



# 第十卷 AWT 的具體物件

## 本卷學習目標

- ✚ 學習選擇表單與下拉選單
- ✚ 學習功能表的製作
- ✚ 學習捲軸的製作
- ✚ 學習各種對話方塊與其相關應用





## 19.1 選擇表單

選擇表單 (list) 可用來一次顯示多個選項。

### 19.1.1 List 類別的建構元與選擇表單

下表列出了 List 類別的建構元與常用的 method :

表 19.1.1 java.awt.List 的建構元與 method

建構元	主要功能
List()	建立選擇表單物件，預設的顯示列數為 4 列
List(int rows)	建立可顯示 rows 列的表單物件
List(int rows, boolean multipleMode)	建立可顯示 rows 列的表單物件，並指定單選或複選
method	主要功能
void add(String item)	加入一個選項到選擇表單中
void add(String item, int index)	加入一個選項到選擇表單中的 index 位置



method	主要功能
<code>void addActionListener(ActionListener l)</code>	將傾聽者/加V 選擇表單
<code>void addItemListener(ItemListener l)</code>	將傾聽者/加V 選擇表單裡特定的項目
<code>void deselect(int index)</code>	取消選擇表單裡第 <code>index</code> 項的選取
<code>String getItem(int index)</code>	取得選擇表單裡的第 <code>index</code> 項目
<code>int getItemCount()</code>	取得選擇表單裡選項的數目
<code>String[] getItems()</code>	取得選擇表單裡選項的所有項目，並以字串陣列的形式傳回
<code>int getRows()</code>	取得可顯示的列數
<code>int getSelectedIndex()</code>	傳回被選取項目的 <code>index</code> 值。若未有項目被選取，或者是多個項目被選取，則傳回 <code>-1</code>
<code>int[] getSelectedIndexes()</code>	以整數陣列的型式傳回所有被選取項目的 <code>index</code> 值。若沒有項目被選取，則傳回長度為 <code>0</code> 的陣列
<code>String getSelectedItem()</code>	傳回被選取項目的名稱
<code>String[] getSelectedItems()</code>	取得所有被選取項目的名稱，並以字串陣列的格式傳回
<code>int getVisibleIndex()</code>	取得由 <code>makeVisible()</code> method 所設定的項目



method	主要功能
<code>boolean isIndexSelected(int index)</code>	查詢第 <code>index</code> 個選項是否被選取的狀態
<code>boolean isMultipleMode()</code>	查詢選擇表單是否允許多選
<code>void makeVisible(int index)</code>	設定第 <code>index</code> 個選項顯示在選擇表單裡
<code>void remove(int position)</code>	移除位置為 <code>position</code> 的項目
<code>void remove(String item)</code>	移除名稱為 <code>item</code> 的項目
<code>void removeAll()</code>	移除所有的 <code>item</code> 的項目
<code>void replaceItem(String str, int index)</code>	將位置為 <code>index</code> 的項目用 <code>str</code> 來取代
<code>void select(int index)</code>	設定第 <code>index</code> 項為被選取的狀態
<code>void setMultipleMode(boolean b)</code>	設定選擇表單是否允許複選

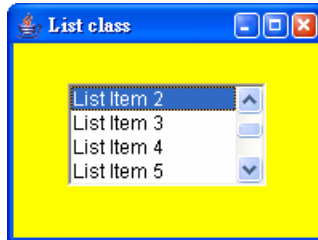


app19\_1 是建立選擇表單的簡單範例。

```
01 // app19_1, 建立選擇表單
02 import java.awt.*;
03 public class app19_1 extends Frame
04 {
05     static app19_1 frm=new app19_1();
06     static List lst=new List(); // 建立選擇表單物件 lst
07     public static void main(String args[])
08     {
09         frm.setLayout(new FlowLayout(FlowLayout.CENTER,10,25));
10         frm.setTitle("List class");
11         for(int i=0;i<=9;i++) // 利用 for 迴圈加入選項
12             lst.add("List Item "+i);
13         lst.select(2); // 選取索引值為 2 的選項
14         frm.setSize(200,150);
15         frm.add(lst);
16         frm.setBackground(Color.yellow);
17         frm.setVisible(true);
18         System.out.println("lst.getRows()= "+lst.getRows());
19         System.out.println("lst.getItemCount()= "+lst.getItemCount());
20     }
21 }
```



```
/* app19_1 OUTPUT-----  
lst.getRows() = 4  
lst.getItemCount() = 10  
-----*/
```





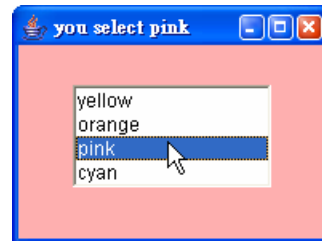
## 19.1.2 選擇表單的物件處理

當選取選擇表單內的選項時，ItemEvent 物件會被觸發。app19\_2 是選擇表單的物件處理範例。

```
01 // app19_2, 選擇表單的物件處理範例
02 import java.awt.*;
03 import java.awt.event.*;
04 public class app19_2 extends Frame implements ItemListener
05 {
06     static app19_2 frm=new app19_2();
07     static List lst=new List(); // 建立選擇表單物件 lst
08     public static void main(String args[])
09     {
10         frm.setLayout(new FlowLayout(FlowLayout.CENTER,10,25));
11         frm.setTitle("Select a color");
12         lst.add("yellow"); // 加入選項到 lst 內
13         lst.add("orange");
14         lst.add("pink");
15         lst.add("cyan");
16         lst.addItemListener(frm); // 設定 frm 為 lst 的傾聽者
17         frm.add(lst);
```



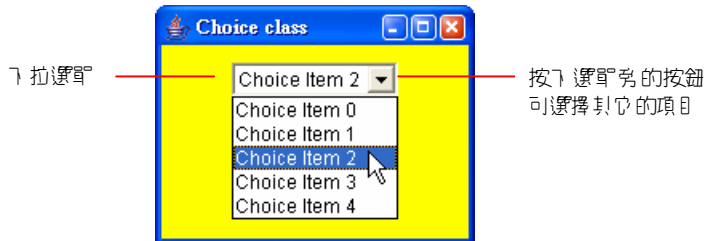
```
18     frm.setSize(200,150);
19     frm.setVisible(true);
20 }
21 public void itemStateChanged(ItemEvent e) // 事件處理的程式碼
22 {
23     String color=lst.getSelectedItem(); // 取得被選取之選項名稱
24     if(color=="yellow") // 如果選項名稱為 yellow
25         frm.setBackground(Color.yellow);
26     else if(color=="orange") // 如果選項名稱為 orange
27         frm.setBackground(Color.orange);
28     else if(color=="pink") // 如果選項名稱為 pink
29         frm.setBackground(Color.pink);
30     else if(color=="cyan") // 如果選項名稱為 cyan
31         frm.setBackground(Color.cyan);
32     frm.setTitle("you select "+color); // 設定視窗 frm 的標題
33 }
34 }
```





## 19.2 下拉選單

下拉選單只能選擇單一 的項目，不能複選：





## 19.2.1 月 Choice 類別의 建構元與 常用의 method

下 表列出了 Choice 類別的建構元與常用의 method :

表 19.2.1 java.awt.Choice 的建構元與 method

建構元	主要功能
Choice()	建立下 拉選擇物件
method	主要功能
void add(String item)	加 V item 到選擇單
void addItemListener(ItemListener l)	加 V 物件傾聽者
String getItem(int index)	取出第 index 項目的名稱
int getItemCount()	傳回下 拉選擇單 選項的總數
int getSelectedIndex()	傳回下 拉選擇單 , 被選取項目的 index 值
String getSelectedItem()	傳回下 拉選擇單 , 被選取項目的名稱
void insert(String item, int index)	在 index 的位置加下 item 選項
void remove(int position)	移除 position 位置的選項

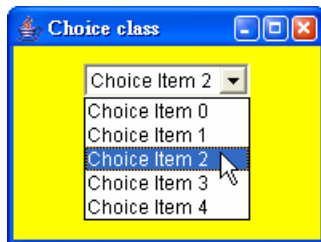


method	主要功能
<code>void remove(String item)</code>	移除名稱為 <code>item</code> 的選項
<code>void removeAll()</code>	移除所有的項目
<code>void select(int pos)</code>	設定 <code>pos</code> 位置的項目為被選取的狀態
<code>void select(String str)</code>	設定名稱為 <code>str</code> 的項目為被選取的狀態



範例 `app19_3` 是以 `Choice` 下拉選單來顯示選項：

```
01 // app19_3, 建立選擇表單
02 import java.awt.*;
03 public class app19_3 extends Frame
04 {
05     static app19_3 frm=new app19_3();
06     static Choice chc=new Choice(); // 建立下拉選單物件 chc
07     public static void main(String args[])
08     {
09         frm.setLayout(new FlowLayout(FlowLayout.CENTER,10,10));
10         frm.setTitle("Choice class");
11         for(int i=0;i<=4;i++) // 利用 for 迴圈加入選項
12             chc.add("Choice Item "+i);
13         frm.add(chc);
14         frm.setSize(200,150);
15         frm.setBackground(Color.yellow);
16         frm.setVisible(true);
17     }
18 }
```





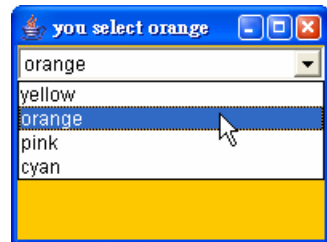
## 19.2.2 下拉選單的物件處理

選取下拉選單內的選項時會觸發 `ItemEvent` 物件。`app19_4` 是下拉選單的物件處理範例。

```
01 // app19_4, 下拉選單的物件處理範例
02 import java.awt.*;
03 import java.awt.event.*;
04 class app19_4 extends Frame implements ItemListener
05 {
06     static app19_4 frm=new app19_4();
07     static Choice chc=new Choice(); // 建立下拉表單物件 chc
08     public static void main(String args[])
09     {
10         BorderLayout br=new BorderLayout();
11         frm.setLayout(br);
12         frm.setTitle("Select a color");
13         chc.add("yellow"); // 加V 選項到下拉表單物件 chc 裡
14         chc.add("orange");
15         chc.add("pink");
16         chc.add("cyan");
17         chc.addItemListener(frm); // 設定 frm 為 chc 的物件傾聽者
```



```
18     frm.setSize(200,150);
19     frm.add(chc,br.NORTH);
20     frm.setVisible(true);
21 }
22 public void itemStateChanged(ItemEvent e)
23 {
24     String color=chc.getSelectedItem();
25     if(color=="yellow")    // 選擇了 yellow 選項
26         frm.setBackground(Color.yellow);
27     else if(color=="orange")    // 選擇了 orange 選項
28         frm.setBackground(Color.orange);
29     else if(color=="pink")      // 選擇了 pink 選項
30         frm.setBackground(Color.pink);
31     else if(color=="cyan")      // 選擇了 cyan 選項
32         frm.setBackground(Color.cyan);
33     frm.setTitle("you select "+color);
34 }
35 }
```



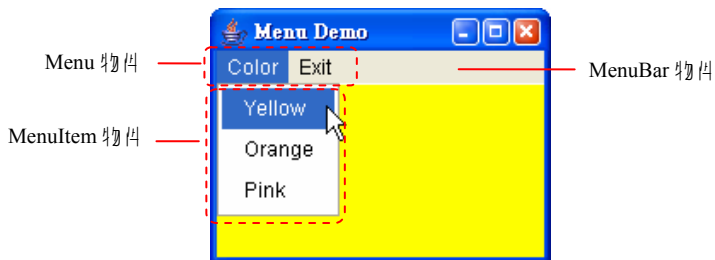


## 19.3 功能表

功能表是由三種功能表類別所建立：

- ✓ MenuBar
- ✓ Menu
- ✓ MenuItem

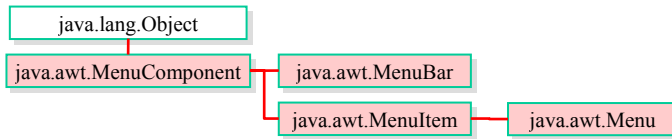
下圖說明了這三種類別所建立之物件於功能表裡所扮演的角色：





MenuItem 與 MenuBar 均是繼承自 MenuComponent 類別。

Menu 則是繼承了 MenuItem 類別，其繼承的關係如下圖所示：





下 表列出了 這三 個功能表類別裡的一些建構子 與 method :

表 19.3.1 java.awt.MenuBar 的建構元與 method

建構元	主要功能
MenuBar()	建立 MenuBar 物件
method	主要功能
Menu add(Menu m)	將 Menu 物件 m 加到 MenuBar 中
Menu getMenu(int i)	取得指定位置的 Menu 物件
int getMenuCount()	取得 MenuBar 中 Menu 物件的總數
void remove(int index)	移除指定位置的 Menu 物件
void remove(MenuComponent m)	移除 MenuComponent 物件 m



表 19.3.2 java.awt.Menu 的建構元與 method

建構元	主要功能
Menu()	建立 Menu 物件
Menu(String label)	建立標題為 label 的 Menu 物件
method	主要功能
MenuItem add(MenuItem mi)	增加 MenuItem 物件 mi 到 Menu 中
void add(String label)	增加標題為 label 的 Menu 物件到 Menu 中
void addSeparator()	在目前的位置增加一行分隔線
MenuItem getItem(int index)	傳回指定位置的 MenuItem 物件
int getItemCount()	傳回目前的 Menu 物件裡，MenuItem 的總數
void insert(MenuItem mi, int index)	在 index 位置插入 MenuItem 物件 mi
void insert(String label, int index)	在 index 位置增加標題為 label 的 Menu 物件
void insertSeparator(int index)	在 index 位置增加一行分隔線
void remove(int index)	移除 index 位置的 MenuItem 物件
void removeAll()	移除 Menu 中所有的 MenuItem 物件



表 19.3.3 java.awt.MenuItem 的建構元與 method

建構元	主要功能
MenuItem()	建立 MenuItem 物件
MenuItem(String label)	建立標題為 label 的 MenuItem 物件
method	主要功能
void addActionListener(ActionListener l)	加入傾聽者物件
String getLabel()	取得 MenuItem 的標題
boolean isEnabled()	查詢 MenuItem 是否可以使用
void setEnabled(boolean b)	設定 MenuItem 可以使用
void setLabel(String label)	設定 MenuItem 的標題為 label



### 19.3.1 建立功能表

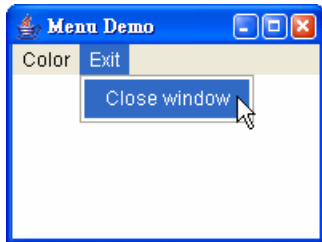
app19\_5 是建立功能表的範例：

```
01 // app19_5, 建立功能表
02 import java.awt.*;
03 public class app19_5
04 {
05     static Frame frm=new Frame("Menu Demo");
06     static MenuBar mb=new MenuBar(); // 建立MenuBar物件
07     static Menu menu1=new Menu("Color"); } 建立Menu物件
08     static Menu menu2=new Menu("Exit"); }
09     static MenuItem mi1=new MenuItem("Yellow"); }
10     static MenuItem mi2=new MenuItem("Orange"); } 建立MenuItem
11     static MenuItem mi3=new MenuItem("Pink"); } 物件
12     static MenuItem mi4=new MenuItem("Close window"); }
13
14     public static void main(String args[])
15     {
16         mb.add(menu1); // 將 menu1 加V mb 中
17         mb.add(menu2); // 將 menu2 加V mb 中
18         menu1.add(mi1); // 將 mi1 加V menu1 中
19         menu1.add(mi2); // 將 mi2 加V menu1 中
```



```
20     menu1.add(mi3);           // 將 mi3 加V menu1 中
21     menu2.add(mi4);           // 將 mi4 加V menu2 中
22     frm.setMenuBar(mb);       // 設定 frm 的功能表為 mb
23     frm.setSize(200,150);
24     frm.setVisible(true);
25 }
26 }
```

下圖為執行本例時，選取 **Exit** 功能表裡 **Close window** 選項的情形：





### 19.3.2 功能表的物件處理

- ✓ 選擇了 MenuItem 類別的物件時便觸發了 ActionEvent 物件。
- ✓ ActionEvent 物件的處理方法可撰寫在 actionPerformed() 裡。

app19\_6 是 處理功能表物件的範例。

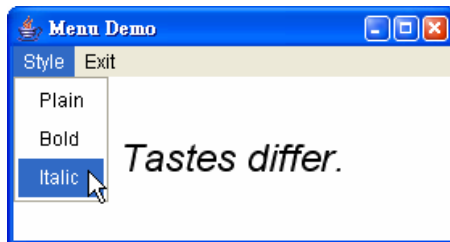
```
01 // app19_6, 功能表的物件處理範例
02 import java.awt.*;
03 import java.awt.event.*;
04 public class app19_6 extends Frame implements ActionListener
05 {
06     static app19_6 frm=new app19_6();
07     static Label lab=new Label("Hello Java",Label.CENTER);
08     static MenuBar mb=new MenuBar(); // 建立 MenuBar 物件
09     static Menu menu1=new Menu("Style"); } 建立 Menu 物件
10     static Menu menu2=new Menu("Exit");
11     static MenuItem mi1=new MenuItem("Plane");
12     static MenuItem mi2=new MenuItem("Bold");
13     static MenuItem mi3=new MenuItem("Italic"); } 建立 MenuItem 物件
```



```
14 static MenuItem mi4=new MenuItem("Close window");
15 public static void main(String args[])
16 {
17     frm.setTitle("Menu Demo");
18     mb.add(menu1);
19     mb.add(menu2);
20     menu1.add(mi1);
21     menu1.add(mi2);
22     menu1.add(mi3);
23     menu2.add(mi4);
24     mi1.addActionListener(frm); // 設定 frm 為 mi1 的物件傾聽者
25     mi2.addActionListener(frm); // 設定 frm 為 mi2 的物件傾聽者
26     mi3.addActionListener(frm); // 設定 frm 為 mi3 的物件傾聽者
27     mi4.addActionListener(frm); // 設定 frm 為 mi4 的物件傾聽者
28     lab.setFont(new Font("Dialog",Font.PLAIN,24));
29     frm.add(lab);
30     frm.setSize(280,150);
31     frm.setMenuBar(mb);
32     frm.setVisible(true);
33 }
34 public void actionPerformed(ActionEvent e) // 物件處理的程式碼
35 {
36     MenuItem mi=(MenuItem) e.getSource(); // 取得觸發物件的物件
```



```
37     if(mi==mi1)           // mi1觸發事件
38         lab.setFont(new Font("Dialog",Font.PLAIN,24));
39     else if(mi==mi2)      // mi2觸發事件
40         lab.setFont(new Font("Dialog",Font.BOLD,24));
41     else if(mi==mi3)     // mi3觸發事件
42         lab.setFont(new Font("Dialog",Font.ITALIC,24));
43     else if(mi==mi4)     // mi4觸發事件
44         frm.dispose();    // 關閉視窗
45 }
46 }
```





## 19.4 捲軸

Scrollbar 類別可用來處理捲軸相關的功能，下表列出了 Scrollbar 類別常用的建構元與 method：

表 19.4.1 java.awt.Scrollbar 的建構元與 method

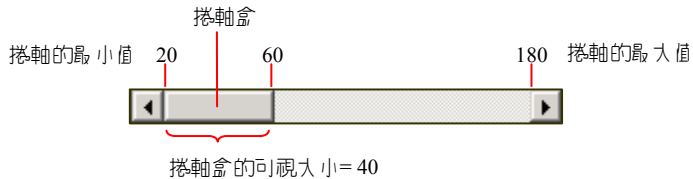
建構元	主要功能
Scrollbar()	建立垂直方向的捲軸
Scrollbar(int orientation)	建立捲軸，並指定方向
Scrollbar(int orientation, int value, int visible, int minimum, int maximum)	建立捲軸，並指定方向、初始值、捲軸盒的可視大小、捲軸的最小與最大值
method	主要功能
void addAdjustmentListener(AdjustmentListener l)	加入 AdjustmentEvent 事件傾聽者
int getMaximum()	取得捲軸的最大值
int getMinimum()	取得捲軸的最小值
int getOrientation()	取得捲軸的方向



method	主要功能
<code>int getValue()</code>	取得目前捲軸的數值
<code>int getVisibleAmount()</code>	取得捲軸盒的可視大小
<code>void setMaximum(int newMaximum)</code>	設定捲軸的最大值
<code>void setMinimum(int newMinimum)</code>	設定捲軸的最小值
<code>void setOrientation(int orientation)</code>	設定捲軸的方向
<code>void setValue(int newValue)</code>	設定目前捲軸的數值
<code>void setValues(int value, int visible, int minimum, int maximum)</code>	設定捲軸的數值、捲軸盒的可視大小、捲軸的最小值與最大值
<code>void setVisibleAmount(int newAmount)</code>	設定捲軸盒的可視大小



下圖中設定捲軸的最小值為 20，最大值為 180，但捲軸盒的大小佔掉了 40，所以這個捲軸只能傳回 20~140 之間的數值。



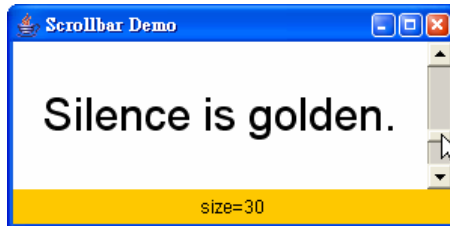


app19\_7 是捲軸的實例應用。

```
01 // app19_7, 捲軸的實例應用
02 import java.awt.*;
03 import java.awt.event.*;
04 public class app19_7 extends Frame implements AdjustmentListener
05 {
06     static app19_7 frm=new app19_7();
07     static Scrollbar scr=new Scrollbar(); // 建立垂直捲軸 scr
08     static Label lab1=new Label("Silence is golden.",Label.CENTER);
09     static Label lab2=new Label("size=20",Label.CENTER);
10
11     public static void main(String args[])
12     {
13         BorderLayout br=new BorderLayout(5,5);
14         frm.setTitle("Scrollbar Demo");
15         frm.setSize(300,150);
16         scr.addAdjustmentListener(frm); // 以 frm 當成 scr 的傾聽者/
17         scr.setValues(20,4,12,40); // 設定 scr 的相關數值
18         frm.add(scr,br.EAST);
19         frm.add(lab1,br.CENTER);
20         frm.add(lab2,br.SOUTH);
21         lab1.setFont(new Font("Dialog",Font.PLAIN,20));
```



```
22     lab2.setBackground(Color.orange);
23     frm.setVisible(true);
24 }
25 public void adjustmentValueChanged(AdjustmentEvent e)
26 {
27     int size=scr.getValue();           // 取得 scr 的數值
28     lab1.setFont(new Font("Dialog",Font.PLAIN,size)); // 設定字型樣式
29     lab2.setText("size="+size);      // 顯示字體大小
30 }
31 }
```

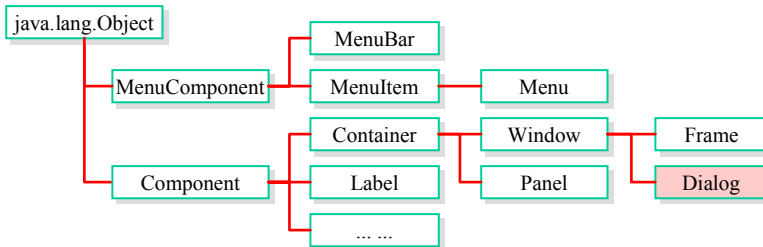




## 19.5 對話方塊

- ✓ 對話方塊 (dialog box) 可用來處理一些簡單的訊息。
- ✓ Dialog 類別可以建立對話方塊。
- ✓ 對話方塊沒有最大化和最小化鈕。

下面為 Dialog 類別的繼承圖：





下表列出了 Dialog 類別常用的建構元和 method :

表 19.5.1 java.awt.Dialog 的建構元與 method

建構元	主要功能
Dialog(Dialog owner)	建立主控對話方塊，指定其擁有者/窗 - 對話方塊
Dialog(Dialog owner, String title)	同上，並加上標題
Dialog(Dialog owner, String title, boolean modal)	建立對話方塊，指定其擁有者/窗 - 對話方塊，並可設定 modal 來指定是否要主控
Dialog(Frame owner)	建立主控對話方塊，指定其擁有者/窗 - 視窗
Dialog(Frame owner, boolean modal)	同上，但可設定 modal 來指定是否要主控
Dialog(Frame owner, String title)	建立主控對話方塊，指定其擁有者/窗 - 視窗，並可設定標題
Dialog(Frame owner, String title, boolean modal)	建立對話方塊，指定其擁有者/窗 - 視窗，設定標題，並可設定 modal 來指定是否要主控



method	主要功能
<code>void dispose()</code>	銷毀對話方塊物件
<code>String getTitle()</code>	取得對話方塊的標題
<code>void hide()</code>	隱藏對話方塊
<code>boolean isModal()</code>	測試對話方塊是否為主控
<code>boolean isResizable()</code>	測試對話方塊是否可以改變大小
<code>void setModal(boolean b)</code>	設定對話方塊為主控
<code>void setResizable(boolean resizable)</code>	設定對話方塊是否可以改變大小
<code>void setTitle(String title)</code>	設定對話方塊的標題
<code>void show()</code>	顯示對話方塊

對話方塊必須指定其擁有者 (owner)。

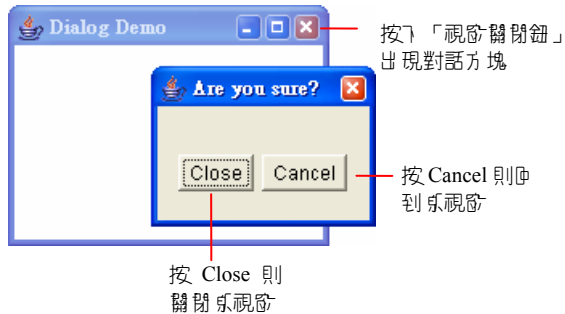


有兩點必須注意：

- (1) 對話方塊有「主控」與「非主控」之分。
- (2) 對話方塊要呼叫 `show() method` 才能顯示它。如果要隱藏，可用 `hide() method`。



app19\_8 是對話方塊的使用範例。



```
01 // app19_8, 對話方塊的實例應用
02 import java.awt.*;
03 import java.awt.event.*;
04 public class app19_8 extends Frame implements ActionListener
05 {
06     static app19_8 frm=new app19_8();
07     static Dialog dlg=new Dialog(frm); // 建立 Dialog 物件 dlg
08     static Button Close_btn=new Button("Close"); // Close 按鈕
09     static Button Cancel_btn=new Button("Cancel"); // Cancel 按鈕
10     static WinLis wlis=new WinLis(); // 建立傾聽者物件 wlis
```



```
11
12 public static void main(String args[])
13 {
14     frm.setTitle("Dialog Demo");
15     frm.setSize(200,150);
16     dlg.setTitle("Are you sure?");    // 設定對話方塊的標題
17     dlg.setSize(140,100);           // 設定對話方塊的大小
18     dlg.setLayout(new FlowLayout(FlowLayout.CENTER,5,30));
19     dlg.add(Close_btn);             // 將 Close_btn 加入對話方塊中
20     dlg.add(Cancel_btn);           // 將 Cancel_btn 加入對話方塊中
21     Cancel_btn.addActionListener(frm); // 設定 Cancel_btn 的傾聽者
22     Close_btn.addActionListener(frm); // 設定 Close_btn 的傾聽者
23     frm.addWindowListener(wlis);    // 設定 frm 的傾聽者
24     frm.setVisible(true);
25 }
26
27 static class WinLis extends WindowAdapter
28 {
29     public void windowClosing(WindowEvent e) // 按下視窗關閉按鈕時
30     {
31         dlg.setLocation(80,30);    // 設定對話方塊的位置
32         dlg.show();                // 顯示對話方塊
33     }
```



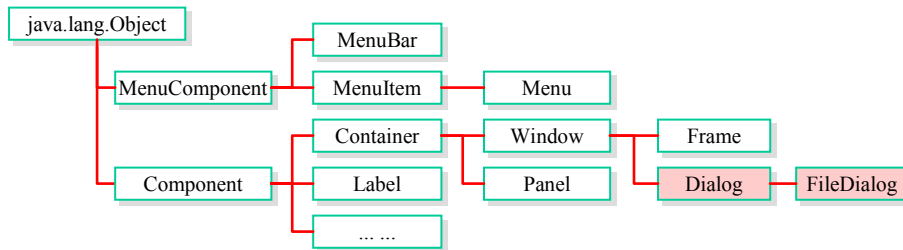
```
34     }
35
36     public void actionPerformed(ActionEvent e) // 按下「對話方塊」的按鈕時
37     {
38         Button btn=(Button) e.getSource(); // 取得被按下的按鈕
39         if(btn==Close_btn) // 如果是 Close 按鈕被按下
40         {
41             dlg.dispose(); // 關閉對話方塊
42             frm.dispose(); // 關閉視窗
43         }
44         else if (btn==Cancel_btn) // 如果是 Cancel 按鈕被按下
45             dlg.hide(); // 隱藏對話方塊
46     }
47 }
```



## 19.6 檔案對話方塊

檔案對話方塊（file dialog）專門用來處理檔案存取等相關事務。

FileDialog 類別可以建立檔案對話方塊，它繼承自 Dialog 類別：





下表列出了 FileDialog 常用的建構元與 method :

表 19.6.1 java.awt.Dialog 的建構元與 method

資料成員	主要功能
static int LOAD	檔案對話方塊為讀取檔案用
static int SAVE	檔案對話方塊為儲存檔案用
建構元	主要功能
FileDialog(Frame parent)	建立讀取用的檔案對話方塊
FileDialog(Frame parent, String title)	建立讀取用的檔案對話方塊，並建立標題
FileDialog(Frame parent, String title, int mode)	建立檔案對話方塊，建立標題，並可指定是儲存或讀取用，若 mode 為 LOAD，則為讀取用，若為 SAVE，則為儲存用
method	主要功能
String getDirectory()	取得檔案對話方塊內所選取的目錄
String getFile()	取得檔案對話方塊內所選取的檔案名稱



method	主要功能
int getMode()	取得檔案對話方塊的模式為 LOAD 或 SAVE
void setDirectory(String dir)	設定 檔案對話方塊的預設開啟目錄
void setFile(String file)	設定 檔案對話方塊的預設開啟檔案
void setMode(int mode)	設定 檔案對話方塊的模式



app19\_8 是使用 FileDialog 類別的範例，其執行流程如下：

(1) 按下 Open 按鈕，開啟檔案對話方塊

(2) 選取檔案所在的目錄

(3) 選取檔案

(4) 按下 開啟鈕

(5) TextArea 物件會顯示出所選擇檔案(train.txt)的內容



```
01 // app19_9, FileDialog 類別的複用
02 import java.awt.*;
03 import java.awt.event.*;
04 import java.io.*;
05 public class app19_9 extends Frame implements ActionListener
06 {
07     static app19_9 frm=new app19_9();
08     static FileDialog fdlg=new FileDialog(frm,"Open"); // 建立 fdlg 物件
09     static Button btn=new Button("Open");
10     static TextArea txa=new TextArea();
11
12     public static void main(String args[])
13     {
14         BorderLayout br=new BorderLayout();
15         frm.setTitle("Dialog Demo");
16         frm.setLayout(br);
17         frm.setSize(200,150);
18         frm.add(txa,br.CENTER);
19         frm.add(btn,br.SOUTH);
20         btn.addActionListener(frm); // 設定 frm 為 btn 的物件傾聽者
21         frm.setVisible(true);
22     }
23 // 當 Open 按鈕按下時，執行下列的程式碼
```



```
24 public void actionPerformed(ActionEvent e)
25 {
26     fdlg.setVisible(true); // 顯示檔案對話方塊
27     // 以下的程式碼是按下檔案對話方塊內的「開啟」鈕之後，才會執行
28     String fname=fdlg.getDirectory()+fdlg.getFile();// 取得路徑與名稱
29     try
30     {
31         FileInputStream fi=new FileInputStream(fname);
32         byte ba[]=new byte[fi.available()];
33         fi.read(ba); // 讀取檔案內容到byte陣列裡
34         txa.setText(new String(ba)); // 將byte陣列內容寫到txa物件裡
35         fi.close();
36     }
37     catch(IOException ioe){};
38 }
39 }
```



-The End-