Swing（三）

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

鍵盤事件

- 在視窗應用程式中，當使用者用鍵盤輸入資料時，其按下或鬆開按鍵皆會觸動鍵盤事件。
- Java在鍵盤的傾聽機制，有兩種方式來處理觸發的事件，分別是「實作KeyListener介面」與「繼承KeyAdapter類別」。
  - 實作KeyListener介面
    1. 需在程式開頭處用import來匯入該套件及其中所有類別。

       import java.awt.event.*;

    2. 定義一個繼承JFrame類別且實作事件傾聽者介面的子類別。

鍵盤事件

3. 對事件來源物件註冊事件傾聽者（即加入傾聽者物件到元件中）。
4. 實作KeyListener介面中的keyPressed(KeyEvent e)、keyTyped(KeyEvent e)及keyReleased(KeyEvent e)三個抽象方法。

   注意：

   - 既使沒有用到，但還是要建立「空」的方法，否則在編譯時，會產生錯誤。

鍵盤事件

- 步驟2～步驟4：

```
class 子類別名稱 extends JFrame implemets ActionListenter
{
    ⋮
  建構子()
  {
                ⋮
    事件來源的物件.addKeyListener(傾聽者物件名稱 );
                ⋮
  }
    ⋮
  class 傾聽者物件名稱 implements KeyListener
  {
      實作三個抽象方法
                ⋮
   }
      ⋮
}
```

鍵盤事件

- 步驟2~步驟4，亦可更改為：

```
class 子類別名稱 extends JFrame implemets ActionListenter
{
    ⋮
  建構子()
  {
                ⋮
    事件來源的物件.addKeyListener(new 傾聽者物件名稱() );
                ⋮
  }
    ⋮
  KeyListener 傾聽者物件名稱 = new KeyListener()
  {
      實作三個抽象方法
                ⋮
  };
      ⋮
}
```

鍵盤事件

- 繼承KeyAdapter類別
  1. 需在程式開頭處用import來匯入該套件及其中所有類別。

     import java.awt.event.*;

  2. 定義一個繼承JFrame類別且實作事件傾聽者介面的子類別。
  3. 對事件來源物件註冊事件傾聽者（即加入傾聽者物件到元件中）。
  4. 定義一個繼承KeyAdapter類別的子類別，並建立處理鍵盤事件的方法。

鍵盤事件

- 步驟2～步驟4：

```
class 子類別名稱 extends JFrame implemets ActionListenter
{
    ⋮
  建構子()
  {
                ⋮
    事件來源的物件.addKeyListener(new 傾聽者物件名稱());
                ⋮
  }
    ⋮
  class 傾聽者物件名稱  extends KeyAdapter
  {
     處理鍵盤事件的方法
                ⋮
   }
      ⋮
}
```

鍵盤事件

- KeyListener介面與KeyAdapter類別皆提供了三種鍵盤處理事件抽象方法：
  - void keyPressed(KeyEvent e)
    - 當鍵盤按鍵被按下時所觸發的事件。
  - void keyReleased(KeyEvent e)
    - 當鍵盤已按下的按鍵被放開時所觸發的事件。
  - void keyTyped(KeyEvent e)
    - 鍵入字元事件，即按下與放開的整個過程。
    - 此時，會產生按鍵的對應字元。
    - 有些按鍵不包括在內。

      例如：

      Alt、Ctrl、Shift、…等。

鍵盤事件

注意：

三個抽象方法均須實作，否則在編譯時會產生錯誤。

- 鍵盤事件類別KeyEvent常用的方法：
  - int getkeyCode()
    - 當按下或放開了某一個按鍵，會傳回一個整數型態的鍵盤碼keyCode。
    - 鍵盤事件類別KeyEvent為常用的按鍵定義了常數代碼，如下：
      - 0 ~ 9數字：
        - KeyEvent.VK_0 ~ KeyEvent.VK_9。
      - A ~ Z字母：
        - KeyEvent.VK_A ~ KeyEvent.VK_Z。

鍵盤事件

- 方向鍵：
  - KeyEvent.VK_UP、KeyEvent.VK_DOWN、KeyEvent.VK_LEFT、KeyEvent.VK_RIGHT。
- 常用鍵：
  - KeyEvent.VK_SHIFT、KeyEvent.VK_CONTROL、KeyEvent.VK_ALT、KeyEvent.VK_ENTER…等。

- String getKeyText(int keyCode)
  - 傳回按鍵名稱字串。
- char getkeyChar()
  - 當按下或放開了某一個按鍵，傳回該按鍵的字元。
- boolean isShiftDown()
  - 當Shift鍵被按時，傳回值true。
- boolean isControlDown()
  - 當Ctrl鍵被按時，傳回值true。

鍵盤事件

- boolean isAltDown()
  - 當Alt鍵被按時，傳回值true。

鍵盤事件

- 程式（實作KeyListener介面）：

```
package CH09_13;

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

class CKeyF extends JFrame implements ActionListener
{
    private JLabel lblKey, lblChar;
    private JTextField txtInput;
    private JButton btnReset;

    public CKeyF()
    {
        setTitle("鍵盤事件偵測");
        setLayout(null);
        setBounds(500, 400, 220, 180);
        setVisible(true);
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
```

鍵盤事件

```
lblKey = new JLabel("按鍵");
lblKey.setBounds(20, 10, 150, 30);
add(lblKey);

txtInput = new JTextField("");
txtInput.setBounds(10, 50, 180, 30);
add(txtInput);
txtInput.addKeyListener(new KeyListInput());

lblChar = new JLabel("字元");
lblChar.setBounds(20, 90, 150, 30);
add(lblChar);

btnReset = new JButton("重來");
btnReset.setBounds(120, 95, 60, 20);
add(btnReset);
btnReset.addActionListener(this);

repaint();
```

鍵盤事件

```
}

class KeyListInput implements KeyListener
{
    public void keyPressed(KeyEvent e)
    {
        lblKey.setText("按鍵：[ " + KeyEvent.getKeyText(e.getKeyCode()) +
                       "] 鍵被按下");
    }

    public void keyTyped(KeyEvent e)
    {
        lblChar.setText("字元：" + e.getKeyChar());
    }

    public void keyReleased(KeyEvent e)
    {
    }
}
```

鍵盤事件

```
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == btnReset)
        {
            txtInput.setText("");
            lblKey.setText("按鍵：");
            lblChar.setText("字元：");
        }
    }
}

public class CH09_13
{
    public static void main(String[] args)
    {
        CKeyF keyF = new CKeyF();
    }
}
```

鍵盤事件

```
1  package CH09_13;
2
3  import javax.swing.*;
4  import java.awt.event.*;
5
6  class CKeyF extends JFrame implements ActionListener
7  {
8     private JLabel lblKey, lblChar;
9     private JTextField txtInput;
10    private JButton btnReset;
11
12    public CKeyF()
13    {
14       setTitle("鍵盤事件偵測");
15       setLayout(null);
16       setBounds(500, 400, 220, 180);
17       setVisible(true);
18       setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
19
20       lblKey = new JLabel("按鍵");
21       lblKey.setBounds(20, 10, 150, 30);
22       add(lblKey);
23
24       txtInput = new JTextField("");
25       txtInput.setBounds(10, 50, 180, 30);
26       add(txtInput);
```

鍵盤事件

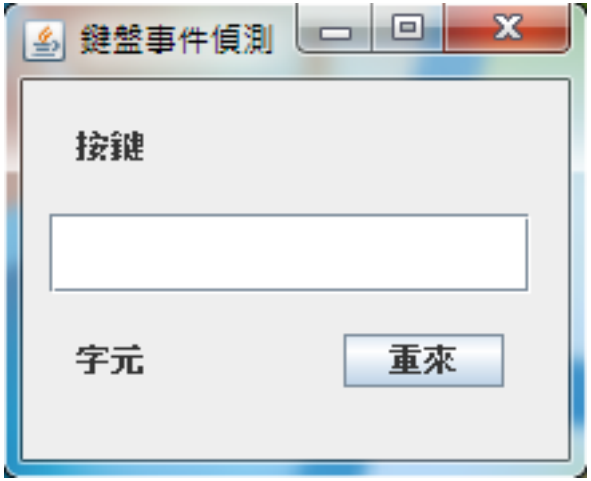
```
27        txtInput.addKeyListener(new KeyListInput());
28
29        lblChar = new JLabel("字元");
30        lblChar.setBounds(20, 90, 150, 30);
31        add(lblChar);
32
33        btnReset = new JButton("重來");
34        btnReset.setBounds(120, 95, 60, 20);
35        add(btnReset);
36        btnReset.addActionListener(this);
37
38        repaint();
39    }
40
41    class KeyListInput implements KeyListener
42    {
43        public void keyPressed(KeyEvent e)
44        {
45            lblKey.setText("按鍵：[ " + KeyEvent.getKeyText(e.getKeyCode()) + " ] 鍵被按下");
46        }
47
48        public void keyTyped(KeyEvent e)
49        {
50            lblChar.setText("字元：" + e.getKeyChar());
51        }
52
```

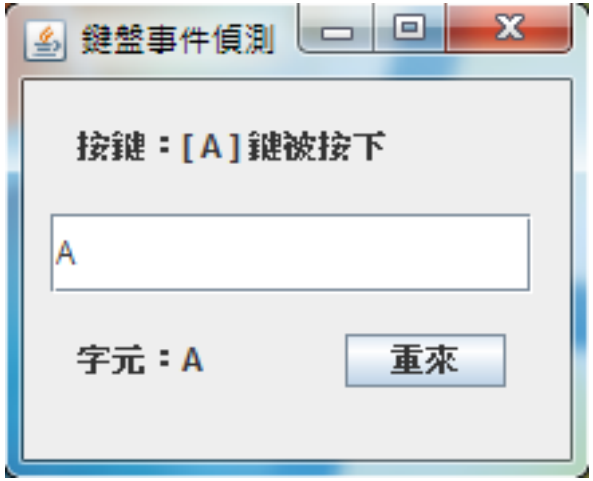
鍵盤事件

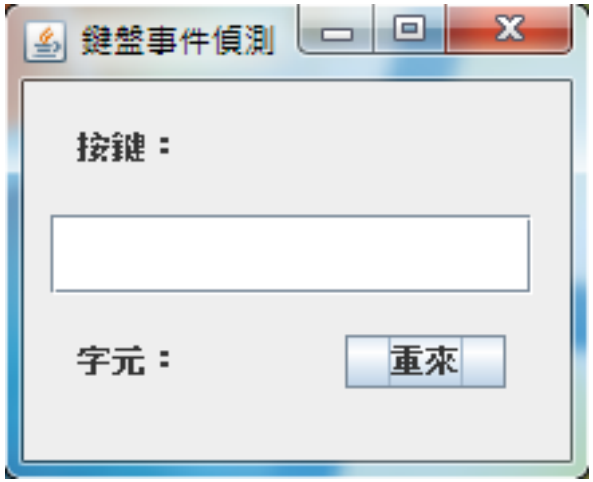
```
53        public void keyReleased(KeyEvent e)
54        {
55        }
56     }
57
58     public void actionPerformed(ActionEvent e)
59     {
60        if (e.getSource() == btnReset)
61        {
62           txtInput.setText("");
63           lblKey.setText("按鍵：");
64           lblChar.setText("字元：");
65        }
66     }
67  }
68
69  public class CH09_13
70  {
71     public static void main(String[] args)
72     {
73        CKeyF keyF = new CKeyF();
74     }
75  }
```

鍵盤事件

- 執行結果：







鍵盤事件

- 說明：
  - 行01：
    - 定義「套件（package）」。
  - 行03：
    - 載入「javax.swing.*」套件。
  - 行04：
    - 載入「java.awt.event.*」套件。
  - 行06~行67：
    - 建立繼承自「JFrame」父類別，並實作「ActionListener」介面的「CKeyF」子類別。
    - 行08~行10：
      - 宣告私有物件成員。
    - 行12~行39：
      - 宣告類別「建構子」「CKeyF」。

鍵盤事件

- 行14~行18：
  - 建立視窗物件。
- 行20~行22：
  - 建立標籤（lblKey）物件。
- 行24~行27：
  - 設定單行文字欄位（txtInput）物件。
  - 行27：

    建立事件來源物件連結事件傾聽者。

    KeyListInput是指自訂的傾聽者物件。
- 行29~行31：
  - 建立標籤（lblChar）物件。
- 行33~行36：
  - 設定按鈕（btnReset）物件。

鍵盤事件

- 行36：

  設定事件來源物件連結事件傾聽者。

  this是指CKeyF類別所建立的視窗物件。

- 行38：
  - 重繪視窗元件。
- 行41~行56：
  - 建立「KeyListInput 」類別，並實作「KeyListener」介面。
  - KeyListener有三個抽象方法，必須要實作。
  - 行43~行46：

    實作keyPressed(KeyEvent e)方法。

    用setText()方法，設定lblKey的值。
  - 行48~行51：

    實作keyTyped(KeyEvent e)方法。

    用setText()方法，設定lblChar的值。

鍵盤事件

注意：

keyTyped(KeyEvent e)無法對應（顯示）Alt、Ctrl、…等按鍵。

- 行53~行55：

  實作keyReleased(KeyEvent e)方法。

注意：

這個方法目前沒有處理事件的敘述（即「空」方法），但依然必須實作，否則編譯時會出現錯誤。

- 行58~行66：
  - 實作ActionListener介面的actionPerformed()方法。
  - 為觸發事件時的處理方法。
  - ActionEvent被觸發時，e就是事件的來源。
  - 行60：

    當e.getSource()取得的事件來源物件為btnReset時，執行此區段程式。

鍵盤事件

行62~行64：

用setText()方法設定txtInput、lblKey、lblChar的值。

- 行73：
  - 利用「CKeyF」的「建構子」，建立視窗物件。

鍵盤事件

- 程式（實作KeyListener介面）：

```
package CH09_14;

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

class CKeyF extends JFrame implements ActionListener
{
    private JLabel lblKey, lblChar;
    private JTextField txtInput;
    private JButton btnReset;

    public CKeyF()
    {
        setTitle("鍵盤事件偵測");
        setLayout(null);
        setBounds(500, 400, 220, 180);
        setVisible(true);
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
```

鍵盤事件

```
lblKey = new JLabel("按鍵");
lblKey.setBounds(20, 10, 150, 30);
add(lblKey);

txtInput = new JTextField("");
txtInput.setBounds(10, 50, 180, 30);
add(txtInput);
txtInput.addKeyListener(KeyListInput);

lblChar = new JLabel("字元");
lblChar.setBounds(20, 90, 150, 30);
add(lblChar);

btnReset = new JButton("重來");
btnReset.setBounds(120, 95, 60, 20);
add(btnReset);
btnReset.addActionListener(this);

repaint();
```

鍵盤事件

```
}

KeyListener KeyListInput = new KeyListener()
{
    public void keyPressed(KeyEvent e)
    {
        lblKey.setText("按鍵：[ " + KeyEvent.getKeyText(e.getKeyCode()) +
                       "] 鍵被按下");
    }

    public void keyTyped(KeyEvent e)
    {
        lblChar.setText("字元：" + e.getKeyChar());
    }

    public void keyReleased(KeyEvent e)
    {
    }
};
```

鍵盤事件

```
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == btnReset)
        {
            txtInput.setText("");
            lblKey.setText("按鍵：");
            lblChar.setText("字元：");
        }
    }
}

public class CH09_14
{
    public static void main(String[] args)
    {
        CKeyF keyF = new CKeyF();
    }
}
```

鍵盤事件

```
1  package CH09_14;
2
3  import javax.swing.*;
4  import java.awt.event.*;
5
6  class CKeyF extends JFrame implements ActionListener
7  {
8     private JLabel lblKey, lblChar;
9     private JTextField txtInput;
10    private JButton btnReset;
11
12    public CKeyF()
13    {
14       setTitle("鍵盤事件偵測");
15       setLayout(null);
16       setBounds(500, 400, 220, 180);
17       setVisible(true);
18       setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
19
20       lblKey = new JLabel("按鍵");
21       lblKey.setBounds(20, 10, 150, 30);
22       add(lblKey);
23
24       txtInput = new JTextField("");
25       txtInput.setBounds(10, 50, 180, 30);
26       add(txtInput);
```

鍵盤事件

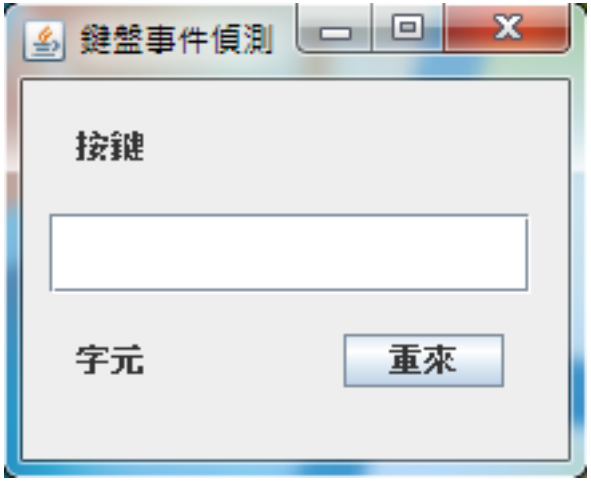
```
27        txtInput.addKeyListener(KeyListInput);
28
29        lblChar = new JLabel("字元");
30        lblChar.setBounds(20, 90, 150, 30);
31        add(lblChar);
32
33        btnReset = new JButton("重來");
34        btnReset.setBounds(120, 95, 60, 20);
35        add(btnReset);
36        btnReset.addActionListener(this);
37
38        repaint();
39    }
40
41    KeyListener KeyListInput = new KeyListener()
42    {
43        public void keyPressed(KeyEvent e)
44        {
45            lblKey.setText("按鍵：[ " + KeyEvent.getKeyText(e.getKeyCode()) + " ] 鍵被按下");
46        }
47
48        public void keyTyped(KeyEvent e)
49        {
50            lblChar.setText("字元：" + e.getKeyChar());
51        }
52
```

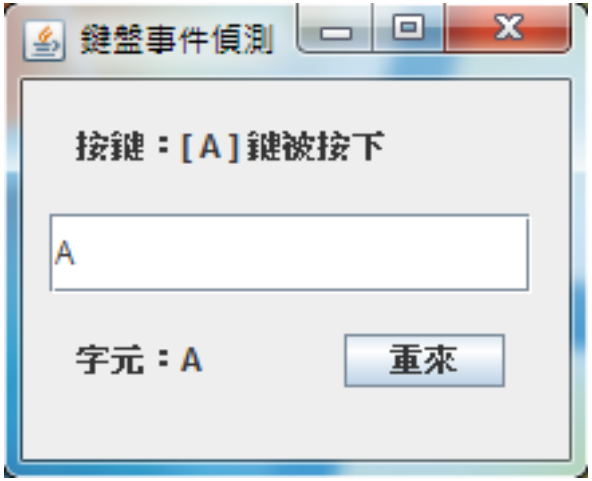
鍵盤事件

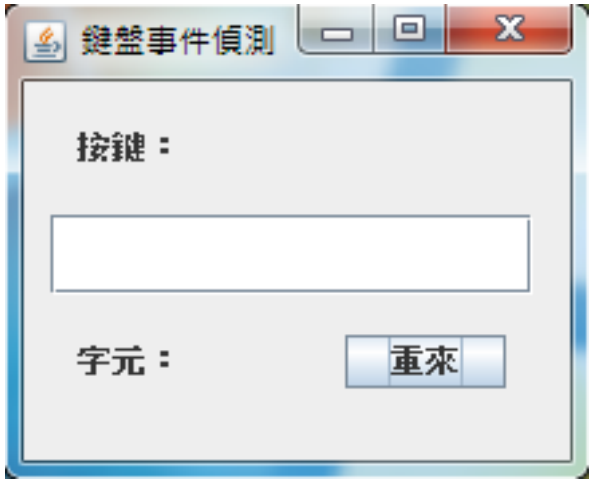
```
53          public void keyReleased(KeyEvent e)
54          {
55          }
56      };
57
58      public void actionPerformed(ActionEvent e)
59      {
60          if (e.getSource() == btnReset)
61          {
62              txtInput.setText("");
63              lblKey.setText("按鍵：");
64              lblChar.setText("字元：");
65          }
66      }
67  }
68
69  public class CH09_14
70  {
71      public static void main(String[] args)
72      {
73          CKeyF keyF = new CKeyF();
74      }
75  }
```

鍵盤事件

- 執行結果：







鍵盤事件

- 說明：
  - 行01：
    - 定義「套件（package）」。
  - 行03：
    - 載入「javax.swing.*」套件。
  - 行04：
    - 載入「java.awt.event.*」套件。
  - 行06~行67：
    - 建立繼承自「JFrame」父類別，並實作「ActionListener」介面的「CKeyF」子類別。
    - 行08~行10：
      - 宣告私有物件成員。
    - 行12~行39：
      - 宣告類別「建構子」「CKeyF」。

鍵盤事件

- 行14~行18：
  - 建立視窗物件。
- 行20~行22：
  - 建立標籤（lblKey）物件。
- 行24~行27：
  - 設定單行文字欄位（txtInput）物件。
  - 行27：

    建立事件來源物件連結事件傾聽者。

    KeyListInput是自訂的傾聽者物件。
- 行29~行31：
  - 建立標籤（lblChar）物件。
- 行33~行36：
  - 設定按鈕（btnReset）物件。

鍵盤事件

- 行36：

  設定事件來源物件連結事件傾聽者。

  this是指CKeyF類別所建立的視窗物件。

- 行38：
  - 重繪視窗元件。
- 行41~行56：
  - 利用「KeyListener()」的「建構子」，建立傾聽者物件「KeyListInpu」。
  - KeyListener有三個抽象方法，必須要實作。
  - 行43~行46：

    實作keyPressed(KeyEvent e)方法。

    用setText()方法，設定lblKey的值。
  - 行48~行51：

    實作keyTyped(KeyEvent e)方法。

    用setText()方法，設定lblChar的值。

鍵盤事件

注意：

keyTyped(KeyEvent e)無法對應（顯示）Alt、Ctrl、…等按鍵。

- 行53~行55：

  實作keyReleased(KeyEvent e)方法。

  注意：

  - 這個方法目前沒有處理事件的敘述（即「空」方法），但依然必須實作，否則編譯時會出現錯誤。
  - 行56的右大括號後面，須加分號（;），否則編譯時會產生錯誤。

- 行58~行66：
  - 實作ActionListener介面的actionPerformed()方法。
  - 為觸發事件時的處理方法。
  - ActionEvent被觸發時，e就是事件的來源。

鍵盤事件

- 行60：

  當e.getSource()取得的事件來源物件為btnReset時，執行此區段程式。

  行62~行64：

  用setText()方法設定txtInput、lblKey、lblChar的值。

- 行73：
  - 利用「CKeyF」的「建構子」，建立視窗物件。

鍵盤事件

- 程式（繼承KeyAdapter類別）：

```
package CH09_15;

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

class CKeyF extends JFrame implements ActionListener
{
    private JLabel lblKey, lblChar;
    private JTextField txtInput;
    private JButton btnReset;

    public CKeyF()
    {
        setTitle("鍵盤事件偵測");
        setLayout(null);
        setBounds(500, 400, 220, 180);
        setVisible(true);
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
```

鍵盤事件

```
lblKey = new JLabel("按鍵");
lblKey.setBounds(20, 10, 150, 30);
add(lblKey);

txtInput = new JTextField("");
txtInput.setBounds(10, 50, 180, 30);
add(txtInput);
txtInput.addKeyListener(new KeyListInput());

lblChar = new JLabel("字元");
lblChar.setBounds(20, 90, 150, 30);
add(lblChar);

btnReset = new JButton("重來");
btnReset.setBounds(120, 95, 60, 20);
add(btnReset);
btnReset.addActionListener(this);

repaint();
```

鍵盤事件

```
}

class KeyListInput extends KeyAdapter
{
    public void keyPressed(KeyEvent e)
    {
        lblKey.setText("按鍵：[ " + KeyEvent.getKeyText(e.getKeyCode()) +
                       "] 鍵被按下");
    }

    public void keyTyped(KeyEvent e)
    {
        lblChar.setText("字元：" + e.getKeyChar());
    }
}

public void actionPerformed(ActionEvent e)
{
    if (e.getSource() == btnReset)
    {
```

鍵盤事件

```
            txtInput.setText("");
            lblKey.setText("按鍵：");
            lblChar.setText("字元：");
        }
    }
}

public class CH09_15
{
    public static void main(String[] args)
    {
        CKeyF keyF = new CKeyF();
    }
}
```

鍵盤事件

```
1  package CH09_15;
2
3  import javax.swing.*;
4  import java.awt.event.*;
5
6  class CKeyF extends JFrame implements ActionListener
7  {
8     private JLabel lblKey, lblChar;
9     private JTextField txtInput;
10    private JButton btnReset;
11
12    public CKeyF()
13    {
14       setTitle("鍵盤事件偵測");
15       setLayout(null);
16       setBounds(500, 400, 220, 180);
17       setVisible(true);
18       setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
19
20       lblKey = new JLabel("按鍵");
21       lblKey.setBounds(20, 10, 150, 30);
22       add(lblKey);
23
24       txtInput = new JTextField("");
25       txtInput.setBounds(10, 50, 180, 30);
26       add(txtInput);
```

鍵盤事件

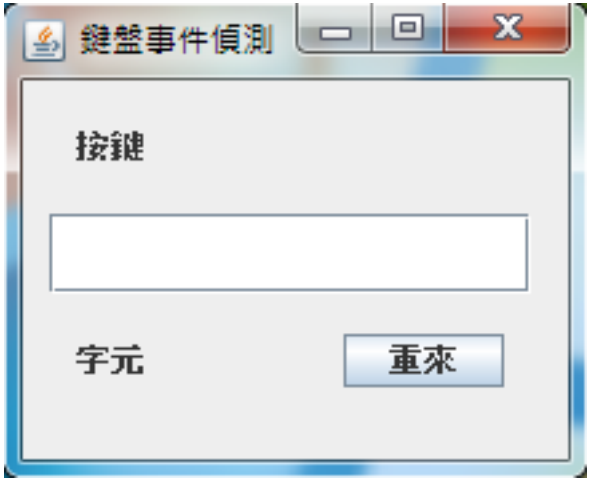
```
27        txtInput.addKeyListener(new KeyListInput());
28
29        lblChar = new JLabel("字元");
30        lblChar.setBounds(20, 90, 150, 30);
31        add(lblChar);
32
33        btnReset = new JButton("重來");
34        btnReset.setBounds(120, 95, 60, 20);
35        add(btnReset);
36        btnReset.addActionListener(this);
37
38        repaint();
39    }
40
41    class KeyListInput extends KeyAdapter
42    {
43        public void keyPressed(KeyEvent e)
44        {
45            lblKey.setText("按鍵：[ " + KeyEvent.getKeyText(e.getKeyCode()) + " ] 鍵被按下");
46        }
47
48        public void keyTyped(KeyEvent e)
49        {
50            lblChar.setText("字元：" + e.getKeyChar());
51        }
52    }
```

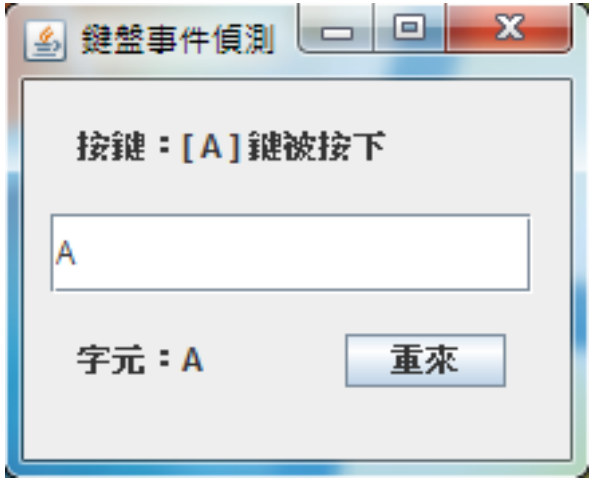
鍵盤事件

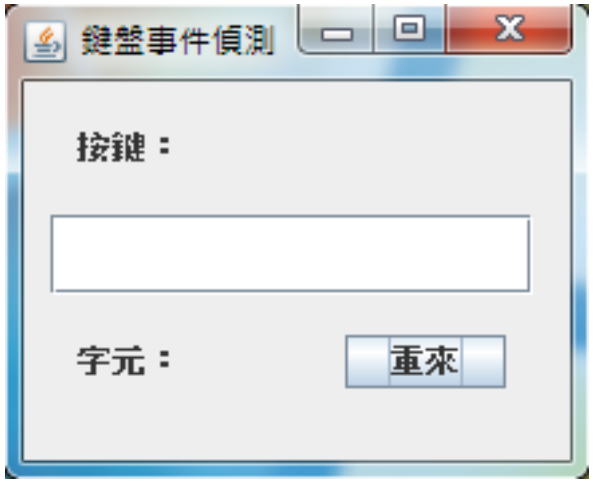
```
53
54    public void actionPerformed(ActionEvent e)
55    {
56       if (e.getSource() == btnReset)
57       {
58          txtInput.setText("");
59          lblKey.setText("按鍵：");
60          lblChar.setText("字元：");
61       }
62    }
63 }
64
65 public class CH09_15
66 {
67    public static void main(String[] args)
68    {
69       CKeyF keyF = new CKeyF();
70    }
71 }
```

鍵盤事件

- 執行結果：







鍵盤事件

- 說明：
  - 行01：
    - 定義「套件（package）」。
  - 行03：
    - 載入「javax.swing.*」套件。
  - 行04：
    - 載入「java.awt.event.*」套件。
  - 行06~行63：
    - 建立繼承自「JFrame」父類別，並實作「ActionListener」介面的「CKeyF」子類別。
    - 行08~行10：
      - 宣告私有物件成員。
    - 行12~行39：
      - 宣告類別「建構子」「CKeyF」。

鍵盤事件

- 行14~行18：
  - 建立視窗物件。
- 行20~行22：
  - 建立標籤（lblKey）物件。
- 行24~行27：
  - 設定單行文字欄位（txtInput）物件。
  - 行27：

    建立事件來源物件連結事件傾聽者。

    KeyListInput是自訂的傾聽者類別。
- 行29~行31：
  - 建立標籤（lblChar）物件。
- 行33~行36：
  - 設定按鈕（btnReset）物件。

鍵盤事件

- 行36：

  設定事件來源物件連結事件傾聽者。

  this是指CKeyF類別所建立的視窗物件。

- 行38：
  - 重繪視窗元件。
- 行41~行52：
  - 建立繼承自「KeyAdapter」父類別的「KeyListInput」子類別。
  - 行43~行46：

    建立keyPressed(KeyEvent e)方法。

    用setText()方法，設定lblKey的值。
  - 行48~行51：

    建立keyTyped(KeyEvent e)方法。

    用setText()方法，設定lblChar的值。

鍵盤事件

- 注意：

  因為是「繼承」KeyAdapter類別，如果用不到父類別中的keyReleased(KeyEvent e) 抽象方法，則不須覆蓋（即 不須建立「空」方法）。

- 行54~行62：
  - 實作ActionListener介面的actionPerformed()方法。
  - 為觸發事件時的處理方法。
  - ActionEvent被觸發時，e就是事件的來源。
  - 行56：

    當e.getSource()取得的事件來源物件為btnReset時，執行此區段程式。

    行58~行60：

    用setText()方法設定txtInput、lblKey、lblChar的值。

- 行69：
  - 利用「CKeyF」的「建構子」，建立視窗物件。

鍵盤事件

- 程式（繼承KeyAdapter類別，傾聽方向鍵）：

```
package CH09_16;

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

class CKeyMoveF extends JFrame
{
   private JLabel lblMove;
   private int pos_x = 70, pos_y = 80;

   CKeyMoveF()
   {
      setTitle("上下左右鍵 移動圖形");
      setLayout(null);
      setBounds(500, 400, 260, 260);
      setVisible(true);
      setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
```

鍵盤事件

```
    ImageIcon icon1 = new ImageIcon("..\\CH09\\src\\CH09_16\\images\\
                                        bomb.gif");

    lblMove = new JLabel(icon1);
    lblMove.setBounds(pos_x, pos_y, 100, 40);
    add(lblMove);

    CkeyMove keyMove = new CkeyMove();
    addKeyListener(keyMove);

    repaint();
}

class CkeyMove extends KeyAdapter
{
    public void keyPressed(KeyEvent e)
    {
        switch (e.getKeyCode())
        {
```

鍵盤事件

```
                case KeyEvent.VK_UP:
                    pos_y -= 2;
                    break;
                case KeyEvent.VK_DOWN:
                    pos_y += 2;
                    break;
                case KeyEvent.VK_LEFT:
                    pos_x -= 2;
                    break;
                case KeyEvent.VK_RIGHT:
                    pos_x += 2;
                    break;
            }
            lblMove.setLocation(pos_x, pos_y);
        }
    }
}

public class CH09_16
{
```

鍵盤事件

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

鍵盤事件

```
1  package CH09_16;
2
3  import javax.swing.*;
4  import java.awt.event.*;
5
6  class CKeyMoveF extends JFrame
7  {
8     private JLabel lblMove;
9     private int pos_x = 70, pos_y = 80;
10
11    CKeyMoveF()
12    {
13       setTitle("上下左右鍵 移動圖形");
14       setLayout(null);
15       setBounds(500, 400, 260, 260);
16       setVisible(true);
17       setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
18
19       ImageIcon icon1 = new ImageIcon("..\\CH09\\src\\CH09_16\\images\\bomb.gif");
20
21       lblMove = new JLabel(icon1);
22       lblMove.setBounds(pos_x, pos_y, 100, 40);
23       add(lblMove);
24
25       CkeyMove keyMove = new CkeyMove();
26       addKeyListener(keyMove);
```

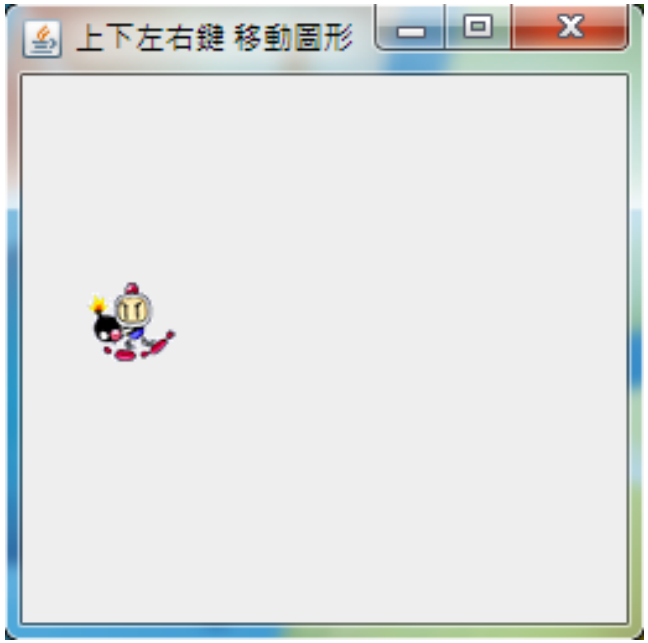
鍵盤事件

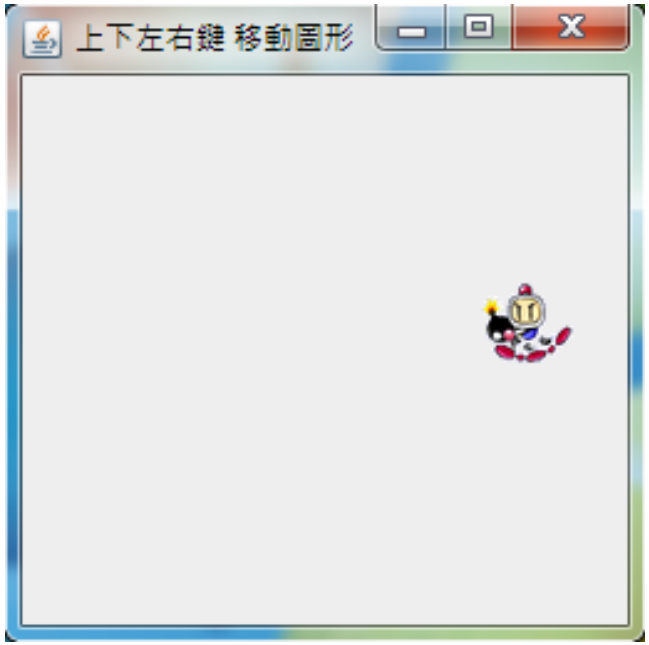
```
27
28      repaint();
29    }
30
31    class CkeyMove extends KeyAdapter
32    {
33      public void keyPressed(KeyEvent e)
34      {
35        switch (e.getKeyCode())
36        {
37        case KeyEvent.VK_UP:
38          pos_y -= 2;
39          break;
40        case KeyEvent.VK_DOWN:
41          pos_y += 2;
42          break;
43        case KeyEvent.VK_LEFT:
44          pos_x -= 2;
45          break;
46        case KeyEvent.VK_RIGHT:
47          pos_x += 2;
48          break;
49        }
50        lblMove.setLocation(pos_x, pos_y);
51      }
52    }
```

鍵盤事件

```
53 }
54
55 public class CH09_16
56 {
57    public static void main(String[] args)
58    {
59       new CKeyMoveF();
60    }
61 }
```

- 執行結果：





鍵盤事件

- 說明：
  - 行01：
    - 定義「套件（package）」。
  - 行03：
    - 載入「javax.swing.*」套件。
  - 行04：
    - 載入「java.awt.event.*」套件。
  - 行06~行53：
    - 建立繼承自「JFrame」父類別的「CKeyMoveF」子類別。
    - 行08~行09：
      - 宣告私有物件及資料成員。
    - 行11~行29：
      - 宣告類別「建構子」「CKeyMoveF」。

鍵盤事件

- 行13~行17：
  - 建立視窗物件。
- 行19：
  - 利用「ImageIcon(String filename)」的「建構子」，建立icon物件。
- 行21~行23：
  - 建立標籤（lblMove）物件。
- 行25：
  - 利用「CkeyMove()」的「建構子」，建立keyMove物件。
- 行26：
  - 建立事件來源物件（keyMove）連結事件傾聽者。
- 行28：
  - 重繪視窗元件。

鍵盤事件

- 行31~行52：
  - 建立繼承自「KeyAdapter」父類別的「CkeyMove」子類別。
  - 行33~行51：

    按下鍵盤時，要處理的事件。

    行35~行49：

    用getKeyCode()方法，取得按下的方向鍵，再決定移動的方向及距離。

    行37~行39：

    按下方向鍵「上」時，Y軸位置減2。

    行40~行42：

    按下方向鍵「下」時，Y軸位置加2。

    行43~行45：

    按下方向鍵「左」時，X軸位置減2。

鍵盤事件

行46~行48：

按下方向鍵「右」時，X軸位置加2。

- 行50：

根據pos_x及pos_y，重新設定標籤的位置。

- 行59：
  - 利用「CKeyMoveF()」的「建構子」，建立匿名的視窗物件。

滑鼠事件

- 在視窗應用程式中，滑鼠是最常使用的輸入工具。
- 常見的滑鼠事件有滑鼠指標的移動、按一下滑鼠鍵、快按兩下滑鼠鍵、滑鼠指標移至或移開.....等。
- 善用滑鼠事件，可以產生各式各樣的互動效果。
- 用滑鼠來處理觸發事件的傾聽機制方式有：
  「繼承MouseAdapter類別」、
  「繼承MouseMotionAdapter類別」、
  「實作MouseListener介面」、
  「實作MouseMotionListener介面」。

滑鼠事件

- MouseListener介面與MouseAdapter類別皆提供了五個滑鼠事件處理方法：
  - void mouseClicked(MouseEvent e)
    - 當滑鼠鍵被按一下時所觸發的事件，包括按下及放開的過程。
  - void mousePressed(MouseEvent e)
    - 當滑鼠鍵被按下時所觸發的事件。
  - void mouseReleased(MouseEvent e)
    - 當已按下的滑鼠鍵被放開時所觸發的事件。
  - void mouseEntered(MouseEvent e)
    - 當滑鼠指標移入來源物件時所觸發的事件。

滑鼠事件

- void mouseExited(MouseEvent e)
  - 當滑鼠指標從來源物件移出來時所觸發的事件。

- MouseMotionListener介面與MouseMotionAdapter類別提供了二個滑鼠事件處理方法：
  - void mouseMoved(MouseEvent e)
    - 當滑鼠指標在來源物件內移動時所觸發的事件。
  - void mouseDragged(MouseEvent e)
    - 當滑鼠指標拖曳來源物件時所觸發的事件。

- 滑鼠事件類別MouseEvent有些常用的方法：
  - int getX()
    - 傳回滑鼠指標在視窗物件內的水平座標。

滑鼠事件

- int getY()
  - 傳回滑鼠指標在視窗物件內的垂直座標。
- int getClickCount()
  - 傳回滑鼠鍵被按了幾下。
- int getButton()
  - 傳回滑鼠被按下或放開的鍵是哪一個按鍵。
    - 傳回1，表示左鍵；
    - 傳回2，表示中鍵；
    - 傳回3，表示右鍵。

滑鼠事件

- 程式（繼承MouseAdapter類別）：

```
package CH09_17;

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

class CMouseAdapF extends JFrame
{
   private JLabel lblShow, lblPic;
   private ImageIcon icon1, icon2, icon3;

   CMouseAdapF()
   {
      setTitle("滑鼠狀態");
      setLayout(null);
      setBounds(500, 400, 220, 200);
      setVisible(true);
      setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
```

滑鼠事件

```
    lblShow = new JLabel("滑鼠狀態：");
    lblShow.setBounds(20, 10, 200, 30);
    add(lblShow);

    icon1 = new ImageIcon("..\\CH09\\src\\CH09_17\\images\\Scissor.gif");
    icon2 = new ImageIcon("..\\CH09\\src\\CH09_17\\images\\Stone.gif");
    icon3 = new ImageIcon("..\\CH09\\src\\CH09_17\\images\\Paper.gif");

    lblPic = new JLabel(icon1);
    lblPic.setBounds(70, 70, 60, 60);
    add(lblPic);
    lblPic.addMouseListener(new CMouseAdap());
}

class CMouseAdap extends MouseAdapter
{
    public void mouseEntered(MouseEvent e)
    {
        lblShow.setText("滑鼠狀態：滑鼠指標移入");
```

滑鼠事件

```
lblPic.setIcon(icon2);
        }

        public void mouseExited(MouseEvent e)
        {
           lblShow.setText("滑鼠狀態：滑鼠指標移出");
           lblPic.setIcon(icon1);
        }

        public void mousePressed(MouseEvent e)
        {
           lblShow.setText("滑鼠狀態：滑鼠按下");
           lblPic.setIcon(icon3);
        }

        public void mouseReleased(MouseEvent e)
        {
           lblShow.setText("滑鼠狀態：滑鼠放開");
           lblPic.setIcon(icon2);
        }
```

滑鼠事件

```
    }
}

public class CH09_17
{
    public static void main(String[] args)
    {
        CMouseAdapF mouseF = new CMouseAdapF();
    }
}
```

滑鼠事件

```
1  package CH09_17;
2
3  import javax.swing.*;
4  import java.awt.event.*;
5
6  class CMouseAdapF extends JFrame
7  {
8     private JLabel lblShow, lblPic;
9     private ImageIcon icon1, icon2, icon3;
10
11    CMouseAdapF()
12    {
13       setTitle("滑鼠狀態");
14       setLayout(null);
15       setBounds(500, 400, 220, 200);
16       setVisible(true);
17       setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
18
19       lblShow = new JLabel("滑鼠狀態：");
20       lblShow.setBounds(20, 10, 200, 30);
21       add(lblShow);
22
23       icon1 = new ImageIcon("..\\CH09\\src\\CH09_17\\images\\Scissor.gif");
24       icon2 = new ImageIcon("..\\CH09\\src\\CH09_17\\images\\Stone.gif");
25       icon3 = new ImageIcon("..\\CH09\\src\\CH09_17\\images\\Paper.gif");
26
```

滑鼠事件

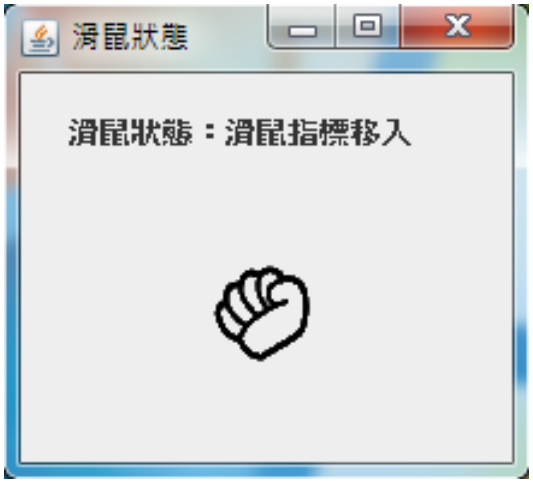
```
27      lblPic = new JLabel(icon1);
28      lblPic.setBounds(70, 70, 60, 60);
29      add(lblPic);
30      lblPic.addMouseListener(new CMouseAdap());
31    }
32
33    class CMouseAdap extends MouseAdapter
34    {
35      public void mouseEntered(MouseEvent e)
36      {
37        lblShow.setText("滑鼠狀態：滑鼠指標移入");
38        lblPic.setIcon(icon2);
39      }
40
41      public void mouseExited(MouseEvent e)
42      {
43        lblShow.setText("滑鼠狀態：滑鼠指標移出");
44        lblPic.setIcon(icon1);
45      }
46
47      public void mousePressed(MouseEvent e)
48      {
49        lblShow.setText("滑鼠狀態：滑鼠按下");
50        lblPic.setIcon(icon3);
51      }
52
```

滑鼠事件

```
53          public void mouseReleased(MouseEvent e)
54          {
55             lblShow.setText("滑鼠狀態：滑鼠放開");
56             lblPic.setIcon(icon2);
57          }
58       }
59    }
60
61    public class CH09_17
62    {
63       public static void main(String[] args)
64       {
65          CMouseAdapF mouseF = new CMouseAdapF();
66       }
67    }
```

滑鼠事件

- 執行結果：



滑鼠事件

- 說明：
  - 行01：
    - 定義「套件（package）」。
  - 行03：
    - 載入「javax.swing.*」套件。
  - 行04：
    - 載入「java.awt.event.*」套件。
  - 行06~行59：
    - 建立繼承自「JFrame」父類別的「CMouseAdapF」子類別。
    - 行08~行09：
      - 宣告私有物件成員。
    - 行11~行31：
      - 宣告類別「建構子」「CMouseAdapF」。

滑鼠事件

- 行13~行17：
  - 建立視窗物件。
- 行19~行21：
  - 建立標籤（lblShow）物件。
- 行23~行25：
  - 利用「ImageIcon(String filename)」的「建構子」，建立icon1、icon2、icon3物件。
- 行27~行30：
  - 建立標籤（lblPic）物件。
  - 行30：

    建立事件來源物件（lblPic）連結事件傾聽者。

    CMouseAdap()是指自訂的傾聽者類別。

滑鼠事件

- 行33~行58：
  - 建立繼承自「MouseAdapter」父類別的「CMouseAdap」子類別。
  - 行35~行39：

    覆蓋父類別的mouseEntered(MouseEvent e)方法。
  - 行41~行45：

    覆蓋父類別的mouseExited(MouseEvent e)方法。
  - 行47~行51：

    覆蓋父類別的mousePressed(MouseEvent e)方法。
  - 行53~行57：

    覆蓋父類別的mouseReleased(MouseEvent e)方法。
- 行65：
  - 利用「CMouseAdapF()」的「建構子」，建立mouseF物件。

滑鼠事件

- 程式（繼承MouseMotionAdapter類別）：

```
package CH09_18;

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

class CMouseAdapF extends JFrame
{
   private JLabel lblX, lblY;

   CMouseAdapF()
   {
      setTitle("顯示滑鼠的指標座標");
      setLayout(null);
      setBounds(500, 400, 260, 200);
      setVisible(true);
      setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

      lblX = new JLabel("x = ");
```

滑鼠事件

```
        lblX.setBounds(140, 140, 50, 20);
        add(lblX);

        lblY = new JLabel("y = ");
        lblY.setBounds(200, 140, 50, 20);
        add(lblY);
        addMouseMotionListener(new CMousePos());
    }

    class CMousePos extends MouseMotionAdapter
    {
        public void mouseMoved(MouseEvent e)
        {
            lblX.setText("x = " + e.getX());
            lblY.setText("y = " + e.getY());
        }
    }
}
```

滑鼠事件

```
public class CH09_18
{
    public static void main(String[] args)
    {
        CMouseAdapF mouseF = new CMouseAdapF();
    }
}
```

滑鼠事件

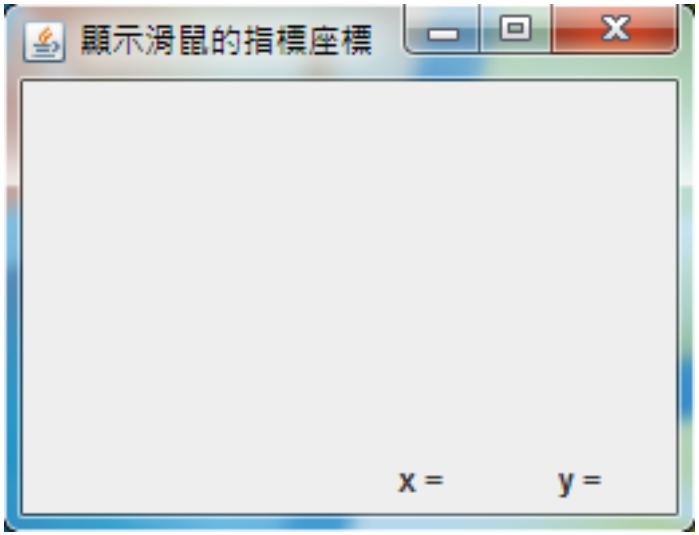
```
1  package CH09_18;
2
3  import javax.swing.*;
4  import java.awt.event.*;
5
6  class CMouseAdapF extends JFrame
7  {
8     private JLabel lblX, lblY;
9
10    CMouseAdapF()
11    {
12       setTitle("顯示滑鼠的指標座標");
13       setLayout(null);
14       setBounds(500, 400, 260, 200);
15       setVisible(true);
16       setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
17
18       lblX = new JLabel("x = ");
19       lblX.setBounds(140, 140, 50, 20);
20       add(lblX);
21
22       lblY = new JLabel("y = ");
23       lblY.setBounds(200, 140, 50, 20);
24       add(lblY);
25       addMouseMotionListener(new CMousePos());
26    }
```

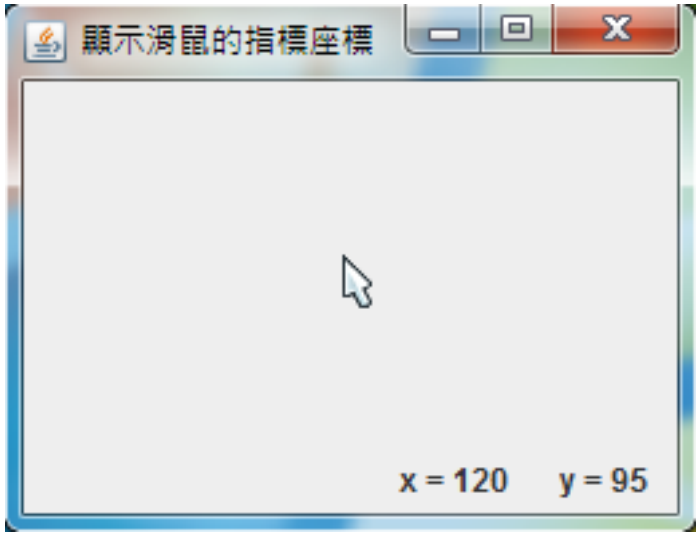
滑鼠事件

```
27
28    class CMousePos extends MouseMotionAdapter
29    {
30       public void mouseMoved(MouseEvent e)
31       {
32          lblX.setText("x = " + e.getX());
33          lblY.setText("y = " + e.getY());
34       }
35    }
36 }
37
38 public class CH09_18
39 {
40    public static void main(String[] args)
41    {
42       CMouseAdapF mouseF = new CMouseAdapF();
43    }
44 }
```

滑鼠事件

- 執行結果：





滑鼠事件

- 說明：
  - 行01：
    - 定義「套件（package）」。
  - 行03：
    - 載入「javax.swing.*」套件。
  - 行04：
    - 載入「java.awt.event.*」套件。
  - 行06~行36：
    - 建立繼承自「JFrame」父類別的「CMouseAdapF」子類別。
    - 行08：
      - 宣告私有物件成員。
    - 行10~行26：
      - 宣告類別「建構子」「CMouseAdapF」。

滑鼠事件

- 行12~行16：
  - 建立視窗物件。
- 行18~行20：
  - 建立標籤（lblX）物件。
- 行22~行25：
  - 建立標籤（lblY）物件。
  - 行25：

    建立事件來源物件連結事件傾聽者。

    CMousePos()是指自訂的傾聽者類別。
- 行28~行35：
  - 建立繼承自「MouseMotionAdapter」父類別的「CMousePos」子類別。
  - 行30~行34：

    覆蓋父類別的mouseMoved(MouseEvent e)方法。

滑鼠事件

- 行42：
  - 利用「CMouseAdapF()」的「建構子」，建立mouseF物件。

滑鼠事件

- 程式（實作MouseListener介面及實作MouseMotionListener介面）：

```
package CH09_19;

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

class CMouseListF extends JFrame implements MouseListener,
MouseMotionListener
{
    private JLabel lblPic;
    private int pos_x = 100, pos_y = 50, x1, x2, y1, y2;
    private boolean is_drag = false;

    CMouseListF()
    {
        setTitle("顯示滑鼠的拖曳");
        setLayout(null);
        setBounds(500, 400, 260, 200);
```

滑鼠事件

```
        setVisible(true);
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        lblPic = new JLabel(new ImageIcon("..\\CH09\\src\\CH09_19\\images
                                                \\Scissor.gif"));
        lblPic.setBounds(pos_x, pos_y, 50, 50);
        add(lblPic);
        lblPic.addMouseListener(this);
        lblPic.addMouseMotionListener(this);
    }

    public void mousePressed(MouseEvent e)
    {
        if (is_drag)
            return;

        if (e.getButton() == 1)
            is_drag = true;
```

滑鼠事件

```
    x1 = e.getX();
    y1 = e.getY();
}

public void mouseReleased(MouseEvent e)
{
    if (!is_drag)
        return;

    is_drag = false;
}

public void mouseDragged(MouseEvent e)
{
    if (!is_drag)
        return;

    x2 = e.getX();
    y2 = e.getY();
```

滑鼠事件

```
        pos_x = pos_x + (x2 - x1);
        pos_y = pos_y + (y2 - y1);

        if (pos_x <= 0)
            pos_x = 0;

        if (pos_x >= 184)
            pos_x = 0;

        if (pos_y <= 0)
            pos_y = 0;

        if (pos_y >= 92)
            pos_y = 92;

        lblPic.setLocation(pos_x, pos_y);
    }

    public void mouseClicked(MouseEvent e)
    {
```

滑鼠事件

```
    }

    public void mouseEntered(MouseEvent e)
    {
    }

    public void mouseExited(MouseEvent e)
    {
    }

    public void mouseMoved(MouseEvent e)
    {
    }
}

public class CH09_19
{
    public static void main(String[] args)
    {
        CMouseListF mouseF = new CMouseListF();
```

滑鼠事件

```
        }
    }
```

滑鼠事件

```
1  package CH09_19;
2
3  import javax.swing.*;
4  import java.awt.event.*;
5
6  class CMouseListF extends JFrame implements MouseListener, MouseMotionListener
7  {
8     private JLabel lblPic;
9     private int pos_x = 100, pos_y = 50, x1, x2, y1, y2;
10    private boolean is_drag = false;
11
12    CMouseListF()
13    {
14       setTitle("顯示滑鼠的拖曳");
15       setLayout(null);
16       setBounds(500, 400, 260, 200);
17       setVisible(true);
18       setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
19
20       lblPic = new JLabel(new ImageIcon("..\\CH09\\src\\CH09_19\\images\\Scissor.gif"));
21       lblPic.setBounds(pos_x, pos_y, 50, 50);
22       add(lblPic);
23       lblPic.addMouseListener(this);
24       lblPic.addMouseMotionListener(this);
25    }
26
```

滑鼠事件

```
27    public void mousePressed(MouseEvent e)
28    {
29       if (is_drag)
30          return;
31
32       if (e.getButton() == 1)
33          is_drag = true;
34
35       x1 = e.getX();
36       y1 = e.getY();
37    }
38
39    public void mouseReleased(MouseEvent e)
40    {
41       if (!is_drag)
42          return;
43
44       is_drag = false;
45    }
46
47    public void mouseDragged(MouseEvent e)
48    {
49       if (!is_drag)
50          return;
51
52       x2 = e.getX();
```

滑鼠事件

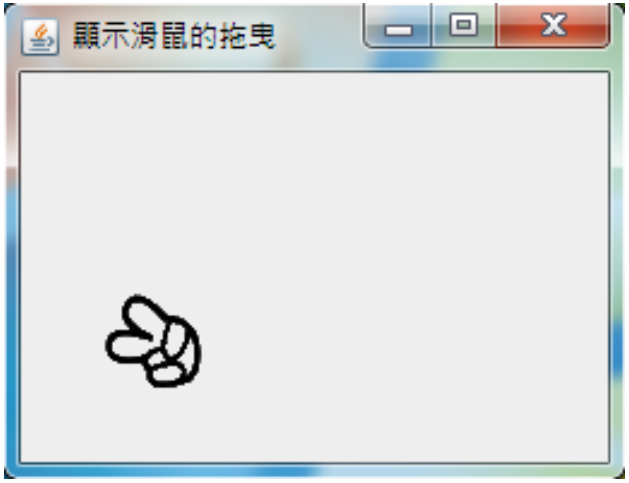
```
53          y2 = e.getY();
54
55          pos_x = pos_x + (x2 - x1);
56          pos_y = pos_y + (y2 - y1);
57
58          if (pos_x <= 0)
59              pos_x = 0;
60
61          if (pos_x >= 184)
62              pos_x = 0;
63
64          if (pos_y <= 0)
65              pos_y = 0;
66
67          if (pos_y >= 92)
68              pos_y = 92;
69
70          lblPic.setLocation(pos_x, pos_y);
71      }
72
73      public void mouseClicked(MouseEvent e)
74      {
75      }
76
77      public void mouseEntered(MouseEvent e)
78      {
```

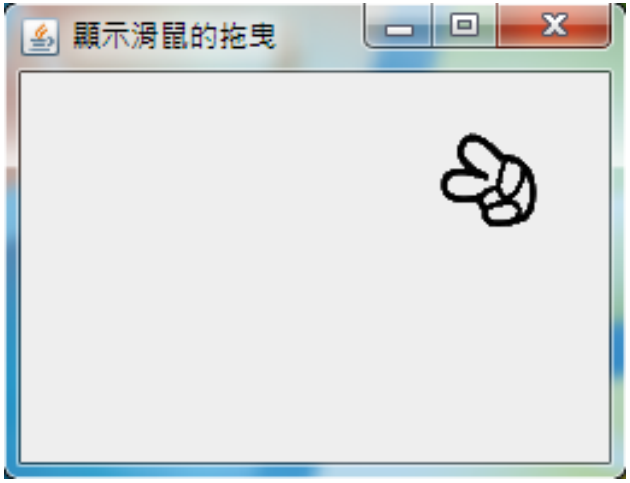
滑鼠事件

```
79      }
80
81      public void mouseExited(MouseEvent e)
82      {
83      }
84
85      public void mouseMoved(MouseEvent e)
86      {
87      }
88  }
89
90  public class CH09_19
91  {
92      public static void main(String[] args)
93      {
94          CMouseListF mouseF = new CMouseListF();
95      }
96  }
```

滑鼠事件

- 執行結果：





滑鼠事件

- 說明：
  - 行01：
    - 定義「套件（package）」。
  - 行03：
    - 載入「javax.swing.*」套件。
  - 行04：
    - 載入「java.awt.event.*」套件。
  - 行06~行88：
    - 建立繼承自「JFrame」父類別，且實作「MouseListener」介面及「MouseMotionListener」介面的「CMouseListF」子類別。
    - 行08~行10：
      - 宣告私有物件成員。

滑鼠事件

- 行12~行25：
  - 宣告類別「建構子」「CMouseListF」。
- 行14~行18：
  - 建立視窗物件。
- 行20行24：
  - 建立標籤（lblPic）物件。
- 行23~行24：
  - 建立事件來源物件連結事件傾聽者。
  - this是指CMouseListF類別所產生的視窗物件。
- 行27~行37：
  - 實作「mousePressed(MouseEvent e)」方法。
- 行39~行45：
  - 實作「mouseReleased(MouseEvent e)」方法。
- 行47~行71：
  - 實作「mouseDragged(MouseEvent e)」方法。

滑鼠事件

- 行73~行75：
  - 實作「mouseClicked(MouseEvent e)」方法。
- 行77~行79：
  - 實作「mouseEntered(MouseEvent e)」方法。
- 行81~行83：
  - 實作「mouseExited(MouseEvent e)」方法。
- 行85~行87：
  - 實作「mouseMoved(MouseEvent e)」方法。

注意：

- 雖然mouseClicked(MouseEvent e)、mouseEntered(MouseEvent e)、mouseExited(MouseEvent e)、mouseMoved(MouseEvent e) 四個方法目前沒有處理事件的敘述（即「空」方法），但依然必須實作，否則編譯時會出現錯誤。

- 行94：
  - 利用「CMouseListF()」的「建構子」，建立mouseF物件。

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。