무료 전자 책

# 배우기

# swing

Free unaffiliated eBook created from
**Stack Overflow contributors.**

#swing

# 1:

Swing JavaFX . JavaFX . : Swing Java . JavaFX Swing .

Swing Event Dispatch Thread . GUI `invokeLater` . Java :

. Swing . Swing "thread safe" . . Swing , API 「thread 」 . thread . Swing . .

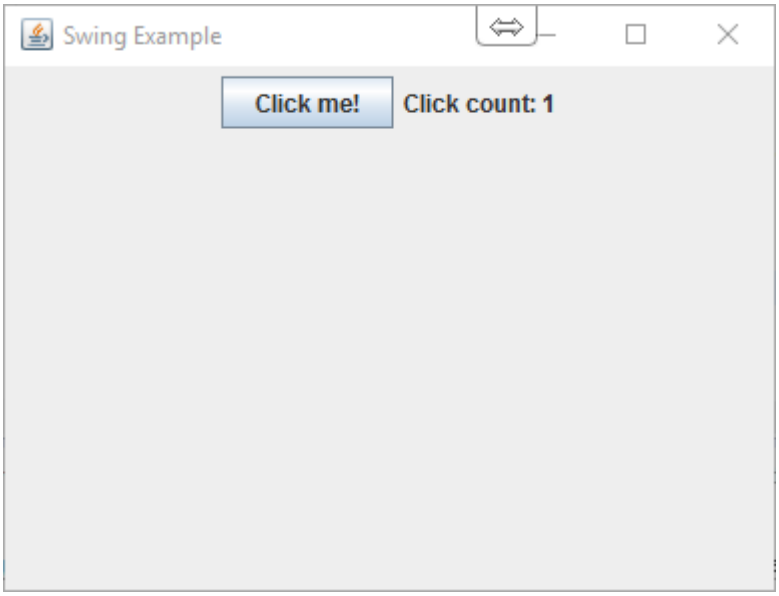
`setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE)` . JVM .

## Examples

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;
import javax.swing.WindowConstants;

/**
 * A very simple Swing example.
 */
public class SwingExample {
    /**
     * The number of times the user has clicked the button.
     */
    private long clickCount;

    /**
     * The main method: starting point of this application.
     *
     * @param arguments the unused command-line arguments.
     */
    public static void main(final String[] arguments) {
        new SwingExample().run();
    }

    /**
     * Schedule a job for the event-dispatching thread: create and show this
     * application's GUI.
     */
    private void run() {
        SwingUtilities.invokeLater(this::createAndShowGui);
    }

    /**
     * Create the simple GUI for this application and make it visible.
     */
    private void createAndShowGui() {
        // Create the frame and make sure the application exits when the user closes
        // the frame.
        JFrame mainFrame = new JFrame("Counter");
        mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
```

```
        // Add a simple button and label.
        JPanel panel = new JPanel();
        JButton button = new JButton("Click me!");
        JLabel label = new JLabel("Click count: " + clickCount);
        panel.add(button);
        panel.add(label);
        mainFrame.getContentPane().add(panel);

        // Add an action listener to the button to increment the count displayed by
        // the label.
        button.addActionListener(actionEvent -> {
            clickCount++;
            label.setText("Click count: " + clickCount);
        });

        // Size the frame.
        mainFrame.setBounds(80, 60, 400, 300);
        //Center on screen
        mainFrame.setLocationRelativeTo(null);
        //Display frame
        mainFrame.setVisible(true);
    }
}
```

"Click me!"  1 .




## "Hello World!"

```
import javax.swing.JFrame;
import javax.swing.SwingUtilities;
import javax.swing.WindowConstants;

public class Main {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            JFrame frame = new JFrame("Hello World!");
            frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
            frame.setSize(200, 100);
            frame.setVisible(true);
        });
```

```
    }
}
```

`main` :
`SwingUtilities.invokeLater` code `() -> {...}` . EDT . Event Dispatch Threads . Swing .

:
`frame` `JFrame` `new JFrame("Hello World!")` . "Hello World!" . . `frame` `EXIT_ON_CLOSE` . . `setSize` `frame` 200 100 . . `setVisible(true)` `frame` .

## "Hello World!"

`java.lang.Runnable` "Hello World!" 1.2 Java :

```
import javax.swing.JFrame;
import javax.swing.SwingUtilities;
import javax.swing.WindowConstants;

public class Main {
    public static void main(String[] args){
        SwingUtilities.invokeLater(new Runnable(){

            @Override
            public void run(){
                JFrame frame = new JFrame("Hello World!");
                frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
                frame.setSize(200, 100);
                frame.setVisible(true);
            }
        });
    }
}
```

: https://riptutorial.com/ko/swing/topic/2191/-

# 2: GridBag

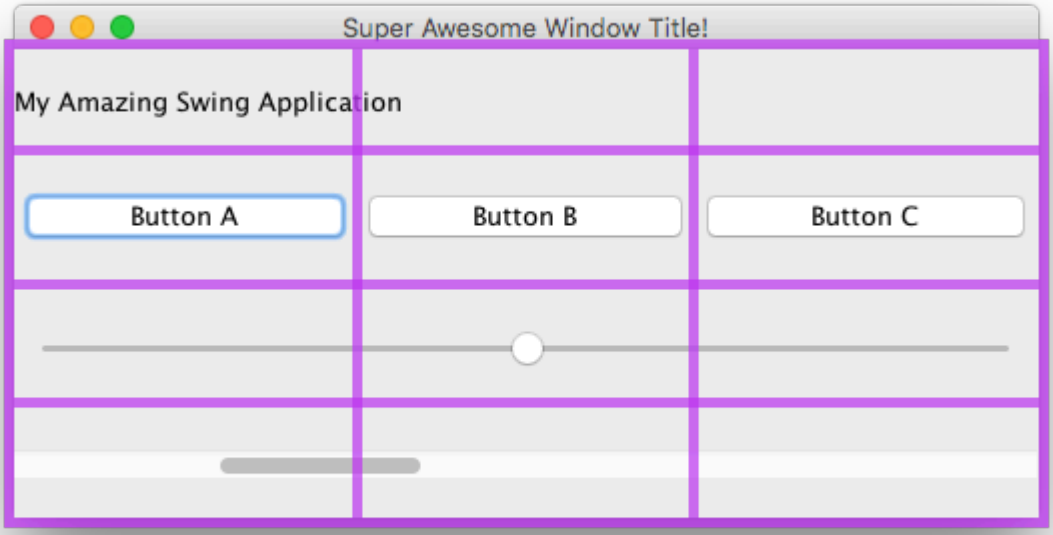
- frame.setLayout ( GridBagLayout ()); //  GridBagLayout .
- pane.setLayout ( GridBagLayout ()); // Panel  GridBagLayout .
- JPanel pane =  JPanel ( GridBagLayout ()); // Panel  GridBagLayout .
- GridBagConstraints c = new GridBagConstraints () // GridBagConstraint .

## Examples

### GridBagLayout  ?

. GridBagLayout  ,  ,  .

.  .




GridBagLayout  6  .

| | | |
|---|---|---|
| JLabel : "  " | 0, 0 | 3, 1 |
| JButton : " A" | 0, 1 | 1, 1 |
| JButton : " B" | 1, 1 | 1, 1 |
| JButton : " C" | 2, 1 | 1, 1 |
| JSlider | 0, 2 | 3, 1 |
| JScrollBar | 0, 3 | 3, 1 |

0, 0 .x() ,y() .

GridBagLayout JFrame .

```
frame.setLayout(new GridBagLayout());
//OR
pane.setLayout(new GridBagLayout());
//OR
JPanel pane = new JPanel(new GridBagLayout()); //Add the layout when creating your content
pane
```

. .

GridBagConstraints .

```
GridBagConstraints c = new GridBagConstraints();
```

1 . .

```
c.weightx = 1;
c.weighty = 1;
```

.

```
c.fill = GridBagConstraints.HORIZONTAL;
```

.

```
GridBagConstraints.NONE //Don't fill components at all
GridBagConstraints.HORIZONTAL //Fill components horizontally
GridBagConstraints.VERTICAL //Fill components vertically
GridBagConstraints.BOTH //Fill components horizontally and vertically
```

, . 5 x 5 . 0, 0 , 1, 1 .

```
JButton button = new JButton("Fancy Button!");
c.gridx = 2;
c.gridy = 1;
c.gridwidth = 5;
c.gridheight = 5;
pane.add(buttonA, c);
```

. .

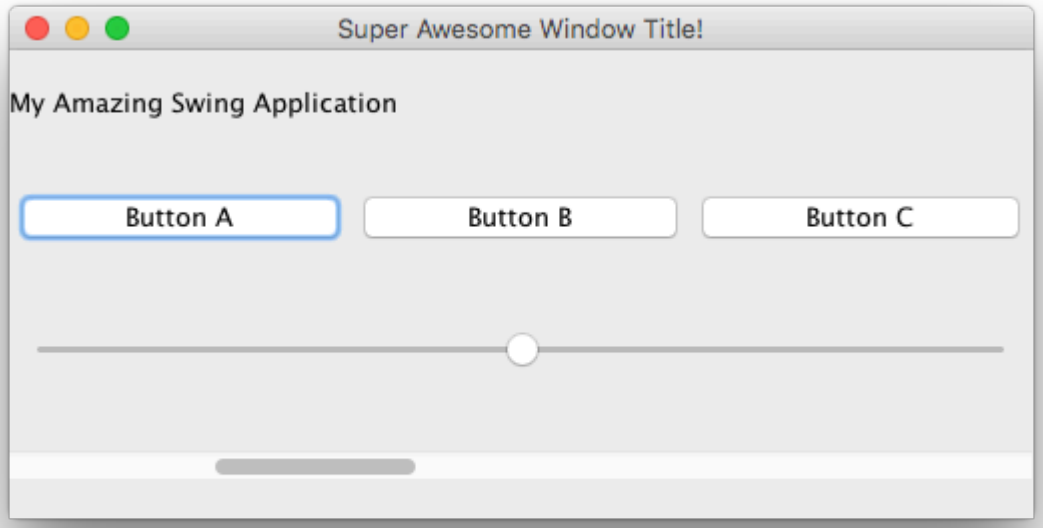
GridBagConstraints . .

---

.

```
JFrame frame = new JFrame("Super Awesome Window Title!"); //Create the JFrame and give it a
title
```

```
frame.setSize(512, 256); //512 x 256px size
frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE); //Quit the application when the
JFrame is closed

JPanel pane = new JPanel(new GridBagLayout()); //Create a pane to house all content, and give
it a GridBagLayout
frame.setContentPane(pane);

GridBagConstraints c = new GridBagConstraints();
c.weightx = 1;
c.weighty = 1;
c.fill = GridBagConstraints.HORIZONTAL;

JLabel headerLabel = new JLabel("My Amazing Swing Application");
c.gridx = 0;
c.gridwidth = 3;
c.gridy = 0;
pane.add(headerLabel, c);

JButton buttonA = new JButton("Button A");
c.gridx = 0;
c.gridwidth = 1;
c.gridy = 1;
pane.add(buttonA, c);

JButton buttonB = new JButton("Button B");
c.gridx = 1;
c.gridwidth = 1;
c.gridy = 1;
pane.add(buttonB, c);

JButton buttonC = new JButton("Button C");
c.gridx = 2;
c.gridwidth = 1;
c.gridy = 1;
pane.add(buttonC, c);

JSlider slider = new JSlider(0, 100);
c.gridx = 0;
c.gridwidth = 3;
c.gridy = 2;
pane.add(slider, c);

JScrollBar scrollBar = new JScrollBar(JScrollBar.HORIZONTAL, 20, 20, 0, 100);
c.gridx = 0;
c.gridwidth = 3;
c.gridy = 3;
pane.add(scrollBar, c);

frame.setVisible(true); //Show the window
```




GridBag : https://riptutorial.com/ko/swing/topic/3698/gridbag-

# 3: JFrame

## Examples

### JFrame

. 10 .

```
private final static long REFRESH_LIST_PERIOD=10 * 60 * 1000; //10 minutes

Timer timer = new Timer(1000, new ActionListener() {

   @Override
   public void actionPerformed(ActionEvent e) {
    if (cnt > 0) {
     cnt = cnt - 1000;
     btnCounter.setText("Remained (" + format.format(new Date(cnt)) + ")");
    } else {
     cnt = REFRESH_LIST_PERIOD;
     //TODO
    }

   }
  });

  timer.start();
```

JFrame : https://riptutorial.com/ko/swing/topic/6745/jframe-

# 4: JList

## Examples

### JList

JList

```
JList myList = new JList(items);
```

, JList ListSelectionModel .

```
ListSelectionModel sm = myList.getSelectionModel();
sm.clearSelection();                      // clears the selection
sm.setSelectionInterval(index, index);    // Sets a selection interval
                                          // (single element, in this case)
```

JList .

```
myList.setSelectionIndex(index);            // sets one selected index
                                            // could be used to define the Default Selection

myList.setSelectedIndices(arrayOfIndexes);  // sets all indexes contained in
                                            // the array as selected
```

JList : https://riptutorial.com/ko/swing/topic/5413/jlist

# 5: MigLayout

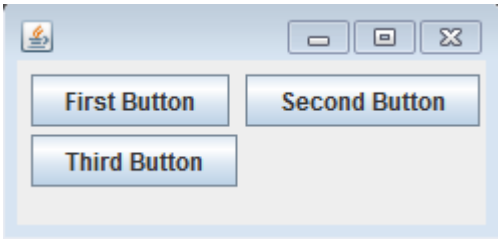
## Examples

3 . (wrap) .

, " ".

```
public class ShowMigLayout {

    // Create the elements
    private final JFrame demo = new JFrame();
    private final JPanel panel = new JPanel();
    private final JButton button1 = new JButton("First Button");
    private final JButton button2 = new JButton("Second Button");
    private final JButton button3 = new JButton("Third Button");

    public static void main(String[] args) {
        ShowMigLayout showMigLayout = new ShowMigLayout();
        SwingUtilities.invokeLater(showMigLayout::createAndShowGui);
    }

    public void createAndShowGui() {
        // Set the position and the size of the frame
        demo.setBounds(400, 400, 250, 120);

        // Tell the panel to use the MigLayout as layout manager
        panel.setLayout(new MigLayout());

        panel.add(button1);
        // Notice the wrapping
        panel.add(button2, "wrap");
        panel.add(button3);

        demo.add(panel);
        demo.setVisible(true);
    }
}
```

:




MigLayout : https://riptutorial.com/ko/swing/topic/2966/miglayout

# 6: MVP

## Examples

### MVP

MVP UI . .

5 :

- - POJO (MVP M)
- - UI (MVP V)
- ViewListener -
- - (MVP P).
- - ""

`count` "".

```
/**
 * A minimal class to maintain some state
 */
public class Model {
    private int count = 0;

    public void addOneToCount() {
        count++;
    }

    public int getCount() {
        return count;
    }
}
```

:

```
/**
 * Provides methods to notify on user interaction
 */
public interface ViewListener {
    public void onButtonClicked();
}
```

UI . UI ( : , ) .

```
/**
 * Provides the UI elements
 */

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
import java.util.ArrayList;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.WindowConstants;

public class View {
    // A list of listeners subscribed to this view
    private final ArrayList<ViewListener> listeners;
    private final JLabel label;

    public View() {
        final JFrame frame = new JFrame();
        frame.setSize(200, 100);
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.setLayout(new GridLayout());

        final JButton button = new JButton("Hello, world!");

        button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(final ActionEvent e) {
                notifyListenersOnButtonClicked();
            }
        });
        frame.add(button);

        label = new JLabel();
        frame.add(label);

        this.listeners = new ArrayList<ViewListener>();

        frame.setVisible(true);
    }

    // Iterate through the list, notifying each listner individualy
    private void notifyListenersOnButtonClicked() {
        for (final ViewListener listener : listeners) {
            listener.onButtonClicked();
        }
    }

    // Subscribe a listener
    public void addListener(final ViewListener listener) {
        listeners.add(listener);
    }

    public void setLabelText(final String text) {
        label.setText(text);
    }
}
```

---

Java8 .

```
        ...
        final Button button = new Button("Hello, world!");
        // In order to do so, our interface must be changed to accept the event parametre
        button.addActionListener((event) -> {
            notifyListeners(ViewListener::onButtonClicked, event);
```

```
            // Example of calling methodThatTakesALong, would be the same as callying:
            // notifyListeners((listener, long)->listener.methodThatTakesALong(long), 10L)
            notifyListeners(ViewListener::methodThatTakesALong, 10L);
        });
        frame.add(button);
        ...

/**
 * Iterates through the subscribed listeneres notifying each listener individually.
 * Note: the {@literal '<T>' in private <T> void} is a Bounded Type Parametre.
 *
 * @param <T>      Any Reference Type (basically a class).
 *
 * @param consumer A method with two parameters and no return,
 *                 the 1st parametre is a ViewListner,
 *                 the 2nd parametre is value of type T.
 *
 * @param data     The value used as parametre for the second argument of the
 *                 method described by the parametre consumer.
 */
private <T> void notifyListeners(final BiConsumer<ViewListener, T> consumer, final T data) {
    // Iterate through the list, notifying each listener, java8 style
    listeners.forEach((listener) -> {

        // Calls the funcion described by the object consumer.
        consumer.accept(listener, data);

        // When this method is called using ViewListener::onButtonClicked
        // the line: consumer.accept(listener,data); can be read as:
        // void accept(ViewListener listener, ActionEvent data) {
        //     listener.onButtonClicked(data);
        // }

    });
}
```

ActionEvent .

```
public interface ViewListener {
    public void onButtonClicked(ActionEvent evt);
    // Example of methodThatTakesALong signature
    public void methodThatTakesALong(long );
}
```

notify-method . . ( :

```
        notifyListeners(ViewListener::methodThatTakesALong, -1L);
```

---

. ( ) . (, ) .

```
/**
 * Responsible to responding to user interaction and updating the view
 */
public class Presenter implements ViewListener {
    private final View view;
    private final Model model;
```

```
    public Presenter(final View view, final Model model) {
        this.view = view;
        view.addListener(this);
        this.model = model;
    }

    @Override
    public void onButtonClicked() {
        // Update the model (ie. the state of the application)
        model.addOneToCount();
        // Update the view
        view.setLabelText(String.valueOf(model.getCount()));
    }
}
```

., . .

```
public class Application {
    public Application() {
        final View view = new View();
        final Model model = new Model();
        new Presenter(view, model);
    }

    public static void main(String... args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new Application();
            }
        });
    }
}
```

MVP : https://riptutorial.com/ko/swing/topic/5154/mvp-

# 7: StyledDocument

- doc.insertString (index, text, attributes); // AttributeSet .

## Examples

### DefaultStyledDocument

```
try {
    StyledDocument doc = new DefaultStyledDocument();
    doc.insertString(0, "This is the beginning text", null);
    doc.insertString(doc.getLength(), "\nInserting new line at end of doc", null);
    MutableAttributeSet attrs = new SimpleAttributeSet();
    StyleConstants.setBold(attrs, true);
    doc.insertString(5, "This is bold text after 'this'", attrs);
} catch (BadLocationException ex) {
    //handle error
}
```

DefaultStyledDocuments . , `StyledDocument` .

### JTextPane StyledDocument

```
try {
        JTextPane pane = new JTextPane();
        StyledDocument doc = new DefaultStyledDocument();
        doc.insertString(0, "Some text", null);
        pane.setDocument(doc); //Technically takes any subclass of Document
    } catch (BadLocationException ex) {
        //handle error
    }
```

JTextPane Swing GUI .

### DefaultStyledDocument

`StyledDocuments` .

```
try {
        //Initialization
        DefaultStyledDocument sourceDoc = new DefaultStyledDocument();
        DefaultStyledDocument destDoc = new DefaultStyledDocument();
        MutableAttributeSet bold = new SimpleAttributeSet();
        StyleConstants.setBold(bold, true);
        MutableAttributeSet italic = new SimpleAttributeSet();
        StyleConstants.setItalic(italic, true);
        sourceDoc.insertString(0, "Some bold text. ", bold);
        sourceDoc.insertString(sourceDoc.getLength(), "Some italic text", italic);

        //This does the actual copying
        String text = sourceDoc.getText(0, sourceDoc.getLength()); //This copies text, but
```

```
loses formatting.
        for (int i = 0; i < text.length(); i++) {
            Element e = destDoc.getCharacterElement(i); //A Elment describes a particular part
of a document, in this case a character
            AttributeSet attr = e.getAttributes(); //Gets the attributes for the character
            destDoc.insertString(destDoc.getLength(), text.substring(i, i+1), attr); //Gets
the single character and sets its attributes from the element
        }
    } catch (BadLocationException ex) {
        //handle error
    }
```

## DefaultStyledDocument RTF

AdvancedRTFEditorKit `DefaultStyledDocument` RTF .

```
try {
   DefaultStyledDocument writeDoc = new DefaultStyledDocument();
   writeDoc.insertString(0, "Test string", null);

   AdvancedRTFEditorKit kit = new AdvancedRTFEditorKit();
   //Other writers, such as a FileWriter, may be used
   //OutputStreams are also an option
   Writer writer = new StringWriter();
   //You can write just a portion of the document by modifying the start
   //and end indexes
   kit.write(writer, writeDoc, 0, writeDoc.getLength());
   //This is the RTF String
   String rtfDoc = writer.toString();

   //As above this may be a different kind of reader or an InputStream
   StringReader reader = new StringReader(rtfDoc);
   //AdvancedRTFDocument extends DefaultStyledDocument and can generally
   //be used wherever DefaultStyledDocument can be.
   //However for reading, AdvancedRTFDocument must be used
   DefaultStyledDocument readDoc = new AdvancedRTFDocument();
   //You can insert at different values by changing the "0"
   kit.read(reader, readDoc, 0);
   //readDoc is now the same as writeDoc
} catch (BadLocationException | IOException ex) {
    //Handle exception
    ex.printStackTrace();
}
```

StyledDocument : https://riptutorial.com/ko/swing/topic/5416/styleddocument

# 8: Swing

. .

```
frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE); //Quit the application when the
JFrame is closed
```

## Examples

### (JFrame)

# JFrame

. JFrame .

```
JFrame frame = new JFrame();
```

. JFrame frame.setTitle(String title) .

```
JFrame frame = new JFrame("Super Awesome Window Title!");
//OR
frame.setTitle("Super Awesome Window Title!");
```

. .

```
frame.setSize(512, 256);
```

pack() .

```
frame.pack();
```

setSize() pack() .

# Window Close

. JFrame JFrame .

```
frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
```

.

```
WindowConstants.DISPOSE_ON_CLOSE //Get rid of the window
WindowConstants.EXIT_ON_CLOSE //Quit the application
WindowConstants.DO_NOTHING_ON_CLOSE //Don't even close the window
WindowConstants.HIDE_ON_CLOSE //Hides the window - This is the default action
```

---

. JPanel frame.setContentPane(Component component) .

```
JPanel pane = new JPanel();
frame.setContentPane(pane);
```

---

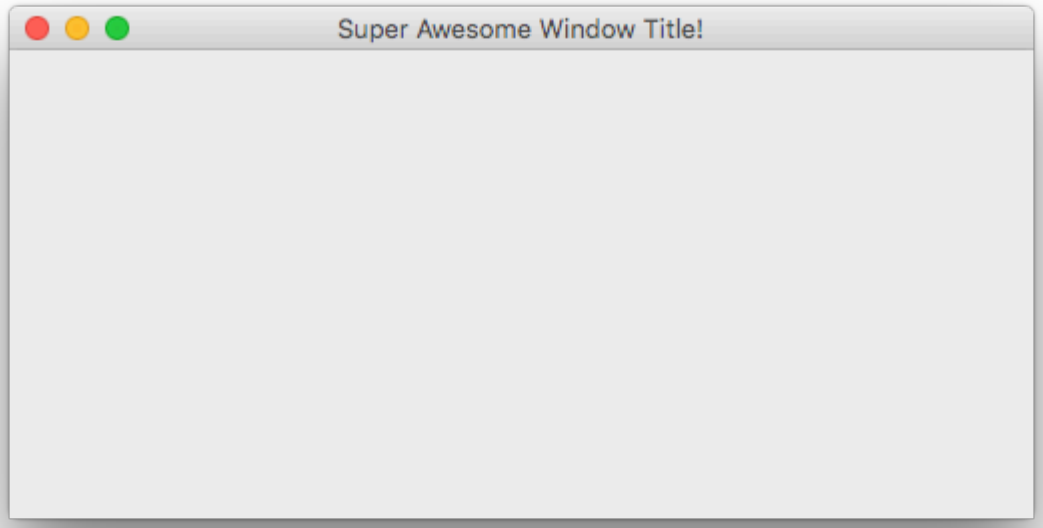
. .

```
frame.setVisible(true);
```

---

.

```
JFrame frame = new JFrame("Super Awesome Window Title!"); //Create the JFrame and give it a
title
frame.setSize(512, 256); //512 x 256px size
frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE); //Quit the application when the
JFrame is closed

JPanel pane = new JPanel(); //Create the content pane
frame.setContentPane(pane); //Set the content pane

frame.setVisible(true); //Show the window
```




.

---

. JFrame . , JButton .

```
JButton button = new JButton();
```

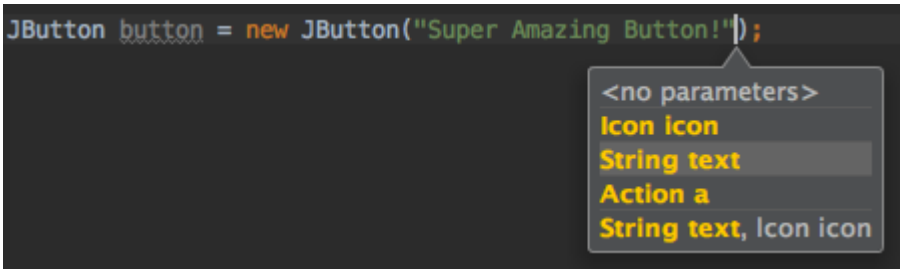
. .

```
JButton button = new JButton("Super Amazing Button!");
```

.

. IDE ( ). .

- IntelliJ IDEA - Windows / Linux : `CTRL + P`
- IntelliJ IDEA - OS X / macOS : `CMD + P`
- Eclipse : `CTRL + SHIFT + Space`
- NetBeans : `CTRL + P`




---

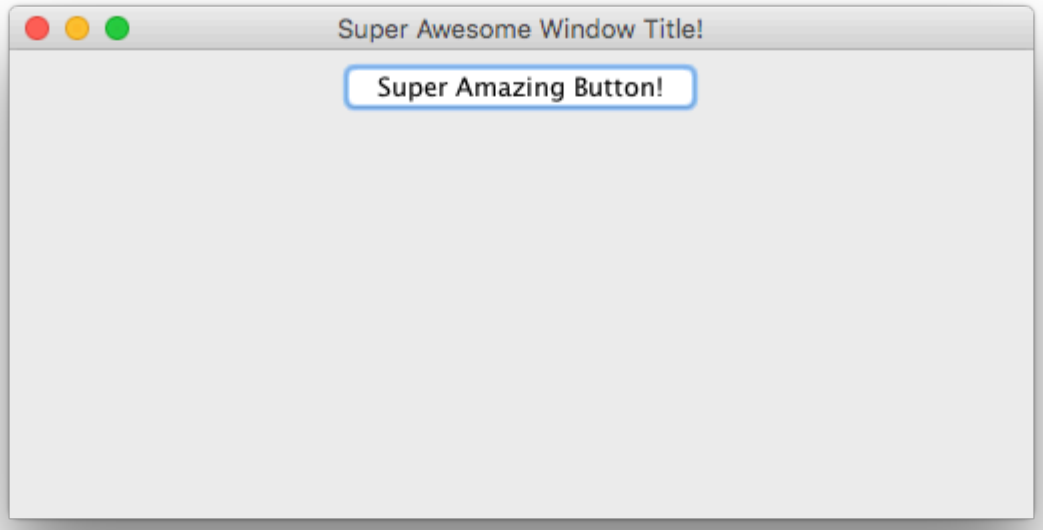
. JFrame .

```
frame.add(button); //Add to your JFrame
//OR
pane.add(button); //Add to your content pane
//OR
myComponent.add(button); //Add to whatever
```

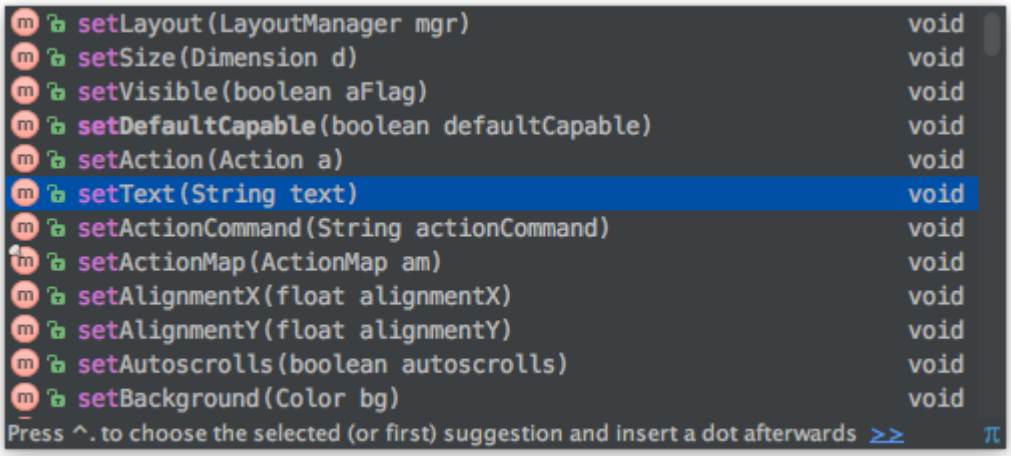
---

, .

```
JFrame frame = new JFrame("Super Awesome Window Title!"); //Create the JFrame and give it a
title
frame.setSize(512, 256); //512 x 256px size
frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE); //Quit the application when the
JFrame is closed

JPanel pane = new JPanel(); //Create the content pane
frame.setContentPane(pane); //Set the content pane

JButton button = new JButton("Super Amazing Button!"); //Create the button
pane.add(button); //Add the button to the content pane

frame.setVisible(true); //Show the window
```




. , `componentName.set` ( IDE ) IDE . IDE `CTRL + Space` .




| | |
|---|---|
| ( minimumSize property ). | `setMinimumSize(Dimension minimumSize)` |
| ( maximumSize property ). | `setMaximumSize(Dimension maximumSize)` |
| ( preferredSize property ). | `setPreferredSize(Dimension preferredSize)` |
| . | `setVisible(boolean aFlag)` |
| . | `setEnabled(boolean enabled)` |
| . | `setFont(Font font)` |
| . | `setToolTipText(String text)` |
| . | `setBackground(Color bg)` |
| ( ) . | `setForeground(Color bg)` |

| | | |
|---|---|---|
| `JLabel` , `JButton` , `JCheckBox` , `JRadioButton` , `JToggleButton` , `JMenu` , `JMenuItem` , `JTextArea` , `JTextField` | . | `setText(String text)` |
| `JProgressBar` , `JScrollBar` , `JSlider` , `JSpinner` | Set . | `setValue(int n)` |
| `JProgressBar` , `JScrollBar` , `JSlider` , `JSpinner` | Set `value` . | `setMinimum(int n)` |
| `JProgressBar` , `JScrollBar` , `JSlider` , `JSpinner` | Set `value` . | `setMaxmimum(int n)` |
| `JCheckBox` , `JToggleBox` | true false ( : ?). | `setSelected(boolean b)` |

| | |
|---|---|
| | JButton |
| | JCheckBox |
| / | JComboBox |
| | JLabel |
| | JList |
| | JMenuBar |
| | JMenu |
| | JMenuItem |
| | JPanel |
| | JProgressBar |
| | JRadioButton |
| | JScrollBar |
| | JSlider |
| / | JSpinner |
| | JTable |
| | JTree |
| / | JTextArea |
| | JTextField |
| | JToolBar |

, ? `ActionListener` , , () .

`ActionListener` .

```
buttonA.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        //Code goes here...
        System.out.println("You clicked the button!");
    }
});
```

Java 8 ...

```
buttonA.addActionListener(e -> {
    //Code
    System.out.println("You clicked the button!");
});
```

# (Java 8 )

```
JFrame frame = new JFrame("Super Awesome Window Title!"); //Create the JFrame and give it a
title
frame.setSize(512, 256); //512 x 256px size
frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE); //Quit the application when the
JFrame is closed

JPanel pane = new JPanel(); //Create a pane to house all content
frame.setContentPane(pane);

JButton button = new JButton("Click me - I know you want to.");
button.addActionListener(e -> {
    //Code goes here
    System.out.println("You clicked me! Ouch.");
});
pane.add(buttonA);

frame.setVisible(true); //Show the window
```

UI . "" .

Swing `LayoutManager` .

LayoutManager .

- 
- GridBag

Swing : https://riptutorial.com/ko/swing/topic/2982/---swing-

# 9:

## Examples

### GridLayout

GridLayout . , , .

- setRows(int rows)
- setColumns(int columns)
- setHgap(int hgap)
- setVgap(int vgap)

.

- GridLayout(int rows, int columns)
- GridLayout(int rows, int columns, int hgap, int vgap)

0 . :

new GridLayout(0, 3)

GridLayout 3 .

---

, GridLayout , , , , , GridLayout .

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.GridLayout;

import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JSpinner;
import javax.swing.SpinnerNumberModel;
import javax.swing.WindowConstants;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

public class GridLayoutExample {

    private GridLayout gridLayout;
    private JPanel gridPanel, contentPane;
    private JSpinner rowsSpinner, columnsSpinner, hgapSpinner, vgapSpinner;

    public void createAndShowGUI() {
        gridLayout = new GridLayout(5, 5, 3, 3);

        gridPanel = new JPanel(gridLayout);
```

```
        final ChangeListener rowsColumnsListener = new ChangeListener() {
            @Override
            public void stateChanged(ChangeEvent e) {
                gridLayout.setRows((int) rowsSpinner.getValue());
                gridLayout.setColumns((int) columnsSpinner.getValue());
                fillGrid();
            }
        };

        final ChangeListener gapListener = new ChangeListener() {
            @Override
            public void stateChanged(ChangeEvent e) {
                gridLayout.setHgap((int) hgapSpinner.getValue());
                gridLayout.setVgap((int) vgapSpinner.getValue());
                gridLayout.layoutContainer(gridPanel);
                contentPane.revalidate();
                contentPane.repaint();
            }
        };

        rowsSpinner = new JSpinner(new SpinnerNumberModel(gridLayout.getRows(), 1, 10, 1));
        rowsSpinner.addChangeListener(rowsColumnsListener);

        columnsSpinner = new JSpinner(new SpinnerNumberModel(gridLayout.getColumns(), 1, 10,
1));
        columnsSpinner.addChangeListener(rowsColumnsListener);

        hgapSpinner = new JSpinner(new SpinnerNumberModel(gridLayout.getHgap(), 0, 50, 1));
        hgapSpinner.addChangeListener(gapListener);

        vgapSpinner = new JSpinner(new SpinnerNumberModel(gridLayout.getVgap(), 0, 50, 1));
        vgapSpinner.addChangeListener(gapListener);

        JPanel actionPanel = new JPanel();
        actionPanel.add(new JLabel("Rows:"));
        actionPanel.add(rowsSpinner);
        actionPanel.add(Box.createHorizontalStrut(10));
        actionPanel.add(new JLabel("Columns:"));
        actionPanel.add(columnsSpinner);
        actionPanel.add(Box.createHorizontalStrut(10));
        actionPanel.add(new JLabel("Horizontal gap:"));
        actionPanel.add(hgapSpinner);
        actionPanel.add(Box.createHorizontalStrut(10));
        actionPanel.add(new JLabel("Vertical gap:"));
        actionPanel.add(vgapSpinner);

        contentPane = new JPanel(new BorderLayout(0, 10));
        contentPane.add(gridPanel);
        contentPane.add(actionPanel, BorderLayout.SOUTH);

        fillGrid();

        JFrame frame = new JFrame("GridLayout Example");
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        frame.setContentPane(contentPane);
        frame.setSize(640, 480);
        frame.setLocationByPlatform(true);
        frame.setVisible(true);
    }

    private void fillGrid() {
```

```
        gridPanel.removeAll();
        for (int row = 0; row < gridLayout.getRows(); row++) {
            for (int col = 0; col < gridLayout.getColumns(); col++) {
                JLabel label = new JLabel("Row: " + row + " Column: " + col);
                label.setHorizontalAlignment(JLabel.CENTER);
                label.setBorder(BorderFactory.createLineBorder(Color.GRAY));
                gridPanel.add(label);
            }
        }
        contentPane.revalidate();
        contentPane.repaint();
    }

    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                new GridLayoutExample().createAndShowGUI();
            }
        });
    }

}
```

: https://riptutorial.com/ko/swing/topic/2780/-

# 10:

## Examples

### UI

Swing (EDT - **E** vent **D** ispatch **T** hread) . UI .

`Thread.sleep` . `javax.swing.Timer` . , `Timer` `JLabel` .

```
  int delay = 2000;//specify the delay for the timer
  Timer timer = new Timer( delay, e -> {
    //The following code will be executed once the delay is reached
    String revertedText = new StringBuilder( label.getText() ).reverse().toString();
    label.setText( revertedText );
  } );
  timer.setRepeats( false );//make sure the timer only runs once
```

`Timer` . UI . 2 .

```
import javax.swing.*;
import java.awt.*;

public final class DelayedExecutionExample {

  public static void main( String[] args ) {
    EventQueue.invokeLater( () -> showUI() );
  }

  private static void showUI(){
    JFrame frame = new JFrame( "Delayed execution example" );

    JLabel label = new JLabel( "Hello world" );
    JButton button = new JButton( "Reverse text with delay" );
    button.addActionListener( event -> {
      button.setEnabled( false );
      //Instead of directly updating the label, we use a timer
      //This allows to introduce a delay, while keeping the EDT free
      int delay = 2000;
      Timer timer = new Timer( delay, e -> {
        String revertedText = new StringBuilder( label.getText() ).reverse().toString();
        label.setText( revertedText );
        button.setEnabled( true );
      } );
      timer.setRepeats( false );//make sure the timer only runs once
      timer.start();
    } );

    frame.add( label, BorderLayout.CENTER );
    frame.add( button, BorderLayout.SOUTH );
    frame.pack();
    frame.setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );
    frame.setVisible( true );
  }
}
```

## UI

Swing EDT (Event Dispatch Thread) . `javax.swing.Timer` EDT `ActionListener` Swing .

`JLabel` 2 .

```
//Use a timer to update the label at a fixed interval
int delay = 2000;
Timer timer = new Timer( delay, e -> {
  String revertedText = new StringBuilder( label.getText() ).reverse().toString();
  label.setText( revertedText );
} );
timer.start();
```

`Timer` . UI 2 .

```
import javax.swing.*;
import java.awt.*;

public final class RepeatTaskFixedIntervalExample {
  public static void main( String[] args ) {
    EventQueue.invokeLater( () -> showUI() );
  }
  private static void showUI(){
    JFrame frame = new JFrame( "Repeated task example" );
    JLabel label = new JLabel( "Hello world" );

    //Use a timer to update the label at a fixed interval
    int delay = 2000;
    Timer timer = new Timer( delay, e -> {
      String revertedText = new StringBuilder( label.getText() ).reverse().toString();
      label.setText( revertedText );
    } );
    timer.start();

    frame.add( label, BorderLayout.CENTER );
    frame.pack();
    frame.setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );
    frame.setVisible( true );
  }
}
```

## UI

`javax.swing.Timer ActionListener Timer ActionListener . , Timer#stop() Timer .`

```
Timer timer = new Timer( delay, new ActionListener() {
  private int counter = 0;
  @Override
  public void actionPerformed( ActionEvent e ) {
    counter++;//keep track of the number of times the Timer executed
    label.setText( counter + "" );
    if ( counter == 5 ){
      ( ( Timer ) e.getSource() ).stop();
    }
  }
```

```
});
```

Timer . 05 UI. Timer .

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public final class RepeatFixedNumberOfTimes {
  public static void main( String[] args ) {
    EventQueue.invokeLater( () -> showUI() );
  }
  private static void showUI(){
    JFrame frame = new JFrame( "Repeated fixed number of times example" );
    JLabel label = new JLabel( "0" );

    int delay = 2000;
    Timer timer = new Timer( delay, new ActionListener() {
      private int counter = 0;
      @Override
      public void actionPerformed( ActionEvent e ) {
        counter++;//keep track of the number of times the Timer executed
        label.setText( counter + "" );
        if ( counter == 5 ){
          //stop the Timer when we reach 5
          ( ( Timer ) e.getSource() ).stop();
        }
      }
    });
    timer.setInitialDelay( delay );
    timer.start();

    frame.add( label, BorderLayout.CENTER );
    frame.pack();
    frame.setDefaultCloseOperation( WindowConstants.EXIT_ON_CLOSE );
    frame.setVisible( true );
  }
}
```

## JFrame

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.SwingUtilities;

public class FrameCreator {

    public static void main(String args[]) {
        //All Swing actions should be run on the Event Dispatch Thread (EDT)
        //Calling SwingUtilities.invokeLater makes sure that happens.
        SwingUtilities.invokeLater(() -> {
            JFrame frame = new JFrame();
            //JFrames will not display without size being set
            frame.setSize(500, 500);

            JLabel label = new JLabel("Hello World");
            frame.add(label);
```

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

}
```

. `add` . Swing Layout Manager .

## JFrame

```
import java.awt.FlowLayout;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.SwingUtilities;

public class CustomFrame extends JFrame {

    private static CustomFrame statFrame;

    public CustomFrame(String labelText) {
        setSize(500, 500);

        //See link below for more info on FlowLayout
        this.setLayout(new FlowLayout());

        JLabel label = new JLabel(labelText);
        add(label);

        //Tells the JFrame what to do when it's closed
        //In this case, we're saying to "Dispose" on remove all resources
        //associated with the frame on close
        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    }

    public void addLabel(String labelText) {
        JLabel label = new JLabel(labelText);
        add(label);
        this.validate();
    }

    public static void main(String args[]) {
        //All Swing actions should be run on the Event Dispatch Thread (EDT)
        //Calling SwingUtilities.invokeLater makes sure that happens.
        SwingUtilities.invokeLater(() -> {
            CustomFrame frame = new CustomFrame("Hello Jungle");
            //This is simply being done so it can be accessed later
            statFrame = frame;
            frame.setVisible(true);
        });

        try {
            Thread.sleep(5000);
        } catch (InterruptedException ex) {
            //Handle error
        }

        SwingUtilities.invokeLater(() -> statFrame.addLabel("Oh, hello world too."));
    }
```

```
}
```

FlowLayout .

```
import java.awt.FlowLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;

public class CustomFrame extends JFrame {

    public CustomFrame(String labelText) {
        setSize(500, 500);

        //See link below for more info on FlowLayout
        this.setLayout(new FlowLayout());

        //Tells the JFrame what to do when it's closed
        //In this case, we're saying to "Dispose" on remove all resources
        //associated with the frame on close
        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        //Add a button
        JButton btn = new JButton("Hello button");
        //And a textbox
        JTextField field = new JTextField("Name");
        field.setSize(150, 50);
        //This next block of code executes whenever the button is clicked.
        btn.addActionListener((evt) -> {
            JLabel helloLbl = new JLabel("Hello " + field.getText());
            add(helloLbl);
            validate();
        });
        add(btn);
        add(field);
    }

    public static void main(String args[]) {
        //All Swing actions should be run on the Event Dispatch Thread (EDT)
        //Calling SwingUtilities.invokeLater makes sure that happens.
        SwingUtilities.invokeLater(() -> {
            CustomFrame frame = new CustomFrame("Hello Jungle");
            //This is simply being done so it can be accessed later
            frame.setVisible(true);
        });
    }

}
```

## "Please wait ..." .

, . JDialog .

```
final JDialog loading = new JDialog(parentComponent);
JPanel p1 = new JPanel(new BorderLayout());
p1.add(new JLabel("Please wait..."), BorderLayout.CENTER);
```

```
loading.setUndecorated(true);
loading.getContentPane().add(p1);
loading.pack();
loading.setLocationRelativeTo(parentComponent);
loading.setDefaultCloseOperation(JDialog.DO_NOTHING_ON_CLOSE);
loading.setModal(true);

SwingWorker<String, Void> worker = new SwingWorker<String, Void>() {
    @Override
    protected String doInBackground() throws InterruptedException
        /** Execute some operation */
    }
    @Override
    protected void done() {
        loading.dispose();
    }
};
worker.execute(); //here the process thread initiates
loading.setVisible(true);
try {
    worker.get(); //here the parent thread waits for completion
} catch (Exception e1) {
    e1.printStackTrace();
}
```

## JButton (Hello World Pt.2)

JFrame ...

.

```
import javax.Swing.*;
```

/

```
import javax.Swing.Jframe;
import javax.Swing.JButton;
```

---

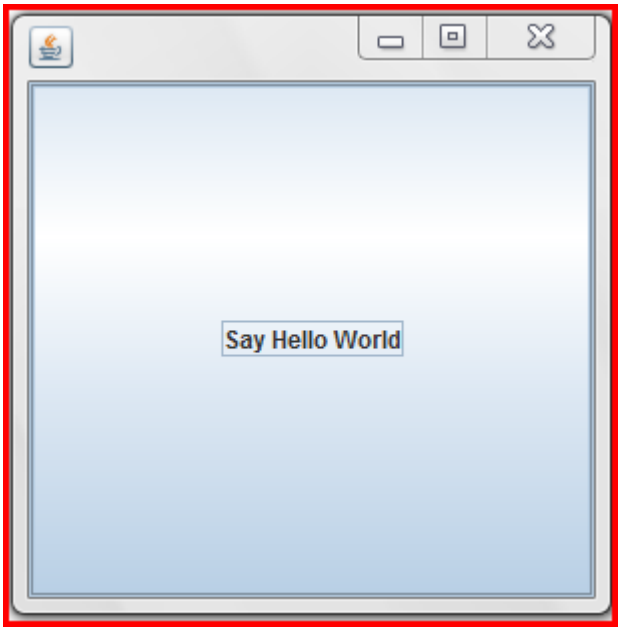
Jbutton ...

```
public static void main(String[] args) {

    JFrame frame = new JFrame(); //creates the frame
    frame.setSize(300, 300);
    frame.setVisible(true);

    ////////////////////////ADDING BUTTON BELOW//////////////////////////////
    JButton B = new JButton("Say Hello World");
    B.addMouseListener(new MouseAdapter() {

        public void mouseReleased(MouseEvent arg0) {
            System.out.println("Hello World");
        }

    });
```

```
    B.setBounds(0, 0,frame.getHeight(), frame.getWidth());
    B.setVisible(true);
    frame.add(B);
    ////////////////////////////////////////////////////////////////////////////
}
```

---

/ .




... "Hello World" .

: https://riptutorial.com/ko/swing/topic/5415/

# 11:

## Examples

```
import static java.awt.BorderLayout.*;
import javax.swing.*;
import java.awt.BorderLayout;

JPanel root = new JPanel(new BorderLayout());

root.add(new JButton("East"), EAST);
root.add(new JButton("West"), WEST);
root.add(new JButton("North"), NORTH);
root.add(new JButton("South"), SOUTH);
root.add(new JButton("Center"), CENTER);

JFrame frame = new JFrame();
frame.setContentPane(root);
frame.pack();
frame.setVisible(true);
```

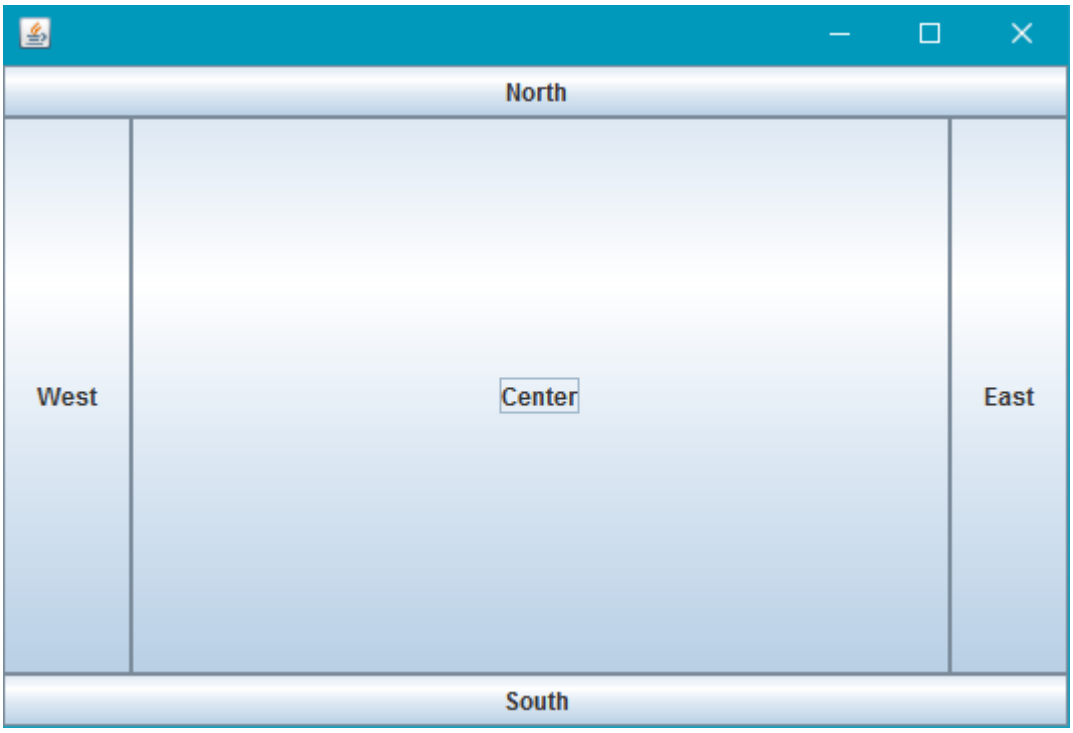
. JPanel .

.

- :
- :
- :

BorderLayout . .

.




```
import javax.swing.*;
import java.awt.FlowLayout;

public class FlowExample {
    public static void main(String[] args){
        SwingUtilities.invokeLater(new Runnable(){

            @Override
            public void run(){
                JPanel panel = new JPanel();
                panel.setLayout(new FlowLayout());

                panel.add(new JButton("One"));
                panel.add(new JButton("Two"));
                panel.add(new JButton("Three"));
                panel.add(new JButton("Four"));
                panel.add(new JButton("Five"));

                JFrame frame = new JFrame();
                frame.setContentPane(Panel);
                frame.pack();
                frame.setVisible(true);
            }
        });
    }
}
```

. . . .

:

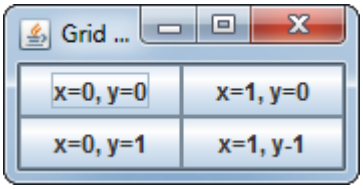

GridLayout .

GridLayout . new GridLayout(3, 2) 3 2 GridLayout .

GridLayout .

```
import javax.swing.*;
import java.awt.GridLayout;

public class Example {
    public static void main(String[] args){
        SwingUtilities.invokeLater(Example::createAndShowJFrame);
    }

    private static void createAndShowJFrame(){
        JFrame jFrame = new JFrame("Grid Layout Example");

        // Create layout and add buttons to show restraints
        JPanel jPanel = new JPanel(new GridLayout(2, 2));
        jPanel.add(new JButton("x=0, y=0"));
        jPanel.add(new JButton("x=1, y=0"));
        jPanel.add(new JButton("x=0, y=1"));
        jPanel.add(new JButton("x=1, y-1"));

        jFrame.setContentPane(jPanel);
        jFrame.pack();
        jFrame.setLocationRelativeTo(null);
        jFrame.setVisible(true);
    }
}
```

JFrame :




: GridLayout

: https://riptutorial.com/ko/swing/topic/5417/-

# 12:  (look and feel)

## Examples

### L & F

L & F  .
L & F  .

```
public class SystemLookAndFeel
{
    public static void main ( final String[] args )
    {
        // L&F installation should be performed within EDT (Event Dispatch Thread)
        // This is important to avoid any UI issues, exceptions or even deadlocks
        SwingUtilities.invokeLater ( new Runnable ()
        {
            @Override
            public void run ()
            {
                // Process of L&F installation might throw multiple exceptions
                // It is always up to you whether to handle or ignore them
                // In most common cases you would never encounter any of those
                try
                {
                    // Installing native L&F as a current application L&F
                    // We do not know what exactly L&F class is, it is provided by the
UIManager
                    UIManager.setLookAndFeel ( UIManager.getSystemLookAndFeelClassName () );
                }
                catch ( final ClassNotFoundException e )
                {
                    // L&F class was not found
                    e.printStackTrace ();
                }
                catch ( final InstantiationException e )
                {
                    // Exception while instantiating L&F class
                    e.printStackTrace ();
                }
                catch ( final IllegalAccessException e )
                {
                    // Class or initializer isn't accessible
                    e.printStackTrace ();
                }
                catch ( final UnsupportedLookAndFeelException e )
                {
                    // L&F is not supported on the current system
                    e.printStackTrace ();
                }

                // Now we can create some natively-looking UI
                // This is just a small sample frame with a single button on it
                final JFrame frame = new JFrame ();
                final JPanel content = new JPanel ( new FlowLayout () );
                content.setBorder ( BorderFactory.createEmptyBorder ( 50, 50, 50, 50 ) );
```

```
                content.add ( new JButton ( "Native-looking button" ) );
                frame.setContentPane ( content );
                frame.setDefaultCloseOperation ( WindowConstants.EXIT_ON_CLOSE );
                frame.pack ();
                frame.setLocationRelativeTo ( null );
                frame.setVisible ( true );
            }
        } );
    }
}
```

JDK L & F (OS -> L & F).

| OS | L & F | L & F |
|---|---|---|
| Solaris, GTK + Linux | GTK + | com.sun.java.swing.plaf.gtk.GTKLookAndFeel |
| Solaris, Linux | | com.sun.java.swing.plaf.motif.MotifLookAndFeel |
| | Windows | com.sun.java.swing.plaf.windows.WindowsLookAndFeel |
| XP | XP | com.sun.java.swing.plaf.windows.WindowsLookAndFeel |
| Windows Vista | Windows Vista | com.sun.java.swing.plaf.windows.WindowsLookAndFeel |
| | | com.apple.laf.AquaLookAndFeel * |
| IBM UNIX | IBM | javax.swing.plaf.synth.SynthLookAndFeel * |
| HP UX | HP | javax.swing.plaf.synth.SynthLookAndFeel * |

* L & F L & F .

## L & F

```
public class CustomLookAndFeel
{
    public static void main ( final String[] args )
    {
        // L&F installation should be performed within EDT (Event Dispatch Thread)
        // This is important to avoid any UI issues, exceptions or even deadlocks
        SwingUtilities.invokeLater ( new Runnable ()
        {
            @Override
            public void run ()
            {
                // Process of L&F installation might throw multiple exceptions
                // It is always up to you whether to handle or ignore them
                // In most common cases you would never encounter any of those
                try
                {
```

```
                    // Installing custom L&F as a current application L&F
                    UIManager.setLookAndFeel ( "javax.swing.plaf.metal.MetalLookAndFeel" );
                }
                catch ( final ClassNotFoundException e )
                {
                    // L&F class was not found
                    e.printStackTrace ();
                }
                catch ( final InstantiationException e )
                {
                    // Exception while instantiating L&F class
                    e.printStackTrace ();
                }
                catch ( final IllegalAccessException e )
                {
                    // Class or initializer isn't accessible
                    e.printStackTrace ();
                }
                catch ( final UnsupportedLookAndFeelException e )
                {
                    // L&F is not supported on the current system
                    e.printStackTrace ();
                }

                // Now we can create some pretty-looking UI
                // This is just a small sample frame with a single button on it
                final JFrame frame = new JFrame ();
                final JPanel content = new JPanel ( new FlowLayout () );
                content.setBorder ( BorderFactory.createEmptyBorder ( 50, 50, 50, 50 ) );
                content.add ( new JButton ( "Metal button" ) );
                frame.setContentPane ( content );
                frame.setDefaultCloseOperation ( WindowConstants.EXIT_ON_CLOSE );
                frame.pack ();
                frame.setLocationRelativeTo ( null );
                frame.setVisible ( true );
            }
        } );
    }
}
```

Swing L & F : Java Look and Feel (L & F)
L & F .

(look and feel) : https://riptutorial.com/ko/swing/topic/3627/--look-and-feel--

# 13: EDT

- SwingWorker <T, V>
- T - SwingWorker doInBackground get
- V - SwingWorker Publish Process
- T doInBackground () - . T.

## Examples

Java main . main . ( short : EDT) . EDT Hello World ! .

EDT . EDT `SwingUtilities.invokeLater` Swing . EDT . EDT EDT . `JFrame` `JFrame` . EDT .

### N JTextArea .

```
import java.awt.EventQueue;
import java.awt.GridLayout;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.util.ArrayList;
import java.util.List;

import javax.swing.JFrame;
import javax.swing.JTextArea;
import javax.swing.SwingWorker;

class PrimeNumbersTask extends SwingWorker<List<Integer>, Integer> {
    private final int numbersToFind;

    private final JTextArea textArea;

    PrimeNumbersTask(JTextArea textArea, int numbersToFind) {
        this.numbersToFind = numbersToFind;
        this.textArea = textArea;
    }

    @Override
    public List<Integer> doInBackground() {
        final List<Integer> result = new ArrayList<>();
        boolean interrupted = false;
        for (int i = 0; !interrupted && (i < numbersToFind); i += 2) {
            interrupted = doIntenseComputing();
            result.add(i);
            publish(i); // sends data to process function
        }
        return result;
    }

    private boolean doIntenseComputing() {
```

```
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            return true;
        }
        return false;
    }

    @Override
    protected void process(List<Integer> chunks) {
        for (int number : chunks) {
            // the process method will be called on the EDT
            // thus UI elementes may be updated in here
            textArea.append(number + "\n");
        }
    }
}

public class SwingWorkerExample extends JFrame {
    private JTextArea textArea;

    public SwingWorkerExample() {
        super("Java SwingWorker Example");
        init();
    }

    private void init() {
        setSize(400, 400);
        setLayout(new GridLayout(1, 1));
        textArea = new JTextArea();
        add(textArea);

        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                dispose();
                System.exit(0);
            }
        });
    }

    public static void main(String args[]) throws Exception {

        SwingWorkerExample ui = new SwingWorkerExample();
        EventQueue.invokeLater(() -> {
            ui.setVisible(true);
        });

        int n = 100;
        PrimeNumbersTask task = new PrimeNumbersTask(ui.textArea, n);
        task.execute(); // run async worker which will do long running task on a
        // different thread
        System.out.println(task.get());
    }
}
```

EDT : https://riptutorial.com/ko/swing/topic/3431/--edt

# 14:

# Examples

## Graphics

Graphics Jpanel Java , , . Graphics JComponent paintComponent(Graphics g) .

**Board**

```
import java.awt.*;
import javax.swing.*;

public class Board extends JPanel{

    public Board() {
        setBackground(Color.WHITE);
    }

    @override
    public Dimension getPreferredSize() {
        return new Dimension(400, 400);
    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        // draws a line diagonally across the screen
        g.drawLine(0, 0, 400, 400);
        // draws a rectangle around "hello there!"
        g.drawRect(140, 180, 115, 25);
    }
}
```

**DrawingCanvas**

```
import javax.swing.*;

public class DrawingCanvas extends JFrame {

    public DrawingCanvas() {

        Board board = new Board();

        add(board); // adds the Board to our JFrame
        pack(); // sets JFrame dimension to contain subcomponents

        setResizable(false);
        setTitle("Graphics Test");
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        setLocationRelativeTo(null); // centers window on screen
    }
```

```
    public static void main(String[] args) {
        DrawingCanvas canvas = new DrawingCanvas();
        canvas.setVisible(true);
    }
}
```

---

setColor  Graphics .

```
g.setColor(Color.BLUE);  // draws a blue square
g.fillRect(10, 110, 100, 100);

g.setColor(Color.RED);  // draws a red circle
g.fillOval(10, 10, 100, 100);

g.setColor(Color.GREEN);  // draws a green triangle
int[] xPoints = {0, 200, 100};
int[] yPoints = {100, 100, 280};
g.fillPolygon(xPoints, yPoints, 3);
```

Graphics drawImage  JComponent .

```
BufferedImage img;
try {
    img = ImageIO.read(new File("stackoverflow.jpg"));
} catch (IOException e) {
    throw new RuntimeException("Could not load image", e);
}
```

```
@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);

    int x = 0;
    int y = 0;

    g.drawImage(img, x, y, this);
}
```

x y  .

## Repaint

MyFrame JFrame main .

```
import javax.swing.JFrame;


public class MyFrame extends JFrame{

    //main method called on startup
    public static void main(String[] args) throws InterruptedException {

        //creates a frame window
```

```
        MyFrame frame = new MyFrame();

        //very basic game loop where the graphics are re-rendered
        while(true){
            frame.getPanel().repaint();

            //The program waits a while before rerendering
            Thread.sleep(12);
        }
    }

    //the MyPanel is the other class and it extends JPanel
    private MyPanel panel;

    //constructor that sets some basic staring values
    public MyFrame(){
        this.setSize(500, 500);
        this.setLocationRelativeTo(null);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //creates the MyPanel with paramaters of x=0 and y=0
        panel = new MyPanel(0,0);
        //adds the panel (which is a JComponent because it extends JPanel)
        //into the frame
        this.add(panel);
        //shows the frame window
        this.setVisible(true);
    }

    //gets the panel
    public MyPanel getPanel(){
        return panel;
    }

}
```

JPanel paintComponent MyPanel

```
import java.awt.Graphics;
import javax.swing.JPanel;


public class MyPanel extends JPanel{

    //two int variables to store the x and y coordinate
    private int x;
    private int y;

    //construcor of the MyPanel class
    public MyPanel(int x, int y){
        this.x = x;
        this.y = y;
    }

    /*the method that deals with the graphics
        this method is called when the component is first loaded,
         when the component is resized and when the repaint() method is
        called for this component
    */
    @Override
```

```
    public void paintComponent(Graphics g){
        super.paintComponent(g);

        //changes the x and y varible values
        x++;
        y++;

        //draws a rectangle at the x and y values
        g.fillRect(x, y, 50, 50);
    }

}
```

: https://riptutorial.com/ko/swing/topic/5153/

| S. No | | Contributors |
|---|---|---|
| 1 | | Community, Freek de Bruijn, Petter Friberg, Vogel612, XavCo7 |
| 2 | GridBag | CraftedCart, Enwired, mayha, Vogel612 |
| 3 | JFrame | SSD |
| 4 | JList | Andreas Fester, Squidward, user6653173 |
| 5 | MigLayout | hamena314, keuleJ |
| 6 | MVP | avojak, ehzawad, Leonardo Pina, sjngm, Squidward |
| 7 | StyledDocument | DonyorM, Squidward |
| 8 | Swing | CraftedCart, mayha, Michael, Vogel612, Winter |
| 9 | | Lukas Rotter, user6653173 |
| 10 | | DarkV1, DonyorM, elias, Robin, Squidward |
| 11 | | explv, J Atkin, mayha, pietrocalzini, recke96, Squidward, XavCo7 |
| 12 | (look and feel) | Mikle Garin |
| 13 | EDT | dpr, isaias-b, rahul tyagi |
| 14 | | Adel Khial, Ashlyn Campbell, Squidward |