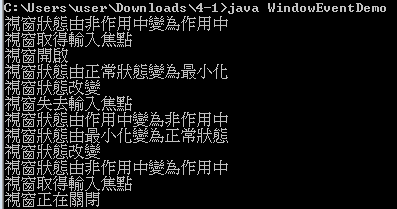
Event

1. 請參考投影片內容，建立以下視窗應用程式

請將1.程式**執行結果**截圖置入作業中、2.程式原始檔置入作業中

1. 使用不同方式來建立與監控windows各項事件
   1. Listener



import java.awt.\*;

import java.awt.event.\*;

public class WindowEventDemo extends java.awt.Frame implements

WindowListener, // 實作WindowListener介面

WindowFocusListener, // 實作WindowFocusListener介面

WindowStateListener { // 實作WindowStateListener介面

// Main method

public static void main(String[] args) {

new WindowEventDemo();

}

// 建構函式

public WindowEventDemo() {

super("Window Event Demo");

// 註冊 WindowFocusListener

this.addWindowFocusListener(this);

// 註冊 WindowListener

this.addWindowListener(this);

// 註冊 WindowStateListener

this.addWindowStateListener(this);

// 設定視窗的大小

this.setSize(new Dimension(200, 200));

// Center the frame

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

Dimension frameSize = this.getSize();

if (frameSize.height > screenSize.height)

frameSize.height = screenSize.height;

if (frameSize.width > screenSize.width)

frameSize.width = screenSize.width;

this.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height - frameSize.height) / 2);

// 顯示視窗

this.setVisible(true);

}

// 實作WindowListener介面之方法

public void windowActivated(WindowEvent e) {

System.out.println("視窗狀態由非作用中變為作用中");

}

public void windowClosed(WindowEvent e) {

System.out.println("視窗已關閉");

}

public void windowClosing(WindowEvent e) {

System.out.println("視窗正在關閉");

dispose();

System.exit(0);

}

public void windowDeactivated(WindowEvent e) {

System.out.println("視窗狀態由作用中變為非作用中");

}

public void windowDeiconified(WindowEvent e) {

System.out.println("視窗狀態由最小化變為正常狀態");

}

public void windowIconified(WindowEvent e) {

System.out.println("視窗狀態由正常狀態變為最小化");

}

public void windowOpened(WindowEvent e) {

System.out.println("視窗開啟");

}

// 實作WindowFocusListener介面之方法

public void windowGainedFocus(WindowEvent e) {

System.out.println("視窗取得輸入焦點");

}

public void windowLostFocus(WindowEvent e) {

System.out.println("視窗失去輸入焦點");

}

// 實作WindowStateListener介面之方法

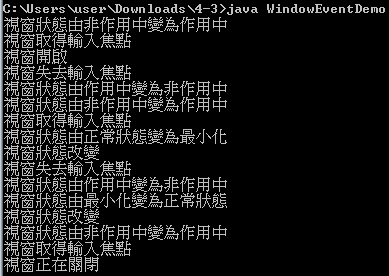
public void windowStateChanged(WindowEvent e) {

System.out.println("視窗狀態改變");

}

}

* 1. Adapter(嘗試)



import java.awt.\*;

import java.awt.event.\*;

public class WindowEventDemo extends java.awt.Frame {

// Main method

public static void main(String[] args) {

new WindowEventDemo();

}

// 建構函式

public WindowEventDemo() {

super("Window Event Demo");

// 自訂繼承WindowAdapter之類別

WindowHandler handler1 = new WindowHandler();

// 註冊 WindowListener

this.addWindowListener(handler1);

// 自訂繼承WindowAdapter之類別

WindowFocusHandler handler2 = new WindowFocusHandler();

// 註冊 WindowFocusListener

this.addWindowFocusListener(handler2);

// 自訂繼承WindowAdapter之類別

WindowStateHandler handler3 = new WindowStateHandler();

// 註冊 WindowStateListener

this.addWindowStateListener(handler3);

// 設定視窗的大小

this.setSize(200, 200);

// Center the frame

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

Dimension frameSize = this.getSize();

if (frameSize.height > screenSize.height)

frameSize.height = screenSize.height;

if (frameSize.width > screenSize.width)

frameSize.width = screenSize.width;

this.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height - frameSize.height) / 2);

// 顯示視窗

this.setVisible(true);

}

}

// 繼承WindowAdapter抽象類別

class WindowHandler extends WindowAdapter {

public void windowActivated(WindowEvent e) {

System.out.println("視窗狀態由非作用中變為作用中");

}

public void windowClosed(WindowEvent e) {

System.out.println("視窗已關閉");

}

public void windowClosing(WindowEvent e) {

System.out.println("視窗正在關閉");

System.exit(0);

}

public void windowDeactivated(WindowEvent e) {

System.out.println("視窗狀態由作用中變為非作用中");

}

public void windowDeiconified(WindowEvent e) {

System.out.println("視窗狀態由最小化變為正常狀態");

}

public void windowIconified(WindowEvent e) {

System.out.println("視窗狀態由正常狀態變為最小化");

}

public void windowOpened(WindowEvent e) {

System.out.println("視窗開啟");

}

}

// 繼承WindowAdapter抽象類別

class WindowFocusHandler extends WindowAdapter {

public void windowGainedFocus(WindowEvent e) {

System.out.println("視窗取得輸入焦點");

}

public void windowLostFocus(WindowEvent e) {

System.out.println("視窗失去輸入焦點");

}

}

// 繼承WindowAdapter抽象類別

class WindowStateHandler extends WindowAdapter {

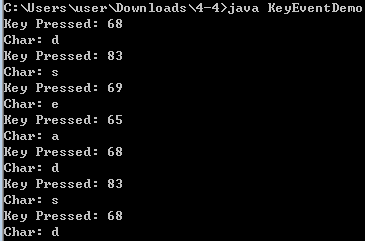
public void windowStateChanged(WindowEvent e) {

System.out.println("視窗狀態改變");

}

}

1. 使用不同方式來建立與監控Keyboard各項事件
   1. Listener



import java.awt.\*;

import java.awt.event.\*;

// 實作KeyListener介面

public class KeyEventDemo extends java.awt.Frame implements KeyListener {

// Main method

public static void main(String[] args) {

new KeyEventDemo();

}

// 建構函式

public KeyEventDemo() {

super("Key Event Demo");

// 註冊 KeyListener

this.addKeyListener(this);

// 設定視窗的大小

this.setSize(200, 200);

// Center the frame

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

Dimension frameSize = this.getSize();

if (frameSize.height > screenSize.height)

frameSize.height = screenSize.height;

if (frameSize.width > screenSize.width)

frameSize.width = screenSize.width;

this.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height - frameSize.height) / 2);

// 顯示視窗

this.setVisible(true);

this.addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

System.exit(0);

}

});

}

public void keyPressed(KeyEvent e) {

int dx = 0;

int dy = 0;

// 判斷是否按下Alt鍵

if (e.isAltDown())

System.out.println("Press Alt key") ;

// 判斷是否按下Ctrl鍵

if (e.isControlDown())

System.out.println("Press Control key") ;

// 判斷是否按下Shift鍵

if (e.isShiftDown())

System.out.println("Press Shift key") ;

// 回傳按鍵所代表的按鍵值

int i = e.getKeyCode() ;

System.out.println("Key Pressed: " + i);

switch(i) {

case KeyEvent.VK\_UP:

dy = -5;

break;

case KeyEvent.VK\_DOWN:

dy = 5;

break;

case KeyEvent.VK\_LEFT:

dx = -5;

break;

case KeyEvent.VK\_RIGHT:

dx = 5;

break;

}

this.setLocation(this.getX() + dx, this.getY() + dy);

}

public void keyReleased(KeyEvent e) {

System.out.println("Char: " + e.getKeyChar()) ;

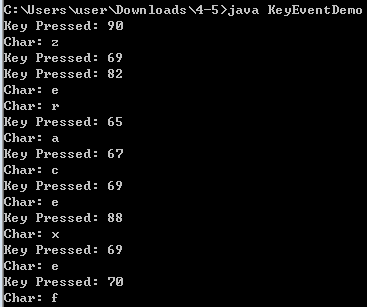
}

public void keyTyped(KeyEvent e) {

}

}

* 1. Adapter(嘗試)



import java.awt.\*;

import java.awt.event.\*;

// 實作KeyListener介面

public class KeyEventDemo extends java.awt.Frame {

// Main method

public static void main(String[] args) {

new KeyEventDemo();

}

// 建構函式

public KeyEventDemo() {

super("Key Event Demo");

// 註冊 KeyListener

KeyHandler handler1 = new KeyHandler();

this.addKeyListener(handler1);

// 設定視窗的大小

this.setSize(200, 200);

// Center the frame

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

Dimension frameSize = this.getSize();

if (frameSize.height > screenSize.height)

frameSize.height = screenSize.height;

if (frameSize.width > screenSize.width)

frameSize.width = screenSize.width;

this.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height - frameSize.height) / 2);

// 顯示視窗

this.setVisible(true);

this.addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

System.exit(0);

}

});

}

}

class KeyHandler extends KeyAdapter {

public void keyTyped(KeyEvent e) {

}

public void keyPressed(KeyEvent e) {

int dx = 0;

int dy = 0;

// 判斷是否按下Alt鍵

if (e.isAltDown())

System.out.println("Press Alt key") ;

// 判斷是否按下Ctrl鍵

if (e.isControlDown())

System.out.println("Press Control key") ;

// 判斷是否按下Shift鍵

if (e.isShiftDown())

System.out.println("Press Shift key") ;

// 回傳按鍵所代表的按鍵值

int i = e.getKeyCode() ;

System.out.println("Key Pressed: " + i);

switch(i) {

case KeyEvent.VK\_UP:

dy = -5;

break;

case KeyEvent.VK\_DOWN:

dy = 5;

break;

case KeyEvent.VK\_LEFT:

dx = -5;

break;

case KeyEvent.VK\_RIGHT:

dx = 5;

break;

}

}

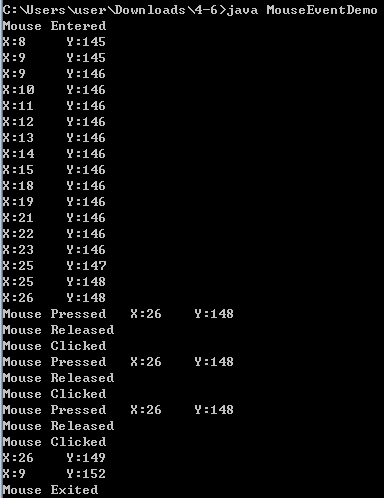
public void keyReleased(KeyEvent e) {

System.out.println("Char: " + e.getKeyChar());

}

}

1. 建立與監控Mouse各項事件
   1. Mouse & Mouse Motion Listener



import java.awt.\*;

import java.awt.event.\*;

public class MouseEventDemo extends java.awt.Frame implements MouseListener, MouseMotionListener{

public static void main(String[] args) {

new MouseEventDemo();

}

public MouseEventDemo() {

super("Mouse Event Demo");

this.addMouseListener(this);

this.addMouseMotionListener(this);

this.setSize(new Dimension(200, 200));

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

Dimension frameSize = this.getSize();

if (frameSize.height > screenSize.height)

frameSize.height = screenSize.height;

if (frameSize.width > screenSize.width)

frameSize.width = screenSize.width;

this.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height - frameSize.height) / 2);

this.setVisible(true);

}

public void mouseClicked(MouseEvent e) {

System.out.println("Mouse Clicked");

}

public void mouseEntered(MouseEvent e) {

System.out.println("Mouse Entered");

}

public void mouseExited(MouseEvent e) {

System.out.println("Mouse Exited");

}

public void mousePressed(MouseEvent e) {

System.out.println("Mouse Pressed" + "\tX:" + e.getX() + "\tY:" + e.getY());

}

public void mouseReleased(MouseEvent e) {

System.out.println("Mouse Released");

}

public void mouseDragged(MouseEvent e) {

System.out.println("Mouse Dragged");

}

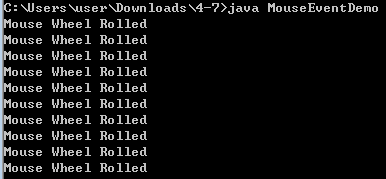
public void mouseMoved(MouseEvent e) {

System.out.println("X:" + e.getX() + "\tY:" + e.getY());

}

}

* 1. Mouse Wheel Listener



import java.awt.\*;

import java.awt.event.\*;

public class MouseEventDemo extends java.awt.Frame implements MouseWheelListener {

public static void main(String[] args) {

new MouseEventDemo();

}

public MouseEventDemo() {

super("Mouse Event Demo");

this.addMouseWheelListener(this);

this.setSize(new Dimension(200, 200));

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

Dimension frameSize = this.getSize();

if (frameSize.height > screenSize.height)

frameSize.height = screenSize.height;

if (frameSize.width > screenSize.width)

frameSize.width = screenSize.width;

this.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height - frameSize.height) / 2);

this.setVisible(true);

}

public void mouseWheelMoved(MouseWheelEvent e) {

System.out.println("Mouse Wheel Rolled");

}

}