Style Panel 1.1 Component Specification

This specification is the same as the one corresponding to v1.0 of the component. The only changes in v1.1 are lied to the appearance of the component. Check the Requirements Specification for more details.

1. Design

The Style Panel component provides a Swing panel that allows the user to set the position, the size, the colors and the font properties for elements and group of elements. It also provides a way to signal the listeners of changes.

The intention of this Style Panel design is to support some styling of some Diagram Interchange classes. Three classes are supported out-of-the-box. They are GraphNode, GraphEdge, and Polyline. If in the future there is another class that needs to be supported by this StylePanel, a new adapter implements StyleObject needs to be created.

To show the style on the StylePanel, an instance of adapter to StyleObject should be created for each object. One or more StyleObject can be set to the StylePanel. StylePanel will determine what styles that are commonly supported and if they have the same value or not. In case of they have different values then a 'blank' value will be shown in the StylePanel.

If an object wants to be notified whenever there is a style change, the object has to implement StyleListener and registers to the StylePanel. When a change happens in the panel, a style event for that change is created and sent to all style listeners registered in the style panel.

1.1 Design Patterns

MVC, this component separates model, view, and controller into different classes.

Adapter, there is a generic style object (an object that can have some styles). An adapter to GraphNode, GraphEdge, and Polyline to StyleObject is provided.

Observer, there is a StyleListener that acts as an Observer.

1.2 Industry Standards

JFC/Swing

1.3 Required Algorithms

Not needed.

1.4 Component Class Overview

*1.4.1 **package** com.topcoder.gui.panels.style*

1.4.1.1 StylePanel

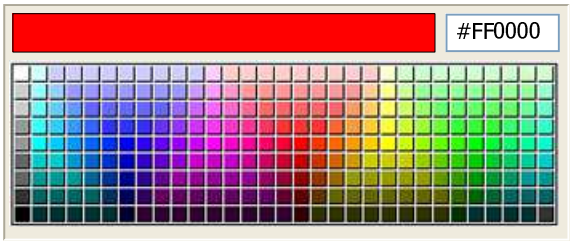
This class is the main class of the component. This class responsible for the main UI of the StylePanel. This panel will be divided into three parts. The top panel contains widgets for changing size and position style. The middle panel contains widgets for coloring. The bottom panel contains widgets for font. There is a separator between each part. All changes will raise event that will fire all listeners.

setStyleObject or setStyleObjectList should be called when this panel have to display style for other components. If one of the style object is not supporting a style, the widget corresponds to the style should not be visible. If all widgets in a part are not visible, this part will also be hidden. If all style objects support the style and have the same value, the widget will shows the value, otherwise, an empty value will be displayed in the style panel.

1.4.1.2 ColorChooserPanel

This is the panel that will be shown when user click on fill color, line color, and text color box.

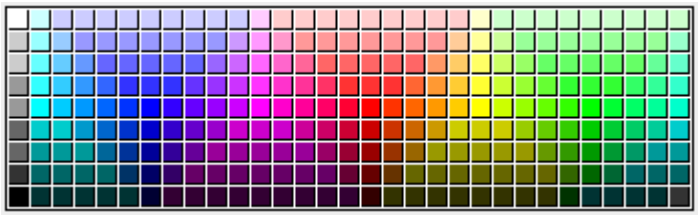
There are three widgets in the panel. The layout used is BorderLayout. In the CENTER, there is a ColorPalettePanel, and in the NORTH, there will be another panel. This panel will use FlowLayout. The first widget is a JButton colored with the last selected color and the second widget is a JTextField contains the RGB value of the last selected color.



1.4.1.3 ColorPalettePanel

This is the panel that will fill the CENTER of the ColorChooserPanel.

This panel will look like this (the default):



The number of rows and columns and the size of each color box can be customized. Please refer to Configuration Parameter section for the detail of this. If the number of rows and columns is defined via configuration, then all of the colors have to be defined also in the configuration. Keep in mind that any inconsistency should trigger StylePanelConfigurationException.

The default number of rows is 9 and the default number of columns is 31. The default size for each color box is 10 x 10 pixels. The default colors are as follow:

Row 1	Row 2	Row 3	Row 4	Row 5	Row 6	Row 7	Row 8	Row 9
255, 255, 255	204, 204, 204	204, 204, 204	153, 153, 153	153, 153, 153	102, 102, 102	102, 102, 102	51, 51, 51	0, 0, 0
204, 255, 255	153, 255, 255	102, 255, 255	51, 255, 255	0, 255, 255	0, 204, 204	0, 153, 153	0, 102, 102	0, 51, 51
204, 204, 255	153, 204, 255	102, 204, 255	51, 204, 255	0, 204, 255	0, 204, 204	0, 153, 153	0, 102, 102	0, 51, 51
204, 204, 255	153, 153, 255	102, 153, 255	51, 153, 255	0, 153, 255	0, 153, 204	0, 153, 153	0, 102, 102	0, 51, 51
204, 204, 255	153, 153, 255	102, 102, 255	51, 102, 255	0, 102, 255	0, 102, 204	0, 102, 153	0, 102, 102	0, 51, 51
204, 204, 255	153, 153, 255	102, 102, 255	51, 51, 255	0, 51, 255	0, 51, 204	0, 51, 153	0, 51, 102	0, 51, 51
204, 204, 255	153, 153, 255	102, 102, 255	51, 51, 255	0, 0, 255	0, 0, 204	0, 0, 153	0, 0, 102	0, 0, 51
204, 204, 255	153, 153, 255	102, 102, 255	51, 51, 255	51, 0, 255	51, 0, 204	51, 0, 153	51, 0, 102	51, 0, 51
204, 204, 255	153, 153, 255	102, 102, 255	102, 51, 255	102, 0, 255	102, 0, 204	102, 0, 153	102, 0, 102	51, 0, 51
204, 204, 255	153, 153, 255	153, 102, 255	153, 51, 255	153, 0, 255	153, 0, 204	153, 0, 153	102, 0, 102	51, 0, 51
204, 204, 255	204, 153, 255	204, 102, 255	204, 51, 255	204, 0, 255	204, 0, 204	153, 0, 153	102, 0, 102	51, 0, 51
255, 204, 255	255, 153, 255	255, 102, 255	255, 51, 255	255, 0, 255	204, 0, 204	153, 0, 153	102, 0, 102	51, 0, 51
255, 204, 204	255, 153, 204	255, 102, 204	255, 51, 204	255, 0, 204	204, 0, 204	153, 0, 153	102, 0, 102	51, 0, 51
255, 204, 204	255, 153, 153	255, 102, 153	255, 51, 153	255, 0, 153	204, 0, 153	153, 0, 153	102, 0, 102	51, 0, 51
255, 204, 204,	255, 153, 153	255, 102, 102	255, 51, 102	255, 0, 102	204, 0, 102	153, 0, 102	102, 0, 102	51, 0, 51
255, 204, 204	255, 153, 153	255, 102, 102	255, 51, 51	255, 0, 51	204, 0, 51	153, 0, 51	102, 0, 51	51, 0, 51
255, 204, 204	255, 153, 153	255, 102, 102	255, 51, 51	255, 0, 0	204, 0, 0,	153, 0, 0	102, 0, 0	51, 0, 0

255, 204, 204	255, 153, 153	255, 102, 102	255, 51, 51	255, 51, 0	204, 51, 0	153, 51, 0	102, 51, 0	51, 51, 0
255, 204, 204	255, 153, 153	255, 102, 102	255, 102, 51	255, 102, 0	204, 102, 0,	153, 102, 0	102, 102, 0	51, 51, 0
255, 204, 204	255, 153, 153	255, 153, 102	255, 153, 51	255, 153, 0	204, 153, 0,	153, 153, 0	102, 102, 0	51, 51, 0
255, 204, 204	255, 204, 153	255, 204, 102	255, 204, 51	255, 204, 0	204, 204, 0,	153, 153, 0	102, 102, 0	51, 51, 0
255, 255, 204	255, 255, 153	255, 255, 102	255, 255, 51	255, 255, 0	204, 204, 0,	153, 153, 0	102, 102, 0	51, 51, 0
204, 255, 204	204, 255, 153	204, 255, 102	204, 255, 51	204, 255, 0	204, 204, 0,	153, 153, 0	102, 102, 0	51, 51, 0
204, 255, 204	153, 255, 153	153, 255, 102	153, 244, 51	153, 255, 0	153, 204, 0,	153, 153, 0	102, 102, 0	51, 51, 0
204, 255, 204	153, 255, 153	102, 255, 102	102, 255, 51	102, 255, 0	102, 204, 0,	102, 153, 0	102, 102, 0	51, 51, 0
204, 255, 204	153, 255, 153	102, 255, 102	51, 255, 51	51, 255, 0	51, 204, 0	51, 153, 0	51, 102, 0	0, 51, 0
204, 255, 204	153, 255, 153	102, 255, 102	51, 255, 51	0, 255, 0	0, 204, 0	0, 153, 0	0, 102, 0	0, 51, 51
204, 255, 204	153, 255, 153	102, 255, 102	51, 255, 51	0, 255, 51	0, 204, 51	0, 153, 51	0, 102, 51	0, 51, 51
204, 255, 204	153, 255, 153	102, 255, 102	51, 255, 102	0, 255, 102	0, 204, 102	0, 153, 102	0, 102, 102	0, 51, 51
204, 255, 204	153, 255, 153	102, 255, 153	51, 255, 153	0, 255, 153	0, 204, 153	0, 153, 153	0, 102, 102	0, 51, 51
204, 255, 204	153, 255, 204	102, 255, 204	51, 255, 204	0, 255, 204	0, 204, 204	0, 153, 153	0, 102, 102	51, 51, 51

1.4.1.4 FontFamilySelector

This class extends JComboBox and acts as a widget to select font family. As the default it will read all fonts that are available in the system. If needed, it can also be configured.

1.4.1.5 FontSizeSelector

This class extends JComboBox and acts as a widget to select font size. As the default it will show several common font sizes. If needed, it can also be configured.

1.4.1.6 StyleObject

The generic class for object that is supported by this component.

1.4.1.7 StyleEvent

An event that will be raised in case of style change.

1.4.1.8 StyleListener

The interface that needs to be implemented by object that wants to be acknowledged in case of style change event.

*1.4.2 **package** com.topcoder.gui.panels.style.styleobject*

1.4.2.1 AbstractStyleObjectAdapter

This class implements StyleObject and provides common operation for the adapter.

1.4.2.2 GraphNodeStyleObjectAdapter

This class extends AbstractStyleObjectAdapter and is an adapter for GraphNode.

1.4.2.3 GraphEdgeStyleObjectAdapter

This class extends AbstractStyleObjectAdapter and is an adapter for GraphEdge.

1.4.2.4 PolylineStyleObjectAdapter

This class extends AbstractStyleObjectAdapter and is an adapter for Polyline.

1.5 Component Exception Definitions

1.5.1 StylePanelException

This class is extending BaseException and is the base exception for the component.

1.5.2 StylePanelConfigurationException

This class is extending **StylePanelException**.

This is the exception that will be thrown if there is some errors happen when creating ColorPalletePanel.

1.5.3 StyleNotSupportedException

This class is extending **StylePanelException**.

This is the exception that will be thrown if there some style is not supported by a StyleObject.

1.6 Thread Safety

Some classes are swing widgets and therefore are not thread safe. But since all of swing operation will be done in Event Dispatching Thread, it should not be a problem. All of the exception classes are thread-safe. The abstract adapter is also not thread safe, and so the children of the abstract adapter. This is also no problem for the component, since they will also be used in Event Dispatching Thread.

2. Environment Requirements

2.1 Environment

At minimum, Java1.5 is required for compilation and executing test cases.

2.2 TopCoder Software Components

- Base Exception 1.0, used as the parent class of all exception in the component.
- Configuration Manager 2.1.5, used to configure some properties in the component.
- Diagram Interchange 1.0. Some classes from this component are used as style objects. Adapters for those classes are also provided in the component.

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.

3. Installation and Configuration

3.1 Package Name

com.topcoder.gui.panels.style

com.topcoder.gui.panels.style.styleobject

(style_panel_1.0.jar)

3.2 Configuration Parameters

Properties used in StylePanel (via ConfigManager)

Parameter	Description	Values
looknfeel.fontName	Font name used in the component	Must be valid in the system (optional)
looknfeel.minimumWidth	Minimum width of the style panel	must be an integer (optional)
looknfeel.minimumHeight	Minimum height of the style panel	must be an integer (optional)
looknfeel.preferredWidth	Preferred width of the style panel	must be an integer bigger / same as minimumWidth (optional)
looknfeel.preferredHeight	Preferred height of the style panel	must be an integer bigger / same as minimumHeight (optional)
propertyKey.widthKey	The key used to store width style in the property	Not empty string (optional)
propertyKey.heightKey	The key used to store height style in the property	Not empty string (optional)
propertyKey.xKey	The key used to store x style in the property	Not empty string (optional)
propertyKey.yKey	The key used to store y style in the property	Not empty string (optional)
propertyKey.fontNameKey	The key used to store font name style in the property	Not empty string (optional)
propertyKey.fontSizeKey	The key used to store font size style in the property	Not empty string (optional)
propertyKey.fillColorKey	The key used to store fill color style in the property	Not empty string (optional)
propertyKey.outlineColorKey	The key used to store outline color style in the property	Not empty string (optional)
propertyKey.textColorKey	The key used to store text color style in the property	Not empty string (optional)

Properties used in ColorPalettePanel (via ConfigManager)

Parameter	Description	Values
palette.width	The width of every palette color.	must be an integer (optional). This became required if palette.height is defined.
palette.height	The height of every palette color.	must be an integer (optional). This became required if palette.width is defined.
palette.columns	The number of columns of palette color.	must be an integer (optional). This became required if palette.rows is defined.
palette.rows	The number of rows of palette color.	must be an integer (optional). This became required if palette.columns is defined.
palette.color.colXrowY	The color for every column and row.	Must be in form of #RRGGBB (optional). This became required if palette.columns & palette.rows are defined. The number of this properties has to be consistent with palette.rows and palette.columns. For example, if palette.rows = 3 and palette.columns = 2, there should be pallete.col0row0, pallete.col0row1, pallete.col0row2, pallete.col1row0, pallete.col1row1, and pallete.col1row2,

Properties used in FontFamilySelector (via ConfigManager)

Parameter	Description	Values
fontFamily.FONT_FAMILY_ID	The font family used in FontFamilySelector.	Must be available in the system (optional)
fontFamily.default	The default font family name	Must be available in the system (optional)

Properties used in FontSizeSelector (via ConfigManager)

Parameter	Description	Values
fontSize.FONT_SIZE_ID	The font size used in FontFamilySelector.	Must be an integer (optional)
fontSize.default	The default font size	Must be an integer (optional)

Default value will be used if one of the properties is missing. If the configuration is not consistent or contains wrong value, StylePanelConfigurationException should be raised. This includes the empty string, unparseable number, and unavailable font.

See sampleconfig.xml for example of complete configuration file.

3.3 Dependencies Configuration

None applicable.

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

Provide the configuration file.

Provide properties file for ConfigManager component.

4.3 Demo

4.3.1 Default namespace demo

```
// Prepare style objects
StyleObject styleObject1 = new GraphNodeStyleObjectAdapter(graphNode);
StyleObject styleObject2 = new GraphEdgeStyleObjectAdapter(graphEdge);
StyleObject styleObject3 = new PolylineStyleObjectAdapter(polyline);

List<StyleObject> list = new ArrayList<StyleObject>();
list.add(styleObject1);
list.add(styleObject2);
list.add(styleObject3);
// using default constructor
StylePanel stylePanel = new StylePanel();

StyleListener theListener = new StyleListerImpl();
stylePanel.addStyleListener(theListener);

// set panel to display information about one style object
stylePanel.setStyleObject(styleObject1);

// after this style panel will contain styleObject1, and it supports all styles.
// user can change any value. In event of change, style event will be sent theListener.

// after this style event about font family change will be sent to theListener
stylePanel.setFontFamilyStyle("Arial");

// after this style event about font size change will be sent to theListener
stylePanel.setFontSizeStyle(12);

// after this style event about fill color change will be sent to theListener
stylePanel.setFillColorStyle(Color.RED);

// after this style event about outline color change will be sent to theListener
stylePanel.setOutlineColorStyle(Color.BLUE);

// after this style event about text color change will be sent to theListener
stylePanel.setTextColorStyle(Color.ORANGE);

// after this style event about size change will be sent to theListener
stylePanel.setSizeStyle(new Dimension(10, 20));

// after this style event about position change will be sent to theListener
stylePanel.setPositionStyle(new Point(10, 20));

// set panel to display information about all style object
stylePanel.setStyleObjectList(list);

// style panel will contain all styles that are related to the
```

```
        // objects in the list.
        // user can only change outline color.
        // In event of change, style event will be sent theListener.

        // after this style event about outline color change will be sent to theListener


        stylePanel.setOutlineColorStyle(Color.BLUE);
```

4.3.2 Custom namespace demo

```
         // Prepare style objects
        StyleObject styleObject1 = new GraphNodeStyleObjectAdapter(graphNode);
        StyleObject styleObject2 = new GraphEdgeStyleObjectAdapter(graphEdge);
        StyleObject styleObject3 = new PolylineStyleObjectAdapter(polyline);

        List<StyleObject> list = new ArrayList<StyleObject>();
        list.add(styleObject1);
        list.add(styleObject2);
        list.add(styleObject3);
        // using constructor with namespace specified
        StylePanel stylePanel = new StylePanel(TestHelper.NAMESPACE);

        StyleListener theListener = new StyleListerImpl();
        stylePanel.addStyleListener(theListener);

        // set panel to display information about one style object
        stylePanel.setStyleObject(styleObject1);

        // after this style panel will contain styleObject1, and it supports all styles.
        // user can change any value. In event of change, style event will be sent theListener.

        // after this style event about font family change will be sent to theListener
        stylePanel.setFontFamilyStyle("Arial");

        // after this style event about font size change will be sent to theListener
        stylePanel.setFontSizeStyle(12);

        // after this style event about fill color change will be sent to theListener
        stylePanel.setFillColorStyle(Color.RED);

        // after this style event about outline color change will be sent to theListener
        stylePanel.setOutlineColorStyle(Color.BLUE);

        // after this style event about text color change will be sent to theListener
        stylePanel.setTextColorStyle(Color.ORANGE);

        // after this style event about size change will be sent to theListener
        stylePanel.setSizeStyle(new Dimension(10, 20));

        // after this style event about position change will be sent to theListener
        stylePanel.setPositionStyle(new Point(10, 20));

        // set panel to display information about all style object
        stylePanel.setStyleObjectList(list);

        // style panel will contain all styles that are related to the
        // objects in the list.
        // user can only change outline color.
        // In event of change, style event will be sent theListener.

        // after this style event about outline color change will be sent to theListener

        stylePanel.setOutlineColorStyle(Color.BLUE);
```

4.3.3 Classic use

```
public class FrameDemo extends JFrame {
    /**
     * <p>
     * The content panel of this frame.
     * </p>
```

```
 */
private JPanel jContentPane;

/**
 * <p>
 * The style panel to be used in this frame.
 * </p>
 */
private StylePanel stylePanel;

/**
 * <p>
 * The check box to be used in this frame.
 * </p>
 *
 * <p>
 * When checked, one GrahEdgeStyleObjectAdapter instance will be created and set the
 * style panel if the user presses ok button.
 * </p>
 */
private JCheckBox graphEdgeCheckbox;

/**
 * <p>
 * The check box to be used in this frame.
 * </p>
 *
 * <p>
 * When checked, one GrahNodeStyleObjectAdapter instance will be created and set the
 * style panel if the user presses ok button.
 * </p>
 */
private JCheckBox graphNodeCheckbox;

/**
 * <p>
 * The check box to be used in this frame.
 * </p>
 *
 * <p>
 * When checked, one PolylineStyleObjectAdapter instance will be created and set the
 * style panel if the user presses ok button.
 * </p>
 */
private JCheckBox polylineCheckbox;

/**
 * <p>
 * This is the default constructor.
 * </p>
 */
public FrameDemo() {
    initialize();
}

/**
 * <p>
 * This method initializes the widgets in this frame.
 * </p>
 */
public void initialize() {
    this.setContentPane(getJContentPane());
    this.setTitle("Frame Demo");
}

/**
 * <p>
 * This method initializes jContentPane.
 * </p>
 *
 * @return the content panel for this frame.
```

```
 */
public JPanel getJContentPane() {
    if (jContentPane == null) {
        jContentPane = new JPanel();
        jContentPane.setLayout(new BoxLayout(jContentPane, BoxLayout.Y_AXIS));

        StylePanel panel = getStylePanel();

        jContentPane.add(getGraphEdgeCheckbox());
        jContentPane.add(getGraphNodeCheckbox());
        jContentPane.add(getPolylineCheckbox());

        JButton button = new JButton("OK");
        button.addActionListener(new UpdateStylePanelAction());

        jContentPane.add(button);

        jContentPane.add(new JSeparator());

        jContentPane.add(panel);
    }
    return jContentPane;
}

/**
 * <p>
 * This method initializes StylePanel.
 * </p>
 *
 * @return the style panel.
 */
public StylePanel getStylePanel() {
    if (stylePanel == null) {
        stylePanel = new StylePanel();
    }

    return stylePanel;
}

/**
 * <p>
 * This method initializes the check box for graph edge.
 * </p>
 *
 * <p>
 * When checked, one GrahEdgeStyleObjectAdapter instance will be created and set the
 * style panel if the user presses ok button.
 * </p>
 *
 * @return the check box for graph edge.
 */
private JCheckBox getGraphEdgeCheckbox() {
    if (graphEdgeCheckbox == null) {
        graphEdgeCheckbox = new JCheckBox("GraphEdge");
    }

    return graphEdgeCheckbox;
}

/**
 * <p>
 * This method initializes the check box for graph node.
 * </p>
 *
 * <p>
 * When checked, one GrahNodeStyleObjectAdapter instance will be created and set the
 * style panel if the user presses ok button.
 * </p>
 *
 * @return the check box for graph node.
 */
```

```
private JCheckBox getGraphNodeCheckbox() {
    if (graphNodeCheckbox == null) {
        graphNodeCheckbox = new JCheckBox("GraphNode");
    }

    return graphNodeCheckbox;
}

/**
 * <p>
 * This method initializes the check box for polyline.
 * </p>
 *
 * <p>
 * When checked, one PolylineStyleObjectAdapter instance will be created and set the
 * style panel if the user presses ok button.
 * </p>
 *
 * @return the check box for polyline.
 */
private JCheckBox getPolylineCheckbox() {
    if (polylineCheckbox == null) {
        polylineCheckbox = new JCheckBox("Polyline");
    }

    return polylineCheckbox;
}

/**
 * <p>
 * This class implements the ActionListener and used for update the style object list
 * for style panel.
 * </p>
 *
 * @author TCSDEVELOPER
 * @version 1.0
 */
private class UpdateStylePanelAction implements ActionListener {
    /**
     * <p>
     * This method handles the button-click event for ok button.
     * </p>
     *
     * <p>
     * When the button is clicked, the style object list of the style panel is updated.
     * </p>
     *
     * @param e an action event that indicates the button is clicked
     */
    public void actionPerformed(ActionEvent e) {
        List<StyleObject> styleObjects = new ArrayList<StyleObject>();
        if (getGraphNodeCheckbox().isSelected()) {
            styleObjects.add(new GraphNodeStyleObjectAdapter(new GraphNode()));
        }

        if (getGraphEdgeCheckbox().isSelected()) {
            styleObjects.add(new GraphEdgeStyleObjectAdapter(new GraphEdge()));
        }

        if (getPolylineCheckbox().isSelected()) {
            styleObjects.add(new PolylineStyleObjectAdapter(new Polyline()));
        }

        stylePanel.setStyleObjectList(styleObjects);
    }

}

/**
 * <p>
 * The entry of the demo program.
```

```
     * </p>
     *
     * @param args the arguments
     */
    public static void main(String[] args) {
        FrameDemo frameDemo = new FrameDemo();
        frameDemo.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frameDemo.pack();
        frameDemo.setVisible(true);
    }

}
```

5. Future Enhancements

None.