Basic Swing components

Swing components are basic building blocks of an application. Swing has a wide range of various components, including buttons, check boxes, sliders, and list boxes.

In this part of the Swing tutorial, we will present JButton, JLabel, JTextField, and JPasswordField.

JButton

JButton is in implementation of a push button. It is used to trigger an action if the user clicks on it.

Displaying text and icons

JButton can display a text, an icon, or both.

ImageIconButtonEx.java

```
package com.zetcode;

import javax.swing.GroupLayout;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;
import java.awt.EventQueue;

public class ImageIconButtonEx extends JFrame {

    public ImageIconButtonEx() {

        initUI();
    }

    private void initUI() {

        var saveIcon = new ImageIcon("src/resources/save.png");
        var homeIcon = new ImageIcon("src/resources/home.png");

        var quitBtn = new JButton("Quit");
        var saveBtn = new JButton(saveIcon);
        var homeBtn = new JButton("Home", homeIcon);

        createLayout(quitBtn, saveBtn, homeBtn);

        setTitle("JButtons");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
    private void createLayout(JComponent... arg) {

        var pane = getContentPane();
        var gl = new GroupLayout(pane);
        pane.setLayout(gl);

        gl.setAutoCreateContainerGaps(true);
        gl.setAutoCreateGaps(true);

        gl.setHorizontalGroup(gl.createSequentialGroup()
                .addComponent(arg[0])
                .addComponent(arg[1])
                .addComponent(arg[2])
        );

        gl.setVerticalGroup(gl.createParallelGroup()
                .addComponent(arg[0])
                .addComponent(arg[1])
                .addComponent(arg[2])
        );

        gl.linkSize(arg[0], arg[1], arg[2]);

        pack();
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(() -> {

            var ex = new ImageIconButtonEx();
            ex.setVisible(true);
        });
    }
}
```

The example shows three buttons: one displays text, one icon, and one both text and icon.

```
var saveIcon = new ImageIcon("src/main/resources/save.png");
```

Many components can be decorated with icons; for this we use the ImageIcon class.

```
var quitBtn = new JButton("Quit");
```

This JButton constructor takes a text as a parameter.

```
var saveBtn = new JButton(saveIcon);
```

In this JButton constructor we pass an icon.

```
JButton homeBtn = new JButton("Home", homeIcon);
```

This button displays text and icon.

```
gl.linkSize(arg[0], arg[1], arg[2]);
```

With the GroupLayout's linkSize() method, we make the button the same size.

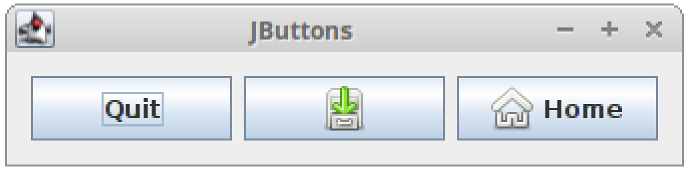


Figure: JButtons

JButton with a mnemonic

A *mnemonic* a key which when combined with the look and feel's mouseless modifier (usually Alt) will activate this button if focus is contained somewhere within this button's ancestor window.

ButtonMnemonicEx.java

```
package com.zetcode;

import javax.swing.GroupLayout;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;

public class ButtonMnemonicEx extends JFrame implements ActionListener {

    public ButtonMnemonicEx() {

        initUI();
    }

    private void initUI() {

        var showBtn = new JButton("Show");
        showBtn.addActionListener(this);
        showBtn.setMnemonic(KeyEvent.VK_S);

        createLayout(showBtn);

        setTitle("JButton");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
    private void createLayout(JComponent... arg) {

        var pane = getContentPane();
        var gl = new GroupLayout(pane);
        pane.setLayout(gl);

        gl.setAutoCreateContainerGaps(true);
        gl.setAutoCreateGaps(true);

        gl.setHorizontalGroup(gl.createSequentialGroup()
                .addComponent(arg[0])
                .addGap(250)
        );

        gl.setVerticalGroup(gl.createParallelGroup()
                .addComponent(arg[0])
                .addGap(150)
        );

        pack();
    }

    @Override
    public void actionPerformed(ActionEvent e) {

        JOptionPane.showMessageDialog(this, "Button clicked",
                "Information", JOptionPane.INFORMATION_MESSAGE);
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(() -> {

            var ex = new ButtonMnemonicEx();
            ex.setVisible(true);
        });
    }
}
```

The button in this example can be activated with a mouse click or with a `Alt + S` keyboard shortcut.

```
public class ButtonMnemonicEx extends JFrame implements ActionListener
```

The ButtonMnemonicEx class implements the ActionListener; it must override the actionPerformed() method where we put the code that is executed after the button is activated.

```
var showBtn = new JButton("Show");
showBtn.addActionListener(this);
```

A new JButton is created. We add an action listener to the button with the addActionListener() method.

```
showBtn.setMnemonic(KeyEvent.VK_S);
```

The setMnemonic() sets a mnemonic key; the 'S' character is underlined.

```
@Override
public void actionPerformed(ActionEvent e) {

    JOptionPane.showMessageDialog(this, "Button clicked",
            "Information", JOptionPane.INFORMATION_MESSAGE);
}
```

When the button is activated, either via a mouse click or via a shortcut, a message dialog is displayed with JOptionPane.showMessageDialog().

JLabel

JLabel is a simple component for displaying text, images or both. It does not react to input events.

Displaying text

The following example displays text.

LabelEx.java

```
package com.zetcode;

import javax.swing.GroupLayout;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Font;

public class LabelEx extends JFrame {

    public LabelEx() {

        initUI();
    }

    private void initUI() {

        var lyrics =  "<html>It's way too late to think of<br>" +
                "Someone I would call now<br>" +
                "And neon signs got tired<br>" +
                "Red eye flights help the stars out<br>" +
                "I'm safe in a corner<br>" +
                "Just hours before me<br>" +
                "<br>" +
                "I'm waking with the roaches<br>" +
                "The world has surrendered<br>" +
                "I'm dating ancient ghosts<br>" +
                "The ones I made friends with<br>" +
                "The comfort of fireflies<br>" +
                "Long gone before daylight<br>" +
                "<br>" +
                "And if I had one wishful field tonight<br>" +
                "I'd ask for the sun to never rise<br>" +
                "If God leant his voice for me to speak<br>" +
                "I'd say go to bed, world<br>" +
                "<br>" +
                "I've always been too late<br>" +
                "To see what's before me<br>" +
                "And I know nothing sweeter than<br>" +
                "Champaign from last New Years<br>" +
                "Sweet music in my ears<br>" +
                "And a night full of no fears<br>" +
                "<br>" +
                "But if I had one wishful field tonight<br>" +
                "I'd ask for the sun to never rise<br>" +
                "If God passed a mic to me to speak<br>" +
                "I'd say stay in bed, world<br>" +
                "Sleep in peace</html>";

        var label = new JLabel(lyrics);
        label.setFont(new Font("Serif", Font.PLAIN, 14));
        label.setForeground(new Color(50, 50, 25));

        createLayout(label);

        setTitle("No Sleep");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    private void createLayout(JComponent... arg) {

        var pane = getContentPane();
        var gl = new GroupLayout(pane);
        pane.setLayout(gl);

        gl.setAutoCreateContainerGaps(true);

        gl.setHorizontalGroup(gl.createSequentialGroup()
                .addComponent(arg[0])
        );

        gl.setVerticalGroup(gl.createParallelGroup()
                .addComponent(arg[0])
        );

        pack();
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(() -> {

            var ex = new LabelEx();
            ex.setVisible(true);
        });
    }
}
```

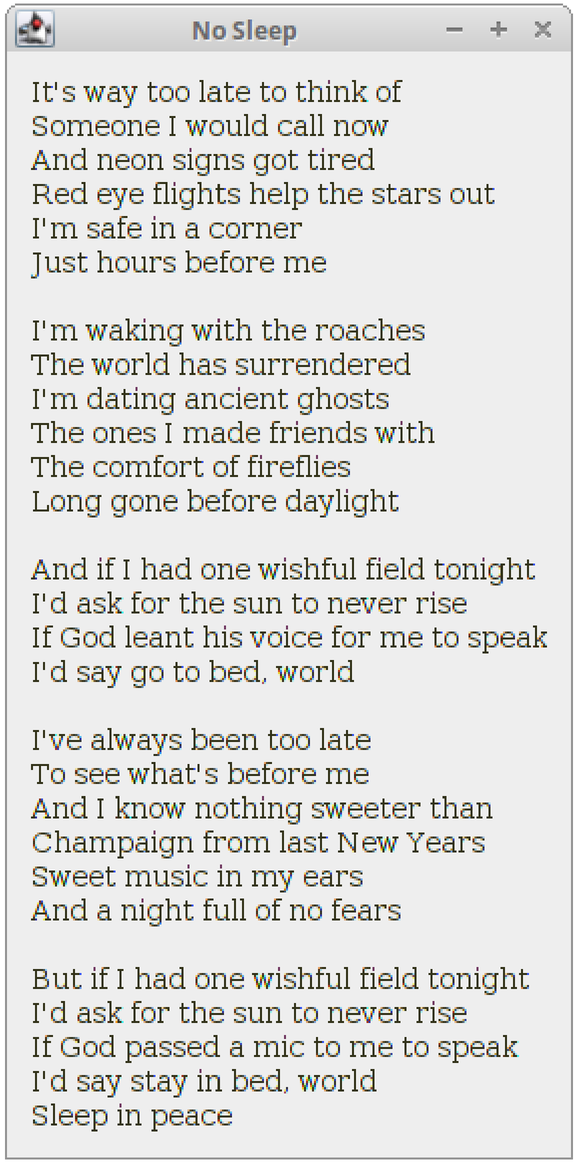
In our example, we show lyrics of a song from Cardigans. We can use HTML tags in JLabel component. We use the <br> tag to separate lines.

```
var label = new JLabel(lyrics);
label.setFont(new Font("Serif", Font.PLAIN, 14));
```

Here we create a label component. We choose a plain Serif font and set its height to 14px.

```
pack();
```

The pack() method resizes the window so that the label component is shown in its preferred size.



```
It's way too late to think of
Someone I would call now
And neon signs got tired
Red eye flights help the stars out
I'm safe in a corner
Just hours before me

I'm waking with the roaches
The world has surrendered
I'm dating ancient ghosts
The ones I made friends with
The comfort of fireflies
Long gone before daylight

And if I had one wishful field tonight
I'd ask for the sun to never rise
If God leant his voice for me to speak
I'd say go to bed, world

I've always been too late
To see what's before me
And I know nothing sweeter than
Champaign from last New Years
Sweet music in my ears
And a night full of no fears

But if I had one wishful field tonight
I'd ask for the sun to never rise
If God passed a mic to me to speak
I'd say stay in bed, world
Sleep in peace
```

Figure: JLabel

Displaying icons

JLabel can be used to display images.

LabelEx2.java

```
package com.zetcode;

import javax.swing.GroupLayout;
import javax.swing.ImageIcon;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.EventQueue;

public class LabelEx2 extends JFrame {

    public LabelEx2() {

        initUI();
    }

    private void initUI() {

        var lbl1 = new JLabel(new ImageIcon("src/resources/cpu.png"));
        var lbl2 = new JLabel(new ImageIcon("src/resources/drive.png"));
        var lbl3 = new JLabel(new ImageIcon("src/resources/laptop.png"));
        var lbl4 = new JLabel(new ImageIcon("src/resources/player.png"));

        createLayout(lbl1, lbl2, lbl3, lbl4);

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

    private void createLayout(JComponent... arg) {
        var pane = getContentPane();
        var gl = new GroupLayout(pane);
        pane.setLayout(gl);

        gl.setAutoCreateContainerGaps(true);
        gl.setAutoCreateGaps(true);

        gl.setHorizontalGroup(gl.createSequentialGroup()
                .addComponent(arg[0])
                .addComponent(arg[1])
                .addComponent(arg[2])
                .addComponent(arg[3])
        );

        gl.setVerticalGroup(gl.createParallelGroup()
                .addComponent(arg[0])
                .addComponent(arg[1])
                .addComponent(arg[2])
                .addComponent(arg[3])
        );

        pack();
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(() -> {

            var ex = new LabelEx2();
            ex.setVisible(true);
        });
    }
}
```

In the example, we use the JLabel component to display four icons.

```
var lbl1 = new JLabel(new ImageIcon("src/main/resources/cpu.png"));
```

JLabel takes an ImageIcon as a parameter. An icon is a fixed-sized image. ImageIcon paints an icon from a GIF, JPEG, or PNG image.

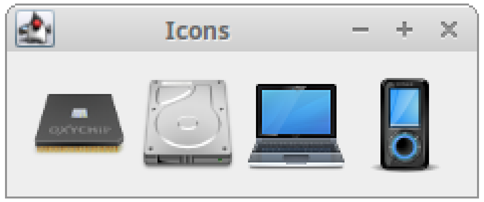


Figure: Displaying icons

JTextField

JTextField is a text component that allows editing of a single line of non-formatted text.

JTextFieldEx.java

```
package com.zetcode;

import javax.swing.GroupLayout;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;
import javax.swing.text.BadLocationException;
import java.awt.EventQueue;
import java.util.logging.Level;
import java.util.logging.Logger;

public class JTextFieldEx extends JFrame {
    private JLabel lbl;

    public JTextFieldEx() {

        initUI();
    }

    private void initUI() {

        var field = new JTextField(15);
        lbl = new JLabel();

        field.getDocument().addDocumentListener(new MyDocumentListener());

        createLayout(field, lbl);

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

    private class MyDocumentListener implements DocumentListener {

        private String text;

        @Override
        public void insertUpdate(DocumentEvent e) {
            updateLabel(e);
        }

        @Override
        public void removeUpdate(DocumentEvent e) {
            updateLabel(e);
        }

        @Override
        public void changedUpdate(DocumentEvent e) {
        }

        private void updateLabel(DocumentEvent e) {

            var doc = e.getDocument();
            int len = doc.getLength();

            try {
                text = doc.getText(0, len);
            } catch (BadLocationException ex) {
                Logger.getLogger(JTextFieldEx.class.getName()).log(
                        Level.WARNING, "Bad location", ex);
            }

            lbl.setText(text);

        }
    }

    private void createLayout(JComponent... arg) {

        var pane = getContentPane();
        var gl = new GroupLayout(pane);
        pane.setLayout(gl);

        gl.setAutoCreateContainerGaps(true);
        gl.setAutoCreateGaps(true);

        gl.setHorizontalGroup(gl.createParallelGroup()
                .addComponent(arg[0])
                .addComponent(arg[1])
                .addGap(250)
        );

        gl.setVerticalGroup(gl.createSequentialGroup()
                .addComponent(arg[0], GroupLayout.DEFAULT_SIZE,
                            GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                    .addComponent(arg[1])
                    .addGap(150)
        );

        pack();
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(() -> {

            var ex = new JTextFieldEx();
            ex.setVisible(true);
        });
    }
}
```

In the example, the text entered into the JTextField is shown immediately in a label component.

```
var field = new JTextField(15);
```

New JTextField is created. The parameter is the number of columns. Note that this value does not set the numbers of characters allowed in the field; the value is used to calculate the preferred width of the field.

```
field.getDocument().addDocumentListener(new MyDocumentListener());
```

We add a document listener to the JTextField. The getDocument() method fetches the model associated with the editor. Each Swing component has a model, which manages its state or data.

```
@Override
public void insertUpdate(DocumentEvent e) {
    updateLabel(e);
}

@Override
public void removeUpdate(DocumentEvent e) {
    updateLabel(e);
}
```

The insertUpdate() and removeUpdate() methods call the updateLabel() method which copies the text from the text field and sets it into the label component.

```
@Override
public void changedUpdate(DocumentEvent e) {
}
```

We are not interested in the changeUpdate() method. This event is generated in styled documents only.

```
private void updateLabel(DocumentEvent e) {

    var doc = e.getDocument();
    int len = doc.getLength();

    try {
        text = doc.getText(0, len);
    } catch (BadLocationException ex) {
        Logger.getLogger(JTextFieldEx.class.getName()).log(
                Level.WARNING, "Bad location", ex);
    }

    lbl.setText(text);
}
```

The document event's getDocument() method is used to get the document of the text field being observed. We get the number of characters using the document's getLength() method. The value is used to copy the text with the document's getText() method. Finally, the text is set to the label with the label's setText() method.

```
gl.setVerticalGroup(gl.createSequentialGroup()
        .addComponent(arg[0], GroupLayout.DEFAULT_SIZE,
                GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
        .addComponent(arg[1])
        .addGap(150)
);
```

We do not want JTextField grow vertically; therefore, we set its maximum value to GroupLayout.PREFERRED_SIZE in the vertical direction.

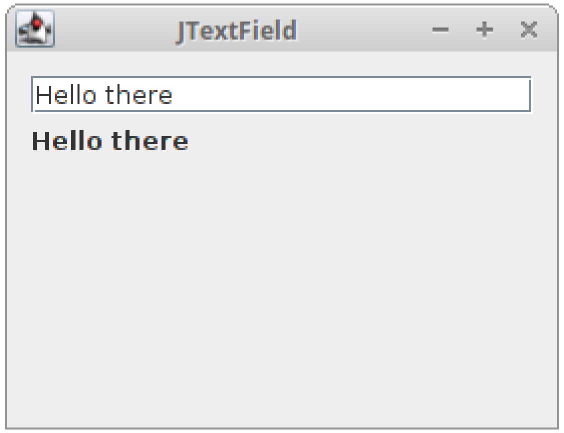


Figure: JTextField

JPasswordField

JPasswordField is a JTextField subclass that does not show the characters that the user types.

PasswordEx.java

```
package com.zetcode;

import javax.swing.AbstractAction;
import javax.swing.GroupLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import java.awt.Component;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.util.Arrays;

import static javax.swing.LayoutStyle.ComponentPlacement.UNRELATED;

public class PasswordEx extends JFrame {

    private JTextField loginField;
    private JPasswordField passField;

    public PasswordEx() {

        initUI();
    }

    private void initUI() {

        var lbl1 = new JLabel("Login");
        var lbl2 = new JLabel("Password");

        loginField = new JTextField(15);
        passField = new JPasswordField(15);

        var submitButton = new JButton("Submit");
        submitButton.addActionListener(new SubmitAction());

        createLayout(lbl1, loginField, lbl2, passField, submitButton);

        setTitle("Login");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    private class SubmitAction extends AbstractAction {

        @Override
        public void actionPerformed(ActionEvent e) {

            doSubmitAction();
        }

        private void doSubmitAction() {

            var login = loginField.getText();
            var passwd = passField.getPassword();

            if (!login.isEmpty() && passwd.length != 0) {

                System.out.format("User %s entered %s password%n",
                        login, String.valueOf(passwd));
            }

            Arrays.fill(passwd, '0');
        }
    }

    private void createLayout(Component... arg) {

        var pane = getContentPane();
        var gl = new GroupLayout(pane);
        pane.setLayout(gl);
        gl.setAutoCreateGaps(true);
        gl.setAutoCreateContainerGaps(true);

        gl.setHorizontalGroup(gl.createSequentialGroup()
                .addGap(50)
                .addGroup(gl.createParallelGroup()
                        .addComponent(arg[0])
                        .addComponent(arg[1])
                        .addComponent(arg[2])
                        .addComponent(arg[3])
                        .addComponent(arg[4]))
                .addGap(50)
        );

        gl.setVerticalGroup(gl.createSequentialGroup()
                .addGap(50)
                .addGroup(gl.createSequentialGroup()
                        .addComponent(arg[0])
                        .addComponent(arg[1], GroupLayout.DEFAULT_SIZE,
                                GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                        .addComponent(arg[2])
                        .addComponent(arg[3], GroupLayout.DEFAULT_SIZE,
                                GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(UNRELATED)
                        .addComponent(arg[4]))
                .addGap(50)
        );

        pack();
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(() -> {

            var ex = new PasswordEx();
```

```
                ex.setVisible(true);
            });
        }
    }
```

The example has a text field, a password field, and a button. The button prints the data entered by the user.

```
passField = new JPasswordField (15);
```

An instance of the JPasswordField is created.

```
var passwd = passField.getPassword();
```

As a security precaution, a password field stores its value as an array of characters, rather than as a string. The array of characters is returned by the getPassword() method. The older getText() method has been deprecated.

```
Arrays.fill(passwd , '0');
```

Once we have finished processing the password, it is recommended to set the array's elements to zero.

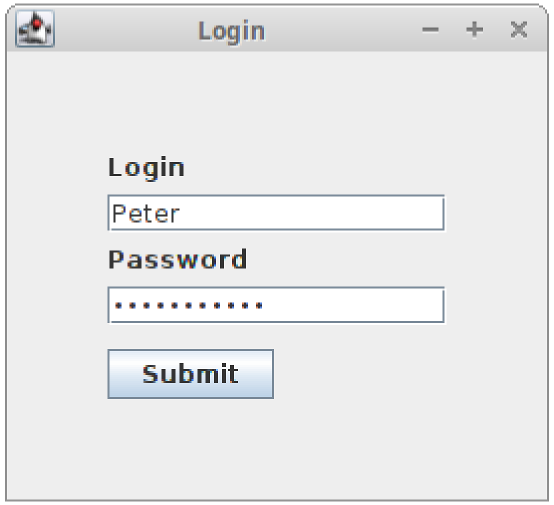


Figure: JPasswordField

In this part of the Java Swing tutorial, we have covered basic Swing components, including JButton, JLabel, JTextField, and JPasswordField.