Popup menus and printing in the SPIRE QLA: technical note

Change Record

ISSUE	DATE	
0.2	7th Februaru 03	Second Version

Table of Contents

1. Introduction

This document describes the classes for creating popup menus within GUIs in the SPIRE QLA. It also covers the printing class, which is used in conjunction with the popup classes.

1.1 Intended readership

This document is intended to be read by developers of the SPIRE QLA and other members of the ISDT and ICCDT.

1.2 Applicability statement

This document is applicable to release 1.0 (and above) of the SPIRE QLA.

1.3 Purpose

This document describes how to add popup menus to GUIs within the SPIRE QLA, and how to include menu items for the special cases of "save" and "print", as well as menu items particular to the individual developer. It is concerned with three classes within the **herschel.spire.qla** package: **SystemPopup**, **SystemPopupMouseHandler** and **PrintableComponent**.

1.4 How to use this document

This document consists of the following sections:

Section 1, this section, explains the purpose and context of this document.
Section 2 provides an overview of the 3 relevant classes.
Section 3 provides an example of how to use the classes described.

1.5 Related documents

Data I/O Design Document

1.6 Conventions

Class names, methods, variables and package names are shown in **bold type**.
File names are shown in *italic type*.
Code is shown in `courier type`.

1.7 Problem reporting instructions

Problems detected in *SystemPopup.java*, *SystemPopupMouseHandler.java* and *PrintableComponent.java* should be reported using the SPIRE SPR / SCR reporting system at http://astro.esa.int/herschel_webapps/servletsuite/ProblemReportServlet?area=spire&mode=displaysubmitpr

Errors detected in this document should be reported to the author at helen.bright@ic.ac.uk.

2. The popup and printing classes

2.1 Overview

SystemPopup and **SystemPopupMouseHandler** are classes which allow the developer to easily add a right-click popup menu to a GUI. Because printing and saving are the most common requirements in a popup menu, special constructors and methods are provided for adding (and removing) these. For other items on the menu, the user must pass a **String** (or a **List** consisting of **Strings**) to the constructor or method, which is converted into the menu item(s). They must also pass an **ActionListener** (or a **List** consisting of **ActionListeners**) to the constructor or method, which is associated with the relevant menu item(s). It is important that if more than one menu item is added, that any given **ActionListener** has the same **List** index as the corresponding **String**. If **Lists** of different lengths are supplied, an **Exception** will be thrown.

The variety of methods provided for adding and removing menu items means that it is easy for the developer to dynamically code different menus for different situations.

PrintableComponent is a class that executes a print command for a **Component** passed in the constructor. The developer does not need to directly instantiate an object of this class, as this is handled by the **SystemPopup** class. Adding a print element to the popup menu actually involves adding two menu items – one for printing the **Component** that is passed to it, and one for printing the surrounding frame.

See the Data I/O Design Document for details of the "save" functionality.

2.2.1 PrintableComponent

2.2.1.1 Purpose

Executes the print command for the **Component** passed to it in the constructor. The developer does not need to directly instantiate an object of this class, as this is handled by the **SystemPopup** class.

2.2.1.2. Constructors

public PrintableComponent(Component c)

2.2.1.3 Methods

public void printComponent()
public void print()
public int print(Graphics g, PageFormat format, int pagenum)
private static void disableDoubleBuffering(Component c)
private static void enableDoubleBuffering(Component c)

2.2.1.4 Restrictions and notes

None.

2.2.2 SystemPopup

2.2.2.1 Purpose

Creates a popup menu to which **MenuItems** and their associated **ActionListeners** can be added. It extends **JPopupMenu**. Special constructors and methods are provided for the print and save commands, as these are the most common menu items required by the developer. The user can also add and remove their own menu items as well.

2.2.2.2 Constructors

public SystemPopup()
public SystemPopup(Component _componentForPrint, DefaultListModel _dlm)
public SystemPopup(String _menuItem, ActionListener _actionListener, Component _componentForPrint, DefaultListModel _dlm)
public SystemPopup(List _menuItemsList, List _actionListenersList)
public SystemPopup(List _menuItemsList, List _actionListenersList, Component _componentForPrint, DefaultListModel _dlm)

2.2.2.3 Methods

private void buildSystemPopup()
public void addItem(String _menuItem, ActionListener _actionListener)
public void addItems(List _menuItemsList, List _actionListenersList)
public void removeItem(String _menuItem, ActionListener _actionListener)
public void removeItems(List _menuItemsList, List _actionListenersList)
public void addPrint(Component _componentForPrint)

public void addSave(DefaultListModel _dlm)
public void removePrint()
public void removeSave()

2.2.2.4 Restrictions and notes

The constructors and methods of the **SystemPopup** class which involve adding the print menu items require the developer to pass a **Component** – this must be the **Component** the developer wishes to be printed. It is important that the developer adds the **SystemPopupMouseHandler** to the same **Component** (see section ***2.2.3.4***). The "print whole frame" menu item which is automatically added, does not require an extra **Component** – this is handled automatically.

Any class implementing the **List** interface can be supplied for those constructors and methods that take **Lists** as arguments, so **ArrayList**, **Vector** and **LinkedList** are all acceptable. It is important that if more than one menu item is added, that any given **ActionListener** has the same **List** index as the corresponding **String**. If **Lists** of different lengths are supplied, an **Exception** will be thrown.

The constructors and methods of the **SystemPopup** class which involve adding the save menu item require the developer to pass a **DefaultListModel** to be saved – see the Data I/O Design Document for details.

Note that the popup menu does not automatically appear when it is constructed. The **show()** method (inherited from **JPopupMenu**) is called by the **SystemPopupMouseHandler** class when the user right-clicks.

If the developer calls one of the remove commands on an item that has not been added, there is no effect on the popup menu.

The three internal classes **PrintActionListener1**, **PrintActionListener2** and **SaveActionListener** deal with the commands for printing and saving.

2.2.3 SystemPopupMouseHandler

2.2.3.1 Purpose

Class that listens for a mouse-click, tests to see if the click is a **PopupTrigger**, and shows a popup menu if it is.
It extends **MouseAdapter**.

2.2.3.2 Constructors

public SystemPopupMouseHandler(Popup _p)

2.2.3.3 Methods

public void mousePressed(MouseEvent me)
public void mouseReleased(MouseEvent me)
public void mouseClicked(MouseEvent me)
public void showSystemPopup(MouseEvent me)

2.2.3.4 Restrictions and Notes

The **showSystemPopup(MouseEvent me)** method calls the **show()** method of the **SystemPopup** class, which is inherited from **JPopupMenu**.

The reason that three different mouse methods are included is that different operating systems have different popup triggers.

It is important that this listener is added to the same **Component** that is passed to the **SystemPopup** class's constructor or its **addPrint(Component _componentForPrint)** method if printing is to be included.

Note that the listener is not automatically added to all sub-components. It is added to **JLabels**, but not to **JTextAreas** or **JButtons** for example. If you require such sub-components to receive the listener, you will need to add it to these additionally.

It is a quirk of the **java.awt.print** api that **JInternalFrames** are printed in full (including the title bar etc), whereas **JFrames** are not. As a consequence the "print whole frame" option has a different result in the single look and feel mode than in the multiple – the former prints the entire window, and the latter just prints its contents. This means that the "print this component" and "print whole frame" can potentially have exactly the same output in the multiple look and feel mode.

3. Example

The following code sample shows a simple example of how the above classes could be used, in a file called *PopupTestCode.java*.

```
1.    package herschel.spire.qla;
2.
3.    import javax.swing.*;
4.    import java.awt.*;
5.    import java.awt.event.*;
6.    import java.util.*;
7.
8.    public class PopupTestCode extends JFrame{
9.
10.         JLabel testLabel = new JLabel("Right-click to test popup menu
construction");
11.         JPanel panel = new JPanel();
12.         JTextArea textarea = new JTextArea("text area", 5, 30);
13.         ListenerOne l1 = new ListenerOne();
14.         ListenerTwo l2 = new ListenerTwo();
```

```
15.         ListenerThree l3 = new ListenerThree();
16.         Vector menuItems = new Vector();
17.         Vector actionListeners = new Vector();
18.         SystemPopup sp;
19.
20.         public PopupTestCode(){
21.               menuItems.add("listener 2");
22.               menuItems.add("listener 3");
23.               actionListeners.add(l2);
24.               actionListeners.add(l3);
25.               sp = new SystemPopup();
26.               sp.addItem("listener 1", l1);
27.               sp.addItems(menuItems, actionListeners);
28.               sp.addPrint(this);
29.               Container c = getContentPane();
30.               c.add(panel);
31.               panel.add(testLabel);
32.               panel.add(textarea);
33.               this.addMouseListener(new SystemPopupMouseHandler(sp));
34.               textarea.addMouseListener(new SystemPopupMouseHandler(sp));
35.               setSize(400, 200);
36.               setLocation(200,300);
37.               show();
38.         }
39.
40.         public class ListenerOne implements ActionListener{
41.               public void actionPerformed(ActionEvent e){
42.                     System.out.println("ListenerOne speaking");
43.               }
44.         }
45.
46.         public class ListenerTwo implements ActionListener{
47.               public void actionPerformed(ActionEvent e){
48.                     System.out.println("ListenerTwo speaking");
49.               }
50.         }
51.
52.         public class ListenerThree implements ActionListener{
53.               public void actionPerformed(ActionEvent e){
54.                     System.out.println("ListenerThree speaking");
55.               }
56.         }
57.
58.   }
```

This code sample creates a simple window with a **JLabel** and a **JTextArea**.

A popup menu is constructed at line 25. In this example it has no arguments, so initially there are no menu items in it. However, it could be constructed with all the required menu items if one of the other constructors were used. The example then demonstrates adding just one item (line 26), adding more than one item at once (line 27) and adding the print items (line 28).

Note that the **Component** passed to the **addPrint()** method is the same as the one that the mouse listener is added to, in this case the whole window. An extra listener is added at line 34 to the **JTextArea**. If this were not in the code, right-clicking on the text area would have no effect. Note, however that the **JLabel** automatically receives the listener.

Because the listener is added to the whole window, the “print this component” and “print whole frame” will have exactly the same effect if the program is run as part of the QLA with the multiple look and feel mode. (see section ***2.2.3.4***).

The internal classes **ListenerOne**, **ListenerTwo** and **ListenerThree** demonstrate how the developer can add their own custom **ActionListeners** to the popup menu.