[TopCoder]

Side Menu v1.0 Component Specification

1. Design

The Java Swing Side Menu component provides a panel in a Swing application that contains a set of controls that can be opened or closed by a user. The menu takes up the full left or right side of an application. This component will be used to house functionality in the UML Tool that isn't always necessary. The user can choose to collapse the side menu to allow for more space for a diagram being viewed.

The design of the Side Menu component allows for maximum flexibility in both the layout and contents of the 'collapsed' panel (what is displayed when the component is not expanded) and 'expanded' panel (what is displayed when the component is expanded).

The component provides a default 'collapsed' panel that displays expansion buttons (using a specified icon) on the top and bottom with a label (using a specified title) in the middle (justified towards the top) - as specified in the prototype images.

The component also provides a default 'expanded' panel (that will replace the 'collapsed' panel when expanded) that features a label (using a specified title) and a collapse button (using a specified icon) along the top with user specified content below it. This panel can have an absolute width applied to it. The width is a 'best effort' attempt by setting the size, preferred size, minimum size and maximum size. However, the layout manager is free to ignore these constraints - making it impossible to enforce this.

In addition to the above, this component supports the following:

1. The ability to hide the various expansion/collapse buttons to allow programmatically expansion/collapse only.
2. The ability to specify a 'click to expand' on the collapsed panel to automatically expand the panel when clicked on (best used when the expansion buttons are hidden).
3. The ability to 'float' the expanded panel on top of other panels. When this occurs, a 'pin' toggle button is added between the title and the collapse button. When the 'pin' button is deselected, the expansion panel will float above the panels. When the 'pin' button is selected, the expansion panel is pinned and will be inserted into the layout.
4. The ability to fully customize the side menu by providing the JComponent(s) used as the expansion and collapse panel.

As mentioned in the forum, a "Visual Studio" look and feel is achievable with these options.

Please note that this component will rely on a new Vertical Label component to handle vertical text. PM has given approval for this component (since its useful in other components) and the docs directory contains a requirement specification documenting the minimal requirements for that component.

1.1 Design Patterns

Strategy - the strategy pattern is used to allow the Side Menu model to vary independently.

1.2 Industry Standards

Java Swing

1.3 Required Algorithms

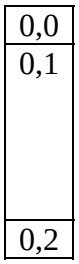
Although there are really no complex algorithms required, showing the layout of each panel is worthwhile.

1.3.1 Main Side Menu panel

The main side menu panel simply is a container for either the collapsed panel or the expanded panel. The side menu panel simply uses a BorderLayout manager and adds/removes the collapsed/expanded panels to the center (BorderLayout.CENTER).

1.3.2 The Default Collapsed Side Menu Panel

The default collapsed side menu panel provides two expansion buttons (on the top and bottom) and a vertical label in between. Both buttons may be hidden if an icon is not assigned to them. This panel uses a GridBagLayout to layout the cells like:



Cell 0,0 (column, row) will hold the top expansion button and has the following GridBagConstraints:

a) Occupies only 1 row and 1 column
b) Does not grow either horizontally or vertically
c) Is centered within the cell

Cell 0,1 (column, row) will hold the vertical label and has the following GridBagConstraints:

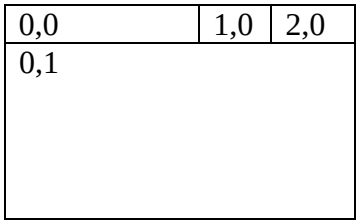
a) Occupies only 1 row and 1 column
b) Grows both horizontally and vertically
c) Is positioned to the north

Cell 0,2 (column, row) will hold the bottom expansion button and has the following GridBagConstraints:

a) Occupies only 1 row and 1 column
b) Does not grow either horizontally or vertically
c) Is centered within the cell

1.3.3 The Default Expanded Side Menu Panel

The default expanded side menu panel provides a label, a pin toggle button (which may be hidden if there is no associated icon), and a collapse button (also may be hidden) at the top with the user specified contents below it. This panel uses a GridBagLayout to layout the cells like:



Cell 0,0 (column, row) will hold the label and has the following GridBagConstraints:

a) Occupies only 1 row and 1 column
b) Grows horizontally
c) Is centered within the cell

Cell 1,0 (column, row) will hold the pin button and has the following GridBagConstraints:

a) Occupies only 1 row and 1 column
b) Does not grow horizontally or vertically
c) Is centered within the cell

Cell 2,0 (column, row) will hold the collapse button and has the following GridBagConstraints:

a) Occupies only 1 row and 1 column
b) Does not grow horizontally or vertically
c) Is centered within the cell

Cell 0,1 (column, row) will hold the user specified contents and has the following GridBagConstraints:

a) Occupies only 1 row and 3 columns
b) Grows both horizontally and vertically
c) Is positioned to the north east

1.3.4 The Floating Expanded Side Menu

When the component has been set to float the expanded side menu, the floating is accomplished like:

```
Set the height of the expansion panel to the height of the Side
Menu Panel (also set's the width to either the preferred width of
the expansion panel or the absolute width specified)

Get the shared instance of the PopupFactory

Create a new Popup from the PopupFactory using the SideMenuPanel
as the parent, the expansion panel for the content at the X/Y
position of the SideMenuPanel

Request focus to the expansion panel
```

The above will overlay the full Side Menu Panel (and the neighbor[s] on the right depending on the width) with the popup.

1.4 Component Class Overview

SideMenu:

This JPanel is the main panel of the component and will manage the collapsed and expansion panels. This panel is simply a container around two other specified components (either application provided or the default components provided with this component). The SideMenuPanel works with a SideMenuModel to determine whether the overall title used, whether it's in an expanded state, whether the expanded panel will float over other panels (or be inserted alongside them) and what the absolute width of the expanded panel will

be. The class also provides a number of convenience methods to modify the state in the model.

SideMenuModel:

This interface defines the contract for the side menu model. The side menu model will be used by the SideMenuPanel to manage the business variables for the component. The menu model will be responsible for listener management, storage of the variables, retrieval/setting of the variables and notification to listeners of changes to the variables. The developer should note that this interface contains constants that represent the property names that will be passed in the property change event.

DefaultSideMenuModel:

This class provides a default implementation of the SideMenuModel interface. This model will use the event manager component to manage the property change listeners, will provide the storage and management of the business variables that will be utilized by the SideMenuPanel (including notification of listeners whenever a variable changes).

AbstractSideMenuPanel:

This abstract class (that inherits from JPanel) can serve as the base class for all SideMenu related panels. This class will provide common services to all panels. Currently, the common services include storage and management of the related side menu model and an internal property change listener to that model (which will notify a protected method that can be overridden to process side menu model property change events). This class will fire a property change event whenever the side menu model itself changes.

DefaultCollapsedSideMenuPanel:

This JPanel can be used as the default collapsed panel used by the side menu. This panel will contain a vertical label (describing the title), an expansion button on the top and bottom (to allow the panel to expand). This panel will work directly with the side menu model to define the label and expand operations. The visibility of the expansion buttons can be controlled independently through the setXXXExpandVisible methods. However, if an icon has not been assigned - both buttons will be hidden. This panel also will allow 'click to expand' operation if enabled - this allows the user to simply click on the panel to expand it. . The application can customize the default buttons by calling getXXXButton and customizing it - or the application can simply set a new button into via the setXXXButton (same applies to the text label).

DefaultExpandedSideMenuPanel:

This JPanel can be used as the default expansion panel used by the side menu panel. This panel will contain a JLabel (describing the title), a 'pin' button

(that will allow the panel to be pinned when pressed down [i.e. non-floating]) and a collapse button (to allow the panel to collapse down. This panel will work directly with the side menu model to define the label and floating/collapse operations. The visibility of the collapse and pin button is controlled by whether they have associated icons (if the button has an icon, the button is visible - if it has no icon, the button is hidden). The application can customize the default buttons by calling getXXXButton and customizing it - or the application can simply set a new button into via the setXXXButton (same applies to the text label).

1.5 Component Exception Definitions

None defined

1.6 Thread Safety

This component is not thread safe since it has mutable state that is not protected by any locking mechanism.

Please note that since this is a Swing graphical component, it is expected that modifications to the component will be done under the AWT event handler

2. Environment Requirements

2.1 Environment

- Development language: Java 1.5
- Compile target: Java 1.5

2.2 TopCoder Software Components

Generic Event Manager version 1.0 is used by the component to provide management and firing of property changes.

Proposed Component: The Vertical Text Label version 1.0 is used by the component to provide vertical text on the collapsed panel.

NOTE: The default location for TopCoder Software component jars is../lib/tcs/COMPONENT_NAME/COMPONENT_VERSION relative to the component installation. Setting the tcs_libdir property in topcoder_global.properties will overwrite this default location.

2.3 Third Party Components

- None

NOTE: The default location for 3rd party packages is ../lib relative to this component installation. Setting the ext_libdir property in topcoder_global.properties will overwrite this default location.

3. Installation and Configuration

3.1 Package Name

com.topcoder.gui.sidemenu holds the main classes

3.2 Configuration Parameters

None needed

3.3 Dependencies Configuration

Please see the dependency components found in section 2.2 for any configuration needs.

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- Execute 'ant test' within the directory that the distribution was extracted to.

4.2 Required steps to use the component

Extract the component distribution

4.3 Demo

4.3.1 Creating a side menu panel by adding it to a panel

Assume there is a panel called appPanel that the side menu will be added to (on the west side via BorderLayout) and we have an icon for both the expanded buttons and collapse buttons :

```
// Get the tools panel (app method)
JPanel toolsPanel = getToolsPanel();

// Create the side menu for it
SideMenuPanel smp =
   new SideMenuPanel("Tools",toolsPanel,expandIcon,collapseIcon);

// Set the width to 100
smp.setAbsoluteWidth(100);
```

```
// Start it in an expanded state
smp.setExpanded(true);

// Add it to our panel
appPanel.add(smp, BorderLayout.CENTER);
```

4.3.2 *Creating a side menu panel that is similar to Visual Studio*

Visual Studio has a 'collapsed' panel that will expand when clicked and the panel can be pinned or floating. The following will provide a look that is similar:

```
// Create a side menu model with a title of tools, non expanded
// with a floating panel (when expanded) with a size of 100
SideMenuModel smm = new DefaultSideMenuModel("Tools", false, true, 100)

// Setup the default collapsed panel using the above model
// with no expansion buttons and setting click to expand to true
DefaultCollapsedSideMenuPanel dcp =
   new DefaultCollapsedSideMenuPanel(smm,null,true);

// Get the tools panel (app method)
JPanel toolsPanel = getToolsPanel();

// Setup the default expansion panel using the above model
// with no expansion buttons and setting click to expand to true
DefaultExpandedSideMenuPanel dep =
   new DefaultExpandedSideMenuPanel(smm,toolsPanel,collapseIcon,pinIcon);

// Create the side menu for it
SideMenuPanel smp = new SideMenuPanel(smm, dcp, dep);

// Add it to our panel
appPanel.add(smp, BorderLayout.CENTER);
```

4.3.3 *Listening for events*

The Side Menu Model can be used to perform custom actions when the menu is expanded or collapsed:

```
// Get the tools panel (app method)
JPanel toolsPanel = getToolsPanel();

// Create the side menu for it
SideMenuPanel smp =
   new DefaultSideMenuPanel("Tools",toolsPanel,expandIcon,collapseIcon);

// Add a handler for expanded events
smp.getSideMenuModel().addPropertyChangeListener(
     new PropertyChangeListener() {
        public void propertyChanged(PropertyChangeEvent evt) {
           // If expansion changed - notify an application class
           if(evt.getPropertyName().equals(SideMenuModel.EXPANDED)) {
              appNotification.toolsExpanded(evt.getNewValue());
           }
        }
     });
```

5. Future Enhancements

Create additional implementations for the collapse and expansion panels.

Provide the ability to expand the collapsed panel when the mouse moves over it.

Provide the ability to automatically close the floating expanded panel when it loses focus.