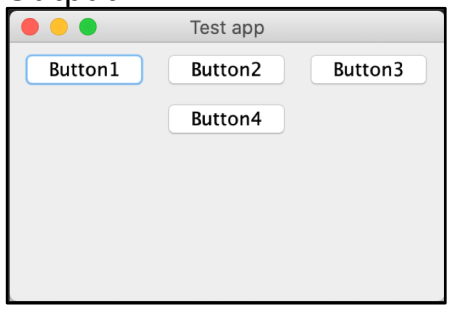
Example: FlowLayout

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

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        setTitle("Test app");
        setSize(300,200);
        setLocation(10,200);

        JPanel content = new JPanel();
        content.setLayout(new FlowLayout());

        JButton btnTest1 = new JButton("Button1");
        content.add(btnTest1);
        JButton btnTest2 = new JButton("Button2");
        content.add(btnTest2);
        JButton btnTest3 = new JButton("Button3");
        content.add(btnTest3);
        JButton btnTest4 = new JButton("Button4");
        content.add(btnTest4);

        setContentPane(content);
    }

    public static void main(String[]args) {
        JFrame myApp = new Test();
        myApp.show();
    }
}
```

Output:



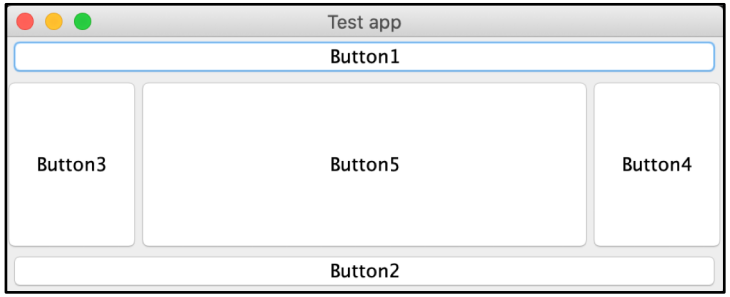

```
setTitle("Test app");
setSize(500,200);
setLocation(10,200);
```

If enough space in the content horizontally then all components are placed in one row

Example: BorderLayout

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

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        setTitle("Test app");
        setSize(500,200);
        setLocation(10,200);

        JPanel content = new JPanel();
        content.setLayout(new BorderLayout());

        JButton btnTest1 = new JButton("Button1");
        content.add(btnTest1, BorderLayout.NORTH);
        JButton btnTest2 = new JButton("Button2");
        content.add(btnTest2, BorderLayout.SOUTH);
        JButton btnTest3 = new JButton("Button3");
        content.add(btnTest3, BorderLayout.WEST);
        JButton btnTest4 = new JButton("Button4");
        content.add(btnTest4, BorderLayout.EAST);
        JButton btnTest5 = new JButton("Button5");
        content.add(btnTest5, BorderLayout.CENTER);

        setContentPane(content);
    }

    public static void main(String[]args) {
        JFrame myApp = new Test();
        myApp.show();
    }
}
```

Output:



Example: CardLayout

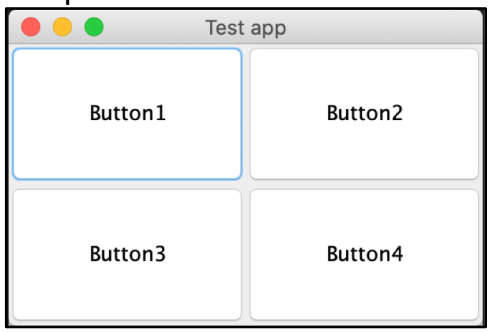
```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        setTitle("Test app");
        setSize(500,200);
        setLocation(10,200);

        JPanel content = new JPanel();
        content.setLayout(new CardLayout());

        JButton btnTest1 = new JButton("Button1");
        content.add(btnTest1);
        JButton btnTest2 = new JButton("Button2");
        content.add(btnTest2);
        JButton btnTest3 = new JButton("Button3");
        content.add(btnTest3);

        setContentPane(content);
    }

    public static void main(String[]args) {
        JFrame myApp = new Test();
        myApp.show();
    }
}
```

Output:

Example: GridLayout

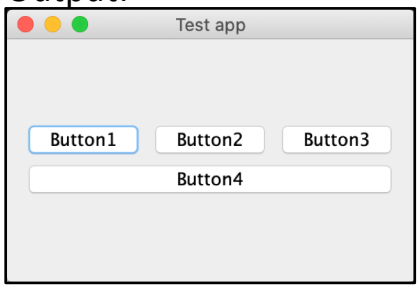
```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        setTitle("Test app");
        setSize(300,200);
        setLocation(10,200);

        JPanel content = new JPanel();
        content.setLayout(new GridLayout(2, 3));

        JButton btnTest1 = new JButton("Button1");
        content.add(btnTest1);
        JButton btnTest2 = new JButton("Button2");
        content.add(btnTest2);
        JButton btnTest3 = new JButton("Button3");
        content.add(btnTest3);
        JButton btnTest4 = new JButton("Button4");
        content.add(btnTest4);

        setContentPane(content);
    }

    public static void main(String[]args) {
        JFrame myApp = new Test();
        myApp.show();
    }
}
```

Output:



Example: GridBagLayout

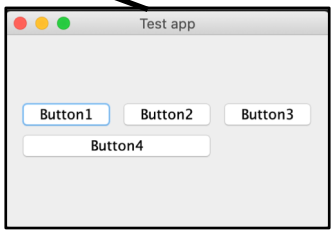
Output:



Need to create *GridBagConstraints* instance to manipulate this layout

A lot of flexibility in layout details

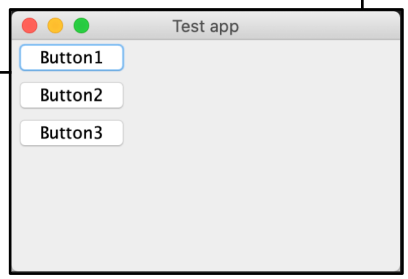
```
JButton btnTest4 = new JButton("Button4");
gbc.gridx = 0;
gbc.gridy = 2;
gbc.gridwidth = 2;
content.add(btnTest4, gbc);
```



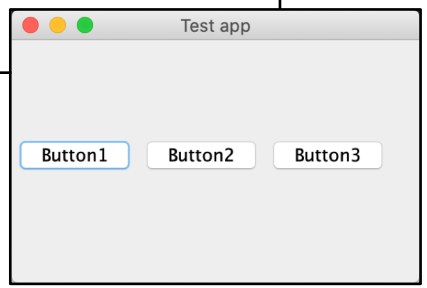

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        setTitle("Test app");
        setSize(300,200);
        setLocation(10,200);

        JPanel content = new JPanel();
        content.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();

        JButton btnTest1 = new JButton("Button1");
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.gridx = 0;
        gbc.gridy = 0;
        content.add(btnTest1, gbc);

        JButton btnTest2 = new JButton("Button2");
        gbc.gridx = 1;
        gbc.gridy = 0;
        content.add(btnTest2, gbc);

        JButton btnTest3 = new JButton("Button3");
        gbc.gridx = 2;
        gbc.gridy = 0;
        content.add(btnTest3, gbc);

        JButton btnTest4 = new JButton("Button4");
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.gridwidth = 3;
        content.add(btnTest4, gbc);

        setContentPane(content);
    }

    public static void main(String[]args) {
        JFrame myApp = new Test();
        myApp.show();
    }
}
```

Example: BoxLayout

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

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        setTitle("Test app");
        setSize(300,200);
        setLocation(10,200);

        JPanel content = new JPanel();
        content.setLayout(new BoxLayout(content, BoxLayout.Y_AXIS));

        JButton btnTest1 = new JButton("Button1");
        content.add(btnTest1);
        JButton btnTest2 = new JButton("Button2");
        content.add(btnTest2);
        JButton btnTest3 = new JButton("Button3");
        content.add(btnTest3);

        setContentPane(content);
    }

    public static void main(String[]args) {
        JFrame myApp = new Test();
        myApp.show();
    }
}
```



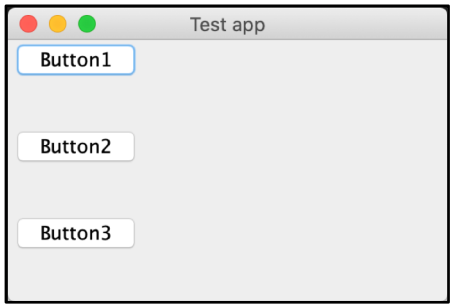

```java
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        setTitle("Test app");
        setSize(300,200);
        setLocation(10,200);

        JPanel content = new JPanel();
        content.setLayout(new BoxLayout(content, BoxLayout.X_AXIS));

        JButton btnTest1 = new JButton("Button1");
        content.add(btnTest1);
        JButton btnTest2 = new JButton("Button2");
        content.add(btnTest2);
        JButton btnTest3 = new JButton("Button3");
        content.add(btnTest3);

        setContentPane(content);
    }

    public static void main(String[]args) {
        JFrame myApp = new Test();
        myApp.show();
    }
}
```



Example: adding empty vertical spaces to BoxLayout

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

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        setTitle("Test app");
        setSize(300,200);
        setLocation(10,200);

        JPanel content = new JPanel();
        content.setLayout(new BoxLayout(content, BoxLayout.Y_AXIS));

        JButton btnTest1 = new JButton("Button1");
        content.add(btnTest1);
        content.add(Box.createRigidArea(new Dimension(0, 30)));
        JButton btnTest2 = new JButton("Button2");
        content.add(btnTest2);
        content.add(Box.createRigidArea(new Dimension(0, 30)));
        JButton btnTest3 = new JButton("Button3");
        content.add(btnTest3);

        setContentPane(content);
    }

    public static void main(String[]args) {
        JFrame myApp = new Test();
        myApp.show();
    }
}
```

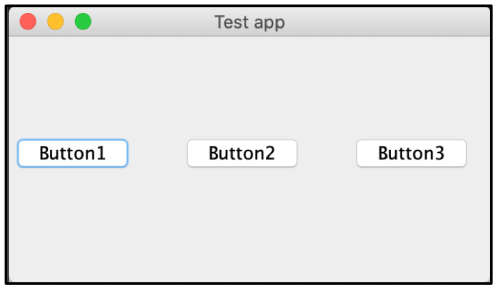
Output:



Example: adding empty horizontal spaces to BoxLayout

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

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        setTitle("Test app");
        setSize(350,200);
        setLocation(10,200);

        JPanel content = new JPanel();
        content.setLayout(new BoxLayout(content, BoxLayout.X_AXIS));

        JButton btnTest1 = new JButton("Button1");
        content.add(btnTest1);
        content.add(Box.createRigidArea(new Dimension(30, 0)));
        JButton btnTest2 = new JButton("Button2");
        content.add(btnTest2);
        content.add(Box.createRigidArea(new Dimension(30, 0)));
        JButton btnTest3 = new JButton("Button3");
        content.add(btnTest3);

        setContentPane(content);
    }

    public static void main(String[]args) {
        JFrame myApp = new Test();
        myApp.show();
    }
}
```

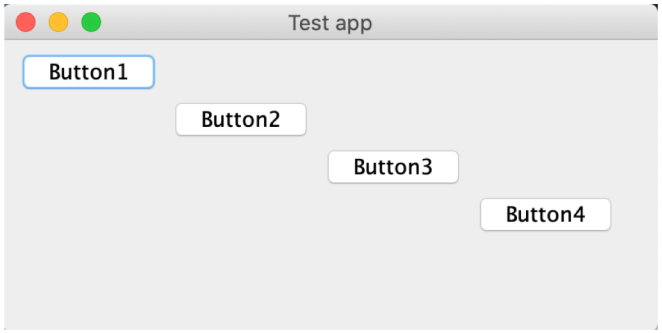
Output:



Example: GroupLayout

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

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        JPanel content = new JPanel();
        GroupLayout groupLayout = new GroupLayout(content);
        groupLayout.setAutoCreateGaps(true);
        groupLayout.setAutoCreateContainerGaps(true);
        content.setLayout(groupLayout);

        JButton btnButton1 = new JButton("Button1");
        JButton btnButton2 = new JButton("Button2");
        JButton btnButton3 = new JButton("Button3");
        JButton btnButton4 = new JButton("Button4");

        content.add(btnButton1);
        content.add(btnButton2);
        content.add(btnButton3);
        content.add(btnButton4);

        groupLayout.setHorizontalGroup(groupLayout.createSequentialGroup().addComponent(btnButton1)
                                                                           .addComponent(btnButton2)
                                                                           .addComponent(btnButton3)
                                                                           .addComponent(btnButton4));

        groupLayout.setVerticalGroup(groupLayout.createSequentialGroup().addComponent(btnButton1)
                                                                         .addComponent(btnButton2)
                                                                         .addComponent(btnButton3)
                                                                         .addComponent(btnButton4));

        add(content);

        setTitle("Test app");
        setSize(400,200);
        setLocation(10,200);
        setVisible(true);
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

Output:

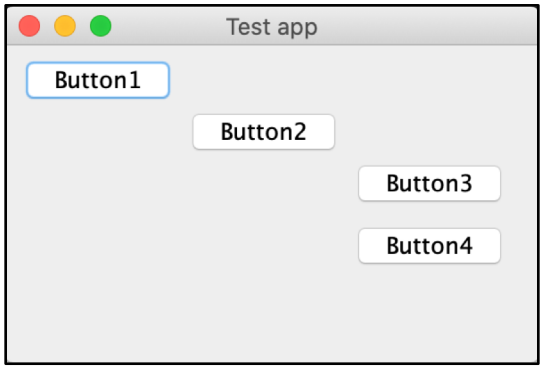


All components added sequentially both vertically and horizontally

Example: GroupLayout with horizontal groups

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

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        JPanel content = new JPanel();
        GroupLayout groupLayout = new GroupLayout(content);
        groupLayout.setAutoCreateGaps(true);
        groupLayout.setAutoCreateContainerGaps(true);
        content.setLayout(groupLayout);

        JButton btnButton1 = new JButton("Button1");
        JButton btnButton2 = new JButton("Button2");
        JButton btnButton3 = new JButton("Button3");
        JButton btnButton4 = new JButton("Button4");

        content.add(btnButton1);
        content.add(btnButton2);
        content.add(btnButton3);
        content.add(btnButton4);

        groupLayout.setHorizontalGroup(groupLayout.createSequentialGroup().addComponent(btnButton1)
                                                                          .addComponent(btnButton2)
                                      .addGroup(groupLayout.createSequentialGroup().addGroup(groupLayout
                                          .createParallelGroup(GroupLayout.Alignment.LEADING)
                                              .addComponent(btnButton3)
                                              .addComponent(btnButton4))));

        groupLayout.setVerticalGroup(groupLayout.createSequentialGroup().addComponent(btnButton1)
                                                                        .addComponent(btnButton2)
                                                                        .addComponent(btnButton3)
                                                                        .addComponent(btnButton4));

        add(content);

        setTitle("Test app");
        setSize(300,200);
        setLocation(10,200);
        setVisible(true);
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

Output:

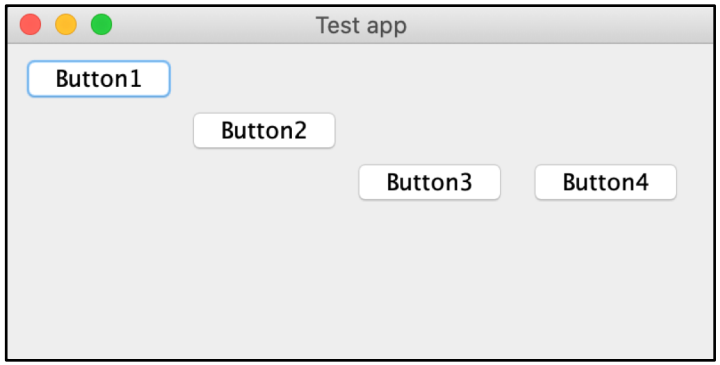


LEADING = left to right

Example: GroupLayout with vertical groups

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

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        JPanel content = new JPanel();
        GroupLayout groupLayout = new GroupLayout(content);
        groupLayout.setAutoCreateGaps(true);
        groupLayout.setAutoCreateContainerGaps(true);
        content.setLayout(groupLayout);

        JButton btnButton1 = new JButton("Button1");
        JButton btnButton2 = new JButton("Button2");
        JButton btnButton3 = new JButton("Button3");
        JButton btnButton4 = new JButton("Button4");

        content.add(btnButton1);
        content.add(btnButton2);
        content.add(btnButton3);
        content.add(btnButton4);

        groupLayout.setHorizontalGroup(groupLayout.createSequentialGroup().addComponent(btnButton1)
                                                                           .addComponent(btnButton2)
                                                                           .addComponent(btnButton3)
                                                                           .addComponent(btnButton4));

        groupLayout.setVerticalGroup(groupLayout.createSequentialGroup().addComponent(btnButton1)
                                                                         .addComponent(btnButton2)
                            .addGroup(groupLayout.createSequentialGroup().addGroup(groupLayout
                                    .createParallelGroup(GroupLayout.Alignment.LEADING)
                                        .addComponent(btnButton3)
                                        .addComponent(btnButton4))));

        add(content);

        setTitle("Test app");
        setSize(400,200);
        setLocation(10,200);
        setVisible(true);
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

Output:



Example: adding sub-panels to the main content panel

Note the use of BorderFactory class in this example

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

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        JPanel content = new JPanel();
        content.setLayout(new BoxLayout(content, BoxLayout.X_AXIS));
        content.setLayout(new FlowLayout());

        JPanel subpanel1 = new JPanel();
        subpanel1.setBorder(BorderFactory.createTitledBorder("Sub-panel 1"));
        BoxLayout subpanel1_layout = new BoxLayout(subpanel1, BoxLayout.Y_AXIS);
        subpanel1.setLayout(subpanel1_layout);
        JLabel lblInput1 = new JLabel("Input field:");
        JTextField txtInput1 = new JTextField(20);
        JButton btnTest1 = new JButton("Process input");
        subpanel1.add(lblInput1);
        subpanel1.add(txtInput1);
        subpanel1.add(btnTest1);

        JPanel subpanel2 = new JPanel();
        subpanel2.setBorder(BorderFactory.createTitledBorder("Sub-panel 2"));
        BoxLayout subpanel2_layout = new BoxLayout(subpanel2, BoxLayout.X_AXIS);
        subpanel2.setLayout(subpanel2_layout);
        subpanel2.setBackground(Color.CYAN);
        JLabel lblInput2 = new JLabel("Input field:");
        JTextField txtInput2 = new JTextField(30);
        JButton btnTest2 = new JButton("Store input");
        subpanel2.add(lblInput2);
        subpanel2.add(txtInput2);
        subpanel2.add(btnTest2);

        content.add(subpanel1);
        content.add(subpanel2);
        add(content);

        setTitle("Test app");
        setSize(600,300);
        setLocation(10,200);
        setVisible(true);
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

Subpanel 1

create title border

each subpanel can have its own layout

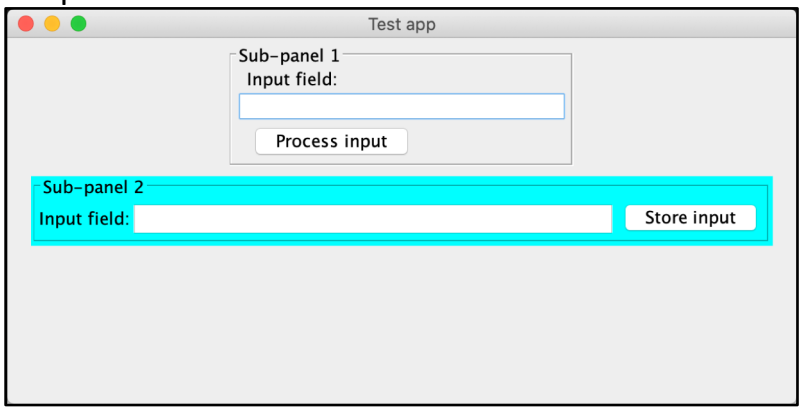
Subpanel 2

Set background of the panel to cyan

Add subpanels to the main panel

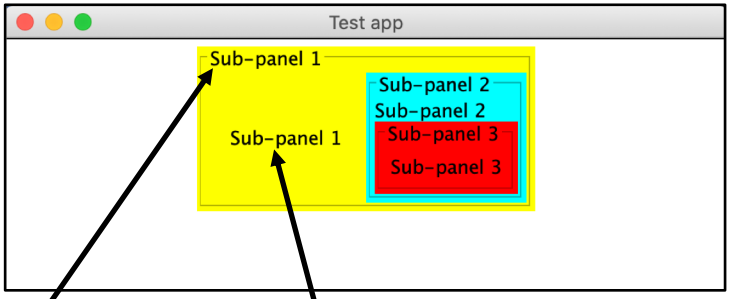
We can also open our app this way or by calling *show()* instead of *setVisible()*

Output:



Example: nested subpanels

Output:



panel title

label inside the panel

Set different background colors for each panel

Each sub-panel can have its own layout

Nest panels inside each other

use of Swing constants

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

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        JPanel content = new JPanel();
        content.setLayout(new BoxLayout(content, BoxLayout.X_AXIS));
        content.setLayout(new FlowLayout());
        content.setBackground(Color.WHITE);
        content.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

        JPanel subpanel1 = new JPanel();
        subpanel1.setBorder(BorderFactory.createTitledBorder("Sub-panel 1"));
        subpanel1.setLayout(new GridLayout(1, 2));
        subpanel1.add(new JLabel("Sub-panel 1", SwingConstants.CENTER));
        subpanel1.setBackground(Color.YELLOW);

        JPanel subpanel2 = new JPanel();
        subpanel2.setBorder(BorderFactory.createTitledBorder("Sub-panel 2"));
        subpanel2.setLayout(new BorderLayout());
        subpanel2.add(new JLabel("Sub-panel 2"), BorderLayout.NORTH);
        subpanel2.setBackground(Color.CYAN);

        JPanel subpanel3 = new JPanel();
        subpanel3.setBorder(BorderFactory.createTitledBorder("Sub-panel 3"));
        subpanel3.setLayout(new FlowLayout());
        subpanel3.add(new JLabel("Sub-panel 3", SwingConstants.RIGHT));
        subpanel3.setBackground(Color.RED);

        subpanel2.add(subpanel3, BorderLayout.SOUTH);
        subpanel1.add(subpanel2);
        content.add(subpanel1);
        add(content);

        setTitle("Test app");
        setSize(500,200);
        setLocation(10,200);
        setVisible(true);
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

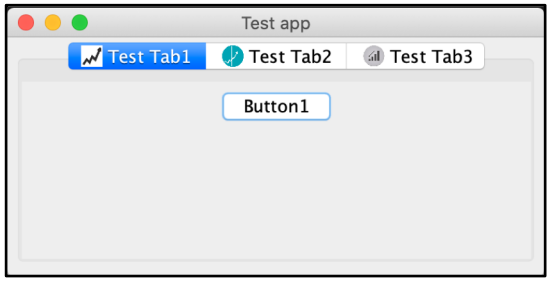
Example: using JTabbedPane container

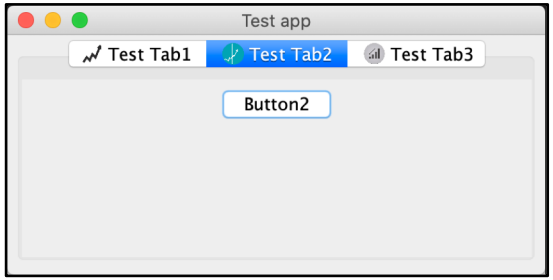
```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.JTabbedPane;

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        JPanel content = new JPanel();
        content.setLayout(new GridLayout(1, 1));

        JButton btnButton1 = new JButton("Button1");
        JButton btnButton2 = new JButton("Button2");
        JButton btnButton3 = new JButton("Button3");

        JPanel panel1 = new JPanel();
        JPanel panel2 = new JPanel();
        JPanel panel3 = new JPanel();

        panel1.add(btnButton1);
        panel2.add(btnButton2);
        panel3.add(btnButton3);

        JTabbedPane tabbedPane = new JTabbedPane(JTabbedPane.TOP);
        tabbedPane.addTab("Test Tab1", new ImageIcon("./my_app_icon.png"), panel1);
        tabbedPane.addTab("Test Tab2", new ImageIcon("./my_app_icon2.png"), panel2);
        tabbedPane.addTab("Test Tab3", new ImageIcon("./my_app_icon3.png"), panel3);

        content.add(tabbedPane);
        add(content);

        setTitle("Test app");
        setSize(400,200);
        setLocation(10,200);
        setVisible(true);
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

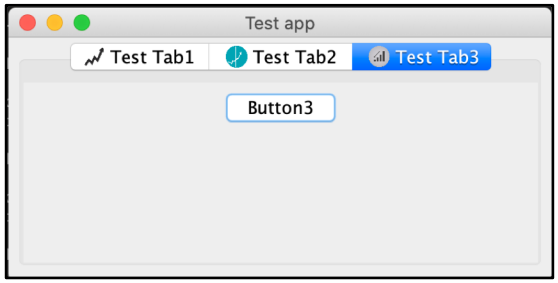
JTabbedPane.LEFT
JTabbedPane.RIGHT
JTabbedPane.BOTTOM
JTabbedPane.TOP

Each tab can have its own panel

Output:







Event handling in Java Swing applications

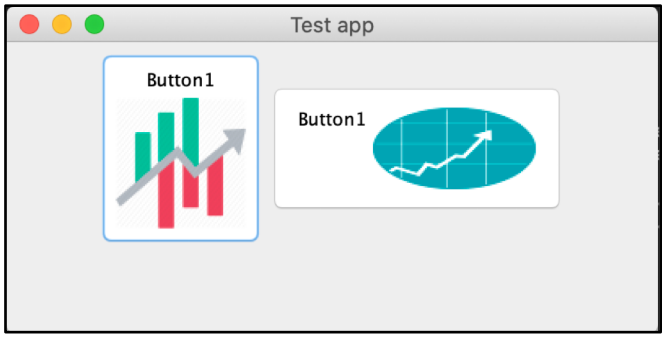
- Event handling is the same as in AWT applications
- Use action listeners and item listeners and add them to appropriate objects

Java Swing components

- JButton – button
- JTextField – text field
- JPasswordField – password field (text is automatically anonymized)
- JLabel – label
- JList – list box
- JCheckBox – check box (multiple selections allowed)
- JRadioButton – radio button (multiple selections are not allowed)
- JComboBox – drop down list
- JSlider – slider bar
- JSpiner – spinner field
- JMenu – drop down menu

Example: working with *JButton* objects

Output:



Load and resize icons (optional)

Can set a number of stylistic options

this button is disabled

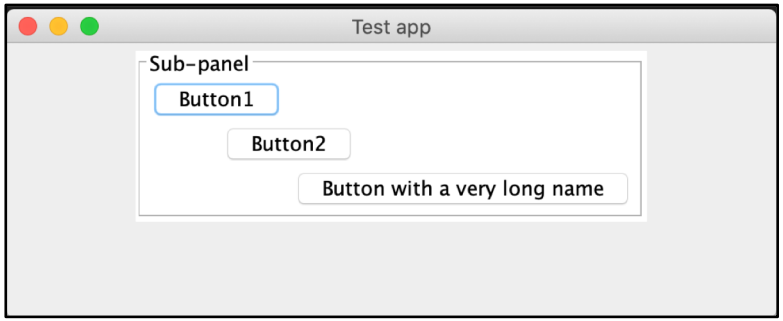
```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.JTabbedPane;

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        JPanel content = new JPanel();
        content.setLayout(new FlowLayout());

        ImageIcon btn1Icon = new ImageIcon("./my_app_icon4.png");
        Image img = btn1Icon.getImage();
        Image newimg = img.getScaledInstance(80, 80, java.awt.Image.SCALE_SMOOTH);
        btn1Icon = new ImageIcon(newimg);

        ImageIcon btn2Icon = new ImageIcon("./my_app_icon2.png");
        img = btn2Icon.getImage();
        newimg = img.getScaledInstance(100, 50, java.awt.Image.SCALE_SMOOTH);
        btn2Icon = new ImageIcon(newimg);

        JButton btnButton1 = new JButton("Button1", btn1Icon);
        JButton btnButton2 = new JButton("Button1", btn2Icon);

        btnButton1.setVerticalTextPosition(AbstractButton.TOP);
        btnButton1.setHorizontalTextPosition(AbstractButton.CENTER);
        btnButton1.setPreferredSize(new Dimension(100, 120));

        btnButton2.setVerticalTextPosition(AbstractButton.TOP);
        btnButton2.setHorizontalTextPosition(AbstractButton.LEFT);
        btnButton2.setActionCommand("disable");
        btnButton2.setPreferredSize(new Dimension(180, 80));

        content.add(btnButton1);
        content.add(btnButton2);
        add(content);

        setTitle("Test app");
        setSize(400,200);
        setLocation(10,200);
        setVisible(true);
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

Example: changing component alignment within the panel

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

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        JPanel content = new JPanel();
        content.setLayout(new BoxLayout(content, BoxLayout.X_AXIS));
        content.setLayout(new FlowLayout());

        JPanel subpanel1 = new JPanel();
        subpanel1.setBorder(BorderFactory.createTitledBorder("Sub-panel"));
        subpanel1.setLayout(new BoxLayout(subpanel1, BoxLayout.Y_AXIS));
        subpanel1.setBackground(Color.WHITE);

        JButton btnButton1 = new JButton("Button1");
        JButton btnButton2 = new JButton("Button2");
        JButton btnButton3 = new JButton("Button with a very long name");

        btnButton1.setAlignmentX(Component.RIGHT_ALIGNMENT);
        btnButton2.setAlignmentX(Component.CENTER_ALIGNMENT);
        subpanel1.add(btnButton1);
        subpanel1.add(btnButton2);
        subpanel1.add(btnButton3);

        content.add(subpanel1);
        add(content);

        setTitle("Test app");
        setSize(500,200);
        setLocation(10,200);
        setVisible(true);
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

Utilize component alignment when adding it to container

Output:



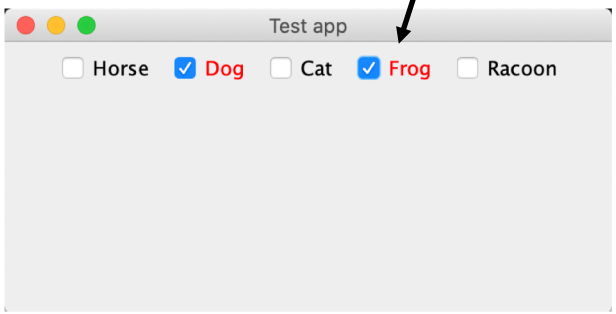
Example: working with JCheckBox objects

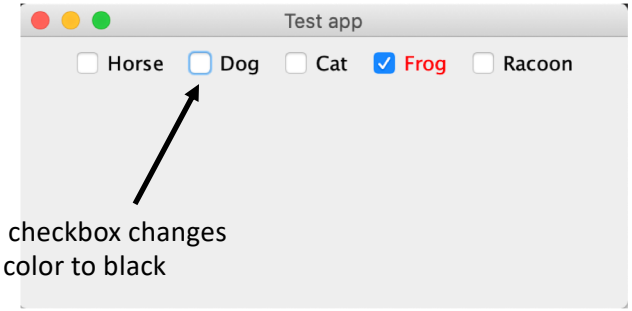
```java
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.JTabbedPane;

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        JPanel content = new JPanel();
        content.setLayout(new FlowLayout());

        JCheckBox chkBox1 = new JCheckBox("Horse");
        JCheckBox chkBox2 = new JCheckBox("Dog");
        JCheckBox chkBox3 = new JCheckBox("Cat");
        JCheckBox chkBox4 = new JCheckBox("Frog");
        JCheckBox chkBox5 = new JCheckBox("Racoon");

        chkBox2.setSelected(true);
        chkBox2.setForeground(Color.RED);

        ItemListener chkListener = new ItemListener(){...};

        chkBox1.addItemListener(chkListener);
        chkBox2.addItemListener(chkListener);
        chkBox3.addItemListener(chkListener);
        chkBox4.addItemListener(chkListener);
        chkBox5.addItemListener(chkListener);

        content.add(chkBox1);
        content.add(chkBox2);
        content.add(chkBox3);
        content.add(chkBox4);
        content.add(chkBox5);
        add(content);

        setTitle("Test app");
        setSize(400,200);
        setLocation(10,200);
        setVisible(true);
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

```java
ItemListener chkListener = new ItemListener(){
    public void itemStateChanged(ItemEvent event) {
        Object source = event.getItemSelectable();

        if (source == chkBox1) {
            if (event.getStateChange() == ItemEvent.DESELECTED){
                chkBox1.setForeground(Color.BLACK);
            }else{
                chkBox1.setForeground(Color.RED);
            }
        }else if (source == chkBox2) {
            if (event.getStateChange() == ItemEvent.DESELECTED){
                chkBox2.setForeground(Color.BLACK);
            }else{
                chkBox2.setForeground(Color.RED);
            }
        }else if (source == chkBox3) {
            if (event.getStateChange() == ItemEvent.DESELECTED){
                chkBox3.setForeground(Color.BLACK);
            }else{
                chkBox3.setForeground(Color.RED);
            }
        }else if (source == chkBox4) {
            if (event.getStateChange() == ItemEvent.DESELECTED){
                chkBox4.setForeground(Color.BLACK);
            }else{
                chkBox4.setForeground(Color.RED);
            }
        }else if (source == chkBox5) {
            if (event.getStateChange() == ItemEvent.DESELECTED){
                chkBox5.setForeground(Color.BLACK);
            }else{
                chkBox5.setForeground(Color.RED);
            }
        }
    }
};
```

Checkboxes and radio buttons use item listeners

Checking the checkbox changes the foreground color to red





Unchecking the checkbox changes the foreground color to black

Example: grouping checkboxes

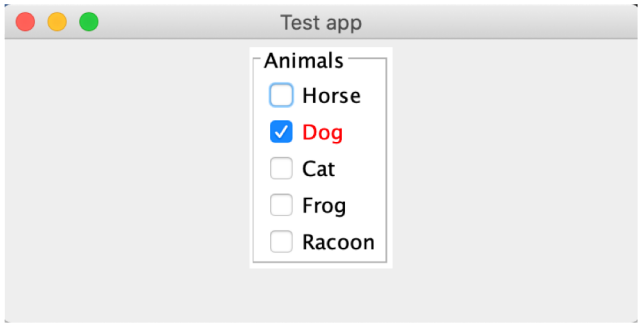
```
public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        JPanel content = new JPanel();
        content.setLayout(new FlowLayout());

        JCheckBox chkBox1 = new JCheckBox("Horse");
        JCheckBox chkBox2 = new JCheckBox("Dog");
        JCheckBox chkBox3 = new JCheckBox("Cat");
        JCheckBox chkBox4 = new JCheckBox("Frog");
        JCheckBox chkBox5 = new JCheckBox("Racoon");

        chkBox2.setSelected(true);
        chkBox2.setForeground(Color.RED);

        ItemListener chkListener = new ItemListener(){...};

        chkBox1.addItemListener(chkListener);
        chkBox2.addItemListener(chkListener);
        chkBox3.addItemListener(chkListener);
        chkBox4.addItemListener(chkListener);
        chkBox5.addItemListener(chkListener);

        JPanel chkButtonGroup = new JPanel();
        chkButtonGroup.setBorder(BorderFactory.createTitledBorder("Animals"));
        chkButtonGroup.setLayout(new BoxLayout(chkButtonGroup, BoxLayout.Y_AXIS));
        chkButtonGroup.setBackground(Color.WHITE);

        chkButtonGroup.add(chkBox1);
        chkButtonGroup.add(chkBox2);
        chkButtonGroup.add(chkBox3);
        chkButtonGroup.add(chkBox4);
        chkButtonGroup.add(chkBox5);
        content.add(chkButtonGroup);
        add(content);

        setTitle("Test app");
        setSize(400,200);
        setLocation(10,200);
        setVisible(true);
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

Same item listener implementation as in the previous slide

Add new sub-panel

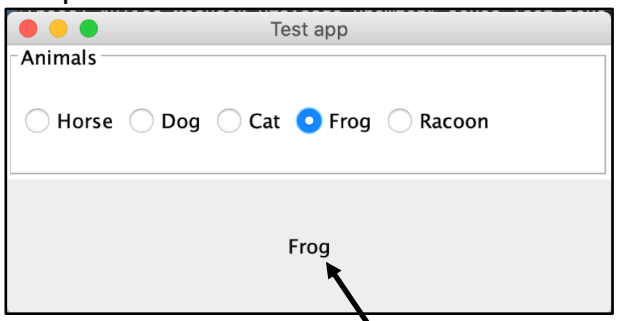
Add checkboxes to the new sub-panel with the desired layout

Output:



Example: working with JRadioButton objects

Output:



As the radio buttons are clicked the selection is reflected in the label below

Text within the label should be centered

Change label text to the text of the selected radio button

Only a single selection is allowed within one button group

```
public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        JPanel content = new JPanel();
        content.setLayout(new GridLayout(2,1));

        JRadioButton rdBox1 = new JRadioButton("Horse");
        JRadioButton rdBox2 = new JRadioButton("Dog");
        JRadioButton rdBox3 = new JRadioButton("Cat");
        JRadioButton rdBox4 = new JRadioButton("Frog");
        JRadioButton rdBox5 = new JRadioButton("Racoon");

        rdBox2.setSelected(true);
        JLabel lblSelection = new JLabel(rdBox2.getText());
        lblSelection.setHorizontalAlignment(JLabel.CENTER);

        ItemListener rdListener = new ItemListener(){
            public void itemStateChanged(ItemEvent event) {
                Object source = event.getItemSelectable();
                JRadioButton rdsource = (JRadioButton) source;
                lblSelection.setText(rdsource.getText());
            }
        };

        rdBox1.addItemListener(rdListener);
        rdBox2.addItemListener(rdListener);
        rdBox3.addItemListener(rdListener);
        rdBox4.addItemListener(rdListener);
        rdBox5.addItemListener(rdListener);

        JPanel rdButtonGroup = new JPanel();
        rdButtonGroup.setBorder(BorderFactory.createTitledBorder("Animals"));
        rdButtonGroup.setLayout(new BoxLayout(rdButtonGroup, BoxLayout.X_AXIS));
        rdButtonGroup.setBackground(Color.WHITE);

        ButtonGroup rdGroup = new ButtonGroup();
        rdGroup.add(rdBox1);
        rdGroup.add(rdBox2);
        rdGroup.add(rdBox3);
        rdGroup.add(rdBox4);
        rdGroup.add(rdBox5);

        rdButtonGroup.add(rdBox1);
        rdButtonGroup.add(rdBox2);
        rdButtonGroup.add(rdBox3);
        rdButtonGroup.add(rdBox4);
        rdButtonGroup.add(rdBox5);
        content.add(rdButtonGroup);
        content.add(lblSelection, Component.CENTER_ALIGNMENT);
        add(content);

        setTitle("Test app");
        setSize(400,200);
        setLocation(10,200);
        setVisible(true);
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

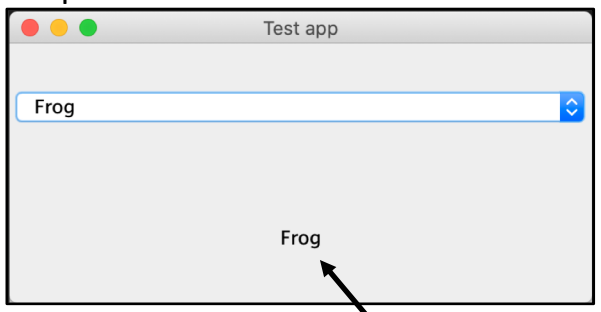
center label position

Example: working with JComboBox objects

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

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        JPanel content = new JPanel();
        content.setLayout(new GridLayout(2,1));

        String[] animals = { "Horse", "Dog", "Cat", "Frog", "Racoon" };
        JComboBox animalList = new JComboBox(animals);
        animalList.setSelectedIndex(2);

        JLabel lblSelection = new JLabel((String)animalList.getSelectedItem());
        lblSelection.setHorizontalAlignment(JLabel.CENTER);

        animalList.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent event) {
                JComboBox cb = (JComboBox)event.getSource();
                String animalString = (String)animalList.getSelectedItem();
                lblSelection.setText(animalString);
                //updateLabel(petName);
            }
        });

        content.add(animalList);
        content.add(lblSelection, Component.CENTER_ALIGNMENT);
        add(content);

        setTitle("Test app");
        setSize(400,200);
        setLocation(10,200);
        setVisible(true);
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

Implement action listener

Output:



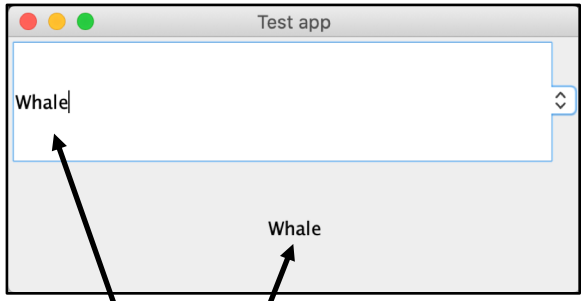
Selection in the drop down list is reflected in the label

Example: working with editable JComboBox objects

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

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        JPanel content = new JPanel();
        content.setLayout(new GridLayout(2,1));

        String[] animals = { "Horse", "Dog", "Cat", "Frog", "Racoon" };
        JComboBox animalList = new JComboBox(animals);
        animalList.setSelectedIndex(2);
        animalList.setEditable(true);

        JLabel lblSelection = new JLabel((String)animalList.getSelectedItem());
        lblSelection.setHorizontalAlignment(JLabel.CENTER);

        animalList.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent event) {
                JComboBox cb = (JComboBox)event.getSource();
                String animalString = (String)animalList.getSelectedItem();
                lblSelection.setText(animalString);
                //updateLabel(petName);
            }
        });

        content.add(animalList);
        content.add(lblSelection, Component.CENTER_ALIGNMENT);
        add(content);

        setTitle("Test app");
        setSize(400,200);
        setLocation(10,200);
        setVisible(true);
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

Changes the behavior of the combo box to allow free text

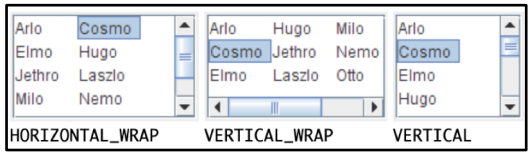
Output:



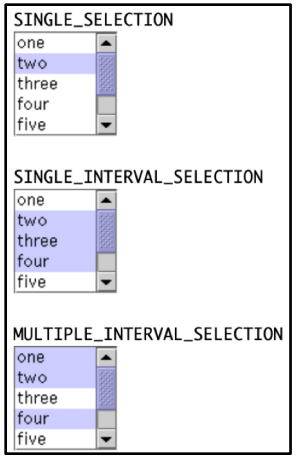
“Whale” is not one of the pre-loaded selections but we are allowed to type it in

Working with JList objects

- Extended tutorial is available here: https://docs.oracle.com/javase/tutorial/uiswing/components/list.html
- Wrap options:



- Selection options:



Working with JList objects (cont'd)

- Use *addListSelectionListener()* method to add a listener
  - *import javax.swing.event.*;*
- Implement
  - *public void valueChanged(ListSelectionEvent event){…}*

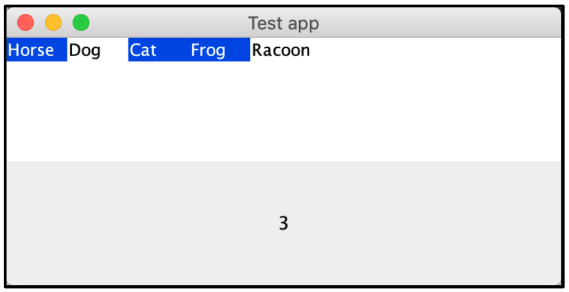
Example: working with JList objects

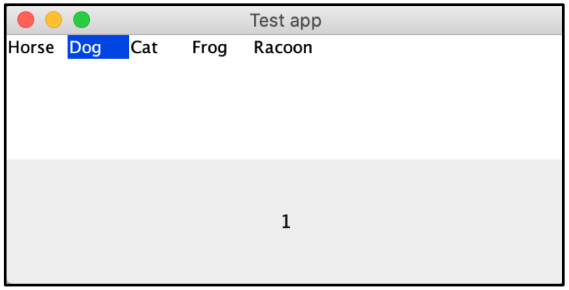
```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.event.*;

public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        JPanel content = new JPanel();
        content.setLayout(new GridLayout(2,1));

        String[] animals = { "Horse", "Dog", "Cat", "Frog", "Racoon" };
        JList animalList = new JList(animals);
        animalList.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
        animalList.setLayoutOrientation(JList.HORIZONTAL_WRAP);
        animalList.setVisibleRowCount(-1);

        JLabel lblSelection = new JLabel();
        lblSelection.setHorizontalAlignment(JLabel.CENTER);

        animalList.addListSelectionListener(new ListSelectionListener(){
            public void valueChanged(ListSelectionEvent event) {
                if (event.getValueIsAdjusting() == false) {
                    if (animalList.getSelectedIndex() >= 0) {
                        Integer size = animalList.getSelectedIndices().length;
                        lblSelection.setText(size.toString());
                    }
                }
            }
        });

        content.add(animalList);
        content.add(lblSelection, Component.CENTER_ALIGNMENT);
        add(content);

        setTitle("Test app");
        setSize(400,200);
        setLocation(10,200);
        setVisible(true);
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

Many stylistic options are available for JList objects

Implement ListSelectionListener

Output:

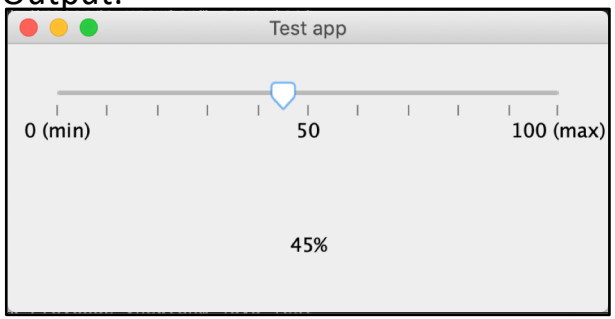




Example: working with JSlider objects

```java
public class Test extends JFrame {
    private static final int SLIDER_MIN = 0;
    private static final int SLIDER_MAX = 100;
    private static final int SLIDER_INIT = 45;

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

        JPanel content = new JPanel();
        content.setLayout(new GridLayout(2,1));

        JSlider percentSlider = new JSlider(JSlider.HORIZONTAL, SLIDER_MIN, SLIDER_MAX, SLIDER_INIT);
        percentSlider.setMajorTickSpacing(10);
        percentSlider.setPaintTicks(true);

        //Create the label table
        Hashtable<Integer, JLabel> labelTable = new Hashtable();
        labelTable.put(0, new JLabel("0 (min)"));
        labelTable.put(50, new JLabel("50"));
        labelTable.put(100, new JLabel("100 (max)"));
        percentSlider.setLabelTable(labelTable);
        percentSlider.setPaintLabels(true);

        JLabel lblSelection = new JLabel();
        lblSelection.setHorizontalAlignment(JLabel.CENTER);
        lblSelection.setText(Integer.toString(this.SLIDER_INIT)+"%");

        percentSlider.addChangeListener(new ChangeListener(){
            public void stateChanged(ChangeEvent event) {
                JSlider source = (JSlider)event.getSource();
                lblSelection.setText(Integer.toString((int)source.getValue())+"%");
            }
        });

        content.add(percentSlider, BorderLayout.CENTER);
        content.add(lblSelection, Component.CENTER_ALIGNMENT);
        add(content);

        setTitle("Test app");
        setSize(400,200);
        setLocation(10,200);
        setVisible(true);
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

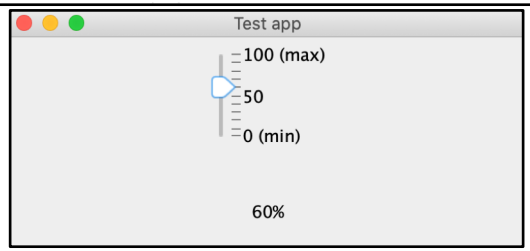
Output:



This block is optional in case we want to add labels to the slider ticks

Listen for changes in the state of the slider bar

```java
JSlider percentSlider = new JSlider(JSlider.VERTICAL, SLIDER_MIN, SLIDER_MAX, SLIDER_INIT);
```



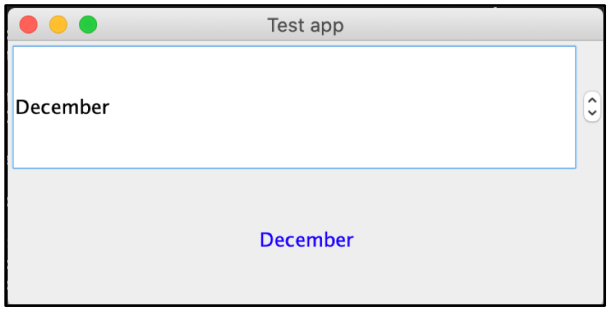
Example: working with JSpinner objects

```
public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        JPanel content = new JPanel();
        content.setLayout(new GridLayout(2,1));

        String[] monthStrings = {"January", "February", "March", "April", "May", "June",
                                 "July", "August", "October", "November", "December"};
        SpinnerListModel monthModel = new SpinnerListModel(monthStrings);
        JSpinner monthsSpinner = new JSpinner(monthModel);

        JLabel lblSelection = new JLabel();
        lblSelection.setHorizontalAlignment(JLabel.CENTER);
        lblSelection.setText((String)monthsSpinner.getValue());
        lblSelection.setForeground(determineColor((String)monthsSpinner.getValue()));

        monthsSpinner.addChangeListener(new ChangeListener(){
            public void stateChanged(ChangeEvent event) {
                JSpinner source = (JSpinner)event.getSource();
                String month = (String)source.getValue();
                lblSelection.setText(month);
                lblSelection.setForeground(determineColor(month));
            }
        });

        content.add(monthsSpinner);
        content.add(lblSelection, Component.CENTER_ALIGNMENT);
        add(content);

        setTitle("Test app");
        setSize(400,200);
        setLocation(10,200);
        setVisible(true);
    }

    private Color determineColor(String month){
        if(month == "December" || month == "January" || month == "February"){
            return(Color.BLUE);
        }else if(month == "March" || month == "April" || month == "May"){
            return(Color.GREEN);
        }else if(month == "June" || month == "July" || month == "August"){
            return(Color.RED);
        }else if(month == "September" || month == "October" || month == "November"){
            return(Color.ORANGE);
        }else{
            return(Color.BLACK);
        }
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

Listen for changes in the state

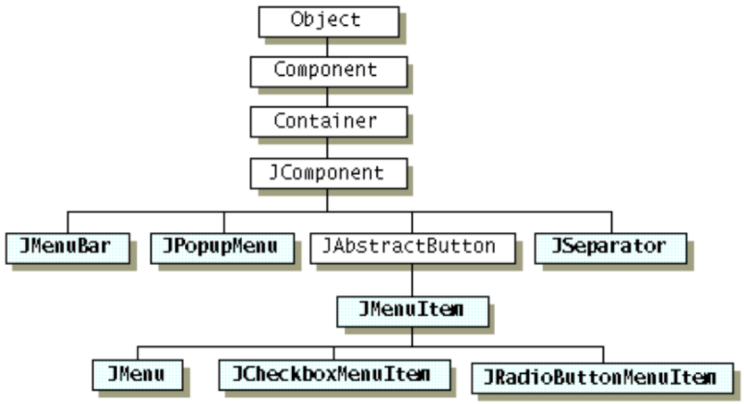
Helper/utility method that helps determine color based on the month selection

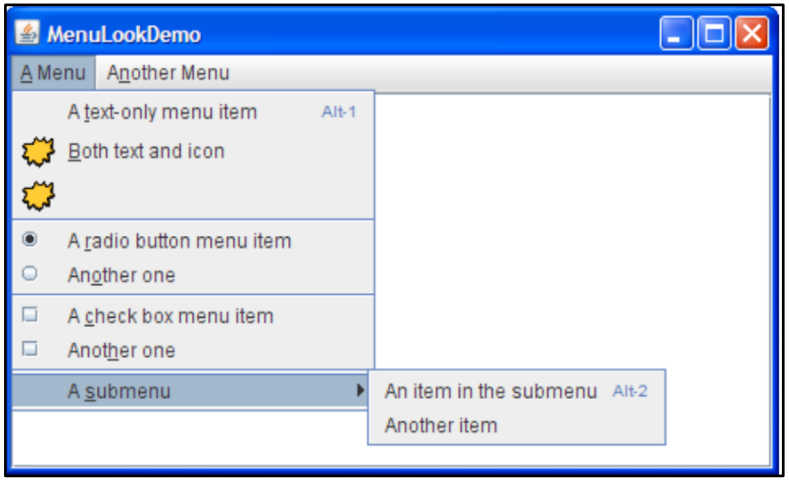
Output:



JMenuBar, JMenu, JMenuItem,

- There is a hierarchy of components when constructing menus
- JMenuBar -> JMenu -> JMenuItem

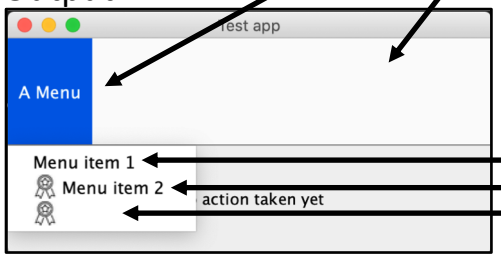




https://docs.oracle.com/javase/tutorial/uiswing/components/menu.html

Example: constructing a menu bar

Output:



Menu item, text only

Menu item, both text and image

Menu item, image only

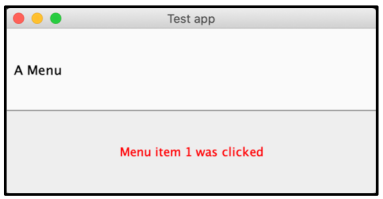
Menu bar

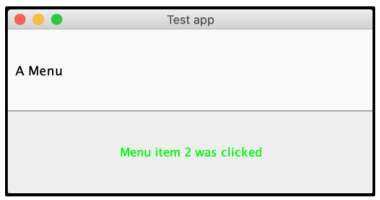
Menu

Menu items

Implement action listener for the menu items

```
public class Test extends JFrame {
    public Test(){
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent event) {
                System.exit(0);
            }
        } );

        JPanel content = new JPanel();
        content.setLayout(new GridLayout(2,1));

        JMenuBar menuBar = new JMenuBar();

        JMenu menu1 = new JMenu("A Menu");
        menuBar.add(menu1);

        // text based menu item:
        JMenuItem menuItem1 = new JMenuItem("Menu item 1");
        menu1.add(menuItem1);

        // both text and image menu item:
        JMenuItem menuItem2 = new JMenuItem("Menu item 2", new ImageIcon("./my_app_icon7.png"));
        menu1.add(menuItem2);

        // image menu item:
        JMenuItem menuItem3 = new JMenuItem(new ImageIcon("./my_app_icon7.png"));
        menu1.add(menuItem3);

        JLabel lblSelection = new JLabel("No action taken yet");
        lblSelection.setHorizontalAlignment(JLabel.CENTER);

        ActionListener menuAL = new ActionListener(){
            public void actionPerformed(ActionEvent event) {
                JMenuItem source = (JMenuItem)event.getSource();

                if(source == menuItem1){
                    lblSelection.setText("Menu item 1 was clicked");
                    lblSelection.setForeground(Color.RED);
                } else if(source == menuItem2){
                    lblSelection.setText("Menu item 2 was clicked");
                    lblSelection.setForeground(Color.GREEN);
                } else if(source == menuItem3){
                    lblSelection.setText("Menu item 3 was clicked");
                    lblSelection.setForeground(Color.BLUE);
                }
            }
        };

        menuItem1.addActionListener(menuAL);
        menuItem2.addActionListener(menuAL);
        menuItem3.addActionListener(menuAL);

        content.add(menuBar);
        content.add(lblSelection, Component.CENTER_ALIGNMENT);
        add(content);

        setTitle("Test app");
        setSize(400,200);
        setLocation(10,200);
        setVisible(true);
    }

    public static void main(String[]args) {
        new Test();
    }
}
```

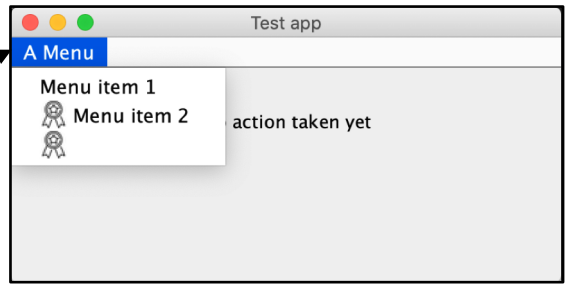




Example: using *setJMenuBar()* method

```
...
//content.add(menuBar);
this.setJMenuBar(menuBar);
content.add(lblSelection, Component.CENTER_ALIGNMENT);
add(content);
...
```

setJMenuBar() JFrame method



Notice that the menu has
a more conventional look

Example: submenus, checkbox menu items, and radio button menu items

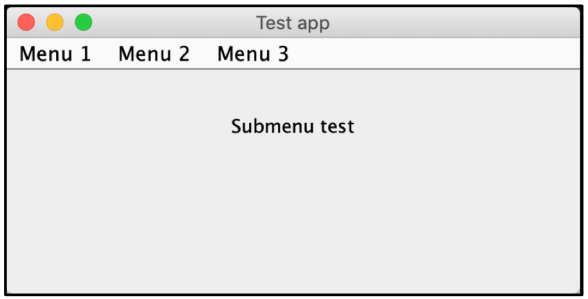
...

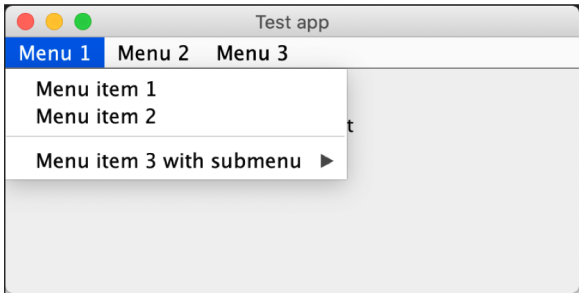
```
// first menu
JMenu menu1 = new JMenu("Menu 1");
menuBar.add(menu1);

// first menu elements
JMenuItem menuItem1 = new JMenuItem("Menu item 1");
menu1.add(menuItem1);

JMenuItem menuItem2 = new JMenuItem("Menu item 2");
menu1.add(menuItem2);

menu1.addSeparator();

JMenu submenu1 = new JMenu("Menu item 3 with submenu");
menu1.add(submenu1);

JMenuItem menuItem3 = new JMenuItem("Menu item 3 submenu 1");
submenu1.add(menuItem3);

JMenuItem menuItem4 = new JMenuItem("Menu item 3 submenu 2");
submenu1.add(menuItem4);

JLabel lblSelection = new JLabel("Submenu test");
lblSelection.setHorizontalAlignment(JLabel.CENTER);
```

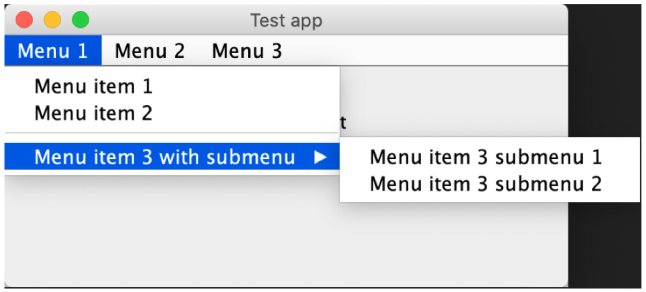
...

We can add separators

Submenu is a *JMenu* object







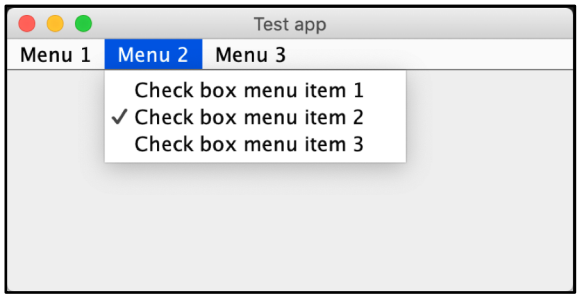
Example: submenus, checkbox menu items, and radio button menu items (cont'd)

...

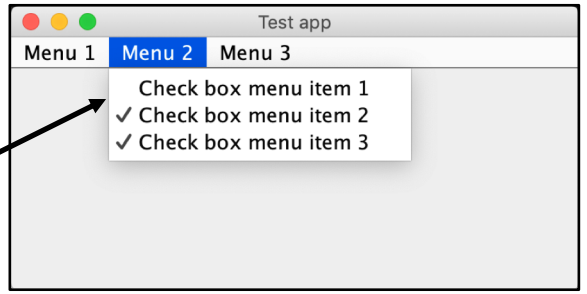
```
// checkbox menu items
JCheckBoxMenuItem cbMenuItem1 = new JCheckBoxMenuItem("Check box menu item 1");
menu2.add(cbMenuItem1);

JCheckBoxMenuItem cbMenuItem2 = new JCheckBoxMenuItem("Check box menu item 2");
cbMenuItem2.setSelected(true);
menu2.add(cbMenuItem2);

JCheckBoxMenuItem cbMenuItem3 = new JCheckBoxMenuItem("Check box menu item 3");
menu2.add(cbMenuItem3);
```

...

We can make default menu selections



Multiple selections are allowed

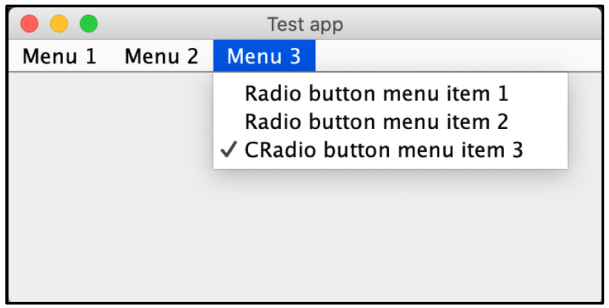


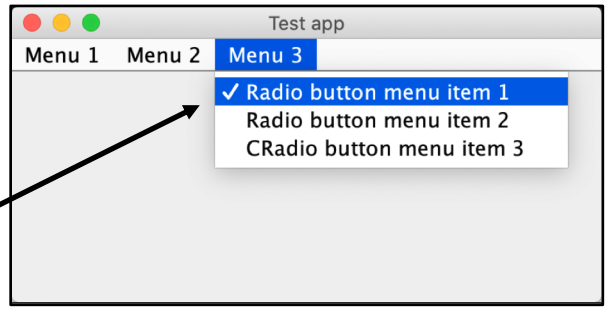
Example: submenus, checkbox menu items, and radio button menu items (cont'd)

...

```
// third menu
JMenu menu3 = new JMenu("Menu 3");
menuBar.add(menu3);

// radio button menu items
ButtonGroup group = new ButtonGroup();
JRadioButtonMenuItem rdMenuItem1 = new JRadioButtonMenuItem("Radio button menu item 1");
menu3.add(rdMenuItem1);
group.add(rdMenuItem1);

JRadioButtonMenuItem rdMenuItem2 = new JRadioButtonMenuItem("Radio button menu item 2");
menu3.add(rdMenuItem2);
group.add(rdMenuItem2);

JRadioButtonMenuItem rdMenuItem3 = new JRadioButtonMenuItem("CRadio button menu item 3");
rdMenuItem3.setSelected(true);
menu3.add(rdMenuItem3);
group.add(rdMenuItem3);
```

...





Multiple selections are not allowed for radio button menu items within the same button group

Troubleshooting tips for your AWT and Java Swing GUI app

- Your program compiles successfully but it does not do what you expect it to do. What now?
- Compile time
  - Did all you include all the proper imports?

- Appearance
  - Did you add your component to the container?
  - Did you set components and/or containers to visible?
  - Is the appropriate size specified?
  - Are you using the appropriate layout?
    - Always helps to sketch out your GUI

- Behavior
  - Did you implement the appropriate listener event handlers?
  - Did you add listeners to appropriate component and/or container objects?
  - Did you add correct listeners?
  - If dealing with radio buttons, did you create a button group?

JavaFX

- Open source Java GUI programming framework introduced in Java 8
- Meant to replace Java Swing
- Advanced visual features and inputs
  - Geared towards developing of rich GUI applications
  - Can create GUI applications, web applications, graphical application
- *javafx* package
- Main idea: scene graph
  - Scene graph is a collection of components/elements (nodes)
  - Arranged in a hierarchical graph

JavaFX applications

- Your GUI application class has to inherit from *Application* class
  - *javafx.application* package
- You start the application by calling *launch()* method in *Application* class
- We must override *start()* method, the entry point into a JavaFX application
  - Called by the *launch()* method
- Controls (visual components) are in the *javafx.scene.control* package

Stage class

- Every JavaFX application has at least one top level stage (primary stage)
  - More than one stages can be created for an application
- Primary stage is always automatically created
  - Stage object is passed in as input parameter into *start()* method
- Stage is used to display a scene
  - Similar to the theater stage in real life
- Scene contains visual nodes (text, shapes, images, controls, animation, etc.)
- Display the application by calling *stage.show();*

Example: basic JavaFX application

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class Test extends Application {
    public static void main(String[]args) {
        Application.launch(args);
    }

    @Override
    public void start(Stage stage)
    {
        VBox root = new VBox();

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }
}
```

Note different imports from what we used before

Test class inherits from Application class

Pass on args to *launch()* method

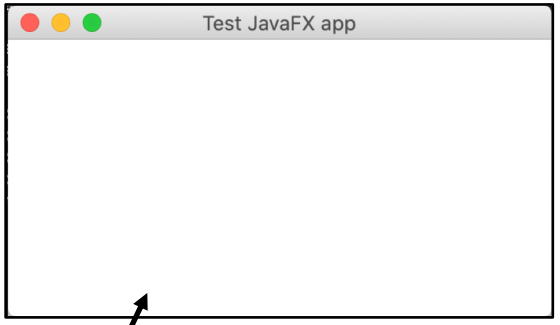
An object of Stage class is passed into *start()* method

Hierarchical graph needs a root, which we attach to a scene

Set stage scene to appropriate scene object

Set some stage settings

Output:

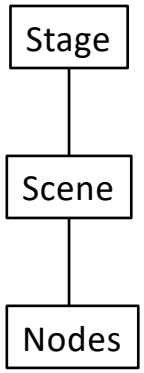


Empty frame

Screen visual components

- Label – represents a label in the scene
- TextField – represents a text field in the scene
- PasswordField – represents a text field with anonymized input
- TextArea – represents a text field that allows multiple line input
- Button – represents a button in the scene
- MenuButton – represents an object that looks like a button but behaves like a menu
- ToggleButton – represents a button that can have two states
- CheckBox – represents a checkbox object (multiple selections allowed)
- RadioButton – represents a radio button object (multiple selections are not allowed)
- ChoiceBox – represents a drop down list
- ComboBox – represents a more sophisticated version of ChoiceBox with many more controls over list selection and behavior
- ListView – represents a list of items
- Menu – represents a menu object

Component hierarchy



Event handling in JavaFX

- Upon every user interaction an event is triggered
- Event handling is done with classes and functionality in *javafx.event* package
- Event types
  - MouseEvent
  - KeyEvent
  - DragEvent
  - MouseDragEvent
  - WindowEvent
  - GestureEvent
  - RotateEvent
  - SwipeEvent
  - ZoomEvent
  - TouchEvent
  - WebEvent
  - …

Actors are similar to AWT and Java Swing event handling

- Target − node on which event occurs (e.g. window, scene, node)
- Source − source from which the event is generated (e.g. mouse, keyboard, etc.)
- Type − type of the event that occurred (e.g. mouse pressed, mouse released, etc.)

Scene graph event handling

- Event travels from bottom up hierarchy chain (from node to scene to stage to root) and if any of the nodes in this chain have an event handler registered to handle this event type then it is executed
  - Event capture
    - Filters are triggered
  - Event bubbling
    - Handlers are triggered
- If no event handler is found as the event travels then it is handled at the root node

Event capture vs. event bubbling (event filters vs. event handlers)

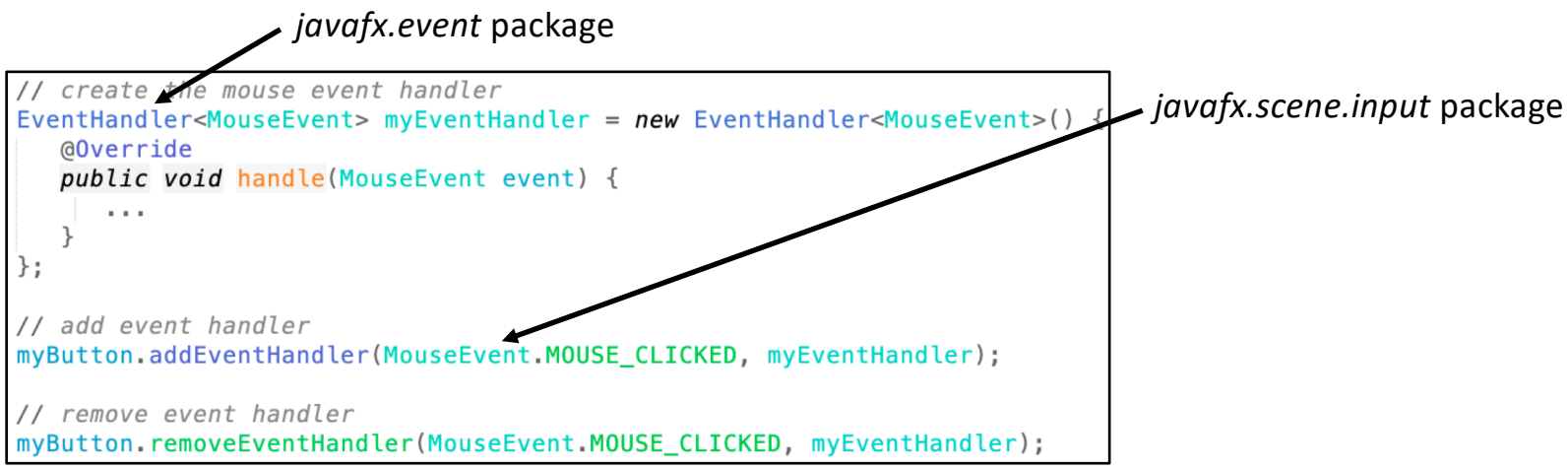
- Event capture
  - Once the event occurs, every node in the chain is polled for the appropriate event filter
  - As soon as one of the filters makes a call to *consume()* method the polling stops and that node becomes the target
  - If no nodes called *consume()* then node at the end of the chain becomes the target
    - Stage -> Scene -> Pane -> Button
    - If no event filters call *consume()* then Button is the target

- Event bubbling
  - The event is then "bubbled" up to the top of the chain looking for an appropriate event handler
    - From event target to the top of the chain
    - Button -> Pane -> Scene -> Stage

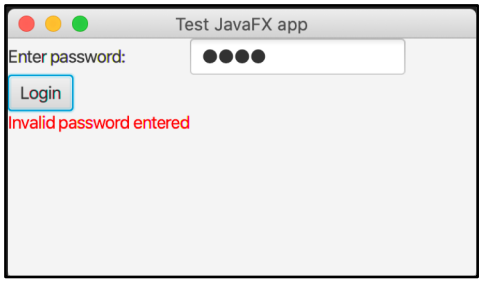
Why have both filters and handlers

- Occur at different point in the process
  - Filters are called during event capture
  - Handlers are called during event bubbling
- Can filter out events you do not want to process/handle before handlers are notified
  - *consume()* is used to filter out the event
- Most of the time you would just use handlers

Adding and removing event handlers



Example: app with a password field



JavaFX related imports

Override the *start()* method to implement our scene

Special type of input field

MouseEvent handler

Change message label text and color depending on the validity of the entered password

Button object

Arrange objects in the scene by rows

Setup the stage

Setup stage options and show it

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.paint.Color;
import javafx.scene.input.*;
import javafx.scene.layout.GridPane;
import javafx.event.*;

public class Test extends Application {
    private final String pwd = "cs151";

    @Override
    public void start(Stage stage)
    {
        Label lblLogin = new Label("Enter password:");
        Label lblMessage = new Label();

        PasswordField pwdField = new PasswordField();
        pwdField.setPromptText("Enter your password");

        EventHandler<MouseEvent> loginEventHandler = new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent event) {
                String enteredPwd = pwdField.getText();

                if(enteredPwd.equals("cs151")){
                    lblMessage.setText("Valid password entered");
                    lblMessage.setTextFill(Color.web("#000000"));
                }else{
                    lblMessage.setText("Invalid password entered");
                    lblMessage.setTextFill(Color.web("#FF0000"));
                }
            }
        };

        Button btnLogin = new Button("Login");
        btnLogin.addEventHandler(MouseEvent.MOUSE_CLICKED, loginEventHandler);


        GridPane root = new GridPane();
        root.addRow(0, lblLogin, pwdField);
        root.addRow(1, btnLogin);
        root.addRow(2, lblMessage);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

    public static void main(String[]args) {
        Application.launch(args);
    }
}
```

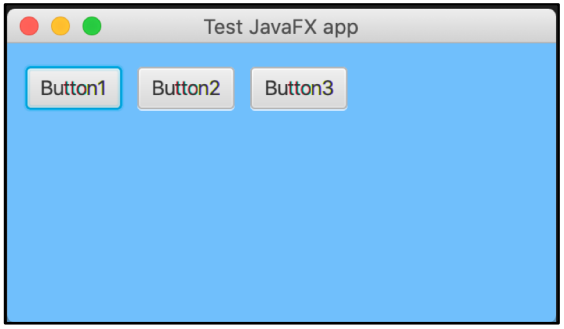
JavaFX layouts

- FlowPane – similar to FlowLayout in Java Swing, objects are arranged in the order they are added.
- TilePane – similar to GridLayout in Java Swing, each cell/tile of the grid is the same size.
- GridPane – arrange elements in columns and rows. Different cell sizes can be arranged.
- HBox – arranges objects in a single horizontal row with configurable padding.
- VBox – similar to HBox but objects are arranged vertically.
- BorderPane – similar to BorderLayout in Java Swing layout. Arranges objects in the following positions: Top, Left, Center, Right, Bottom.
- StackPane – objects are stacked on top of each other (provides ability to overlay objects on top of each other; e.g. text over image).
- AnchorPane - anchor nodes to the top, bottom, left side, right side, or center of the pane. These positions are maintained/anchored when the window is resized.

Example: HBox layout

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Button btnButton1 = new Button("Button1");
        Button btnButton2 = new Button("Button2");
        Button btnButton3 = new Button("Button3");

        HBox hboxLayout = new HBox();
        hboxLayout.setPadding(new Insets(15, 12, 15, 12));
        hboxLayout.setSpacing(10);
        hboxLayout.setStyle("-fx-background-color: #81BEF7;");

        hboxLayout.getChildren().addAll(btnButton1, btnButton2, btnButton3);

        Scene scene = new Scene(hboxLayout);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

    public static void main(String[]args) {
        Application.launch(args);
    }
}
```

Output:



Create new instance of HBox layout and setup appearance settings

Change background color

Add nodes to the layout

Add layout to the scene

Add the scene to the stage

Example: VBox layout

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Button btnButton1 = new Button("Button1");
        Button btnButton2 = new Button("Button2");
        Button btnButton3 = new Button("Button3");

        VBox vboxLayout = new VBox();
        vboxLayout.setPadding(new Insets(15, 12, 15, 12));
        vboxLayout.setSpacing(10);
        vboxLayout.setStyle("-fx-background-color: #CC2EFA;");

        vboxLayout.getChildren().addAll(btnButton1, btnButton2, btnButton3);

        Scene scene = new Scene(vboxLayout);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

    public static void main(String[]args) {
        Application.launch(args);
    }
}
```

VBox instead of HBox

Output:

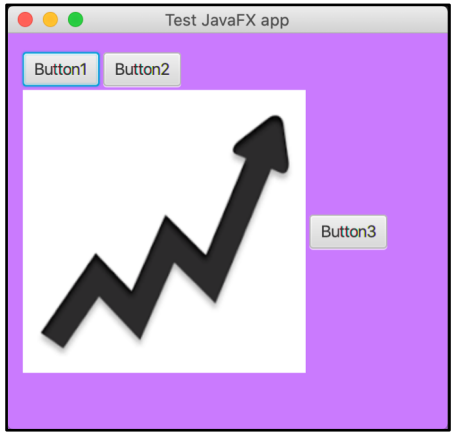
Example: FlowPane layout

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.scene.image.*;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Button btnButton1 = new Button("Button1");
        Button btnButton2 = new Button("Button2");
        Button btnButton3 = new Button("Button3");
        ImageView imgImage1 = new ImageView(
            new Image(Test.class.getResourceAsStream(
            "./my_app_icon.png")));

        FlowPane flowPaneLayout = new FlowPane();
        flowPaneLayout.setPadding(new Insets(15, 12, 15, 12));
        flowPaneLayout.setPrefWrapLength(170);
        flowPaneLayout.setVgap(3);
        flowPaneLayout.setHgap(3);
        flowPaneLayout.setStyle("-fx-background-color: #BE81F7;");

        flowPaneLayout.getChildren().addAll(btnButton1, btnButton2, imgImage1, btnButton3);

        Scene scene = new Scene(flowPaneLayout);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

    public static void main(String[]args) {
        Application.launch(args);
    }
}
```

Need to import this package in order to work with images

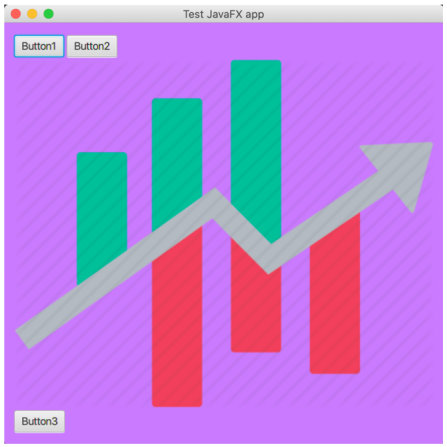
Create new instance of FlowPane layout and setup appearance settings

Output:



Example: FlowPane layout (cont'd)

The positions of nodes change based on their size:



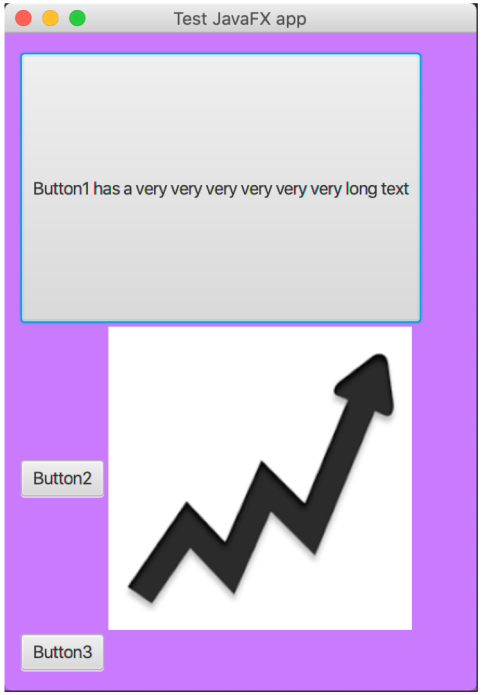
A differently sized image causes a different position of Button3

We now have a much bigger button so position of the nodes changes again

```
Button btnButton1 = new Button("Button1 has a very very very very very very very long text");
btnButton1.setMinWidth(100);
btnButton1.setMinHeight(200);
//btnButton1.setWrapText(true);
Button btnButton2 = new Button("Button2");
Button btnButton3 = new Button("Button3");
ImageView imgImage1 = new ImageView(
    new Image(Test.class.getResourceAsStream(
    "./my_app_icon.png")));
```

We could wrap the text of the button if we wanted

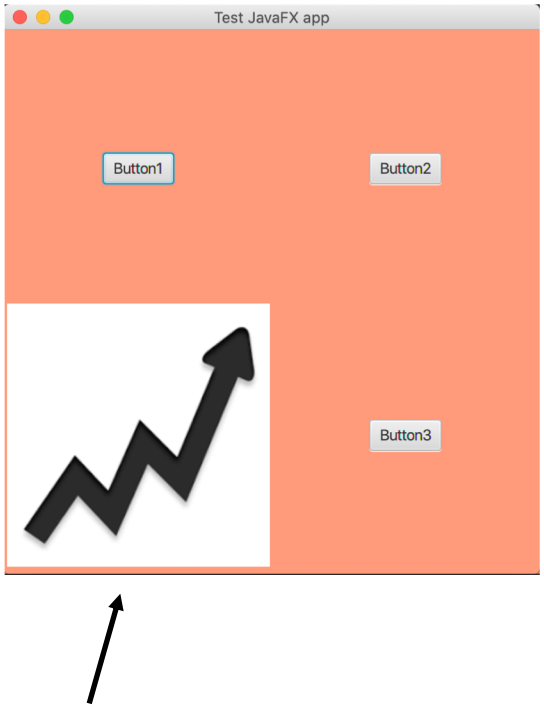
Output:



Example: TilePane layout

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.scene.image.*;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Button btnButton1 = new Button("Button1");
        Button btnButton2 = new Button("Button2");
        Button btnButton3 = new Button("Button3");
        ImageView imgImage1 = new ImageView(
            new Image(Test.class.getResourceAsStream(
            "./my_app_icon.png")));

        TilePane tileLayout = new TilePane();
        tileLayout.setPadding(new Insets(6, 3, 6, 3));
        tileLayout.setVgap(3);
        tileLayout.setHgap(3);
        tileLayout.setPrefColumns(2);
        tileLayout.setStyle("-fx-background-color: #F79F81;");

        tileLayout.getChildren().addAll(btnButton1, btnButton2, imgImage1, btnButton3);

        Scene scene = new Scene(tileLayout);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

    public static void main(String[]args) {
        Application.launch(args);
    }
}
```

Output:



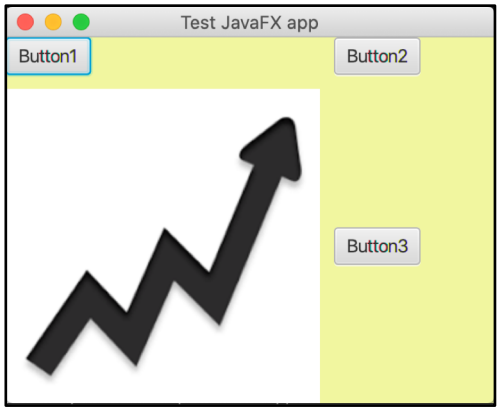
Nodes are arranged in the grid

Example: GridPane layout (adding rows)

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.scene.image.*;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Button btnButton1 = new Button("Button1");
        Button btnButton2 = new Button("Button2");
        Button btnButton3 = new Button("Button3");
        ImageView imgImage1 = new ImageView(
            new Image(Test.class.getResourceAsStream(
            "./my_app_icon.png")));


        GridPane gridLayout = new GridPane();
        gridLayout.setVgap(10);
        gridLayout.setHgap(10);

        gridLayout.addRow(0, btnButton1, btnButton2);
        gridLayout.addRow(1, imgImage1, btnButton3);
        gridLayout.setMinSize(350, 250);
        gridLayout.setStyle("-fx-background-color: #F2F5A9;");

        Scene scene = new Scene(gridLayout);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

    public static void main(String[]args) {
        Application.launch(args);
    }
}
```

Add nodes in rows

Output:

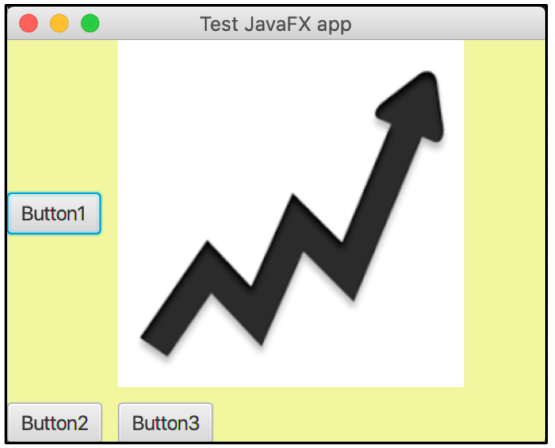


Example: GridPane layout (adding columns)

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.scene.image.*;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Button btnButton1 = new Button("Button1");
        Button btnButton2 = new Button("Button2");
        Button btnButton3 = new Button("Button3");
        ImageView imgImage1 = new ImageView(
            new Image(Test.class.getResourceAsStream(
            "./my_app_icon.png")));


        GridPane gridLayout = new GridPane();
        gridLayout.setVgap(10);
        gridLayout.setHgap(10);

        gridLayout.addColumn(0, btnButton1, btnButton2);
        gridLayout.addColumn(1, imgImage1, btnButton3);
        gridLayout.setMinSize(350, 250);
        gridLayout.setStyle("-fx-background-color: #F2F5A9;");

        Scene scene = new Scene(gridLayout);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

    public static void main(String[]args) {
        Application.launch(args);
    }
}
```

Add nodes in columns

Output:

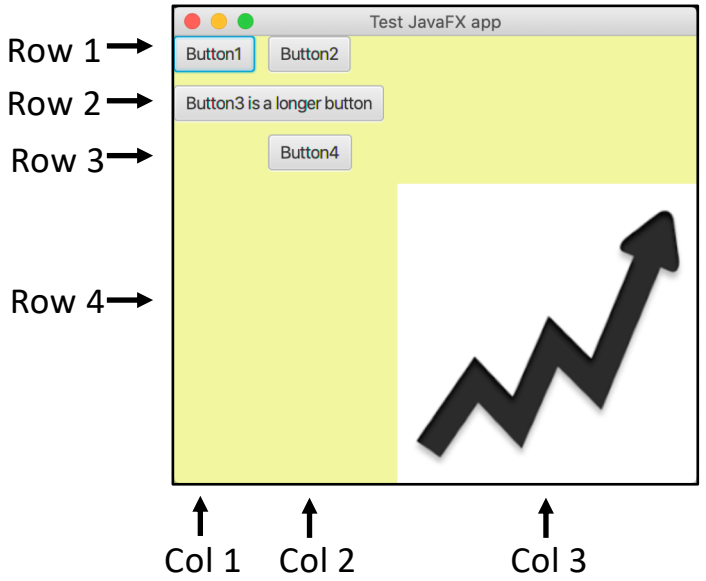


Example: GridPane layout (adding individual nodes)

```java
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.scene.image.*;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Button btnButton1 = new Button("Button1");
        Button btnButton2 = new Button("Button2");
        Button btnButton3 = new Button("Button3 is a longer button");
        Button btnButton4 = new Button("Button4");
        ImageView imgImage1 = new ImageView(
            new Image(Test.class.getResourceAsStream(
            "./my_app_icon.png")));


        GridPane gridLayout = new GridPane();
        gridLayout.setVgap(10);
        gridLayout.setHgap(10);

        gridLayout.add(btnButton1, 0, 0, 1, 1);
        gridLayout.add(btnButton2, 1, 0, 1, 1);
        gridLayout.add(btnButton3, 0, 1, 2, 1);
        gridLayout.add(btnButton4, 1, 2, 1, 1);
        gridLayout.add(imgImage1, 2, 3, 1, 1);
        gridLayout.setMinSize(350, 250);
        gridLayout.setStyle("-fx-background-color: #F2F5A9;");

        Scene scene = new Scene(gridLayout);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

    public static void main(String[]args) {
        Application.launch(args);
    }
}
```

Add nodes in columns

Output:



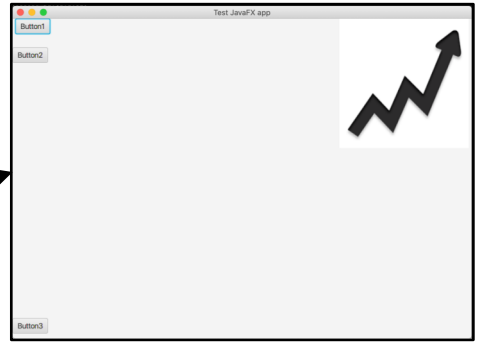
From Java API:

```java
public void add(Node child,
                int columnIndex,
                int rowIndex,
                int colspan,
                int rowspan)
```

Example: AnchorPane layout

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.scene.image.*;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Button btnButton1 = new Button("Button1");
        Button btnButton2 = new Button("Button2");
        Button btnButton3 = new Button("Button3");
        ImageView imgImage1 = new ImageView(
            new Image(Test.class.getResourceAsStream(
            "./my_app_icon.png")));

        AnchorPane anchorPaneLayout = new AnchorPane();
        anchorPaneLayout.getChildren().addAll(btnButton1, btnButton2, imgImage1, btnButton3);
        AnchorPane.setLeftAnchor(btnButton1, 5.0);
        AnchorPane.setTopAnchor(btnButton2, 50.0);
        AnchorPane.setRightAnchor(imgImage1, 5.0);
        AnchorPane.setBottomAnchor(btnButton3, 8.0);

        Scene scene = new Scene(anchorPaneLayout);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

    public static void main(String[]args) {
        Application.launch(args);
    }
}
```

Specify how far from the border to anchor the node



Nodes move as the window is resized; without anchoring nodes would stay in place while the window gets bigger