Basic Swing components II

Swing components are basic building blocks of an application. Swing toolkit has a wide range of various components, including buttons, check boxes, sliders, list boxes. Everything a programmer needs for his job. In this section of the tutorial, we will describe several useful components.

JList Component

`JList` is a component that displays a list of objects. It allows the user to select one or more items.

```
package com.zetcode;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GraphicsEnvironment;

import javax.swing.BorderFactory;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.SwingUtilities;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;


public class ListExample extends JFrame {

    private JLabel label;
    private JList list;


    public ListExample() {

        initUI();
    }

    private void initUI() {

        JPanel panel = new JPanel();
        panel.setLayout(new BorderLayout());
```

```
        panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

        GraphicsEnvironment ge =
            GraphicsEnvironment.getLocalGraphicsEnvironment();

        String[] fonts = ge.getAvailableFontFamilyNames();

        list = new JList(fonts);
        list.addListSelectionListener(new ListSelectionListener() {
            @Override
            public void valueChanged(ListSelectionEvent e) {
                if (!e.getValueIsAdjusting()) {
                    String name = (String) list.getSelectedValue();
                    Font font = new Font(name, Font.PLAIN, 12);
                    label.setFont(font);
                }
            }
        });

        JScrollPane pane = new JScrollPane();
        pane.getViewport().add(list);
        pane.setPreferredSize(new Dimension(250, 200));
        panel.add(pane);

        label = new JLabel("Aguirre, der Zorn Gottes");
        label.setFont(new Font("Serif", Font.PLAIN, 12));
        add(label, BorderLayout.SOUTH);

        add(panel);

        pack();
        setTitle("JList");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                ListExample ex = new ListExample();
                ex.setVisible(true);
            }
        });
```

```
    }
}
```

In our example, we will display a `JList` and `JLabel` components. The list component contains a list of all available font family names on our system. If we select an item from the list, the label will be displayed in a font, we have chosen.

```
GraphicsEnvironment ge =
    GraphicsEnvironment.getLocalGraphicsEnvironment();

String[] fonts = ge.getAvailableFontFamilyNames();
```

Here we obtain all possible font family names on our system.

```
list = new JList(fonts);
```

We create a `JList` component.

```
public void valueChanged(ListSelectionEvent e) {
    if (!e.getValueIsAdjusting()) {
```

Events in list selection are grouped. We receive events for both selecting and deselecting. To filter only the selecting events, we use the `getValueIsAdjusting()` method.

```
String name = (String) list.getSelectedValue();
Font font = new Font(name, Font.PLAIN, 12);
label.setFont(font);
```

We get the selected item and set a new font for the label.

```
JScrollPane pane = new JScrollPane();
pane.getViewport().add(list);
```

`JLabel` component is not scrollable by default. We put the list into the `JScrollPane` to make it scrollable.

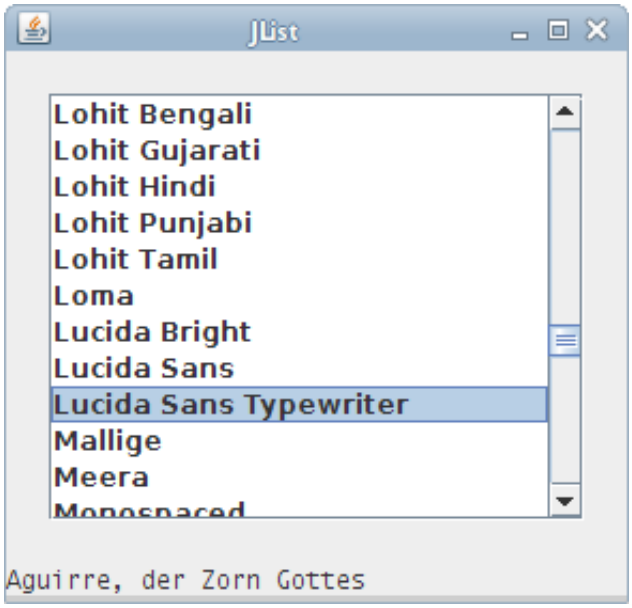

Figure: JList

JTextArea component

A `JTextArea` is a multiline text area that displays plain text. It is lightweight component for working with text. The component does not handle scrolling. For this task, we use `JScrollPane` component.

```
package com.zetcode;

import java.awt.BorderLayout;
import java.awt.Dimension;

import javax.swing.BorderFactory;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.SwingUtilities;

public class TextAreaExample extends JFrame {

    public TextAreaExample() {

        initUI();
    }

    private void initUI() {

        JPanel panel = new JPanel();
        panel.setLayout(new BorderLayout());
        panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));
```

```
        JScrollPane pane = new JScrollPane();
        JTextArea area = new JTextArea();

        area.setLineWrap(true);
        area.setWrapStyleWord(true);
        area.setBorder(BorderFactory.createEmptyBorder(8, 8, 8, 8));

        pane.getViewport().add(area);
        panel.add(pane);

        add(panel);

        setTitle("JTextArea");
        setSize(new Dimension(350, 300));
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                TextAreaExample ex = new TextAreaExample();
                ex.setVisible(true);
            }
        });
    }
}
```

The example shows a simple `JTextArea` component.

```
JTextArea area = new JTextArea();
```

This is the constructor of the `JTextArea` component.

```
area.setLineWrap(true);
```

The `setLineWrap()` makes the lines wrapped if they are too long to fit the text area's width.

```
area.setWrapStyleWord(true);
```

Here we specify, how is line going to be wrapped. In our case, lines will be wrapped at word boundaries—white spaces.

```
area.setBorder(BorderFactory.createEmptyBorder(8, 8, 8, 8));
```

We put some border around the text in the component.

```
pane.getViewport().add(area);
```

To make the text scrollable, we put the `JTextArea` component into the `JScrollPane` component.

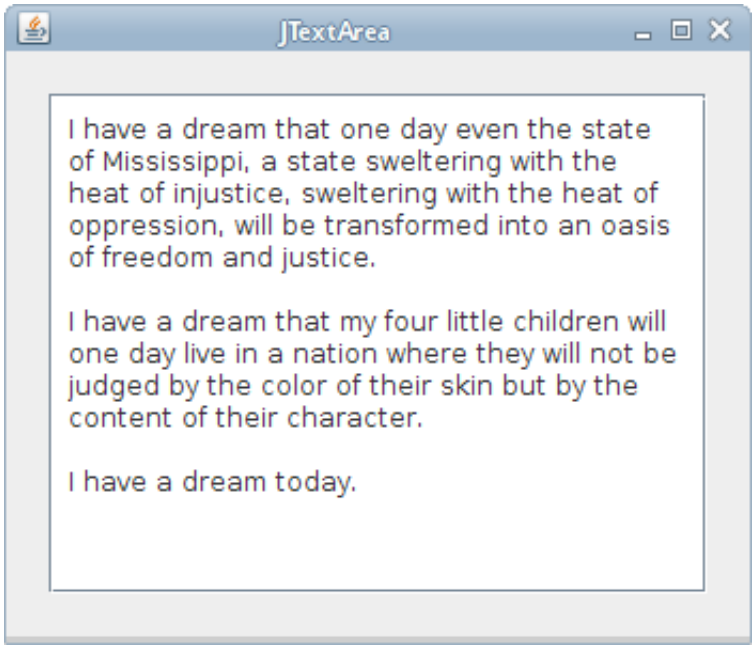

Figure: JTextAra

JTextPane component

`JTextPane` component is a more advanced component for working with text. The component can do some complex formatting operations over the text. It can display also HTML documents.

```
package com.zetcode;

import java.awt.BorderLayout;

import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.swing.BorderFactory;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextPane;
import javax.swing.SwingUtilities;

public class TextPaneExample extends JFrame {
```

```
    JTextPane textPane;

    public TextPaneExample() {

        initUI();
    }

    private void initUI() {

        JPanel panel = new JPanel();
        panel.setLayout(new BorderLayout());
        panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

        JScrollPane pane = new JScrollPane();
        textPane = new JTextPane();

        textPane.setContentType("text/html");
        textPane.setEditable(false);

        textPane.setBorder(BorderFactory.createEmptyBorder(8, 8, 8, 8));

        loadFile();

        pane.getViewport().add(textPane);
        panel.add(pane);

        add(panel);
        pack();

        setTitle("JTextPane");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
    }

    private void loadFile() {

        try {
            String cd = System.getProperty("user.dir") + "/";
            textPane.setPage("File:///" + cd + "test.html");
        } catch (IOException ex) {

Logger.getLogger(TextPaneExample.class.getName()).log(Level.SEVERE,
                    null, ex);
        }
```

```
    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                TextPaneExample ex = new TextPaneExample();
                ex.setVisible(true);
            }
        });
    }
}
```

This is the HTML code that we are loading into the `JTextPane` component. The component does not handle scrolling.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>A simple html document</title>
</head>
<body>

<h2>A simple html document</h2>


<p>
<b>JTextPane</b> can display HTML documents.
</p>

<br>

<pre>
 JScrollPane pane = new JScrollPane();
 JTextPane textpane = new JTextPane();

 textpane.setContentType("text/html");
 textpane.setEditable(false);
</pre>

<br>
<small>The Java Swing tutorial, 2013</small>

</body>
```

```
</html>
```

In our example we show a `JTextPane` component and load a HTML document. Example shows formatting capabilities of the component.

```
JTextPane textpane = new JTextPane();

textpane.setContentType("text/html");
textpane.setEditable(false);
```

We create a `JTextPane` component, set the content of the component to be a HTML document and disable editing.

```
private void loadFile() {

    try {
        String cd = System.getProperty("user.dir") + "/";
        textPane.setPage("File:///" + cd + "test.html");
    } catch (IOException ex) {
        Logger.getLogger(TextPaneExample.class.getName()).log(Level.SEVERE,
                null, ex);
    }
}
```

Here we determine the current working directory of the user. We load a HTML document into the pane.

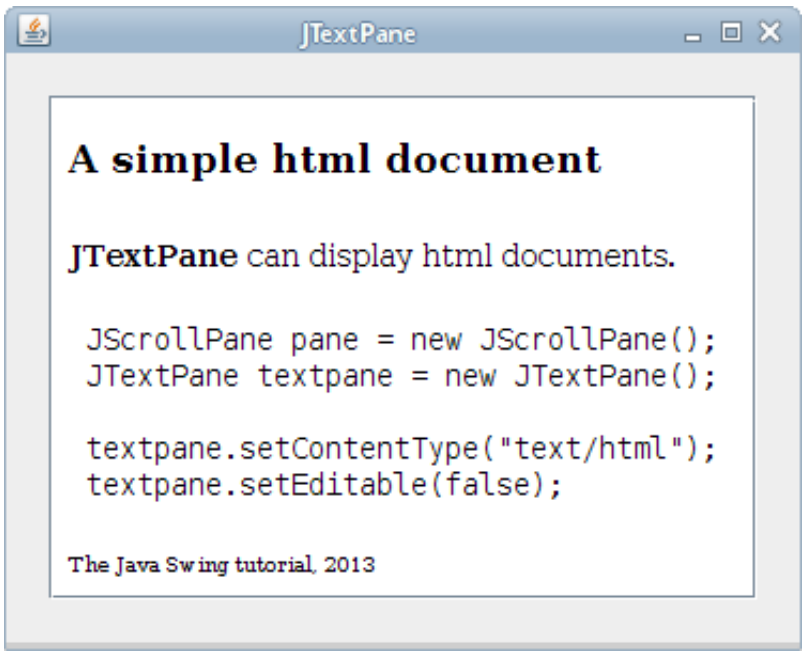

Figure: JTextPane

In this chapter, we have continued covering basic Swing components.