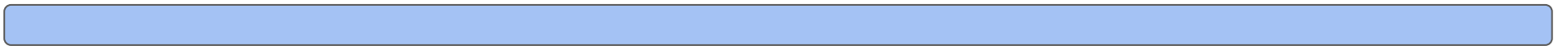


GUI intro



Java grafinės bibliotekos

- AWT (Abstract Window Toolkit)
- Swing

Nuorodos

<https://docs.oracle.com/javase/tutorial/deployment/applet/>

<http://www.realapplets.com/tutorial/>

https://www.tutorialspoint.com/java/java_applet_basics.htm

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

AWT:

https://www.tutorialspoint.com/awt/awt_quick_guide.htm

Apletų privalumai

- Nereikalinga instaliacija kliento kompiuteryje
- Nepriklauso nuo platformos

Klasių hierarchijos pavyzdys

java.lang.Object

java.awt.Component

java.awt.Container

java.awt.Panel

java.applet.Applet

javax.swing.JApplet

Paprasciausias appletas

```
package lt.codeacademy.first;

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

public class Applet1 extends JApplet {
    public void init() {
        getContentPane().add(new JLabel("Apletas"));
    }
}
```

Eclipse:
Run As -> 1 Java Applet



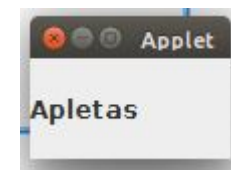
Apleto vykdymas naršyklėje ***

```
1<html>
2  <head>
3    <title>Tinklapio vardas</title>
4  </head>
5  <body>
6    <applet code=lt.codeadademy.first.Applet1 width=100
    height=100>
7      Naršyklė nepalaiko apletų!
8    </applet>
9</body>
10</html>
```

Vykdymas iš komandinės eilutės

```
public static void main(String[] args) {  
    Applet1 applet = new Applet1();  
    JFrame frame = new JFrame("Applet1");  
    frame.setDefaultCloseOperation(JFrame.EXIT_O  
N_CLOSE);  
    frame.getContentPane().add(applet);  
    frame.setSize(100, 50);  
    applet.init();  
    applet.start();  
    frame.setVisible(true);  
}
```

Eclipse:
Run As -> 1 Java App



Apletų archyvavimas** - deprecated

```
> javac It/codeacademy/first/Button2.java
```

```
> jar cf archyvas.jar It/codeacademy/first/Button2.class
```

Html žymę applet papildyti atributu `archive=archyvas.jar`

```
<html>
<head>
<title>Tinklapis vardas</title>
</head>
<body>
```

```
    <APPLET CODE=Button2 CODEBASE="lt.codeacademy.first" WIDTH=300
        HEIGHT=150 archive="archyvas.jar"> Jusu narsykle nepalaiko Java.
</APPLET>
```

```
</body>
</html>
```

Klasės JApplet metodai

- `init()`
- `start()`
- `stop()`
- `destroy()`

Life Cycle of an Applet

Four methods in the Applet class gives you the framework on which you build any applet –

- **init** – This method is intended for whatever initialization is needed for your applet. It is called after the param tags inside the applet tag have been processed.
- **start** – This method is automatically called after the browser calls the init method. It is also called whenever the user returns to the page containing the applet after having gone off to other pages.
- **stop** – This method is automatically called when the user moves off the page on which the applet sits. It can, therefore, be called repeatedly in the same applet.
- **destroy** – This method is only called when the browser shuts down normally. Because applets are meant to live on an HTML page, you should not normally leave resources behind after a user leaves the page that contains the applet.
- **paint** – Invoked immediately after the start() method, and also any time the applet needs to repaint itself in the browser. The paint() method is actually inherited from the java.awt.

```
Shapes *paint* g.draw*
```

```
import java.applet.*;
```

```
import java.awt.*;
```

```
public class Shapes extends Applet{
```

```
    int x=300,y=100,r=50;
```

```
    public void paint(Graphics g) {
```

```
        g.drawLine(30,300,200,10);
```

```
        g.drawOval(x-r,y-r,100,100);
```

```
        g.drawRect(400,50,200,100);
```

```
    }
```

```
}
```

Mygtuko talpinimas

```
package lt.codeadademy.first;

import java.awt.Container;
import java.awt.FlowLayout;

import javax.swing.JApplet;
import javax.swing.JButton;

public class Button1 extends JApplet {
    private JButton b1 = new JButton("Button 1");
    private JButton b2 = new JButton("Button 2");

    public void init() {
        Container cp = getContentPane();
        cp.setLayout(new FlowLayout());
        cp.add(b1);
        cp.add(b2);
    }
}
```

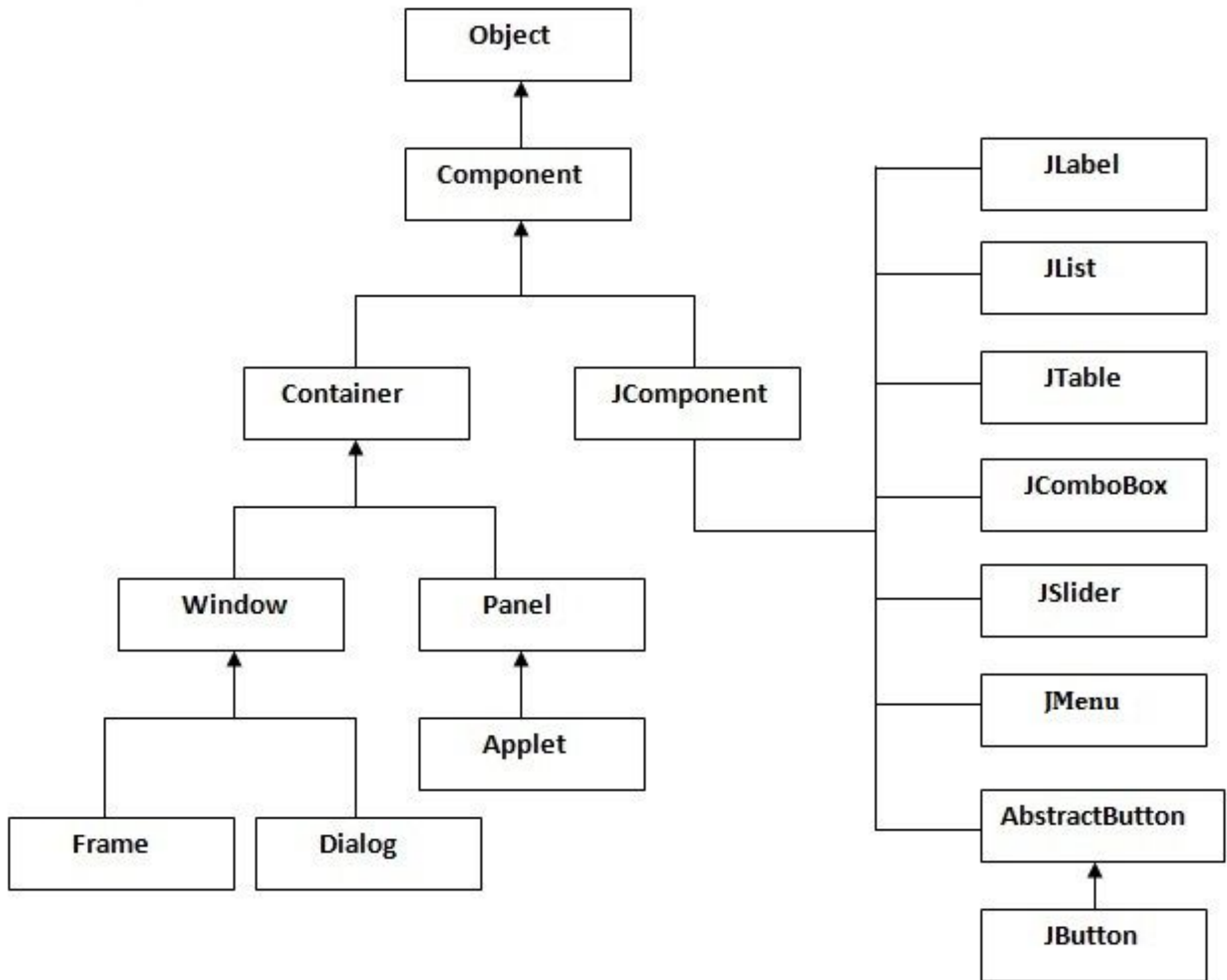
Įvykiai

Anoniminė vidinė klasė
įvykių gaudymui:

```
class ButtonListener  
implements ActionListener
```

```
package It.codeacademy.first;  
import java.awt.*;  
import java.awt.event.*;  
import javax.swing.*;  
public class Button2 extends JApplet {  
    private JButton b1 = new JButton("Button 1");  
    private JButton b2 = new JButton("Button 2");  
    private JTextField txt = new JTextField(10);  
    class ButtonListener implements ActionListener {  
        public void actionPerformed(ActionEvent e) {  
            String name = ((JButton) e.getSource()).getText();  
            txt.setText(name);  
        }  
    }  
    private ButtonListener bli = new ButtonListener();  
    public void init() {  
        b1.addActionListener(bli);  
        b2.addActionListener(bli);  
        Container cp = getContentPane();  
        cp.setLayout(new FlowLayout());  
        cp.add(b1);  
        cp.add(b2);  
        cp.add(txt);  
    }  
}
```

swing



JOptionPane.show*

```
package It.codeacademy.first;

import javax.swing.JOptionPane;

public class DialogsTest {

    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null, "Labas Vakaras!!!");
    }

}
```

JOptionPane.showInputDialog(

```
import javax.swing.JOptionPane;
```

```
public class DialogsTest {
```

```
    public static void main(String[] args) {  
        JOptionPane.showMessageDialog(null, "Labas Vakaras!!!");
```

```
        String a = JOptionPane.showInputDialog("Iveks a:");  
        System.out.println(a);  
        JOptionPane.showMessageDialog(null, a);
```

```
    }
```

```
}
```

U1

Mini skaičiuotuvai:

paprašyti vartotojo įvesti du skaičius

Programa apskaičiuoja jų sumą.

PS jei įvesta reikšmė negali būti paversta į skaičių, programa turi paprašyti dar kartą įvesti reikšmę.

PS

```
JOptionPane.showMessageDialog(null, "Labas Vakaras!!!");
```

```
String a = JOptionPane.showInputDialog("Ivesk a:");
```

PVZ Dialogs

```
package lt.codeacademy.first;

import javax.swing.JOptionPane;

public class Dialogs {

    public static void main(String[] args) {
        String a = JOptionPane.showInputDialog("Pirma reikšmė?");
        String b = JOptionPane.showInputDialog("Antra reikšmė?");

        int intA = Integer.parseInt(a);
        int intB = Integer.parseInt(b);

        JOptionPane.showMessageDialog(null, "a+b=" + (a + b));
        JOptionPane.showMessageDialog(null, "a+b=" + (intA + intB));
    }
}
```

Simplest GUI programming:

JOptionPane

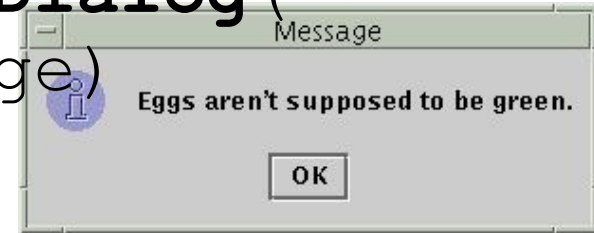
An option pane is a simple dialog box for graphical input/output

- advantages:
 - simple
 - flexible (in some ways)
 - looks better
- disadvantages:
 - created with static methods;
not very object-oriented
 - not very powerful (just simple dialog boxes)



Types of JOptionPane

- `public static void showMessageDialog(
Component parent, Object message)
Displays a message on a dialog
with an OK button.`
- `public static int showConfirmDialog(
Component parent, Object message)
Displays a message and list of
choices Yes, No, Cancel`
- `public static String showInputDialog(
Component parent, Object message)
Displays a message and text
field for input, and returns the
value entered as a String.`



JOptionPane examples 1

- `showMessageDialog` analogous to `System.out.println` for displaying a simple message

```
import javax.swing.*;
```

```
class MessageDialogExample {  
    public static void main(String[] args) {  
        JOptionPane.showMessageDialog(null,  
            "How's the weather?");  
        JOptionPane.showMessageDialog(null,  
            "Second message");  
    }  
}
```

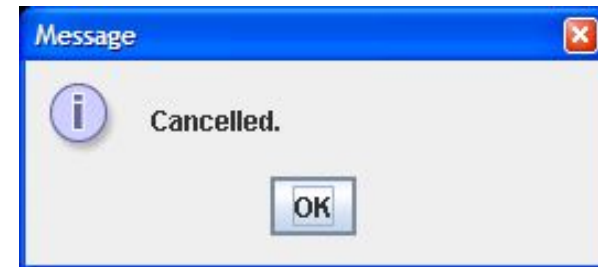
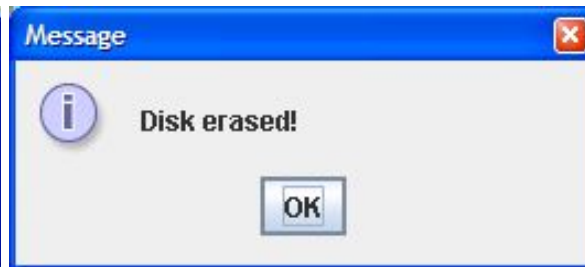


JOptionPane examples 2

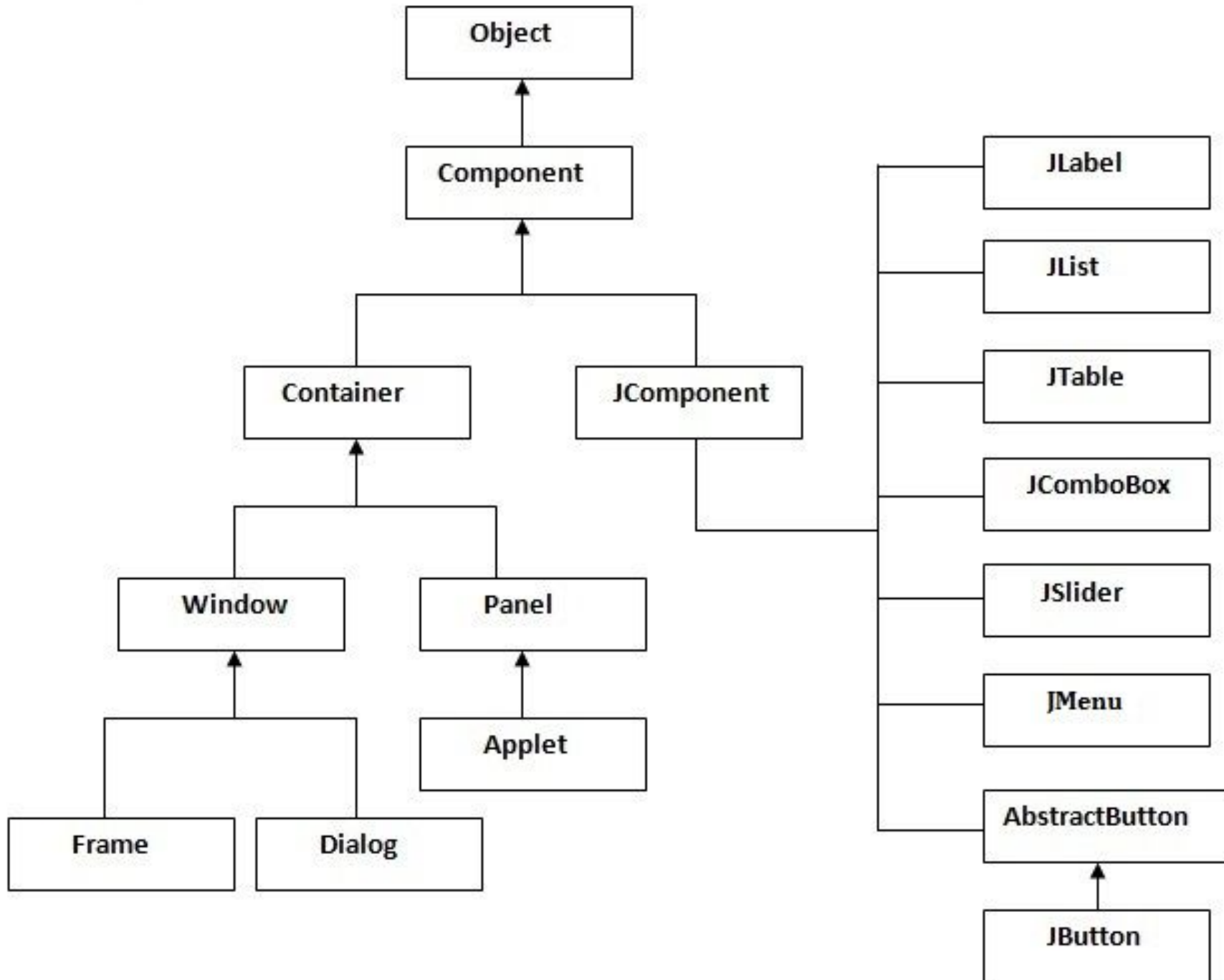
- `showConfirmDialog` analogous to a `System.out.print` that prints a question, then reading an input value from the user (can only be one of the provided choices)

```
import javax.swing.*;

class ConfirmDialogExample {
    public static void main(String[] args) {
        int choice = JOptionPane.showConfirmDialog(null,
            "Erase your hard disk?");
        if (choice == JOptionPane.YES_OPTION) {
            JOptionPane.showMessageDialog(null, "Disk erased!");
        } else {
            JOptionPane.showMessageDialog(null, "Cancelled.");
        }
    }
}
```



Hierarchy of Java Swing classes



Onscreen GUI elements

- **windows:** actual first-class citizens of desktop;
also called top-level containers
examples: frame, dialog box
- **components:** GUI widgets
examples: button, text box, label
- **containers:** logical grouping for components
example: panel



Java GUI: AWT and Swing

- Sun's initial idea: create a set of classes/methods that can be used to write a multi-platform GUI (Abstract Windowing Toolkit, or **AWT**)
 - problem: not powerful enough; limited; a bit clunky to use
- Second edition (JDK v1.2): **Swing**
 - a newer library written from the ground up that allows much more powerful graphics and GUI construction
- Drawback: Both exist in Java now; easy to get them mixed up;

Swing component hierarchy

```
java.lang.Object
  +-- java.awt.Component
    +-- java.awt.Container
      |
      +-- javax.swing.JComponent
        |
        +-- javax.swing.JButton
        +-- javax.swing.JLabel
        +-- javax.swing.JMenuBar
        +-- javax.swing.JOptionPane
        +-- javax.swing.JPanel
        +-- javax.swing.JTextArea
        +-- javax.swing.JTextField
      +-- java.awt.Window
        +-- java.awt.Frame
          +-- javax.swing.JFrame
```

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

Commonly used Methods of Component class

Method	Description
<code>public void add(Component c)</code>	add a component on another component.
<code>public void setSize(int width,int height)</code>	sets size of the component.
<code>public void setLayout(LayoutManager m)</code>	sets the layout manager for the component.
<code>public void setVisible(boolean b)</code>	sets the visibility of the component. It is by default false.

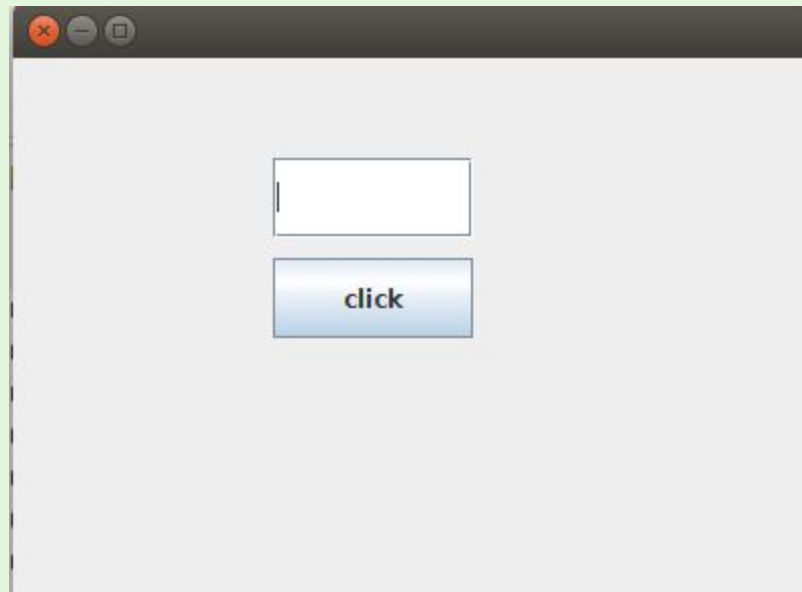
Simple Java Swing Example

```
import javax.swing.*;
```

```
public class FirstSwingExample {  
    public static void main(String[] args) {  
        JFrame f = new JFrame();// creating instance of JFrame  
  
        JButton b = new JButton("click");// creating instance of JButton  
1      b.setBounds(130, 100, 100, 40);// x axis, y axis, width, height  
  
        f.add(b);// adding button in JFrame  
  
2      f.setSize(400, 500);// 400 width and 500 height  
3      f.setLayout(null);// using no layout managers  
        f.setVisible(true);// making the frame visible  
    }  
}
```

JTextField Test

Formoje patalpinti mygtuką ir teksto lauką:



```
JTextField t = new JTextField();
```

`t.setText("Veikia");` - nurodome ką išvesti

`t.getText();` - nuskaityme kas įvesta

Mygtuko talpinimas

```
package lt.codeadademy.first;

import java.awt.Container;
import java.awt.FlowLayout;

import javax.swing.JApplet;
import javax.swing.JButton;

public class Button1 extends JApplet {
    private JButton b1 = new JButton("Button 1");
    private JButton b2 = new JButton("Button 2");

    public void init() {
        Container cp = getContentPane();
        cp.setLayout(new FlowLayout());
        cp.add(b1);
        cp.add(b2);
    }
}
```


Įvykiai

Anoniminė vidinė klasė
įvykių gaudymui:

```
class ButtonListener  
implements ActionListener
```

```
package lt.codeacademy.first;  
import java.awt.*;  
import java.awt.event.*;  
import javax.swing.*;  
public class Button2 extends JApplet {  
    private JButton b1 = new JButton("Button 1");  
    private JButton b2 = new JButton("Button 2");  
    private JTextField txt = new JTextField(10);  
    class ButtonListener implements ActionListener {  
        public void actionPerformed(ActionEvent e) {  
            String name = ((JButton) e.getSource()).getText();  
            txt.setText(name);  
        }  
    }  
    private ButtonListener bli = new ButtonListener();  
    public void init() {  
        b1.addActionListener(bli);  
        b2.addActionListener(bli);  
        Container cp = getContentPane();  
        cp.setLayout(new FlowLayout());  
        cp.add(b1);  
        cp.add(b2);  
        cp.add(txt);  
    }  
}
```

<http://www.javatpoint.com/java-swing>

trumpai

JPanel

`JPanel` is a Swing's lightweight container which is used to group a set of components together. `JPanel` is a pretty simple component which, normally, does not have a GUI (except when it is being set an opaque background or has a visual border).

In this article, we summarize the common practices when working with `JPanel` in Swing. At the end, we will create a sample program looks like this:

<http://www.codejava.net/java-se/swing/jpanel-basic-tutorial-and-examples>

JFrame



A frame is a graphical window that can be used to hold other components

- `public JFrame()` or `public JFrame(String title)`
Creates a frame with an optional title.
- `public void setTitle(String text)`
Puts the given text in the frame's title bar.
- `public void setDefaultCloseOperation(int op)`
Makes the frame perform the given action when it closes. Common value:
`JFrame.EXIT_ON_CLOSE`
- `public void add(Component comp)`
Places the given component or container inside the frame.
 - *How would we add more than one component to the frame?*
- `public void pack()`
Resizes the frame to fit the components inside it.
- **NOTE:** Call `setVisible(true)` to make a frame appear on the screen after creating it.

JButton, JLabel

*The most common component—
a button is a clickable onscreen
region that the user interacts with
to perform a single command*



*A text label is simply a string of text
displayed on screen in a graphical
program. Labels often give infor-
mation or describe other components*



- `public JButton(String text)`
`public JLabel(String text)`
Creates a new button / label with the given string as its text.
- `public String getText()`
Returns the text showing on the button / label.
- `public void setText(String text)`
Sets button / label's text to be the given string.

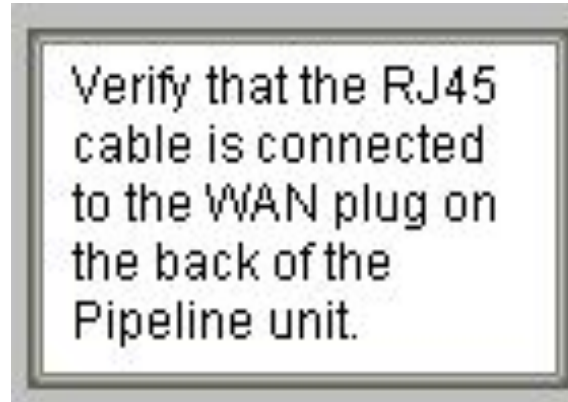
JTextField, JTextArea

A text field is like a label, except that the text in it can be edited and modified by the user. Text fields are commonly used for user input, where the user types information in the field and the program reads it

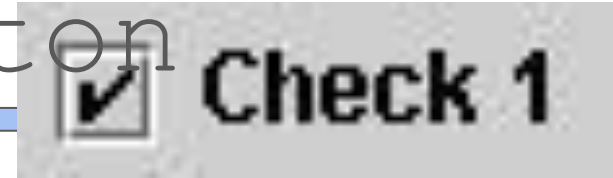


A text area is a multi-line text field

- `public JTextField(int columns)`
- `public JTextArea(int lines, int columns)`
Creates a new text field that is the given number of columns (letters) wide.
- `public String getText()`
Returns the text currently in the field.
- `public void setText(String text)`
Sets field's text to be the given string.



JCheckBox, JRadioButton



A check box is a toggleable button with two states: checked and unchecked

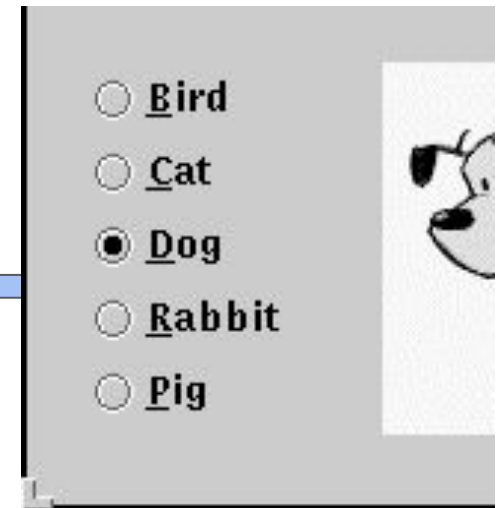


A radio button is a button that can be selected; usually part of a group of mutually-exclusive radio buttons (1 selectable at a time)

- `public JCheckBox / JRadioButton(String text)`
`public JCheckBox(String text, boolean isChecked)`
Creates checked/unchecked check box with given text.
- `public boolean isSelected()`
Returns true if check box is checked.
- `public void setSelected(boolean selected)`
Sets box to be checked/unchecked.

ButtonGroup

A logical group of radio buttons that ensures that only one is selected at a time



- `public ButtonGroup()`
- `public void add(JRadioButton button)`
- The `ButtonGroup` is not a graphical component, just a logical group; the `RadioButtons` themselves are added to the container, not the `ButtonGroup`

Icon/ImageIcon



Allows you to put a picture on a button, label or other component

- `public class ImageIcon implements Icon`
 - `public ImageIcon(String filename)`
 - `public ImageIcon(URL address)`
- **in** `JButton`, `JRadioButton`, `JCheckBox`, `JLabel`, etc...
 - **constructor that takes an** `Icon`
 - `public void setIcon(Icon)`
 - `public void setSelectedIcon(Icon)`
 - `public void setRolloverIcon(Icon)`

JScrollPane



A special container that holds a component, using scrollbars to allow that component to be seen

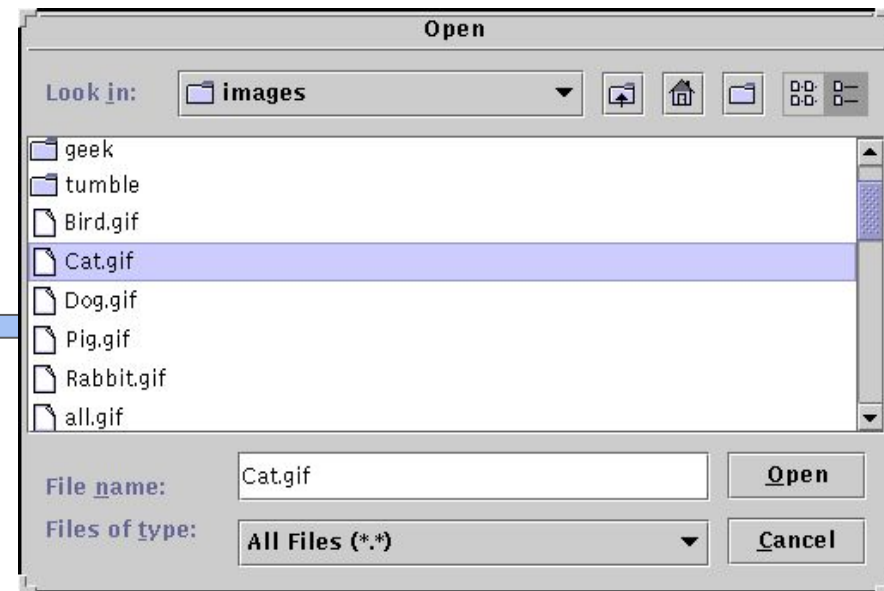
- `public JScrollPane(Component comp)`
Wraps the given component with scrollbars.

After constructing the scroll pane, add the scroll pane to the container, not the original component.

```
contentPane.add(new JScrollPane(textarea),  
                BorderLayout.CENTER);
```

JFileChooser

A special dialog box that allows the user to select one or more files/folders

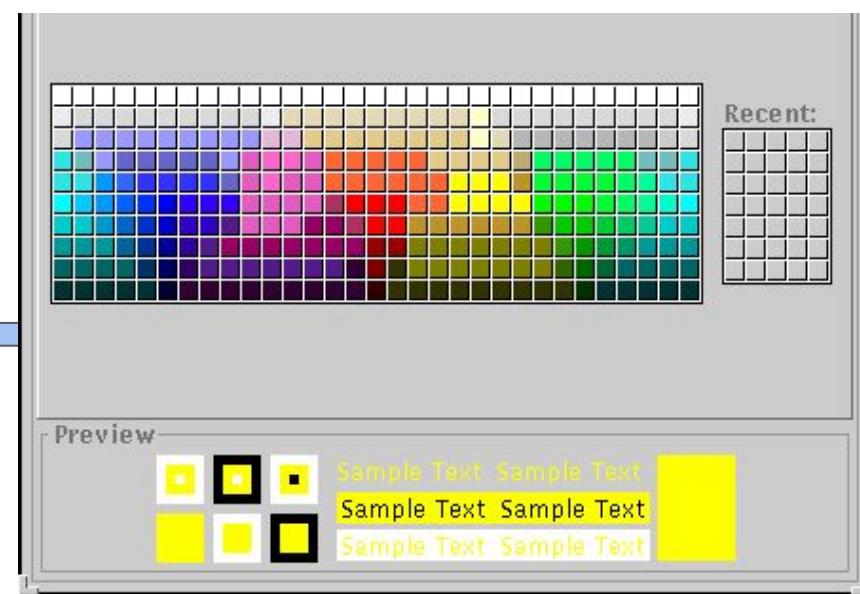


- `public JFileChooser()`
- `public JFileChooser(String currentDir)`
- `public int showOpenDialog(Component parent)`
- `public int showSaveDialog(Component parent)`
- `public File getSelectedFile()`
- `public static int APPROVE_OPTION, CANCEL_OPTION`
Possible result values from showXxxDialog(..).

```
JFileChooser chooser = new JFileChooser();
int result = chooser.showSaveDialog(this);
if (result == JFileChooser.APPROVE_OPTION)
    this.saveData(chooser.getSelectedFile().getName());
```

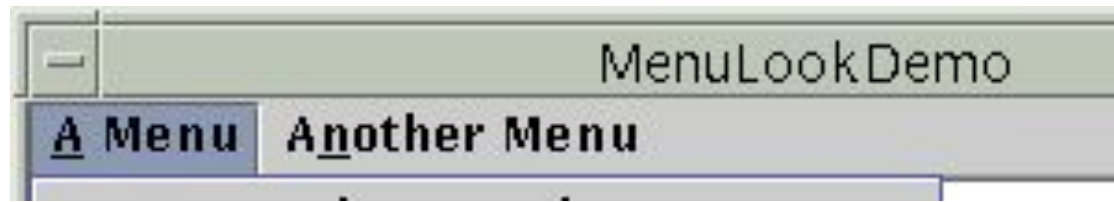
JColorChooser

Another special dialog that lets the user pick from a palette of colors



- `public JColorChooser()`
- `public JColorChooser(Color initial)`
- `public Color showDialog(Component parent, String title, Color initialColor)`
 - returns null if user chose Cancel option

JMenuBar



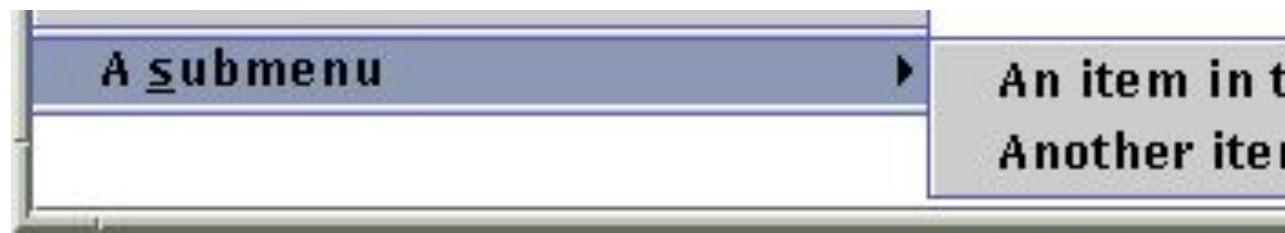
The top-level container that holds menus; can be attached to a frame

- `public JMenuBar()`
- `public void add(JMenu menu)`

Usage: in `JFrame`, the following method exists:

- `public void setJMenuBar(JMenuBar bar)`

JMenu



A menu to hold menu items; menus can contain other menus (Composite)

- `public JMenu(String text)`
- `public void add(JMenuItem item)`
- `public void addSeparator()`
- `public void setMnemonic(int mnemonic)`



JMenuItem

A text-only menu item

Alt+1



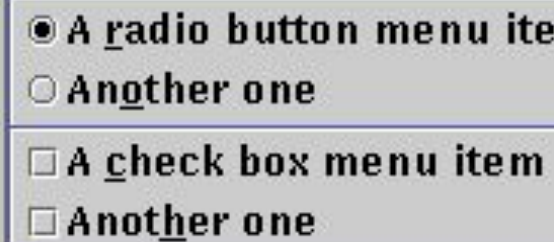
Both text and icon

*An entry in a frame's Menu bar, which
can be clicked to perform commands*

- `public JMenuItem(String text)`
- `public JMenuItem(String text, Icon icon)`
- `public JMenuItem(String text, int mnemonic)`

- `public void addActionListener(
 ActionListener al)`
- `public void setAccelerator(KeyStroke ks)`
- `public void setEnabled(boolean b)`
- `public void setMnemonic(int mnemonic)`

JCheckBoxMenuItem / JRadioButtonMenuItem



Radio button and checkbox-like menu items

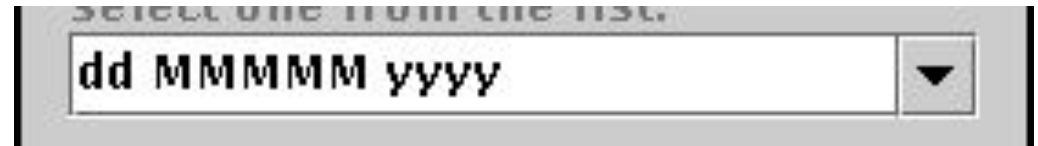
- `public J_____MenuItem(String text)`
- `public J_____MenuItem(String text, boolean select)`
- `public J_____MenuItem(String text, Icon icon)`
- `public J_____MenuItem(String text, Icon icon, boolean selected)`
- `public void addActionListener(ActionListener al)`
- `public boolean isSelected()`
- `public void setSelected(boolean b)`

Recall: in a `ButtonGroup`, the following method exists:

- `public void add(AbstractButton button)`

These two classes extend `AbstractButton`!

JComboBox



- `public JComboBox()`
- `public JComboBox(Vector items)`
- `public JComboBox(ComboBoxModel model)`

Constructs a combo box. Can optionally pass a vector or model of items. (See `DefaultComboBoxModel` for a model implementation.)

- `public void addActionListener(
 ActionListener al)`

Causes an action event to be sent to listener `al` when the user selects or types a new item in the

JComboBox: Managing Items

- `public void addItem(Object item)`
- `public Object getItemAt(int index)`
- `public void removeAllItems()`
- `public void removeItem(Object item)`
- `public void removeItemAt(int index)`

JComboBox: Selected Item

- `public int getSelectedIndex()`
- `public Object getSelectedItem()`
- `public void setSelectedItem(Object item)`
- `public void setSelectedIndex(int index)`
- `public void setEnabled(boolean enabled)`
- `public void setEditable(boolean editable)`

If editable, the user can type new arbitrary values into the combo box.

JComboBox Code Example

```
final JComboBox box = new JComboBox();
box.addItem("Marty");    box.addItem("Tom");
box.addItem("Jessica");
box.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent
event) {
        JOptionPane.showMessageDialog(null,
            "You chose " + box.getSelectedItem());
    }
});
getContentPane().add(box,
    BorderLayout.NORTH);
```

JTabbedPane



A container that can hold many "tab" subcontainers, each with components in it

- `public JTabbedPane()`
- `public JTabbedPane(int tabAlignment)`
Constructs a new tabbed pane. Defaults to having the tabs on top; can be set to `JTabbedPane.BOTTOM`, `JTabbedPane.LEFT`, `JTabbedPane.RIGHT`, etc.
- `public void addTab(String title, Component comp)`
- `public void addTab(String title, Icon icon, Component comp)`
- `public void addTab(String title, Icon icon, Component comp, String tooltip)`
Adds the given component as a tab in this tabbed pane. Can optionally use an icon and/or tool tip.

JTabbedPane methods

- `public void insertTab(String title, Icon icon, Component comp, String tooltip, int index)`
- `public void remove(Component comp)`
- `public void remove(int index)`
- `public void removeAll()`
- `public void setSelectedComponent(Component c)`
- `public void setSelectedIndex(int index)`

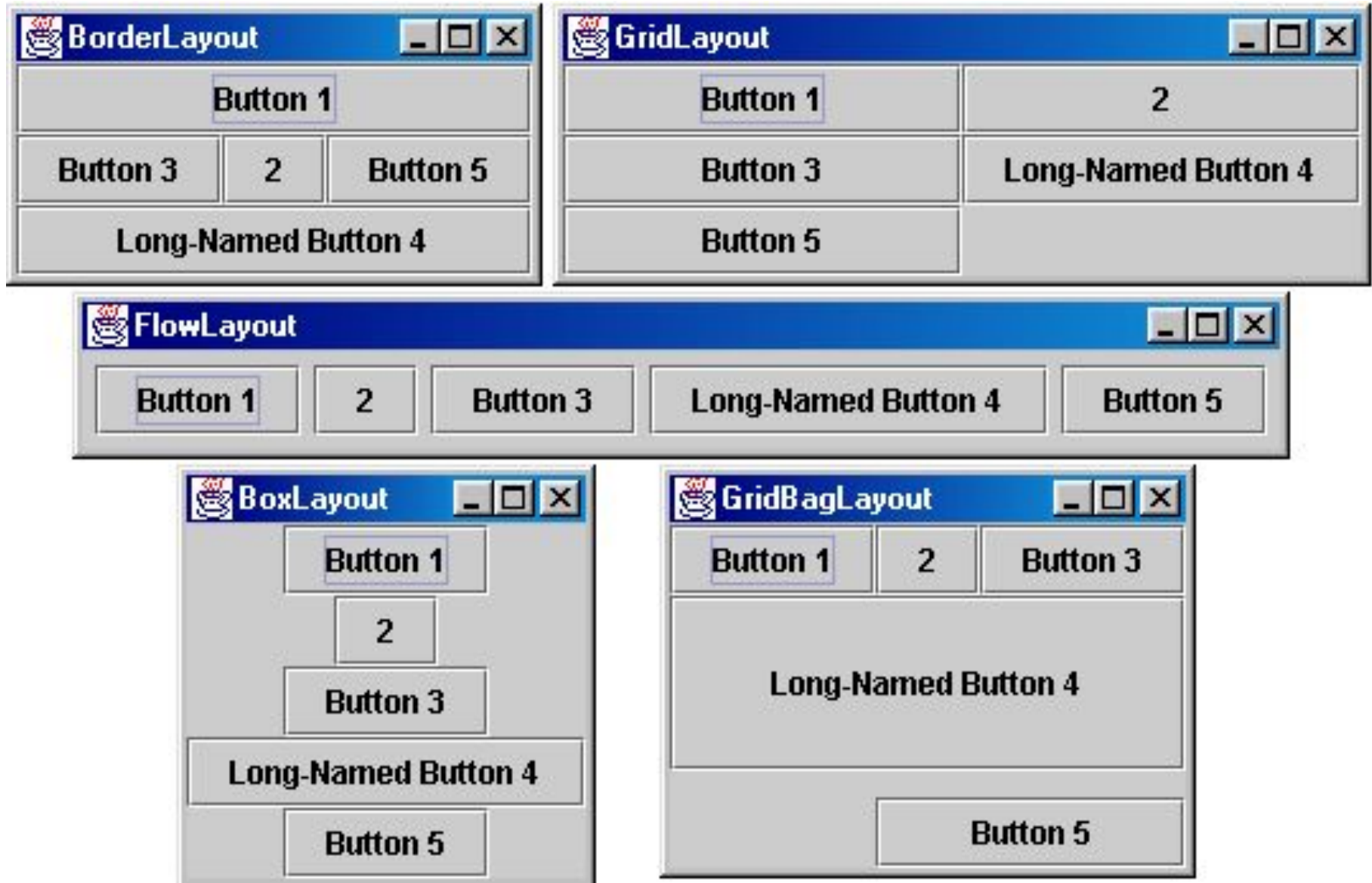
Problem: positioning, resizing

How does the programmer specify where each component sits in the window, how big each component should be, and what the component should do if the window is resized/moved/maximized/etc?

- **Absolute positioning** (C++, C#, others):
Specify exact pixel coordinates for every component
- **Layout managers** (Java):
Have special objects that decide where to position each component based on some criteria
 - What are benefits or drawbacks to each approach?

Containers with layout

- The idea: Place many components into a special component called a **container**, then add the container to the JFrame



Container

container: *an object that holds components; it also governs their positions, sizes, and resize behavior*

- `public void add(Component comp)`
`public void add(Component comp, Object info)`
Adds a component to the container, possibly giving extra information about where to place it.
- `public void remove(Component comp)`
Removes the given component from the container.
- `public void setLayout(LayoutManager mgr)`
Uses the given layout manager to position the components in the container.
- `public void validate()`
You should call this if you change the contents of a container that is already on the screen, to make it re-do

JPanel

A panel is our container of choice; it is a subclass of Container, so it inherits the methods from the previous slide and defines these additional methods (among others):

- `public JPanel()`
Constructs a panel with a default flow layout.
- `public JPanel(LayoutManager mgr)`
Constructs a panel that uses the given layout manager.

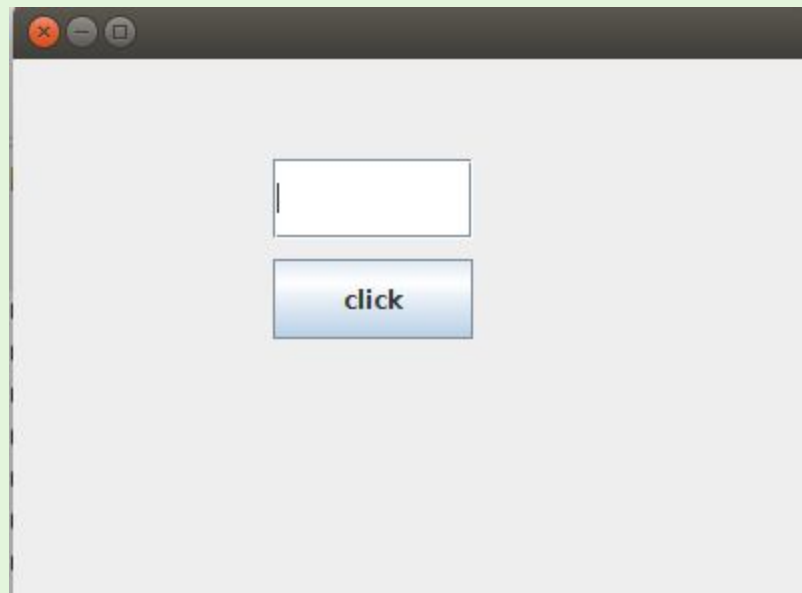
Preferred size of components

- Swing component objects each have a certain size they would "like" to be--just large enough to fit their contents (text, icons, etc.)
- This is called the *preferred size* of the component
- Some types of layout managers (e.g. `FlowLayout`) choose to size the components inside them to the preferred size; others (e.g. `BorderLayout`, `GridLayout`) disregard the preferred size and use some other scheme



U3 JTextField Test

Formoje patalpinti mygtuką ir teksto lauką:



```
JTextField t = new JTextField();
```

`t.setText("Veikia");` - nurodome ką išvesti

`t.getText();` - nuskaityme kas įvesta

JTextField

```
package It.codeacademy.first;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JTextField;

public class JTextFieldTest {

    public static void main(String[] args) {
        JFrame f = new JFrame();// creating instance of JFrame

        JButton b = new JButton("click");// creating instance of JButton
        b.setBounds(130, 100, 100, 40);// x axis, y axis, width, height

        f.add(b);// adding button in JFrame

        JTextField t = new JTextField();
        t.setBounds(130, 50, 100, 40);
        f.add(t);

        f.setSize(400, 500);// 400 width and 500 height
        f.setLayout(null);// using no layout managers
        f.setVisible(true);// making the frame visible
    }
}
```

JButton

public void **addActionListener**(ActionListener a): is used to add the action listener to this object.

```
public interface ActionListener extends EventListener {
```

```
/**
```

```
 * Invoked when an action occurs.
```

```
*/
```

```
public void actionPerformed(ActionEvent e);
```

```
}
```


U4 actionPerformed

Sukurit klasę kuri implementuoja `ActionListener` interfeisą. Perrašyti `actionPerformed` metodą. Metode išvesti tekstini pranešimą į konsolę.

1. Apsirašyti naują klasę
2. Sukurti tos klasės objektą
3. Užduoties U2 mygtukui iškviešti **`addActionListener`** metodą su katik sukurtu objektu
4. Patestuoti.

MyListener

```
class MyListener implements ActionListener {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        System.out.println("Veikia");  
    }  
}
```

JTextField

```
package It.codeacademy.first;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JTextField;

public class JTextFieldTest {

    public static void main(String[] args) {
        JFrame f = new JFrame();// creating instance of JFrame

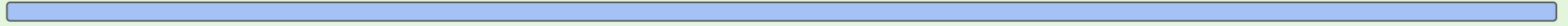
        JButton b = new JButton("click");// creating instance of JButton
        b.setBounds(130, 100, 100, 40);// x axis, y axis, width, height

        f.add(b);// adding button in JFrame

        b.addActionListener(new MyListener());

        f.setSize(400, 500);// 400 width and 500 height
        f.setLayout(null);// using no layout managers
        f.setVisible(true);// making the frame visible
    }
}
```

U5 bevardé



Re-factor your code to use Anonymous ActionListener class

Swing įvykių modelis

- Komponentai gali generuoti įvykius
- Kiekvieno tipo įvykiui atstovauja skirtinga klasė, pvz., KeyEvent
- Įvykius gaudo ir apdoroja listener klasės (klausytojai), kurios realizuoja atitinkamus listener interfeisus, pvz., KeyListener
- Įvykius generuojantis komponentas ir įvykius apdorojanti listener klasė susiejami kviečiant komponento metodą `addXXXListener(XXXListener)`,
- kur XXX – įvykio tipas

Dar kartą apie anoniminės vidinės klasės panaudojimą įvykiams gaudyti

```
1 JButton b = new JButton("Button1");
2 b.addActionListener(new ActionListener() {
3     public void actionPerformed(ActionEvent e) {
4         //veiksmas
5     }
6 });
```

Swing įvykių modelis

- Visi Swing komponentai turi atitinkamus `addXXXListener()` ir `removeXXXListener()` metodus
- Konkrečias įvykių/klausytojų/metodų
 - tipų reikšmes žr. kitoje skaidrėje
- Galima kurti savo įvykius/klausytojus
-

Įvykių/klausytojų/metodų tipai ir juos palaikantys komponentai

Action	JButton, JList, JTextField, JMenuItem (JMenu, ...)
Adjustment	Adjustable interface (JScrollbar)
Component	Component (JButton, JCheckBox, Container, JPanel, JApplet, JLabel, JTextArea, JTextField, ...)
Container	Container (JPanel, JApplet, Window, JDialog, JFrame, ...)
Focus	Component (...)
Key	Component (...)

Įvykių/klausytojų/metodų tipai ir juos palaikantys komponentai

Mouse	Component (...)
MouseMotion	Component (...)
Window	Window (JDialog, JFrame, ...)
Item	ItemSelectable interface (JCheckBox, JCheckBoxMenuItem, JComboBox, JList)
Text	JTextComponent (JTextArea, JTextField)

Listener interfeisų metodai

ActionListener	actionPerformed(ActionEvent)
AdjustmentListener	adjustmentValueChanged(AdjustmentEvent)
ComponentListener ComponentAdapter	componentHidden(ComponentEvent) componentShown(ComponentEvent) componentMoved(ComponentEvent) componentResized(ComponentEvent)
ContainerListener ContainerAdapter	componentAdded(ContainerEvent) componentRemoved(ContainerEvent)
FocusListener FocusAdapter	focusGained(FocusEvent) focusLost(FocusEvent)
ItemListener	itemStateChanged(ItemEvent)

Listener interfeisų metodai

KeyListener KeyAdapter	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)
MouseListener MouseAdapter	mouseClicked(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mousePressed(MouseEvent) mouseReleased(MouseEvent)
MouseMotionListener MouseMotionAdapter	mouseDragged(MouseEvent) mouseMoved(MouseEvent)
WindowListener WindowAdapter	windowOpened(WindowEvent) windowActivated(WindowEvent) windowIconified(WindowEvent) ...

U6

setText(String text) ActionListener

Mygtuko paspaudimas turi išvesti atsitiktinį skaičių



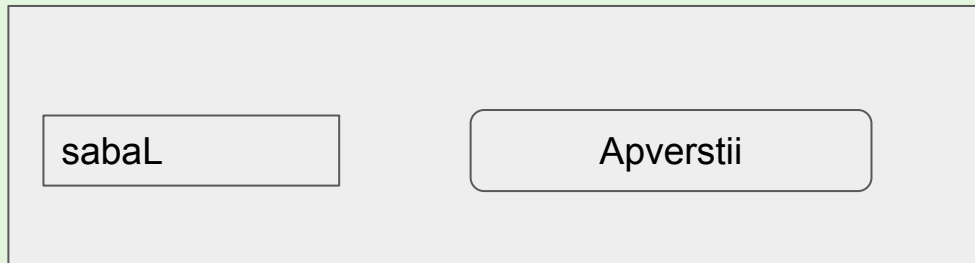
`TextField pirmas = new TextField(5);`

`@param columns` the number of columns to use to calculate the preferred width; if columns is set to zero, the preferred width will be whatever naturally results from the component implementation

U7 setText(String text)

1. Vartotojas įveda tekstą į lauką
2. Paspaudžiamas mygtikas
3. Teksto laukelyje atsiranda tekstas atvirkščia tvarka.

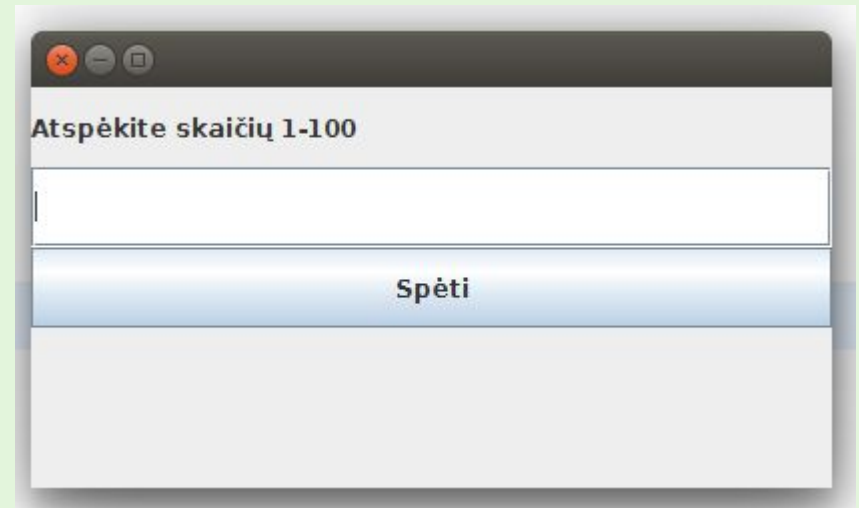
t.setText("Veikia"); - nurodome ką išvesti
t.getText(); - nuskaityme kas įvesta



The image shows a graphical user interface (GUI) window with a light purple background. Inside the window, there are two elements: a text input field on the left containing the text "sabaL" and a button on the right with the text "Apverstii".

U8 Žaidimas: šilta-šalta

**Program sugeneruoja atsitiktinį skaičių,
Vartotojas bando jį atspėti. Jei spėjamas
skačius yra mažesnis parodo raudona spalva,
jei didesnis - mėlyna, jei atspėja - geltona**



Žaidimas: šilta-šalta

```
class Zaidimas extends JFrame {
    JPanel panel = new JPanel();
    JTextField text = new JTextField();
    JLabel uzrasas = new JLabel("Atspėkite skaičių 1-100");
    Random r = new Random();
    int sk = r.nextInt(100) + 1;
    int kartai;
    Zaidimas() {
        add(panel);
        panel.setSize(new Dimension(400, 200));
        panel.setLayout(new GridLayout(5, 1));

        panel.add(uzrasas);
        panel.add(text);

        JButton bt = new JButton("Spėti");
        panel.add(bt);

        bt.addActionListener(new Spejimas());

        setBounds(200, 200, 400, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }
}
```

Žaidimas: šilta-šalta

```
class Spejimas implements ActionListener {
```

```
    @Override
```

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

```
        kartai++;
```

```
        int spejimas = Integer.parseInt(text.getText());
```

```
        if (sk == spejimas) {
```

```
            uzrasas.setText("Atspėjote");
```

```
            panel.setBackground(Color.YELLOW);
```

```
        } else {
```

```
            uzrasas.setText("Spėjimo kartai: " + kartai);
```

```
            if (sk > spejimas) {
```

```
                panel.setBackground(Color.RED);
```

```
            } else {
```

```
                panel.setBackground(Color.BLUE);
```

```
            }
```

```
        }
```

```
    }
```

```
}
```

```
}
```


BorderFactory

```
panele.setBorder(BorderFactory  
    .createTitledBorder("Paneles pavadinimas"));
```

```
class Pa extends JFrame {
```

```
    Pa() {
```

```
        JPanel panele = new JPanel();
```

```
        panele.setBorder(BorderFactory.createTitledBorder("Paneles pavadinimas"));
```

```
        getContentPane().add(panele); //add(panele);
```

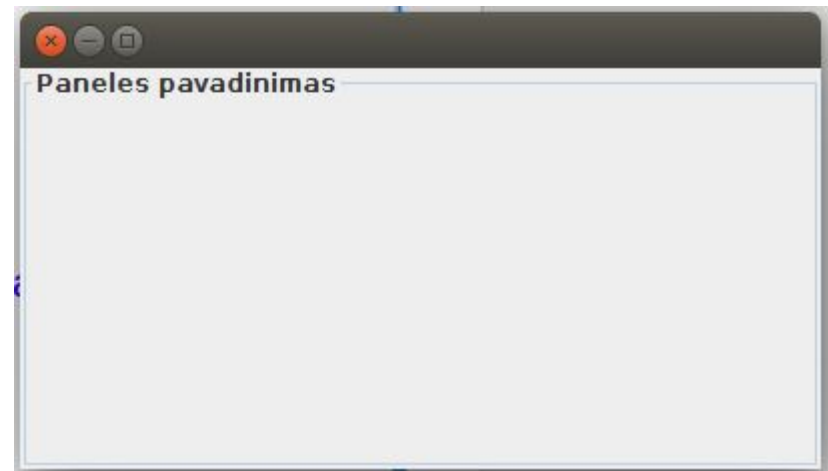
```
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        setBounds(200, 200, 400, 200);
```

```
        setVisible(true);
```

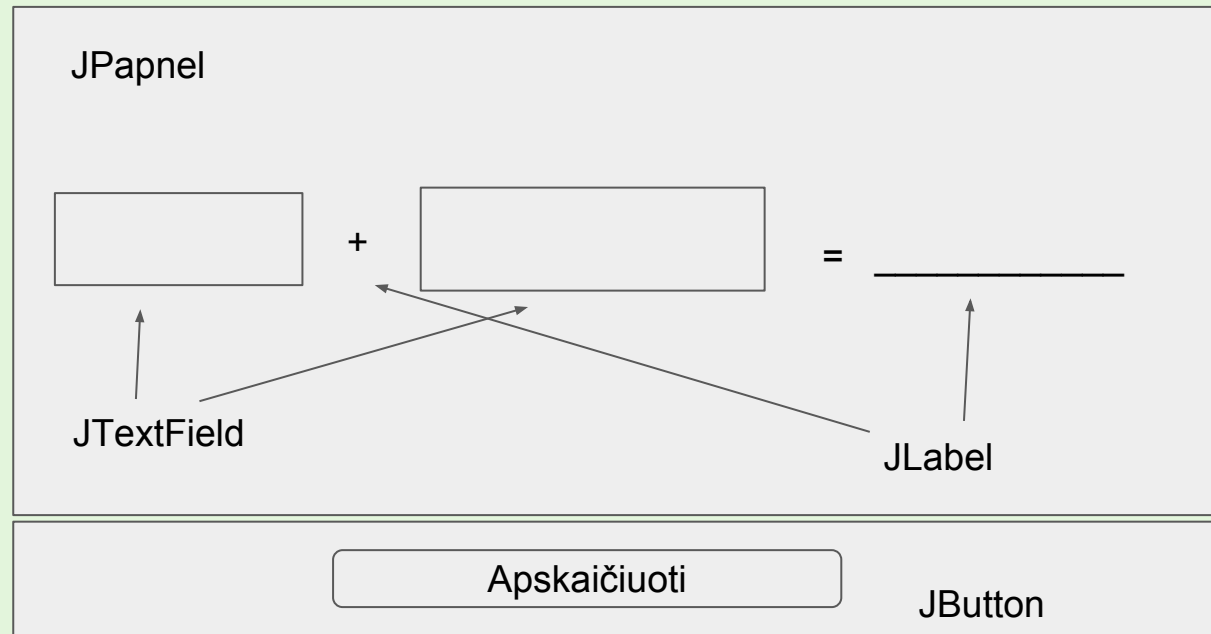
```
    }
```

```
}
```



U9 sveikų skaičių sumos skaičiuotuvas

JFrame ->



```
setLayout(new BorderLayout(getContentPane(), BorderLayout.Y_AXIS));  
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
setBounds(200, 200, 400, 200);  
setVisible(true);
```

-> eclipse

```
class CalcU7 extends JFrame {
    JTextField pirmas = new JTextField(5);
    JTextField antras = new JTextField(5);
    JLabel atsakymas = new JLabel(" = _____");
    CalcU7() {
        JPanel panele1 = new JPanel();
        panele1.setBorder(BorderFactory.createTitledBorder("Paneles pavadinimas"));
        panele1.add(pirmas);
        panele1.add(new JLabel(" + "));
        panele1.add(antras);
        panele1.add(atsakymas);

        JPanel panele2 = new JPanel();
        JButton myg = new JButton("Apskaičiuoti");
        panele2.add(myg);

        add(panele1);
        add(panele2);

        setLayout(new BoxLayout(getContentPane(), BoxLayout.Y_AXIS));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(200, 200, 400, 200);
        setVisible(true);
    }
}
```

```
myg.addActionListener(new ManoKlausytonas());
```

```
class ManoKlausytonas implements ActionListener {
```

```
    @Override
```

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

```
        atsakymas.setText(" = "
```

```
            + (Integer.parseInt(pirmas.getText())
```

```
                + Integer.parseInt(antras.getText())));
```

```
    }
```

```
}
```

ND

-
- Susikurkite apletą (klasę, paveldėtą iš JApplet). Metode init įdėkite tekstinio laukelio ir mygtuko sukūrimą bei talpinimą. Naudodami anoniminę vidinę klasę įdėkite įvykių gaudytoją. Tekstiniame laukelyje turi būti rodomas skaitliukas, kiek kartų buvo nuspaustas mygtukas