GUI Programming with Swing

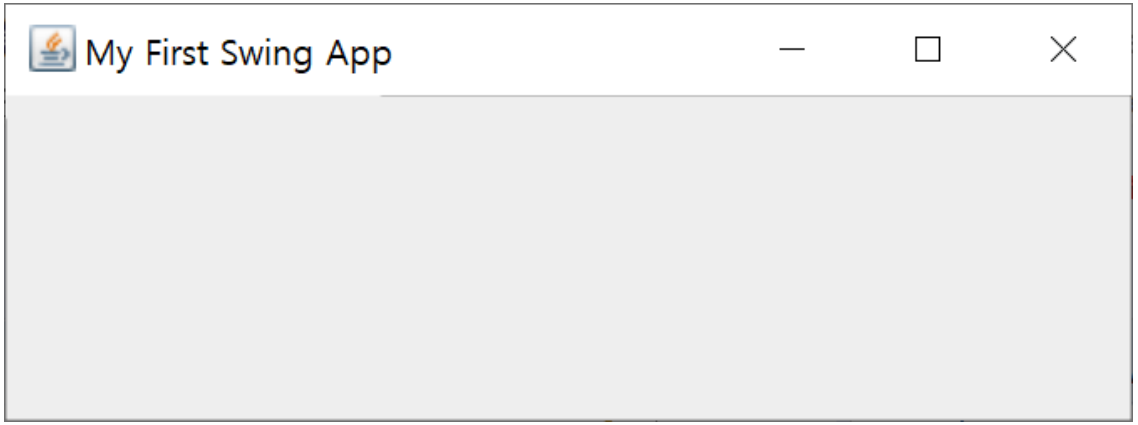
Core Java Volume I – Fundamentals

- Chapter 10. Graphical User Interface Programming
- Chapter 11. User Interface Components with Swing

Swing

- **Swing** is a widget toolkit for Java.
- It is part of Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs.
- Swing was developed to provide a more sophisticated set of GUI components than the earlier AWT(Abstract Window Toolkit).
- It can be compared with MFC and WinForm in MS Windows Platform.

Your First Swing Application




```
import javax.swing.JFrame;

public class HelloSwingWorld {
   public static void main(String[] args) {
       JFrame frame = new JFrame("My First Swing App");
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       frame.setVisible(true);
       frame.setSize(400, 150) ;
   }
}
```

Creating & Positioning a Frame

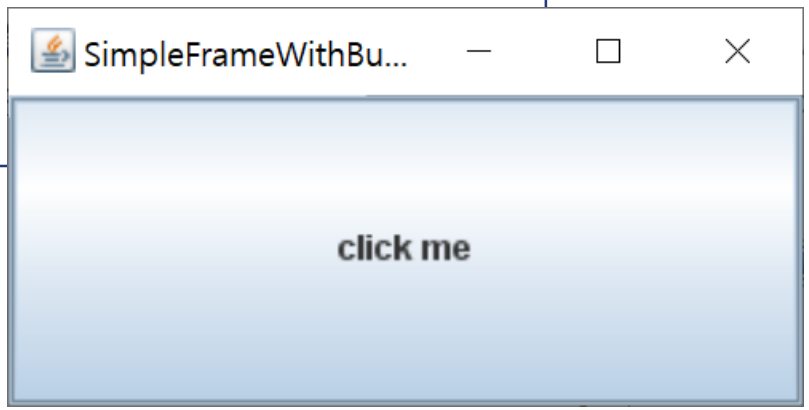
- A frame window so that
  - Its area is one-fourth that of the whole screen
  - It is centered in the middle of the screen

```
import java.awt.*;

import javax.swing.*;

public class CenteredFrameMain {
   public static void main(String[] args) {
      CenteredFrame frame = new CenteredFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
```

```
class CenteredFrame extends JFrame {
   public CenteredFrame() {
       setTitle("CenteredFrame");

      // get screen dimensions
      Toolkit kit = Toolkit.getDefaultToolkit(); // java.awt.Toolkit
      Dimension screenSize = kit.getScreenSize();
      int screenHeight = screenSize.height;
      int screenWidth = screenSize.width;

      // center frame in screen
      setSize(screenWidth / 2, screenHeight / 2);
      setLocation(screenWidth / 4, screenHeight / 4);

      // set frame icon
      Image img = kit.getImage("icon.gif");
      setIconImage(img);
   }
}
```

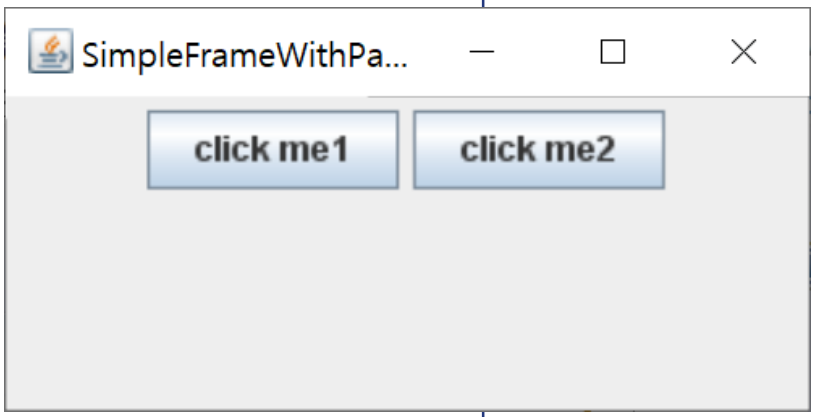
Frame with Button

```
import javax.swing.*;

public class HelloSwingWorldWithButton {
  public static void main(String[] args) {
      JFrame frame = new JFrame("SimpleFrameWithButton");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
      frame.setSize(400, 150) ;

      JButton button = new JButton("click me");
      frame.add(button);
  }
}
```



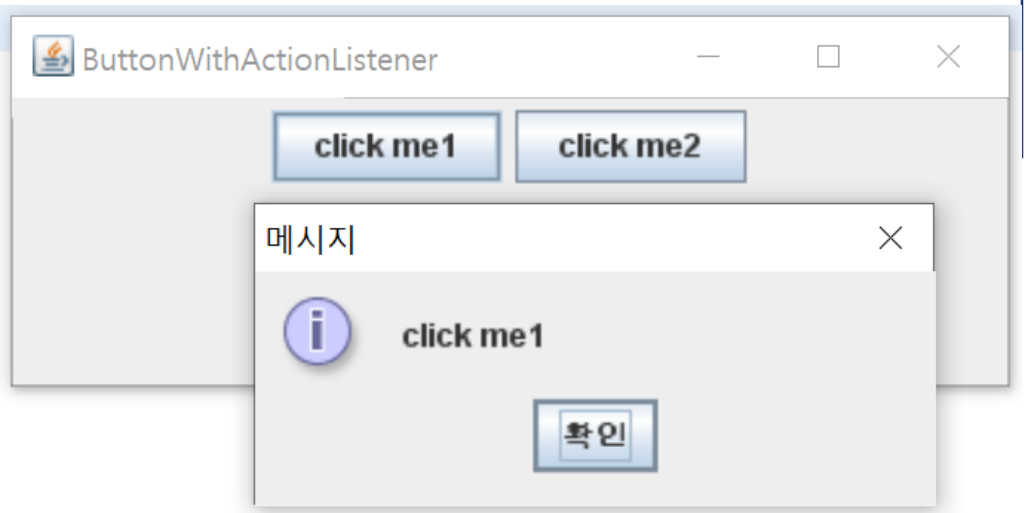
Frame with Panel

```
import javax.swing.*;

public class HelloSwingWorldWithPanel {
  public static void main(String[] args) {
      JFrame frame = new JFrame("SimpleFrameWithPanel");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
      frame.setSize(300, 150) ;

      JPanel panel = new JPanel() ;
      frame.add(panel) ;
      JButton button1 = new JButton("click me1");
      panel.add(button1);
      JButton button2 = new JButton("click me2");
      panel.add(button2);
  }
}
```



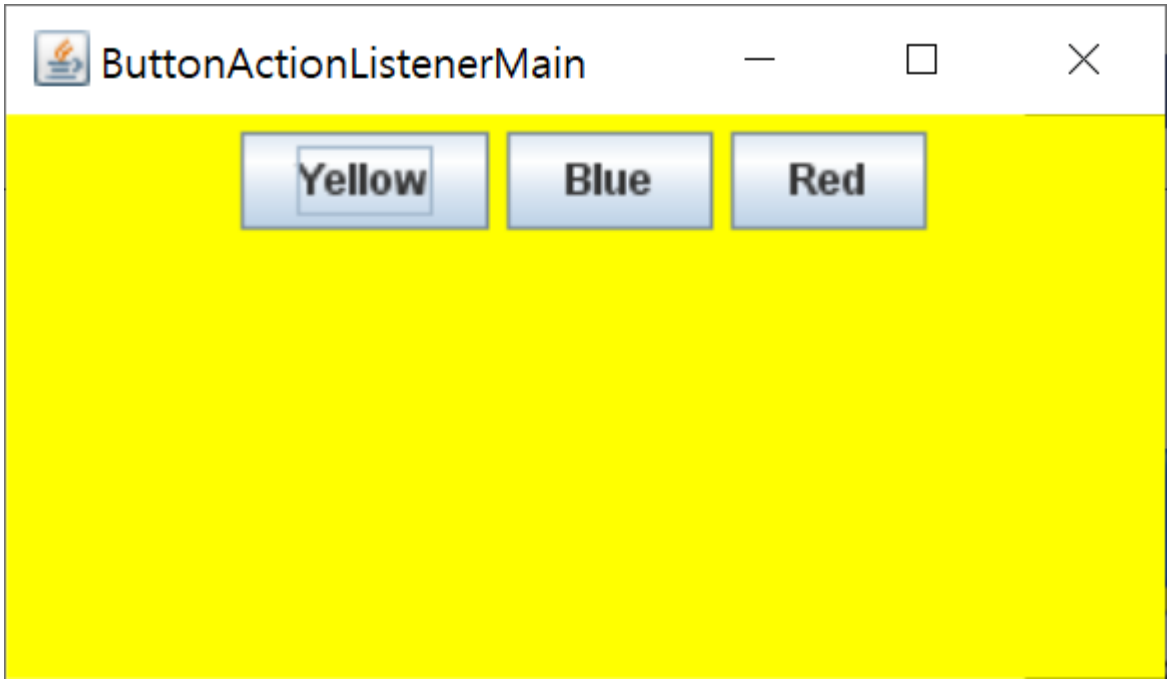
Frame with Button Event Handler

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

import javax.swing.*;

public class FrameWithButtonActionListenerMain {
    public static void main(String[] args) {
        FrameWithButtonActionListener frame =
            new FrameWithButtonActionListener ("ButtonWithActionListener");
    }
}
```



```
class FrameWithButtonActionListener extends JFrame implements ActionListener {
    public FrameWithButtonActionListener (String title) {
        setTitle(title) ;          setVisible(true);    setSize(400, 150) ;

        JPanel panel = new JPanel() ;
        add(panel) ;
        JButton button1 = new JButton("click me1");
        button1.addActionListener(this) ;
        panel.add(button1);
        JButton button2 = new JButton("click me2");
        button2.addActionListener(this) ;
        panel.add(button2);
    }
    @Override
    public void actionPerformed(ActionEvent event) {
        System.out.println(event) ;

        String cmd = event.getActionCommand() ;
        //String cmd = ((JButton) event.getSource()).getText() ;

        JOptionPane.showMessageDialog(null, cmd) ;
    }
}
```

Action Listener Class



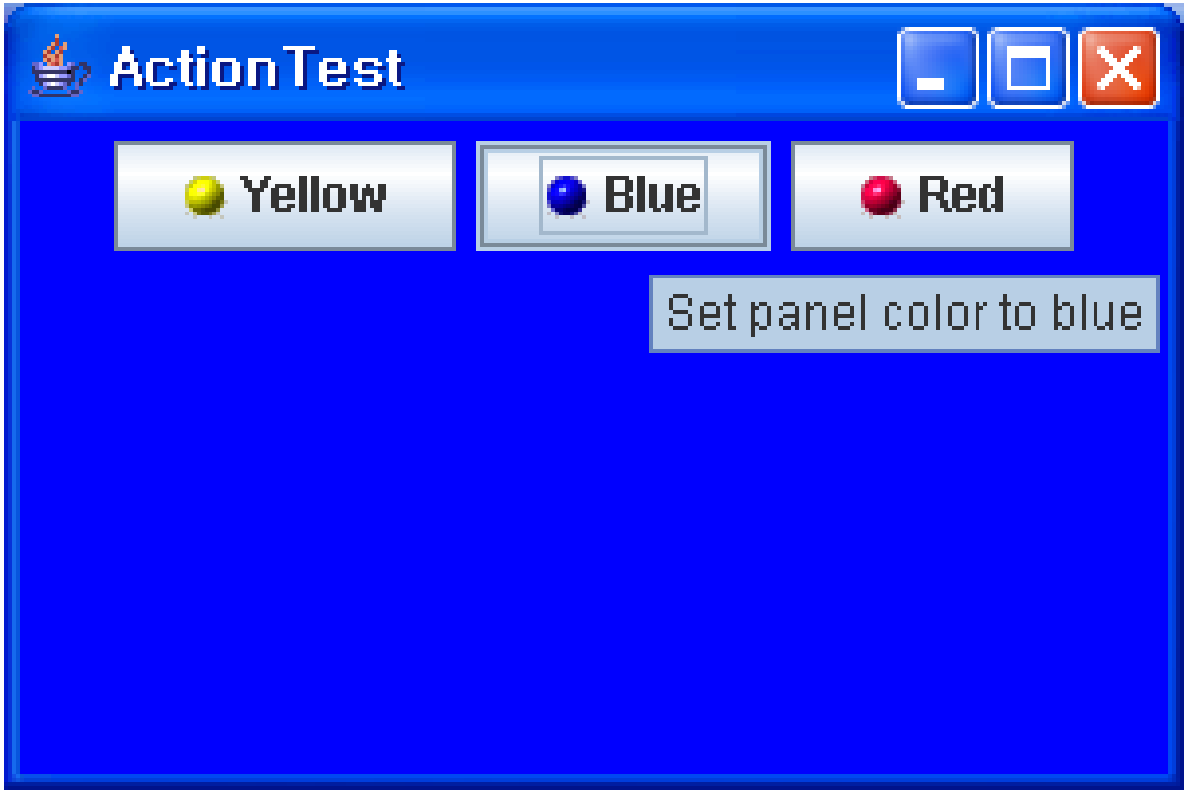
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ButtonActionListenerMain {
  public static void main(String[] args) {
     ButtonActionListenerFrame frame =
        new ButtonActionListenerFrame("ButtonActionListenerMain");
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     frame.setVisible(true);
  }
}

class ButtonActionListenerFrame extends JFrame {
  private static final int DEFAULT_WIDTH = 300;
  private static final int DEFAULT_HEIGHT = 200;

  public ButtonActionListenerFrame(String title) {
     setTitle(title);
     setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

     ButtonPanelWithActionListenerObject panel =
        new ButtonPanelWithActionListenerObject();
     add(panel);
  }
}
```

```
class ButtonPanelWithActionListenerObject extends JPanel {
  public ButtonPanelWithActionListenerObject() {
     // create buttons
     JButton yellowButton = new JButton("Yellow");
     JButton blueButton = new JButton("Blue");
     JButton redButton = new JButton("Red");

     // add buttons to panel
     add(yellowButton); add(blueButton); add(redButton);

     // create button action listener objects
     SetColorListener setYellow = new SetColorListener(Color.YELLOW);
     SetColorListener setBlue = new SetColorListener(Color.BLUE);

     // associate action listener objects with buttons
     yellowButton.addActionListener(setYellow);
     blueButton.addActionListener(setBlue);
     redButton.addActionListener((ActionEvent event) -> setBackground(Color.RED));
  }
  private class SetColorListener implements ActionListener {
     public SetColorListener(Color color) { backgroundColor = color; }
     public void actionPerformed(ActionEvent event) {
        setBackground(backgroundColor);
     }
     private Color backgroundColor;
  }
}
```

```
class ButtonPanelWithActionListener extends JPanel implements ActionListener {
   …
  public void actionPerformed(ActionEvent  event) {
     Object source = event.getSource() ;
     if ( source == yellowButton ) {
          setBackground(Color.YELLOW) ;
         …
  }
  …
  yellowButton.addActionListener(this) ;
  …
```

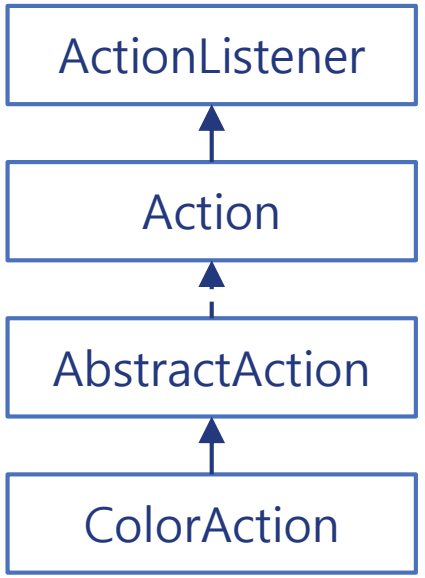
```
class ButtonPanelWithActionListener extends JPanel implements ActionListener {
   …
  public void actionPerformed(ActionEvent  event) {
     String command = event.getActionCommand() ;
     if ( command.equals("Yellow") ) {
          setBackground(Color.YELLOW) ;
         …
  }
  …
  yellowButton.addActionListener(this) ;
  …
```

Actions



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

public class ActionMain {
   public static void main(String[] args) {
      ActionFrame frame = new ActionFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
class ActionFrame extends JFrame {
   private static final int DEFAULT_WIDTH = 300;
   private static final int DEFAULT_HEIGHT = 200;

   public ActionFrame() {
      setTitle("ActionTest");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
      ActionPanel panel = new ActionPanel();
      add(panel);
   }

}
```

```
class ActionPanel extends JPanel {

  class ColorAction extends AbstractAction {
     //AbstractAction implements all methods of interface Action except for actionPerformed
     public ColorAction(String name, Icon icon, Color c) {
        putValue(Action.NAME, name);      // displayed on buttons and menu items
        putValue(Action.SMALL_ICON, icon);
        putValue(Action.SHORT_DESCRIPTION, // for display in a tooltip
                          "Set panel color to " + name.toLowerCase());
        putValue("color", c);
     }
     @Override
     public void actionPerformed(ActionEvent event) {
        setBackground((Color) getValue("color"));
     }
  }

  public ActionPanel() {
     Action yellowAction = // name, icon, color
        new ColorAction("Yellow", new ImageIcon("yellow-ball.gif"), Color.YELLOW);
     Action blueAction =
        new ColorAction("Blue", new ImageIcon("blue-ball.gif"), Color.BLUE);
     Action redAction =
        new ColorAction("Red", new ImageIcon("red-ball.gif"), Color.RED);
```



```
    // add buttons for these actions  to the panel
    add(new JButton(yellowAction));
    add(new JButton(blueAction));
    add(new JButton(redAction));

    // javax.swing.InputMap and javax.swing.ActionMap
    InputMap imap = getInputMap(JComponent.WHEN_FOCUSED);
    imap.put(KeyStroke.getKeyStroke("ctrl Y"), "panel.yellow");
    imap.put(KeyStroke.getKeyStroke("ctrl B"), "panel.blue");
    imap.put(KeyStroke.getKeyStroke("ctrl R"), "panel.red");

    // void put(KeyStroke keyStroke, Object actionMapKey)
    // Adds a binding for keyStroke to actionMapKey.

    // ActionMap provides mappings from Objects (called keys or Action names) to Actions
    ActionMap amap = getActionMap();
    amap.put("panel.yellow", yellowAction);
    amap.put("panel.blue", blueAction);
    amap.put("panel.red", redAction);
    // put(Object key, Action action): Adds a binding for key to action.
}
```

KeyStroke

KeyStroke **getKeyStroke**(int keyCode, int modifiers, boolean onKeyRelease)

The **keycode** constants can be used to specify the key code. For example:
- java.awt.event.KeyEvent.VK_ENTER
- java.awt.event.KeyEvent.VK_TAB
- java.awt.event.KeyEvent.VK_SPACE

The **modifiers** consist of any combination of:
- java.awt.event.InputEvent.SHIFT_MASK (1)
- java.awt.event.InputEvent.CTRL_MASK (2)
- java.awt.event.InputEvent.META_MASK (4)
- java.awt.event.InputEvent.ALT_MASK (8)

"INSERT"	getKeyStroke(KeyEvent.VK_INSERT, 0);
"control DELETE"	getKeyStroke(KeyEvent.VK_DELETE, InputEvent.CTRL_MASK);
"alt shift X"	getKeyStroke(KeyEvent.VK_X, InputEvent.ALT_MASK | InputEvent.SHIFT_MASK);
"alt shift released X"	getKeyStroke(KeyEvent.VK_X, InputEvent.ALT_MASK | InputEvent.SHIFT_MASK, true);
"typed a"	getKeyStroke('a');

Java Tutorial: Creating a GUI with JFC/Swing

https://docs.oracle.com/javase/tutorial/uiswing/index.html

Using Swing Components: Example

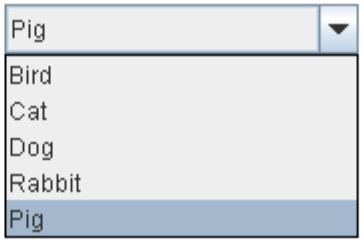
https://docs.oracle.com/javase/tutorial/uiswing/examples/components/index.html

Basic Controls

- Simple components get input from the user

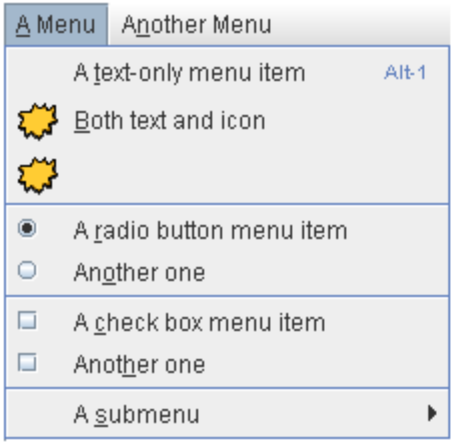
JButton

JCheckBox



JComboBox

JList



JMenu

JRadioButton

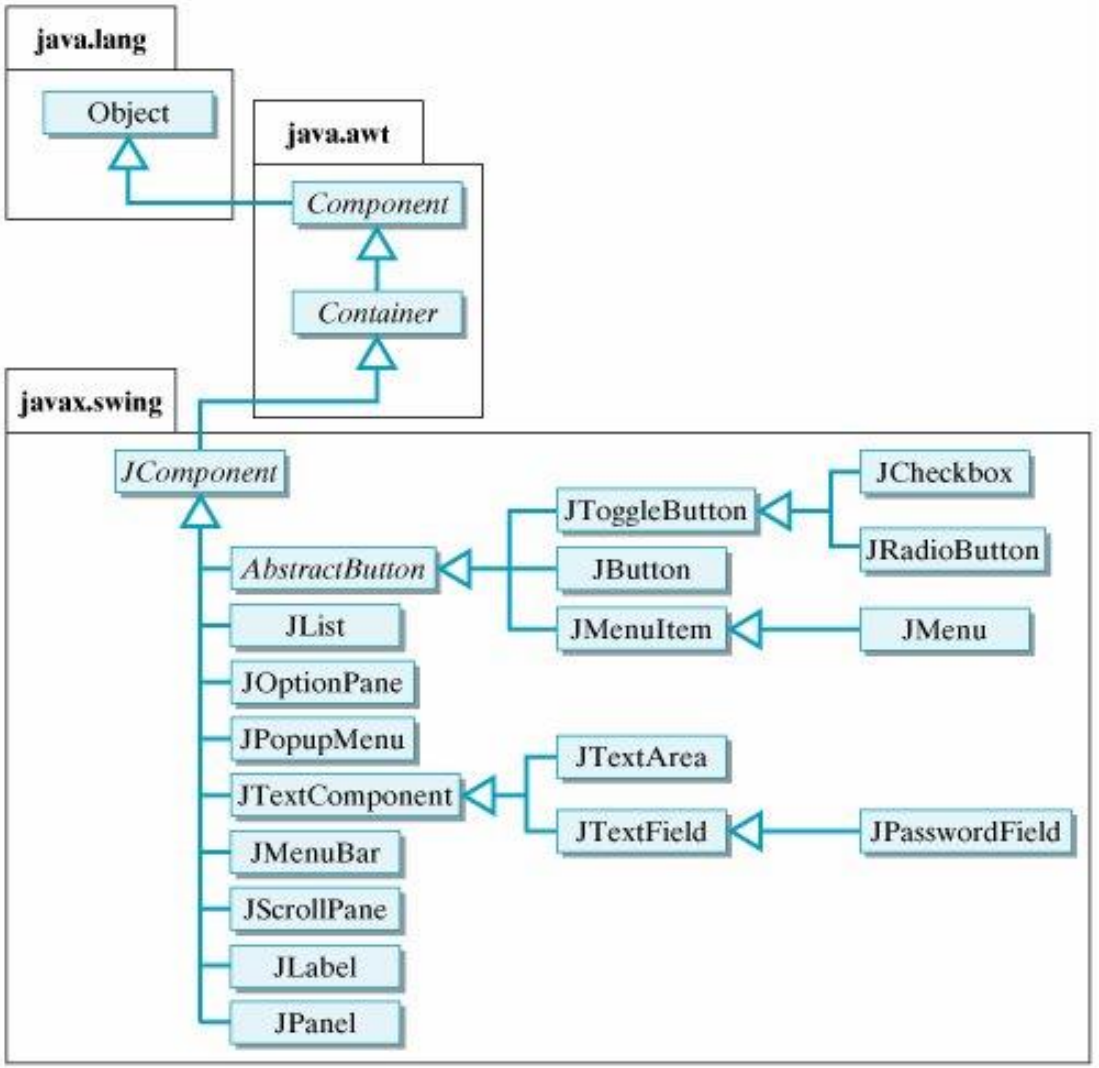
JSlider

JSpinner

JTextField

JPasswordField

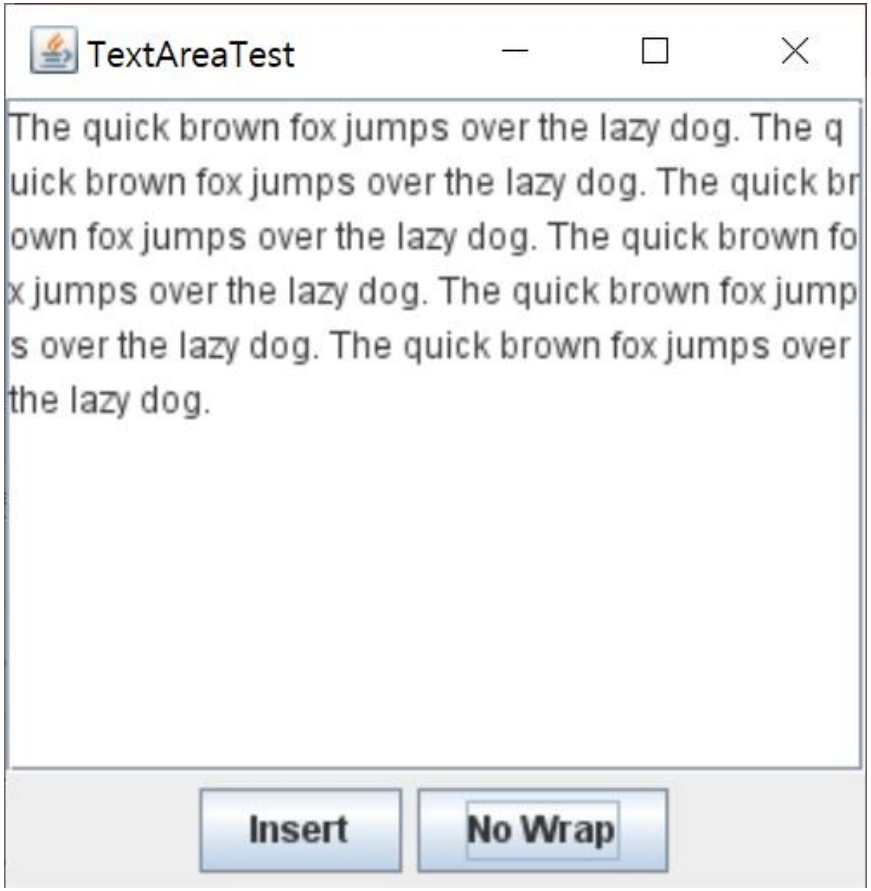
Swing Classes



Contents

- Text Input
  - Text Fields, Formatted Text, Text Area
- Choice Components
  - Checkboxes, Radio Buttons, Borders
  - Combo Boxes, Sliders, JSpinner
- Menus
  - Menu building, icons in menu items, keyboard mnemonics and accelerators, toolbars, tooltips
- Dialog Boxes
  - Option dialogs, file dialogs, color choosers
- Layout Management

Text Area



How to Use Text Areas in Java Tutorial
https://docs.oracle.com/javase/tutorial/uiswing/components/textarea.html

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class TextAreaTest  {
   public static void main(String[] args) {
      TextAreaFrame frame = new TextAreaFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
class TextAreaFrame extends JFrame {
   public TextAreaFrame() {
      setTitle("TextAreaTest");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
      buttonPanel = new JPanel();

      JButton insertButton = new JButton("Insert");
      buttonPanel.add(insertButton);
      insertButton.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent event) {
            textArea.append("The quick brown fox jumps over the lazy dog. ");
          }
       });
```

```
insertButton.addActionListener(
   (ActionEvent event) ->
   textArea.append("The quick brown fox jumps over the lazy dog. ")
);
```

```
    wrapButton = new JButton("Wrap");  buttonPanel.add(wrapButton);
    wrapButton.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent event) {
         final boolean wrap = !textArea.getLineWrap();
         textArea.setLineWrap(wrap);
         wrapButton.setText(wrap ? "No Wrap" : "Wrap");
       }
     });
```

```
wrapButton.addActionListener(
  (ActionEvent event) -> {
    final boolean wrap = !textArea.getLineWrap();
    textArea.setLineWrap(wrap);
    wrapButton.setText(wrap ? "No Wrap" : "Wrap");
  }
);
```

```
    add(buttonPanel, BorderLayout.SOUTH);
    textArea = new JTextArea(8, 40); // JTextArea(int rows, int columns)
    scrollPane = new JScrollPane(textArea);
    add(scrollPane, BorderLayout.CENTER);
  }
  public static final int DEFAULT_WIDTH = 300, DEFAULT_HEIGHT = 300;
  private JTextArea textArea;
  private JScrollPane scrollPane;
  private JPanel buttonPanel;
  private JButton wrapButton;
}
```

Choice Components

- Checkboxes
- Radio Buttons
- Combo Boxes

Checkboxes

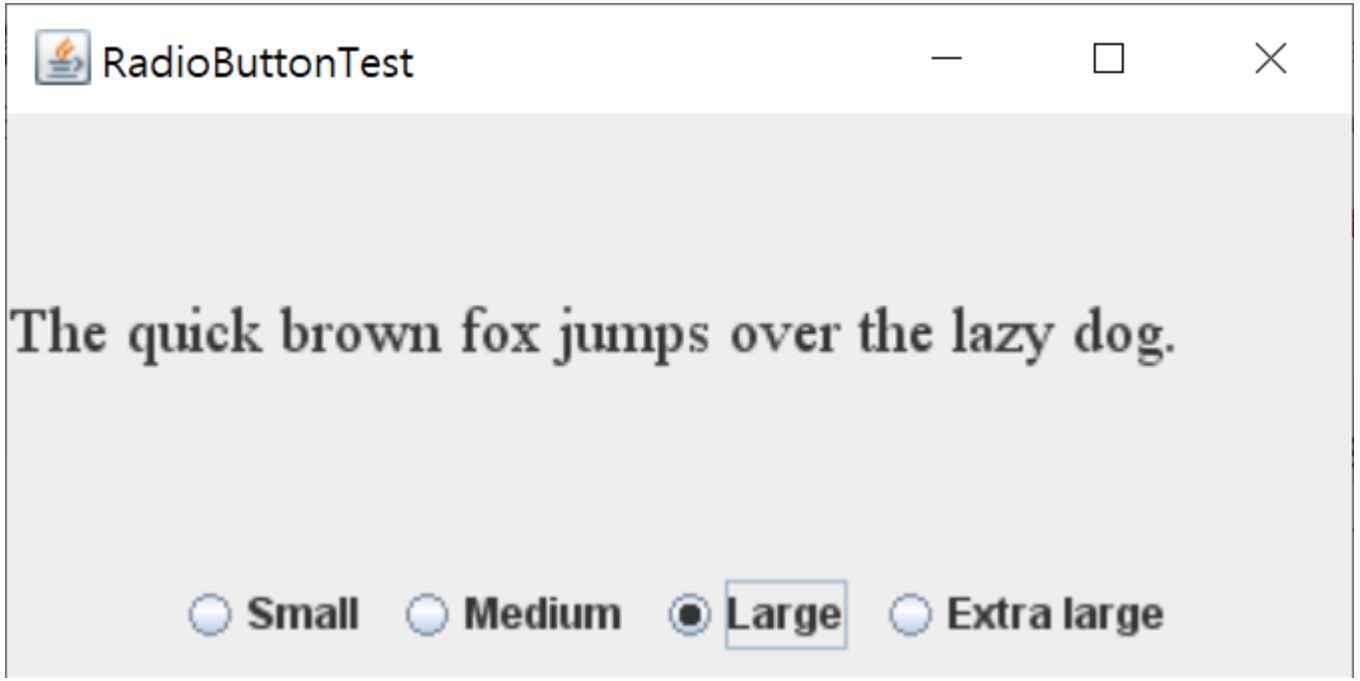

CheckBoxTest
The quick brown fox jumps over the lazy dog.
Bold
Italic

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class CheckBoxTest {
   public static void main(String[] args) {
      CheckBoxFrame frame = new CheckBoxFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
class CheckBoxFrame extends JFrame  {
  public static final int DEFAULT_WIDTH = 300;
   public static final int DEFAULT_HEIGHT = 200;
   private JLabel label;
   private JCheckBox bold, italic;

   private static final int FONTSIZE = 24;
   public CheckBoxFrame() {
      setTitle("CheckBoxTest"); setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
      label = new JLabel("The quick brown fox jumps over the lazy dog.");
      label.setFont(new Font("Serif", Font.PLAIN, FONTSIZE));
      // Font(String name, int style, int size)
      // Creates a new Font from the specified name, style and point size.
      // GraphicsEnvironment.getAvailableFontFamilyNames()

      add(label, BorderLayout.CENTER); // Border Layout is the default layout in JFrame
```

```
    ActionListener listener = new ActionListener() {
        public void actionPerformed(ActionEvent event) {
          int mode = 0;
          if (bold.isSelected()) mode += Font.BOLD;
          if (italic.isSelected()) mode += Font.ITALIC;
          label.setFont(new Font("Serif", mode, FONTSIZE));
        }
      };
    JPanel buttonPanel = new JPanel();

    bold = new JCheckBox("Bold");
    bold.addActionListener(listener);
    buttonPanel.add(bold);

    italic = new JCheckBox("Italic");
    italic.addActionListener(listener);
    buttonPanel.add(italic);

    add(buttonPanel, BorderLayout.SOUTH);
  }
}
```

```
ActionListener listener = (ActionEvent event) -> {
  int mode = 0;
  if (bold.isSelected()) mode += Font.BOLD;
  if (italic.isSelected()) mode += Font.ITALIC;
  label.setFont(new Font("Serif", mode, FONTSIZE));
};
```

Radio Buttons

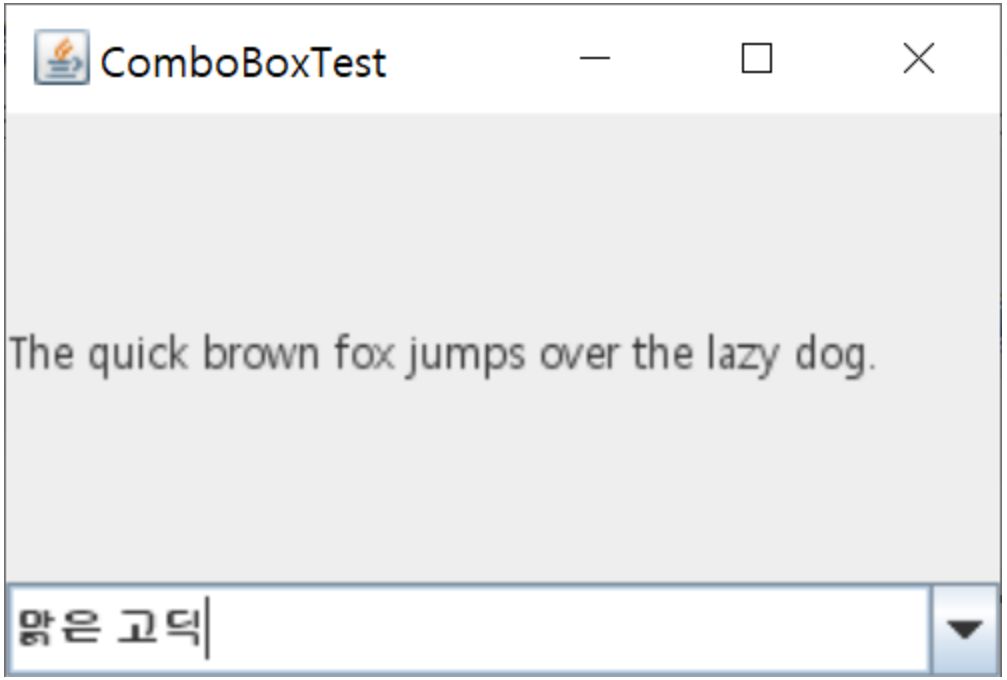


```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
 public class RadioButtonTest {
   public static void main(String[] args) {
      RadioButtonFrame frame = new RadioButtonFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
class RadioButtonFrame extends JFrame {
  public static final int DEFAULT_WIDTH = 400, DEFAULT_HEIGHT = 200;
   private JPanel buttonPanel;
  private JLabel label;
   private static final int DEFAULT_SIZE = 18;

   public RadioButtonFrame() {
      setTitle("RadioButtonTest");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
      label = new JLabel("The quick brown fox jumps over the lazy dog.");
      label.setFont(new Font("Serif", Font.PLAIN, DEFAULT_SIZE));
      add(label, BorderLayout.CENTER);
      buttonPanel = new JPanel();
     add(buttonPanel, BorderLayout.SOUTH);
```

```
    ButtonGroup group = new ButtonGroup();
    addRadioButton(group, "Small", 8);
    addRadioButton(group, "Medium", 12);
    addRadioButton(group, "Large", 18);
    addRadioButton(group, "Extra large", 36);
  }

  public void addRadioButton(ButtonGroup group, String name, final int size) {
    boolean selected = (size == DEFAULT_SIZE);
    JRadioButton button = new JRadioButton(name, selected);
    group.add(button);
    buttonPanel.add(button);

    ActionListener listener = new  ActionListener() {
        public void actionPerformed(ActionEvent event) {
          // size refers to the final parameter of the addRadioButton method
          label.setFont(new Font("Serif", Font.PLAIN, size));
        }
      };
    button.addActionListener(listener);
  }
}
```

```
button.addActionListener(
  (ActionEvent e) ->
  label.setFont(new Font("Serif", Font.PLAIN, size))
);
```

Combo Boxes

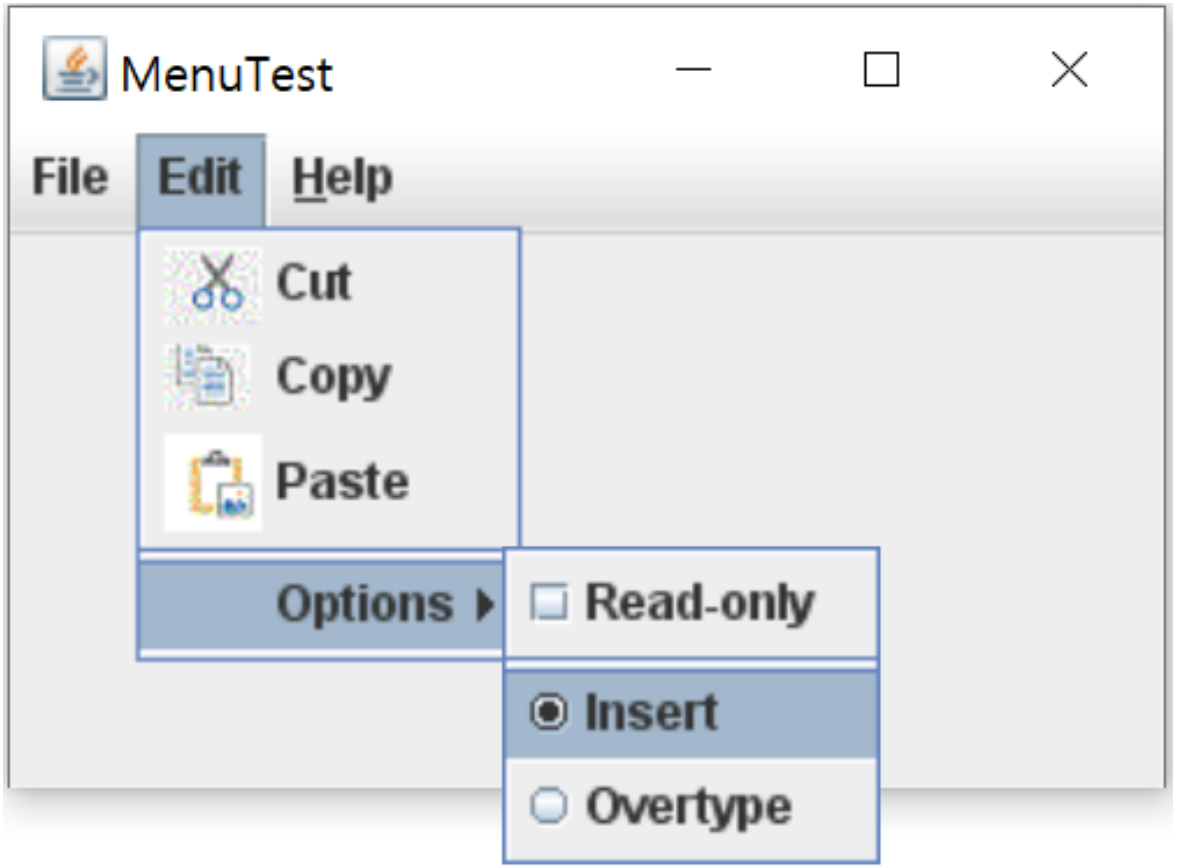

ComboBoxTest
The quick brown fox jumps over the lazy dog.
맑은 고딕

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class ComboBoxTest {
   public static void main(String[] args) {
      ComboBoxFrame frame = new ComboBoxFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
class ComboBoxFrame extends JFrame {
  public static final int DEFAULT_WIDTH = 300;
  public static final int DEFAULT_HEIGHT = 200;

  private JLabel label;
  private static final int DEFAULT_SIZE = 12;

  public ComboBoxFrame() {
     setTitle("ComboBoxTest"); setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

     label = new JLabel("The quick brown fox jumps over the lazy dog.");
     label.setFont(new Font("Serif", Font.PLAIN, DEFAULT_SIZE));
     add(label, BorderLayout.CENTER);
```

```
    JComboBox<String> fontNameCombo = new JComboBox<>();
    add(fontNameCombo, BorderLayout.SOUTH);

    fontNameCombo.setEditable(true);
    fontNameCombo.addItem("Serif");
    fontNameCombo.addItem("SansSerif");
    fontNameCombo.addItem("Monospaced");
    fontNameCombo.addItem("Dialog");
    fontNameCombo.addItem("DialogInput");
    fontNameCombo.addItem("맑은 고딕");

    fontNameCombo.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent event) {
            label.setFont(new Font(
                                (String) fontNameCombo.getSelectedItem(),
                                Font.PLAIN, DEFAULT_SIZE));
        }
    });
  }
}
```

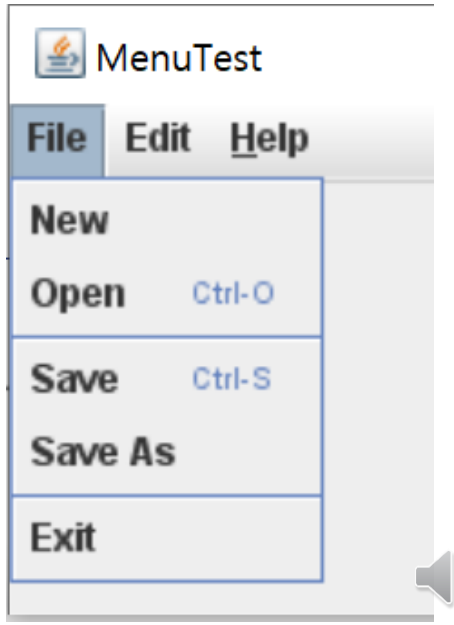
```
faceCombo.addActionListener(
  (ActionEvent event) ->
    label.setFont(new Font(
      (String) fontNameCombo.getSelectedItem(),
      Font.PLAIN,  DEFAULT_SIZE))
);
```

Menus

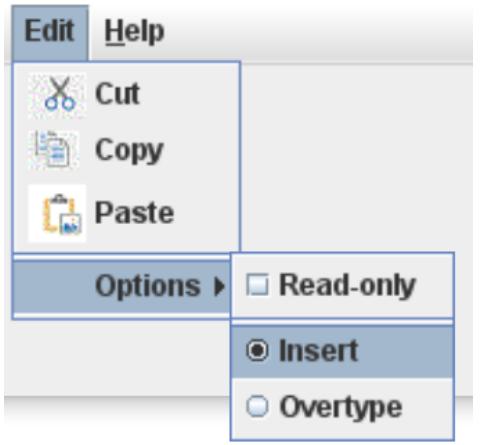

MenuTest
File
Edit
Help
Cut
Copy
Paste
Options
Read-only
Insert
Overtype

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
public class MenuTest {
   public static void main(String[] args) {
      MenuFrame frame = new MenuFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}

class MenuFrame extends JFrame {
  public static final int DEFAULT_WIDTH = 300;
   public static final int DEFAULT_HEIGHT = 200;

   public MenuFrame() {
      setTitle("MenuTest");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
```

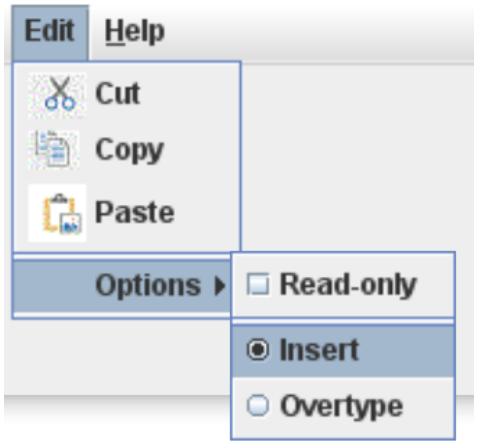
```
JMenu fileMenu = new JMenu("File");

JMenuItem newItem = fileMenu.add(new TestAction("New"));

JMenuItem openItem = fileMenu.add(new TestAction("Open")); // CTRL + O
openItem.setAccelerator(
    KeyStroke.getKeyStroke(KeyEvent.VK_O, InputEvent.CTRL_MASK));

fileMenu.addSeparator();
Action saveAction = new TestAction("Save"); // CTRL + S
JMenuItem saveItem = fileMenu.add(saveAction);
saveItem.setAccelerator(
    KeyStroke.getKeyStroke(KeyEvent.VK_S, InputEvent.CTRL_MASK));

Action saveAsAction = new TestAction("Save As");
JMenuItem saveAsItem = fileMenu.add(saveAsAction);
fileMenu.addSeparator();

fileMenu.add(new  AbstractAction("Exit") {
      public void actionPerformed(ActionEvent event) {
         System.exit(0);
    }
});
```



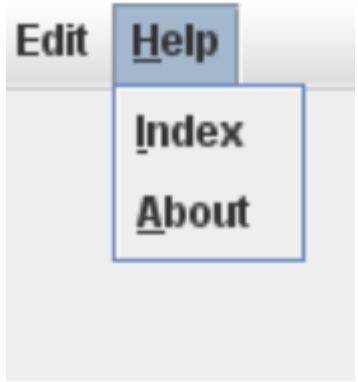
```
// demonstrate icons
Action cutAction = new TestAction("Cut");
cutAction.putValue(Action.SMALL_ICON, new ImageIcon("cut.gif"));
Action copyAction = new TestAction("Copy");
copyAction.putValue(Action.SMALL_ICON, new ImageIcon("copy.gif"));
Action pasteAction = new TestAction("Paste");
pasteAction.putValue(Action.SMALL_ICON, new ImageIcon("paste.gif"));

JMenu editMenu = new JMenu("Edit");
editMenu.add(cutAction);
editMenu.add(copyAction);
editMenu.add(pasteAction);

editMenu.addSeparator();
```

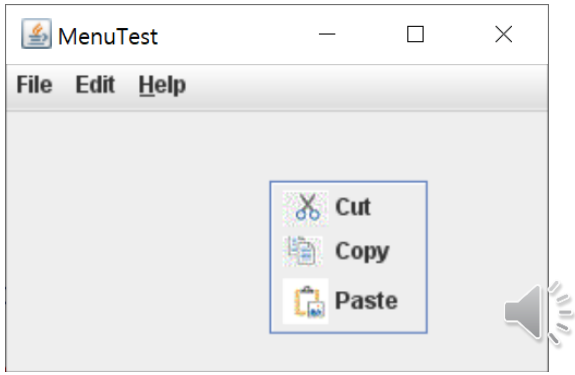


```
// demonstrate nested menus
JMenu optionMenu = new JMenu("Options");
editMenu.add(optionMenu);

JCheckBoxMenuItem readonlyItem = new JCheckBoxMenuItem("Read-only");
readonlyItem.addActionListener(new
   ActionListener() {
      public void actionPerformed(ActionEvent event) {
         boolean saveOk = !readonlyItem.isSelected();
         saveAction.setEnabled(saveOk);
         saveAsAction.setEnabled(saveOk);
      }
   });

ButtonGroup group = new ButtonGroup();
JRadioButtonMenuItem insertItem = new JRadioButtonMenuItem("Insert");
insertItem.setSelected(true);
JRadioButtonMenuItem overtypeItem = new JRadioButtonMenuItem("Overtype");
group.add(insertItem);
group.add(overtypeItem);

optionMenu.add(readonlyItem);
optionMenu.addSeparator();
optionMenu.add(insertItem);
optionMenu.add(overtypeItem);
```

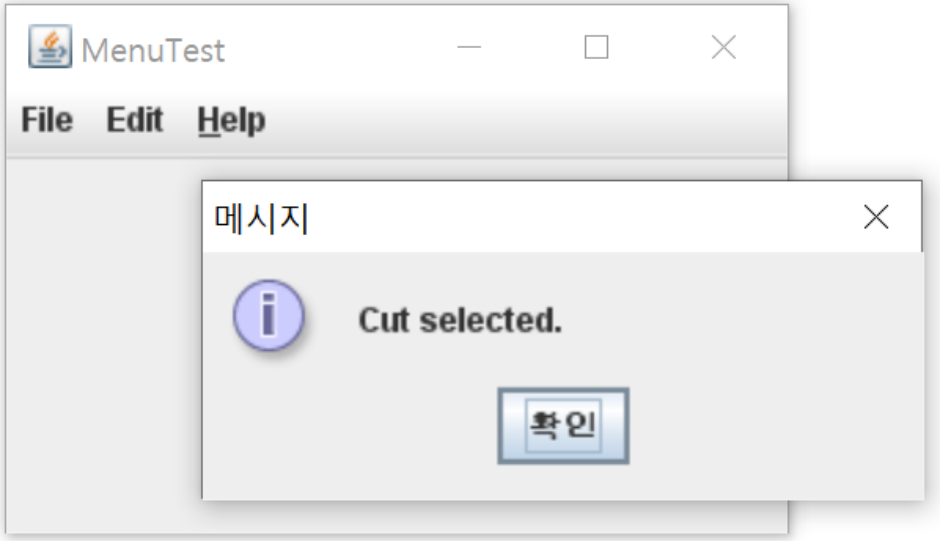


```
    // demonstrate mnemonics
    JMenu helpMenu = new JMenu("Help");
    helpMenu.setMnemonic('H');
    JMenuItem indexItem = new JMenuItem("Index");
    indexItem.setMnemonic('I');
    helpMenu.add(indexItem);
    // you can also add the mnemonic key to an action
    Action aboutAction = new TestAction("About");
    aboutAction.putValue(Action.MNEMONIC_KEY, new Integer('A'));
    helpMenu.add(aboutAction);

    // add all top-level menus to menu bar
    JMenuBar menuBar = new JMenuBar();
    setJMenuBar(menuBar);
    menuBar.add(fileMenu); menuBar.add(editMenu); menuBar.add(helpMenu);

    // demonstrate pop-ups
    JPopupMenu popup = new JPopupMenu();
    popup.add(cutAction); popup.add(copyAction); popup.add(pasteAction);

    JPanel panel = new JPanel();
    panel.setComponentPopupMenu(popup);
    add(panel);
  }
}
```





```
class TestAction extends AbstractAction {
   public TestAction(String name) { super(name); }

   public void actionPerformed(ActionEvent event) {
      JOptionPane.showMessageDialog(null, getValue(Action.NAME) + " selected.");
   }
}
```

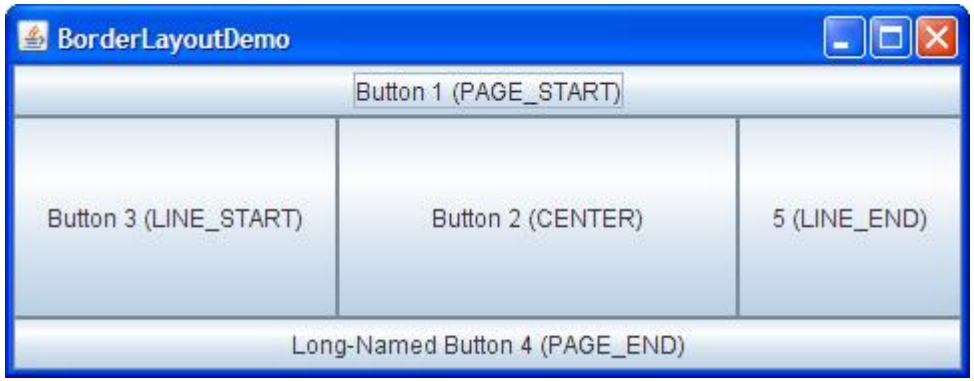


Layout Management

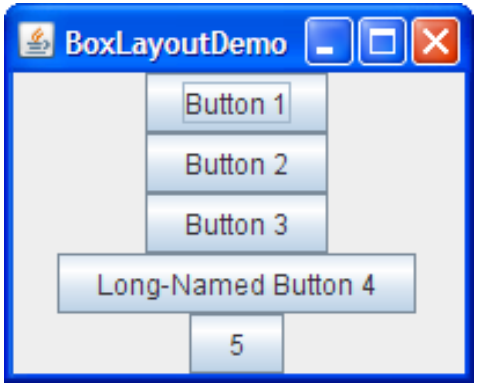
- A layout manager determines the size and position of the components within a container.
  - BorderLayout
  - BoxLayout
  - FlowLayout
  - CardLayout
  - GridLayout
  - GridBagLayout
  - GroupLayout
  - SpringLayout

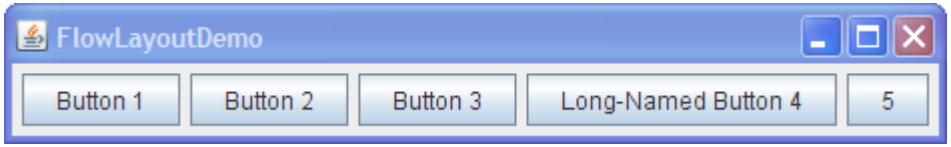
https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

Layout Management

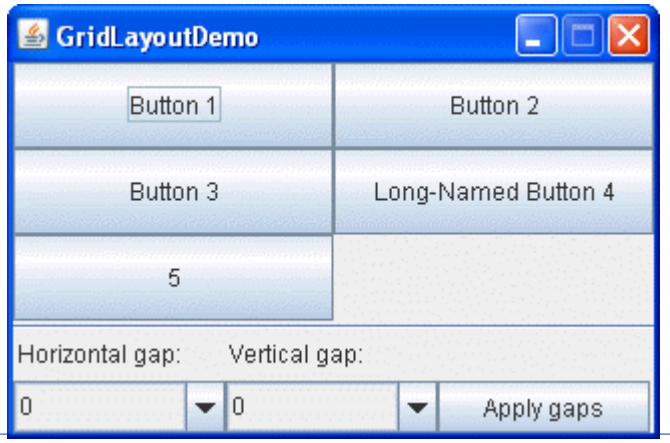


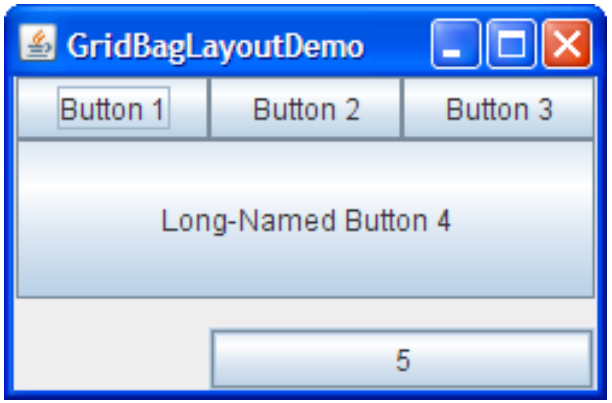
JFrame is initialized to use a BorderLayout





FlowLayout is the default layout manager for every JPanel



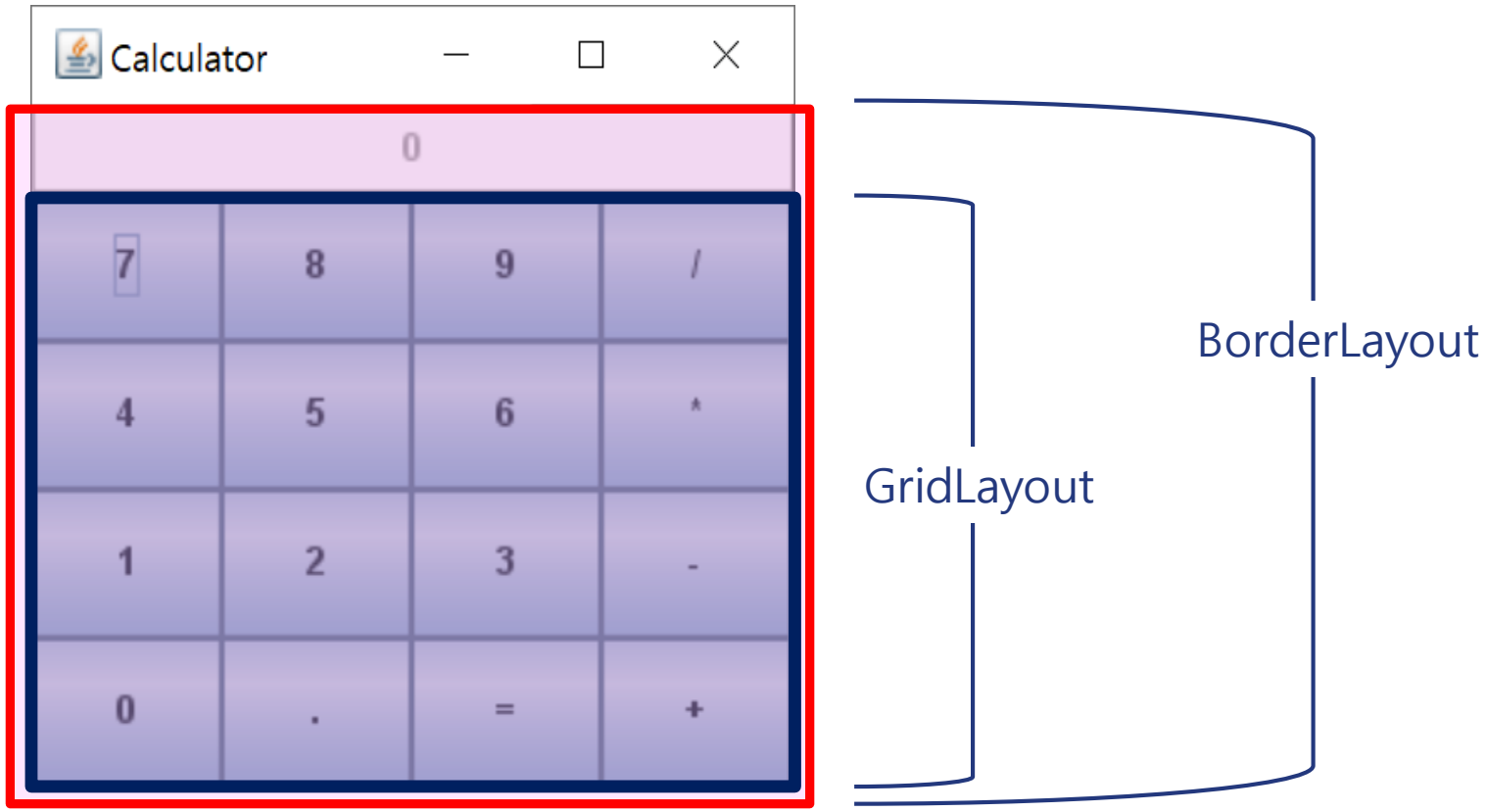


Layout Management

Layout	Description
BorderLayout	places components in up to five areas: top, bottom, left, right, and center
BoxLayout	puts components in a single row or column
FlowLayout	lays out components in a single row, starting a new row if its container is not sufficiently wide
GridLayout	makes a bunch of components equal in size and displays them in the requested number of rows and columns
GridBagLayout	aligns components by placing them within a grid of cells, allowing components to span more than one cell

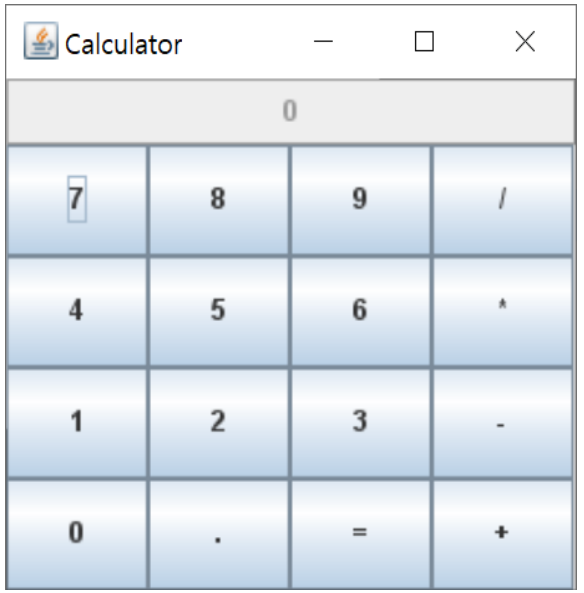
Layout Management: Example

- Calculator with BorderLayout Manger and GridLayout Manager



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

public class Calculator {
   public static void main(String[] args) {
      CalculatorFrame frame = new CalculatorFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
class CalculatorFrame extends JFrame {
  private JButton display;
  private JPanel panel;
  private double result;
  private String lastCommand;
  private boolean start;
   public CalculatorFrame() {
      setTitle("Calculator");
      CalculatorPanel panel = new CalculatorPanel();
      add(panel);
      pack(); // Causes this Window to be sized to fit the layouts of its subcomponents.
   }
}
```

```
class CalculatorPanel extends JPanel {// the default layout manager: flow layout
 public CalculatorPanel() {
   setLayout(new BorderLayout());  // North, West, Center, East, South

   result = 0; lastCommand = "="; start = true;

   display = new JButton("0"); display.setEnabled(false);
   add(display, BorderLayout.NORTH);

   ActionListener insert = new InsertAction();
   ActionListener command = new CommandAction();
   // The grid layout arranges all components in rows and columns like a spreadsheet.
   commandPanel = new JPanel();
   add(commandPanel, BorderLayout.CENTER);

   commandPanel.setLayout(new GridLayout(4, 4));

   addButton("7", insert); addButton("8", insert);
   addButton("9", insert); addButton("/", command);
   addButton("4", insert); addButton("5", insert);
   addButton("6", insert); addButton("*", command);
   addButton("1", insert); addButton("2", insert);
   addButton("3", insert); addButton("-", command);
   addButton("0", insert); addButton(".", insert);
   addButton("=", command); addButton("+", command);
 }
```



```
  private void addButton(String label, ActionListener listener) {
    JButton button = new JButton(label);
    button.addActionListener(listener); commandPanel.add(button);
  }
  private class InsertAction implements ActionListener {
    public void actionPerformed(ActionEvent event) {
      String input = event.getActionCommand();
      if (start)  { display.setText(""); start = false; }
      display.setText(display.getText() + input);
    }
  }
  private class CommandAction implements ActionListener {
    public void actionPerformed(ActionEvent event) {
      String command = event.getActionCommand();
      if (start) {
        if (command.equals("-") ) { display.setText(command);  start = false;  }
        else  lastCommand = command;
      } else {
        calculate(Double.valueOf(display.getText()));
        lastCommand = command;  start = true;
      }
    }
    private void calculate(double x) {
      if (lastCommand.equals("+")) result += x;
      else if (lastCommand.equals("-")) result -= x;
      else if (lastCommand.equals("*")) result *= x;
      else if (lastCommand.equals("/")) result /= x;
      else if (lastCommand.equals("=")) result = x;
      display.setText("" + result);
    }
  }
}
```

Q&A