TextField 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
--------------------*/
```

用TextArea建立文字區 (1/2)

- 文字區（text area）
  - 可呈現多行文字，並具有自動換行的功能
  - 與捲軸（scroll bars）搭配，拉動捲軸觀看文件的內容
- 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，並指定列與行分別可供 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

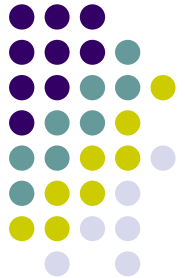
用TextArea建立文字區 (2/2)

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()	取得文字區的行數（以字元數為單位）
int getRows()	取得文字區的列數（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)	設定文字區的行數（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17.6.3 最後一個建構元可設定捲軸的顯示方式：

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

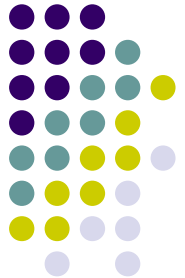
資料成員　（field）	主要功能
SCROLLBARS_BOTH	設定文字區有垂直與水平捲軸
SCROLLBARS_HORIZONTAL_ONLY	設定文字區只有水平捲軸
SCROLLBARS_NONE	設定文字區沒有捲軸
SCROLLBARS_VERTICAL_ONLY	設定文字區只有垂直捲軸



TextArea類別的應用

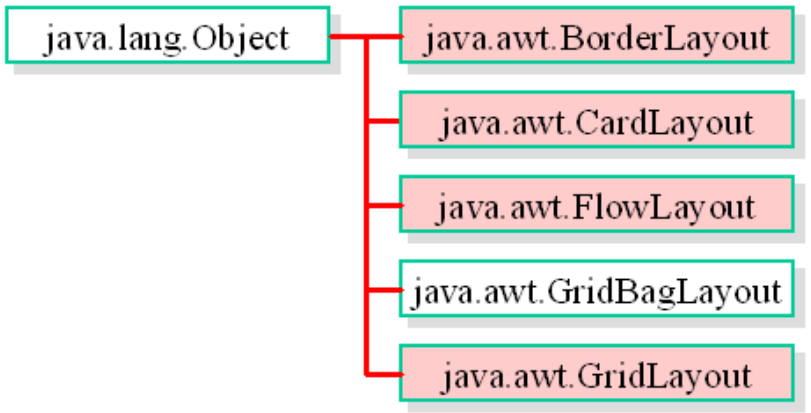
- 以一個實例來說明TextArea類別的應用：

```
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  }
```



版面配置

- 「版面配置」（layout）
  - 是指視窗上的物件遵循一定的規則來排列
  - 會隨著視窗的大小來改變物件大小與位置
- 下圖繪出AWT用來處理版面配置的類別以及它們之間的繼承關係：



使用BorderLayout類別 (1/2)

- 下表列出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

使用BorderLayout類別 (2/2)

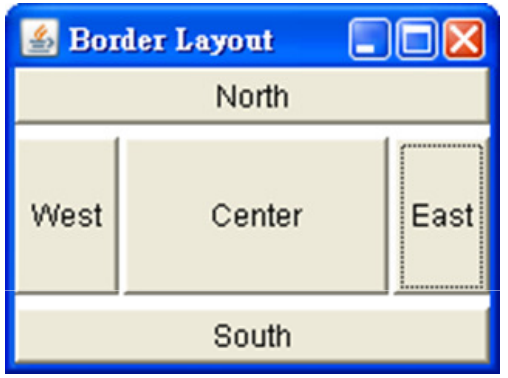
- 使用「邊界版面配置」（border layout）時，
  - 須在add() method裡指定物件擺設的位置
  - 版面上的物件大小會根據視窗的尺寸而定
  - 捲軸常會用「邊界版面配置」
- 下表列出BorderLayout類別常用的成員與其主要功能：

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

邊界版面配置的使用

- 下面是「邊界版面配置」的範例




```
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  }
```

此處frm呼叫的add() method是從父類別Container繼承而來的add() method：

```
void add(Component comp, Object constraints)
```

使用CardLayout類別

- AWT以CardLayout類別處理多層版面配置
  - 下表列出常用的建構元與method：

「多層版面配置」
✓ 把每個物件視為視窗中的一層
✓ 每個物件會佈滿整個視窗

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

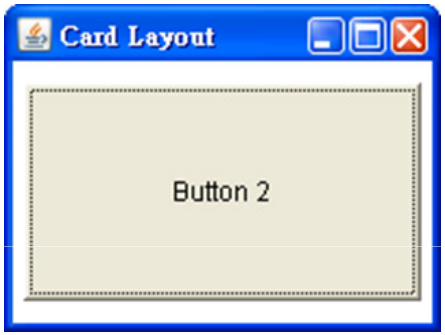
method	主要功能
void first(Container parent)	顯示 Container 中的第一個物件
int getHgap()	取得 CardLayout 的水平間距
int getVgap()	取得 CardLayout 的垂直間距
void last(Container parent)	顯示 Container 中的最後一個物件
void next(Container parent)	顯示下一個物件
void previous(Container parent)	顯示前一個物件
void removeLayoutComponent (Component comp)	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

CardLayout類別的用法

- 來看看下面的簡單的範例：

```
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);      // 使用多層版面配置
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



使用FlowLayout類別

- FlowLayout類別處理流動版面配置的相關事宜
  - 下表列出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 個單位
FlowLayout(int align)	建立 FlowLayout 類別的物件，物件的垂直與水平間距皆為 5 個單位，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

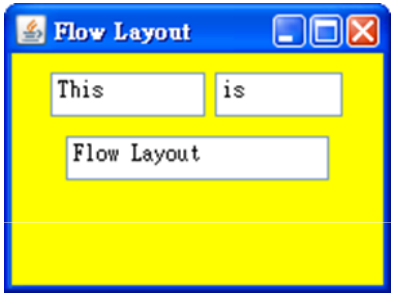
「流動式版面配置」

✓ 依視窗的大小，將物件由左而右、由上而下的次序排列

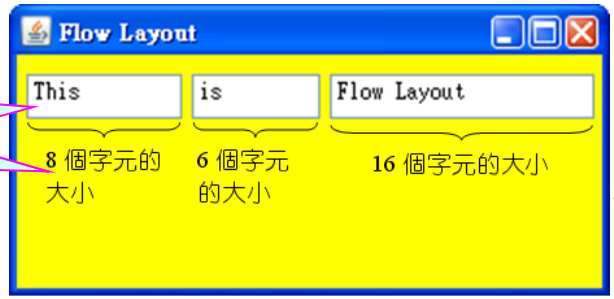
流動式版面配置的範例

- 下面的範例是在「流動式版面配置」裡建立文字方塊

```
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  }
```



視窗拉大後，視窗內的物件也會重新排列



使用GridLayout類別

- AWT利用GridLayout類別處理方格式版面配置
  - 下表列出常用的建構元與method：

「方格式版面配置」
✓ 物件呈方格的形式排列

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

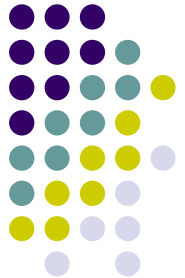
建構元	主要功能
GridLayout()	建立 GridLayout 類別的物件，將物件配置在同一列的數個格子內
GridLayout(int rows, int cols)	建立 GridLayout 類別的物件，將物件配置在 rows 列，cols 行的數個格子內
GridLayout(int rows, int cols, int hgap, int vgap)	建立 GridLayout 類別的物件，將物件配置在 rows 列，cols 行的數個格子內，並指定水平間距為 hgap，垂直間距為 vgap

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

GridLayout類別的使用範例

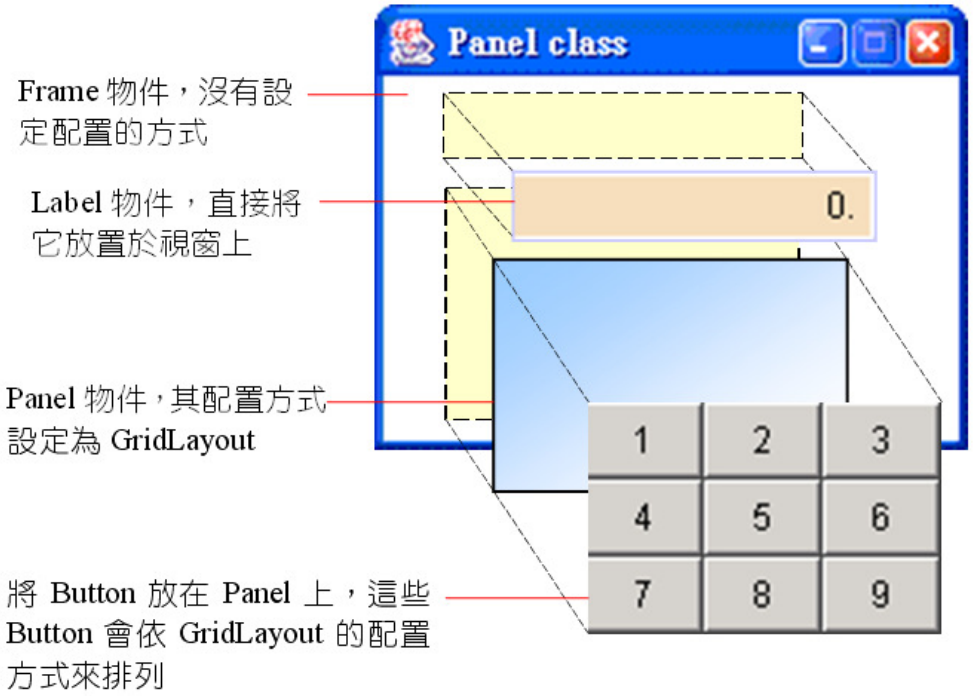
- 下面的範例是在視窗中，配置3列5行的按鈕

```
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  }
```

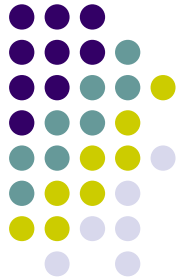


Panel面板 (1/2)

- 面板（panel）可以用來盛裝物件
  - 每個面板都擁有自己的屬性，包括
    - 顏色、大小、版面配置



左圖是計算機的物件、面板與視窗的擺設圖



Panel面板 (2/2)

- AWT利用Panel類別來處理面板的相關運作
  - 下表只列出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

繪製小計算機

- app17_18示範以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  }
```