Menu

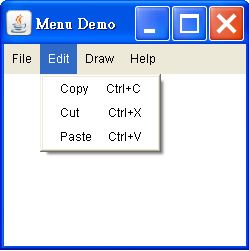
1. 請參考投影片內容，建立以下視窗應用程式

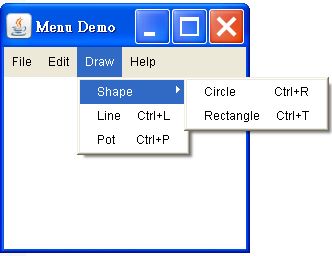
請將1.程式**執行結果**截圖置入作業中、2.程式**原始檔**置入作業中

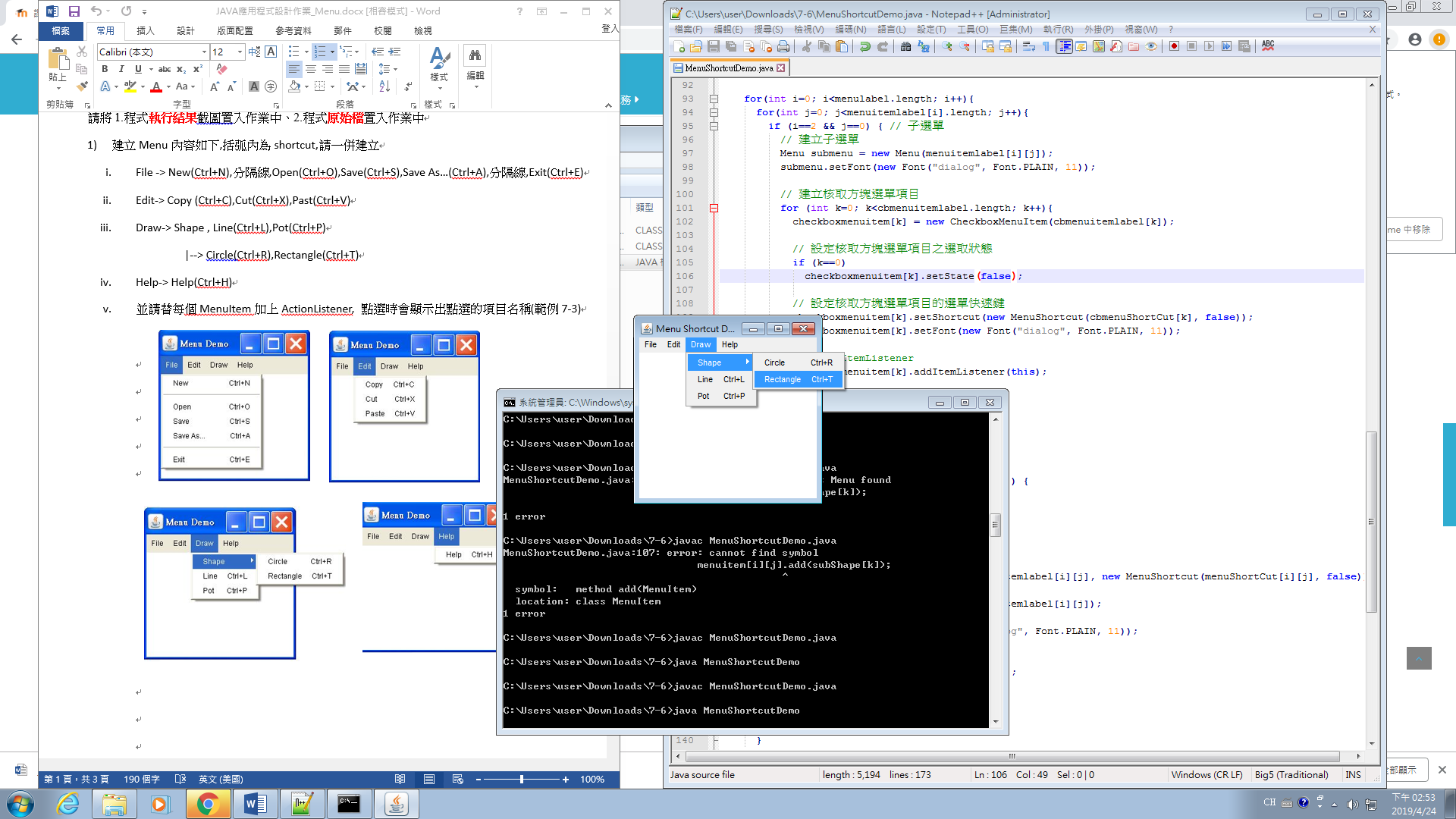
1. 建立Menu內容如下,括弧內為shortcut,請一併建立
   1. File -> New(Ctrl+N),分隔線,Open(Ctrl+O),Save(Ctrl+S),Save As…(Ctrl+A),分隔線,Exit(Ctrl+E)
   2. Edit-> Copy (Ctrl+C),Cut(Ctrl+X),Past(Ctrl+V)
   3. Draw-> Shape , Line(Ctrl+L),Pot(Ctrl+P)

|--> Circle(Ctrl+R),Rectangle(Ctrl+T)

* 1. Help-> Help(Ctrl+H)
  2. 並請替每個MenuItem加上ActionListener, 點選時會顯示出點選的項目名稱(範例7-3)







import java.awt.\*;

import java.awt.event.\*;

public class MenuShortcutDemo extends java.awt.Frame implements ActionListener, ItemListener {

String menulabel[]={"File", "Edit", "Draw", "Help"};

String menuitemlabel[][]={

{"New", "-", "Open", "Save", "Save As...", "-", "Exit"},

{"Copy", "Cut", "Paste"},

{"Shape", "Line", "Pot"},

{"Help"},

};

String cbmenuitemlabel[]={"Circle", "Rectangle"};

int menuShortCut[][]={

{KeyEvent.VK\_N, 0, KeyEvent.VK\_O, KeyEvent.VK\_S, KeyEvent.VK\_A, 0,KeyEvent.VK\_E},

{KeyEvent.VK\_C, KeyEvent.VK\_X, KeyEvent.VK\_V},

{0, KeyEvent.VK\_L, KeyEvent.VK\_P},

{KeyEvent.VK\_H}

};

int cbmenuShortCut[]={KeyEvent.VK\_R, KeyEvent.VK\_T};

Menu menu[] = new Menu[4];

MenuItem menuitem[][] = new MenuItem[4][12];

CheckboxMenuItem checkboxmenuitem[] = new CheckboxMenuItem[2];

// Demo only

Label label;

public static void main(String args[]){

new MenuShortcutDemo();

}

// 建構函式

public MenuShortcutDemo() {

super("Menu Shortcut Demo");

// 定義 Layout Manager 為 BorderLayout

setLayout(new BorderLayout());

// Demo only

label = new Label();

add(label, BorderLayout.SOUTH);

// 建立選單列

MenuBar menuBar = createMenuBar();

// 定義視窗使用者介面之選單列

setMenuBar(menuBar);

// 設定視窗的大小

this.setSize(250, 250);

// 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);

}

});

}

private MenuBar createMenuBar() {

// 建立選單列

MenuBar menuBar = new MenuBar();

// 建立選單

for (int i=0; i<menulabel.length; i++){

// 建立選單

menu[i] = new Menu(menulabel[i]);

menu[i].setFont(new Font("dialog", Font.PLAIN, 11));

// 新增選單至選單列

menuBar.add(menu[i]);

}

for(int i=0; i<menulabel.length; i++){

for(int j=0; j<menuitemlabel[i].length; j++){

if (i==2 && j==0) { // 子選單

// 建立子選單

Menu submenu = new Menu(menuitemlabel[i][j]);

submenu.setFont(new Font("dialog", Font.PLAIN, 11));

// 建立核取方塊選單項目

for (int k=0; k<cbmenuitemlabel.length; k++){

checkboxmenuitem[k] = new CheckboxMenuItem(cbmenuitemlabel[k]);

// 設定核取方塊選單項目之選取狀態

if (k==0)

checkboxmenuitem[k].setState(false);

// 設定核取方塊選單項目的選單快速鍵

checkboxmenuitem[k].setShortcut(new MenuShortcut(cbmenuShortCut[k], false));

checkboxmenuitem[k].setFont(new Font("dialog", Font.PLAIN, 11));

// 註冊 ItemListener

checkboxmenuitem[k].addItemListener(this);

// 新增核取方塊選單項目至子選單

submenu.add(checkboxmenuitem[k]);

}

menu[i].add(submenu);

}

else if (menuitemlabel[i][j].equals("-")) {

// 新增分隔線

menu[i].addSeparator();

}

else {

// 建立選單項目並使用選單快速鍵

if (menuShortCut[i][j] != 0)

menuitem[i][j] = new MenuItem(menuitemlabel[i][j], new MenuShortcut(menuShortCut[i][j], false));

else

menuitem[i][j] = new MenuItem(menuitemlabel[i][j]);

menuitem[i][j].setFont(new Font("dialog", Font.PLAIN, 11));

// 註冊 ActionListener

menuitem[i][j].addActionListener(this);

// 新增選單項目

menu[i].add(menuitem[i][j]);

}

}

}

return menuBar;

}

// 實作ActionListener介面之方法

public void actionPerformed(ActionEvent e) {

// 取得產生動作事件的選單項目

MenuItem menuitem = (MenuItem)e.getSource();

if (menuitem.getLabel().equals("Exit")) { // Exit

System.exit(0);

}

else {

label.setText("Select " + menuitem.getLabel() + " Menu Item.");

}

}

public void itemStateChanged(ItemEvent e) {

// 取得產生項目事件的物件

CheckboxMenuItem cbmi = (CheckboxMenuItem)(e.getItemSelectable());

for (int i=0; i < checkboxmenuitem.length; i++) {

if(cbmi == checkboxmenuitem[i]) {

checkboxmenuitem[i].setState(true);

label.setText("Select " + checkboxmenuitem[i].getLabel() + " Checkbox Menu Item.");

}

else

checkboxmenuitem[i].setState(false);

}

}

}

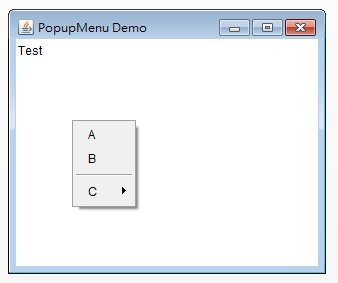
1. 建立Popup Menu內容如下
   1. A
   2. B
   3. 分隔線
   4. C -> D

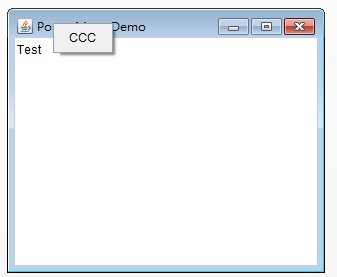
E

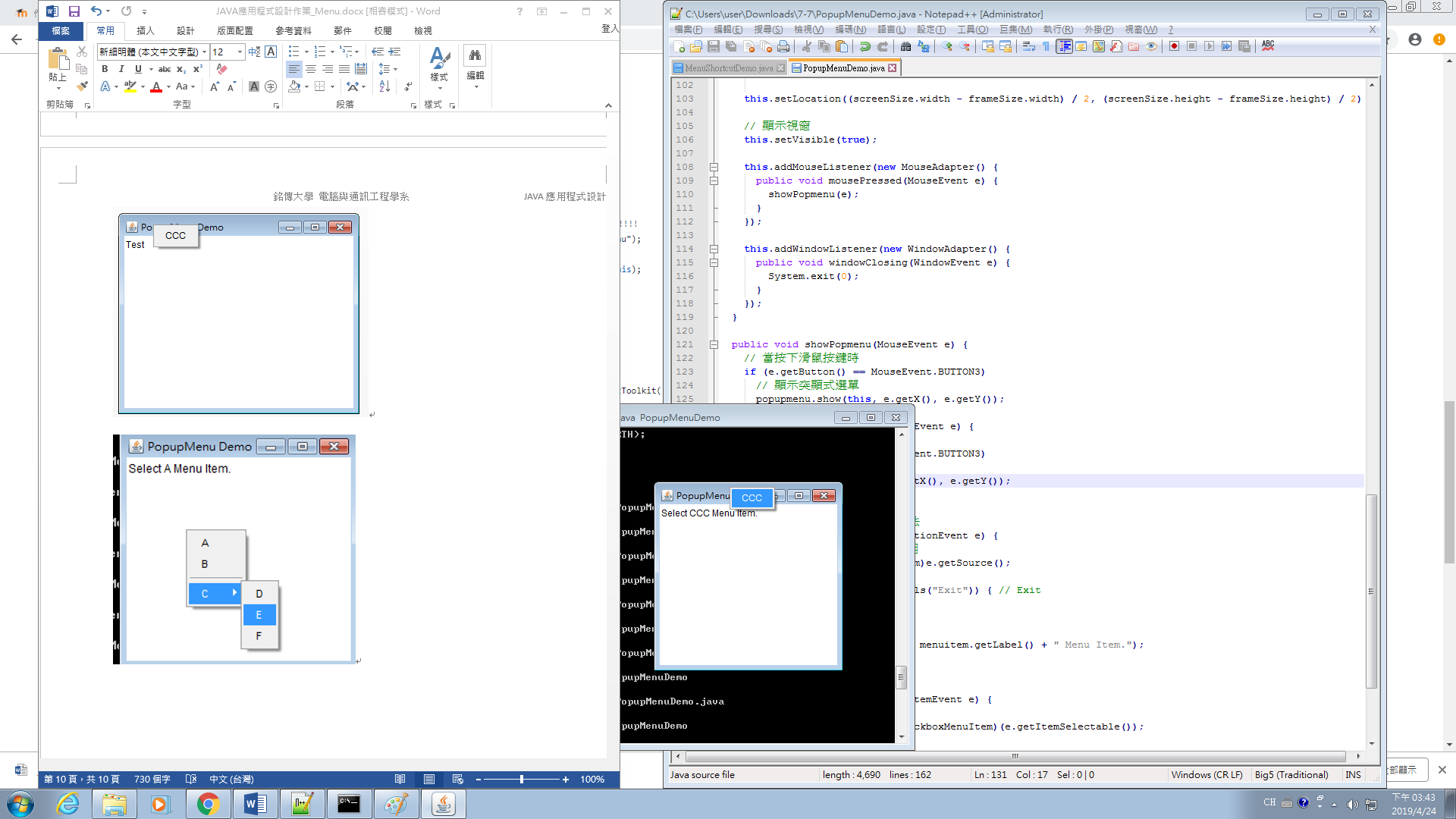
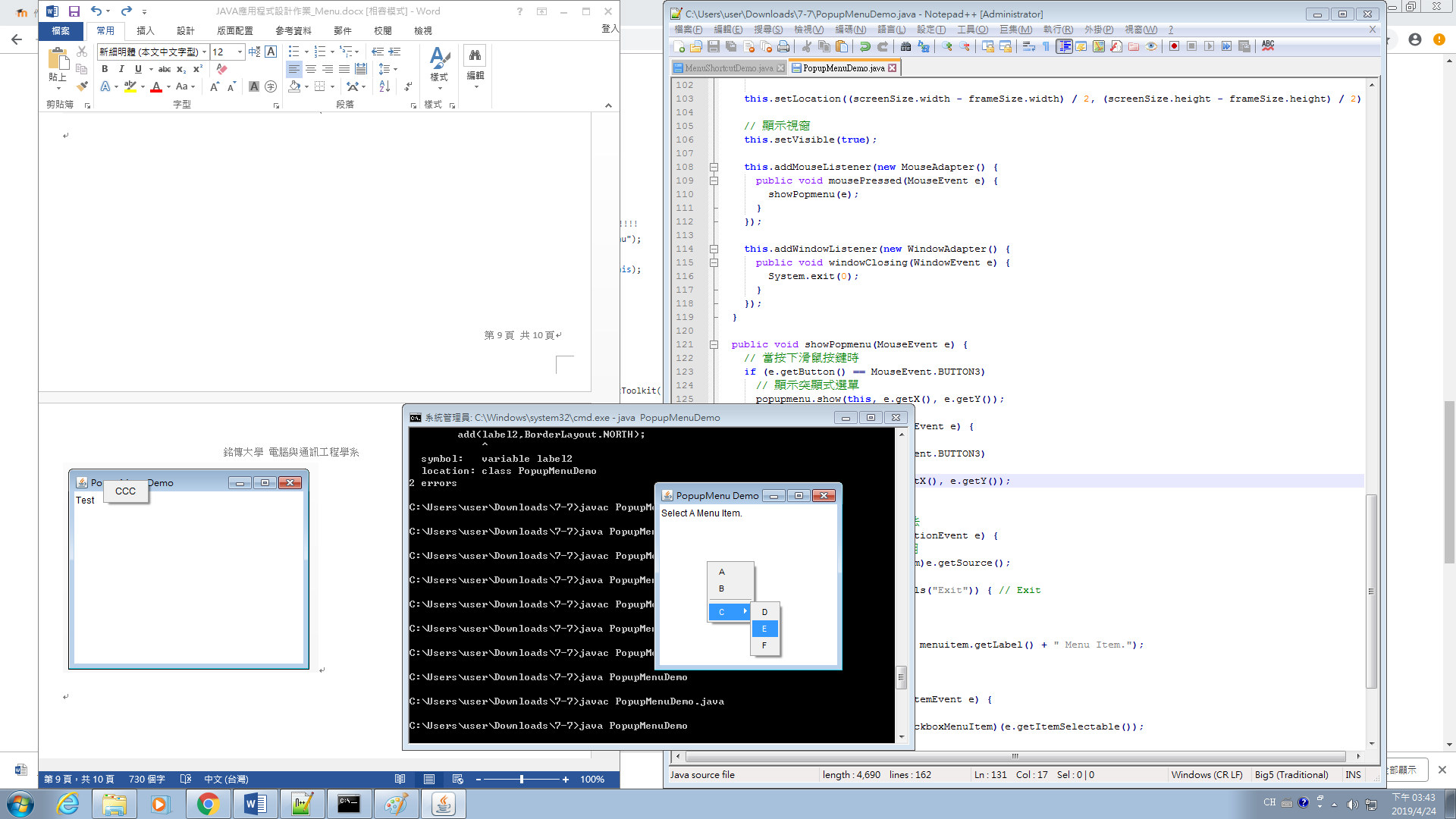
F

並請替每個MenuItem加上ActionListener, 點選時會顯示出點選的項目名稱

在不同的區域點滑鼠右鍵,會顯示不同的Menu







import java.awt.\*;

import java.awt.event.\*;

public class PopupMenuDemo extends java.awt.Frame implements ActionListener, ItemListener {

PopupMenu popupmenu ,popupmenu2;

String menuitemlabel[]={

"A", "B", "-", "C"};

String cbmenuitemlabel[]={"D", "E", "F"};

MenuItem menuitem[] = new MenuItem[4];

CheckboxMenuItem checkboxmenuitem[] = new CheckboxMenuItem[3];

MenuItem menuitem2;

// Demo only

Label label;

public static void main(String args[]){

new PopupMenuDemo();

}

// 建構函式

public PopupMenuDemo() {

super("PopupMenu Demo");

// Demo only

label = new Label("Test");

label.addMouseListener(new MouseAdapter() {

public void mousePressed(MouseEvent e) {

showPopmenu2(e);

}

});

add(label, BorderLayout.NORTH);

// 建立突顯式選單

popupmenu = new PopupMenu("popup menu");

for(int i=0; i<menuitemlabel.length; i++){

if (i==3) { // 子選單

// 建立子選單

Menu submenu = new Menu(menuitemlabel[i]);

submenu.setFont(new Font("dialog", Font.PLAIN, 11));

// 建立核取方塊選單項目

for (int j=0; j<cbmenuitemlabel.length; j++){

checkboxmenuitem[j] = new CheckboxMenuItem(cbmenuitemlabel[j]);

// 設定核取方塊選單項目之選取狀態

if (j==0)

checkboxmenuitem[j].setState(false);

checkboxmenuitem[j].setFont(new Font("dialog", Font.PLAIN, 11));

// 註冊 ItemListener

checkboxmenuitem[j].addItemListener(this);

// 新增核取方塊選單項目至子選單

submenu.add(checkboxmenuitem[j]);

}

// 新增子選單至突顯式選單中

popupmenu.add(submenu);

}

else if (menuitemlabel[i].equals("-")) {

// 新增分隔線

popupmenu.addSeparator();

}

else {

// 建立選單項目

menuitem[i] = new MenuItem(menuitemlabel[i]);

menuitem[i].setFont(new Font("dialog", Font.PLAIN, 11));

// 註冊 ActionListener

menuitem[i].addActionListener(this);

// 新增選單項目至突顯式選單中

popupmenu.add(menuitem[i]);

}

}

// 定義突顯式選單

add(popupmenu);

popupmenu2 = new PopupMenu("popup menu2");

menuitem2 = new MenuItem("CCC");

menuitem2.addActionListener(this);

popupmenu2.add(menuitem2);

add(popupmenu2);

// 設定視窗的大小

this.setSize(250, 250);

// 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.addMouseListener(new MouseAdapter() {

public void mousePressed(MouseEvent e) {

showPopmenu(e);

}

});

this.addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

System.exit(0);

}

});

}

public void showPopmenu(MouseEvent e) {

// 當按下滑鼠按鍵時

if (e.getButton() == MouseEvent.BUTTON3)

// 顯示突顯式選單

popupmenu.show(this, e.getX(), e.getY());

}

public void showPopmenu2(MouseEvent e) {

// 當按下滑鼠按鍵時

if (e.getButton() == MouseEvent.BUTTON3)

// 顯示突顯式選單

popupmenu2.show(this, e.getX(), e.getY());

}

// 實作ActionListener介面之方法

public void actionPerformed(ActionEvent e) {

// 取得產生動作事件的選單項目

MenuItem menuitem = (MenuItem)e.getSource();

if (menuitem.getLabel().equals("Exit")) { // Exit

System.exit(0);

}

else {

label.setText("Select " + menuitem.getLabel() + " Menu Item.");

}

}

public void itemStateChanged(ItemEvent e) {

// 取得產生項目事件的物件

CheckboxMenuItem cbmi = (CheckboxMenuItem)(e.getItemSelectable());

for (int i=0; i < checkboxmenuitem.length; i++) {

if(cbmi == checkboxmenuitem[i]) {

checkboxmenuitem[i].setState(true);

label.setText("Select " + checkboxmenuitem[i].getLabel() + " Checkbox Menu Item.");

}

else

checkboxmenuitem[i].setState(false);

}

}

}