Chapter 7 Java and UI

javacose@qq.com

Xiang Zhang

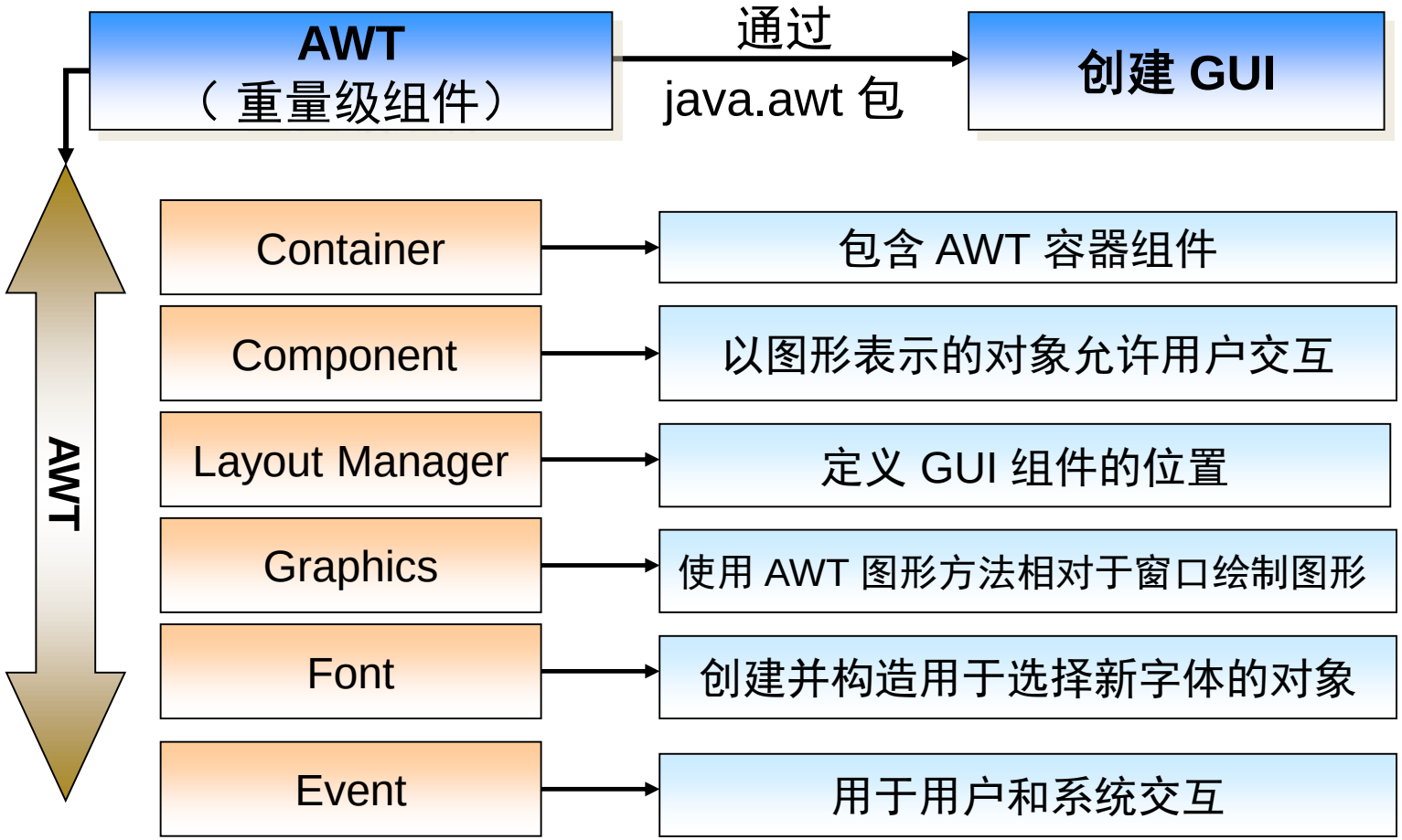
Content

- AWT and Swing Introduction
- Swing Container (JFrame, JPanel)
- Swing Components
- Layout Manager
- Event and Event-based Programming
- Menu

AWT Introduction

- Abstract Window Tookit
- Basic UI component of Java
- Early Technology of Java
  - Limited component
  - Different appearance in different platform
  - No pop-up menu, scrolling pane
  - No clipboard, print ability, keyboard navigation…
- java.awt

AWT Introduction



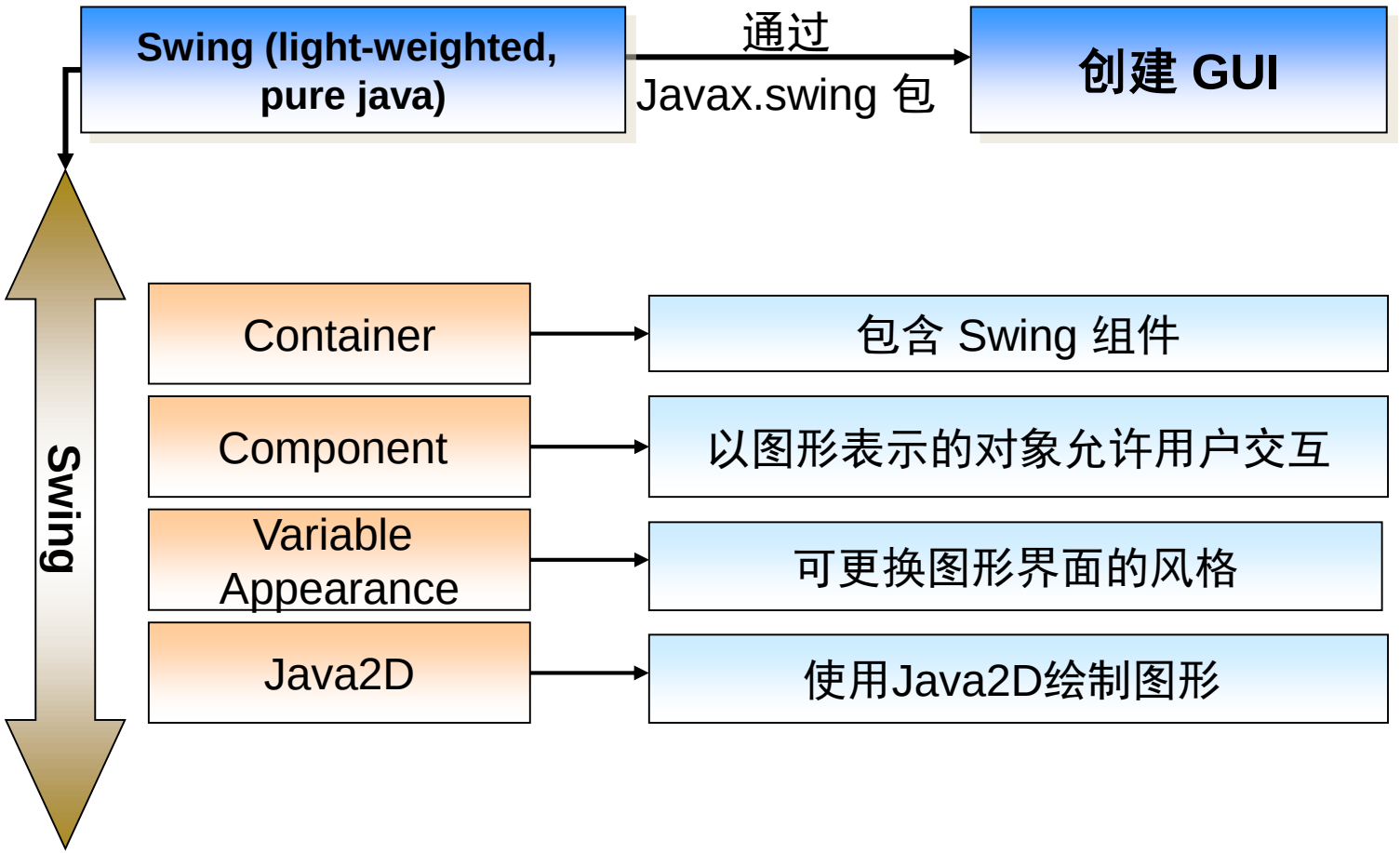
AWT Introduction

- Lessons of AWT
  - Not fully featured (a very short development cycle)
  - Heavy-weighted components
  - Native Interface 原生界面
  - For AWT: GCD(Greatest Common Divisor) principle applied
  - For Swing, LCD(Lowest Common Denominator) applied
  - IBM: SWT

Swing Introduction

- Overcome AWTs Shortage
  - Pure Java
  - Swing package is based on AWT
  - Swing is slower than AWT
- javax.swing

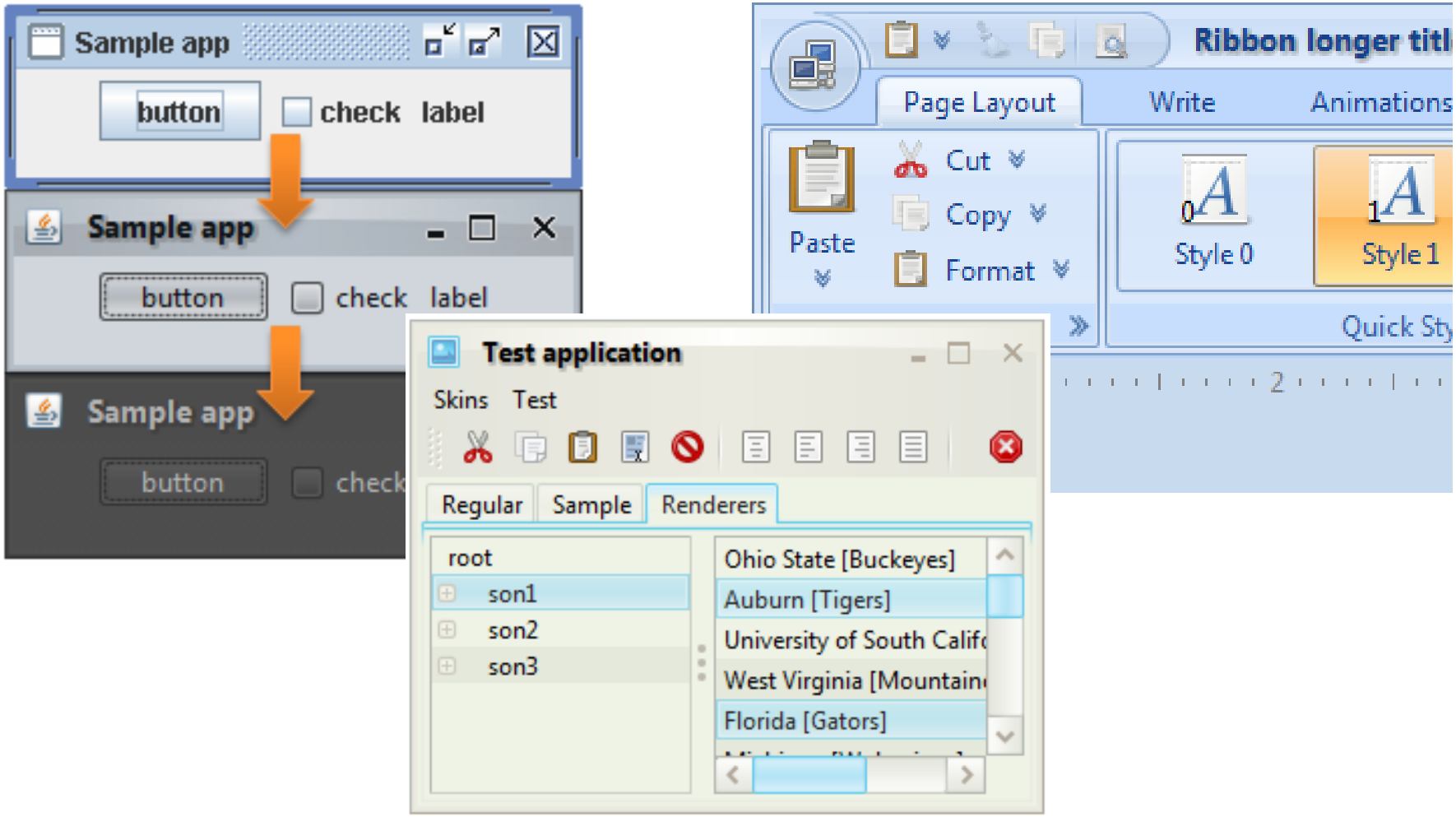
Swing Introduction



AWT vs. Swing

	AWT	Swing
Devoloper	Sun JDK	Sun JDK
Native	Yes	No
Implemen-tation	Heavy-weighted ; GCD ; Invoke OS Component	Light-weighted ; Top-level container invoke OS component; most component is in pure java
Portablity	Appearance and Behavior depend on OS	Independent with OS
Speed	Fast	Slow before Jdk1.4, but faster now
Component	No abundant	Abundant
Visual Development	No	Jbuilder , Netbeans , Eclipse VE

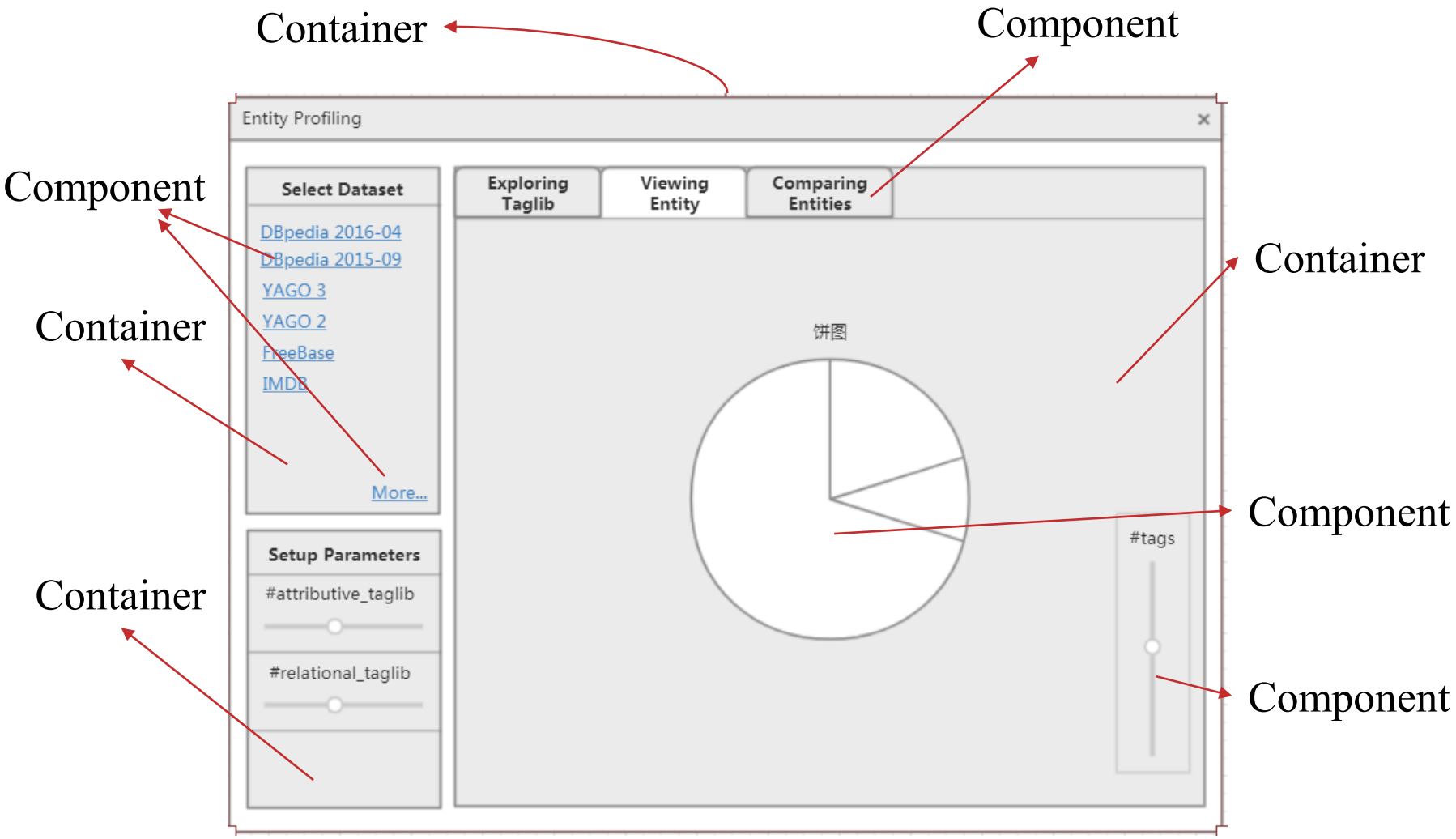
Swing is NOT Out!



Self-study

- AWT / Swing / SWT / JFace Comparation
- Substance / JIDE: Swing Look&Feel
- Reference
  - 《SWT/JFace in Action》
  - 《Eclipse in Action》

The Basic Idea of a UI



Swing Container

- Basic Steps to Create GUI using Swing:
  - Step 1: create a Frame (a window)

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

  - Step 2: create a Component (here is a button)

```
JButton button = new JButton("Click me");
```

Swing Container

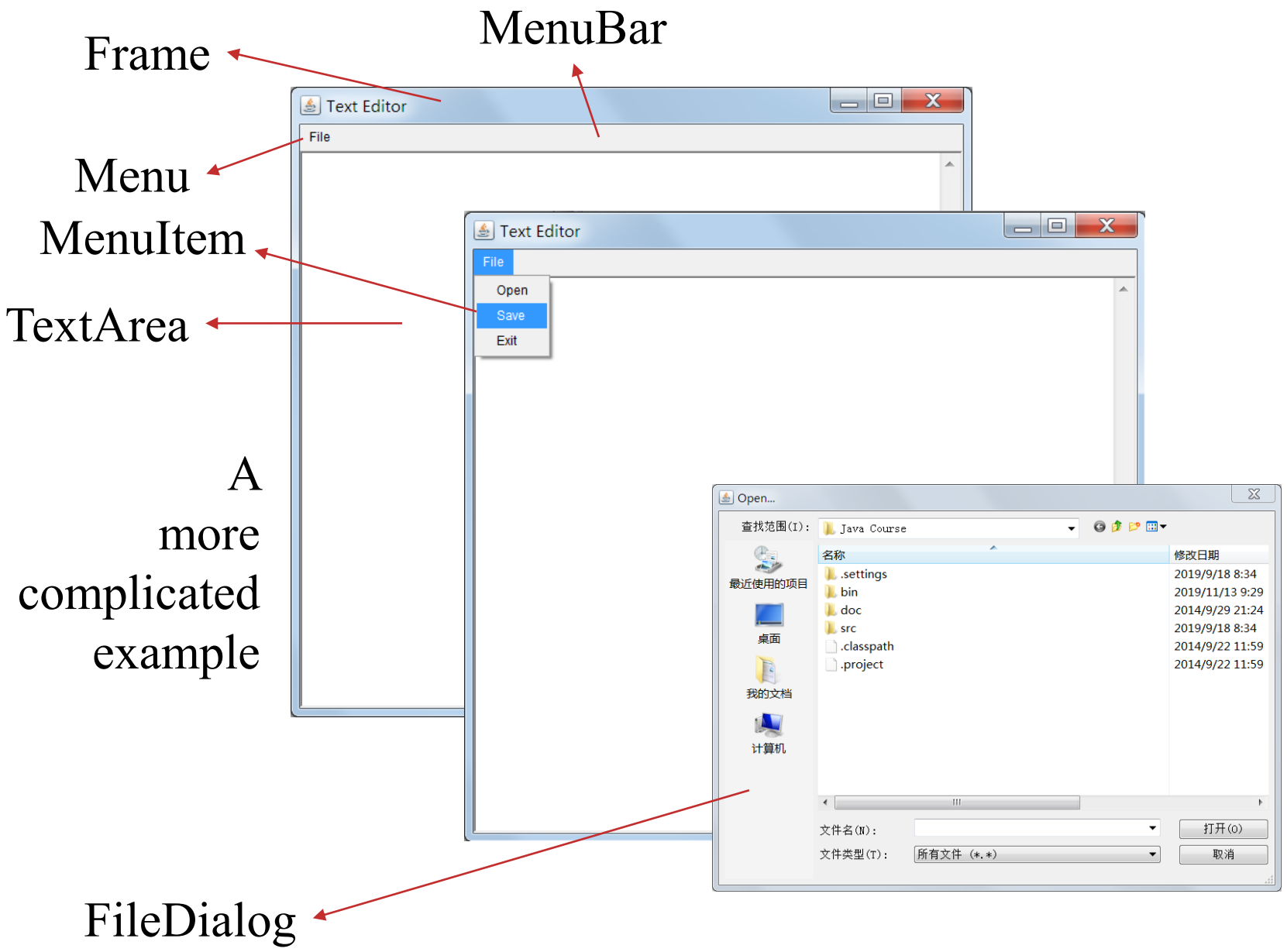
- Step 3: add the component into a Pane of Frame

```
frame.getContentPane().add(BorderLayout.EAST, button);
```

- Step 4: show the Frame (set its size and make it visible)

```
frame.setSize(300, 300);
frame.setVisible(true);
```

```
public static void main(String[] args) {
    JFrame frame = new JFrame();
    JButton button = new JButton("Click me");
    frame.getContentPane().add(BorderLayout.EAST, button);
    frame.setSize(300,300);
    frame.setVisible(true);
}
```


Frame
MenuBar
Menu
MenuItem
TextArea
A more complicated example
FileDialog
Text Editor
File
Text Editor
File
Open
Save
Exit
Open...
查找范围(I):
Java Course
名称
修改日期
.settings
2019/9/18 8:34
bin
2019/11/13 9:29
doc
2014/9/29 21:24
src
2019/9/18 8:34
.classpath
2014/9/22 11:59
.project
2014/9/22 11:59
最近使用的项目
桌面
我的文档
计算机
文件名(N):
文件类型(T):
所有文件 (*.*)
打开(O)
取消

Step1: import classes

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;

public class TextEditor {
    private Frame f;
    private MenuBar mb;
    private Menu m;
    private TextArea ta;
    private MenuItem openItem, saveItem, closeItem;
    private FileDialog openDia, saveDia;
    private File file;
```

Step2: declare containers and components

```
public TextEditor() {
    f = new Frame("Text Editor");
    f.setBounds(300, 100, 650, 600);
    f.setVisible(true);
    mb = new MenuBar();
    ta = new TextArea();
    m = new Menu("File");
    openItem = new MenuItem("Open");
    saveItem = new MenuItem("Save");
    closeItem = new MenuItem("Exit");
    m.add(openItem);
    m.add(saveItem);
    m.add(closeItem);
    mb.add(m);
    addEvent();
    openDia = new FileDialog(f, "Open...", FileDialog.LOAD);
    saveDia = new FileDialog(f, "Save.", FileDialog.SAVE);
    f.setMenuBar(mb);
    f.add(ta);
}
```

Step3: constructor (put components in containers)

Step4: add events to make it interactive

we created an anonymous class here!!!

```
private void addEvent() {
    openItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            openDia.setVisible(true);
            String dirPath = openDia.getDirectory();
            String fileName = openDia.getFile();
            if (dirPath == null || fileName == null)
                return;
            ta.setText("");
            file = new File(dirPath, fileName);
            try {
                BufferedReader bufr = new BufferedReader(new FileReader(file));
                String line = null;
                while ((line = bufr.readLine()) != null) {
                    ta.append(line + "\r\n");
                }
                bufr.close();
            } catch (Exception e1) {
                throw new RuntimeException("error reading");
            }
        }
    });
```

Step5: add the event if the menu item **Open** is selected

Step6: add the event if the menu item **Save** is selected

```
saveItem.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (file == null) {
            saveDia.setVisible(true);
            String dirPath = saveDia.getDirectory();
            String fileName = saveDia.getFile();
            if (dirPath == null || fileName == null)
                return;
            file = new File(dirPath, fileName);
        }
        BufferedWriter buf;
        try {
            buf = new BufferedWriter(new FileWriter(file));
            String text = ta.getText();
            buf.write(text);
            buf.flush();
            buf.close();
        } catch (IOException e1) {
            throw new RuntimeException("读取失败");
        }

    }
});
```

```
        closeItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    } // this is the closing brace of method addEvent
```

Step7: add the event if the menu item **Close** is selected, and close the method *addEvent*()

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

Step8: a very simple *main* method

Inner Class


```
public class OuterClass {

    int a = 5;
    static int b = 6;
    void show() {
        System.out.println("Outer Class");
    }

    public class InnerClass {
        void use() {
            System.out.println("a=" + a);
            System.out.println("b=" + a);
            show();
        }
    }

    public static void main(String[] args) {
        InnerClass inner = new OuterClass().new InnerClass();
        inner.use();
    }
}
```

They are visible to the InnerClass

An inner class

Static Member in Inner Class


```
public class InnerClass {
    int c = 10;
    static int d = 10;
    static final int e = 10;
    static void method1() {}
    void method2() {}
```

Non-static variable is allowed

static variable is not allowed

static constant is allowed

static method is not allowed

Non-static method is allowed

Using Inner Class From Outside


```
public class Application {
    public static void main() {
        OuterClass out = new OuterClass();
        OuterClass.InnerClass in = new OuterClass().new InnerClass();
    }
}
```

Private Inner Class

```
public class OuterClass {

    int a = 5;
    static int b = 6;
    void show() {
        System.out.println("Outer Class")
    }

    private class InnerClass {
        void use() {
            System.out.println("a=" + a);
            System.out.println("b=" + a);
            show();
        }
    }
}

public class Application {
    public static void main() {
        OuterClass out = new OuterClass();
        OuterClass.InnerClass in = new OuterClass().
                new InnerClass();
    }
}
```

Not visible to outside

Why We Need Inner Class

- Inner classes are used for logical grouping of classes.
- In cases where your class B is used only by class A, it's better to put Class B as an inner class to Class A.
- This improves encapsulation and readability of the code.

Four Types of Inner Class

- member inner class 成员内部类
- static inner class 静态内部类
- local inner class 局部内部类
- anonymous inner class 匿名内部类

Member Inner Class


```
public class OuterClass {

    int a = 5;
    static int b = 6;
    void show() {
        System.out.println("Outer Class");
    }

    public class InnerClass {
        void use() {
            System.out.println("a=" + a);
            System.out.println("b=" + a);
            show();
        }
    }

    public static void main(String[] args) {
        InnerClass inner = new OuterClass().new InnerClass();
        inner.use();
    }
}
```

as a member of OuterClass

Static Inner Class


```
public class OuterClass {

    static class InnerClass {
        void print() {
            System.out.println("Inner Class");
        }
    }

    public static void main(String[] args) {
        InnerClass inner = new OuterClass.InnerClass();
        inner.print();
    }
}
```

No need to create an object of OuterClass

Local Inner Class

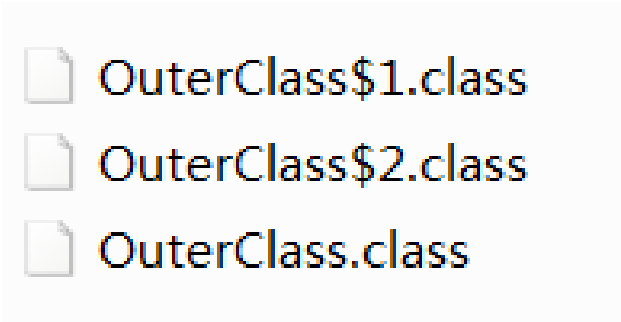

```
public class OuterClass {

    public void show() {
        class InnerClass {
            void print() {
                System.out.println("Inner Class");
            }
        }

        InnerClass in = new InnerClass();
        in.print();
    }
}
```

Only visible inside the show()

Anonymous Inner Class

```
public class OuterClass {

    ArrayList<String> list;
    public OuterClass(){
        list = new ArrayList<String>();
        list.add("a");list.add("c");list.add("b");list.add("d");
    }
    public void sortAscAndDes() {
        list.sort(new Comparator<String>() {
            public int compare(String arg0, String arg1) {
                return arg0.compareTo(arg1);
            }
        }); //implement Comparator interface in an anonymous class
        System.out.println(list.toString());
        list.sort(new Comparator<String>() {
            public int compare(String arg0, String arg1) {
                return -arg0.compareTo(arg1);
            }
        });
        System.out.println(list.toString());
    }
    public static void main(String[] args) {
        OuterClass out = new OuterClass();
        out.sortAscAndDes();
    }
}
```

Class File of Anonymous Class



Swing Container

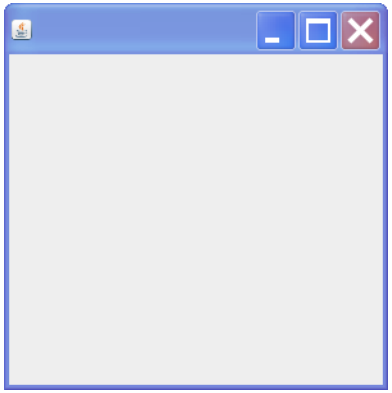
JButton

JComboBox

JCheckBox

add

Frame



Panel

Component

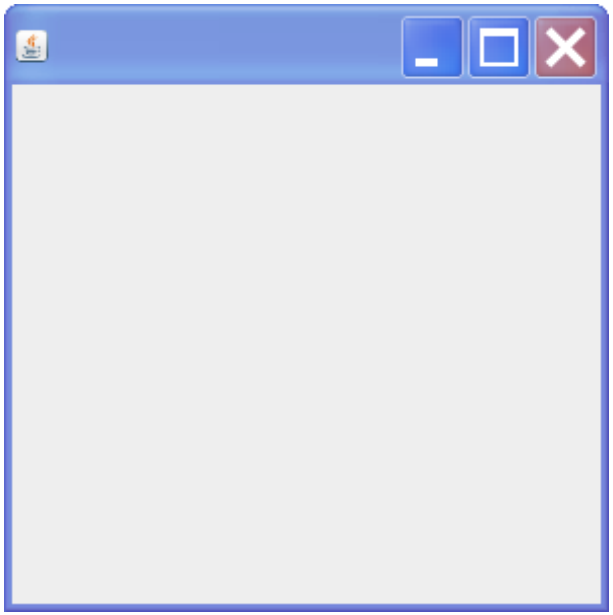
Container

Container

- JFrame
- JPanel
- JSplitPane
- JScrollPane

JFrame

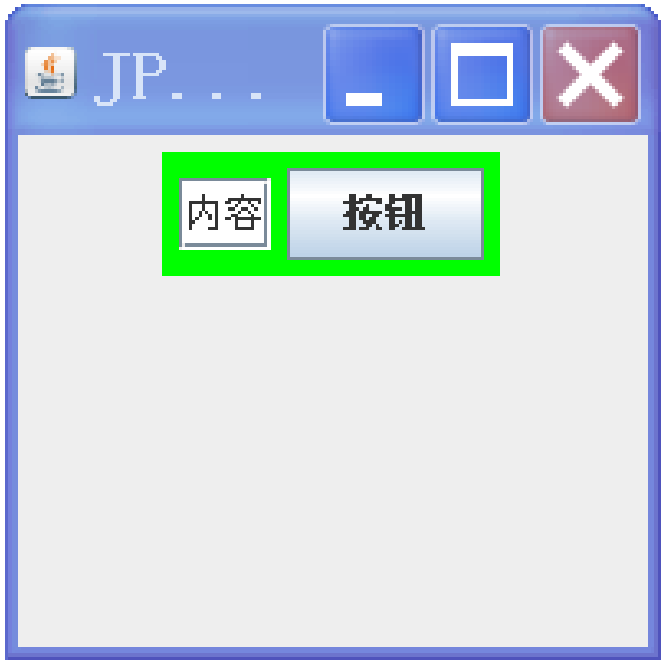
- To Create a Window in Swing Program
- Including the Rim, Title, Icon and Min/Max/Close
- Constructor
  - JFrame()
  - JFrame(String title)



JPanel

- Middle-level Container
- Combining Small Light-weighted Component
- Constructor
  - JPanel()
  - JPanel(boolean isDoubleBuffered)
  - JPanel(LayoutManager layout)
  - JPanel(LayoutManager layout, boolean isDoubleBuffered)

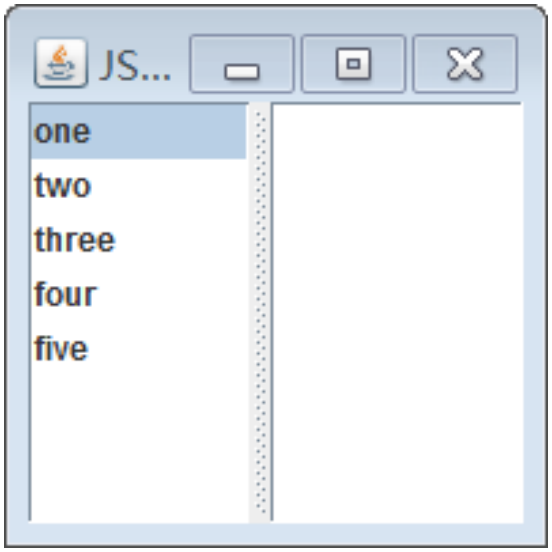
```
public static void main(String[] args)
{
    JFrame f = new JFrame("JPanel example");
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    Container cp = f.getContentPane();
    cp.setLayout(new FlowLayout());
    JPanel p1 = new JPanel();
    p1.setBackground(Color.green);
    cp.add(p1);
    p1.add(new JTextField("内容"));
    p1.add(new JButton("按钮"));
    f.setSize(200, 200);
    f.setVisible(true);
}
```



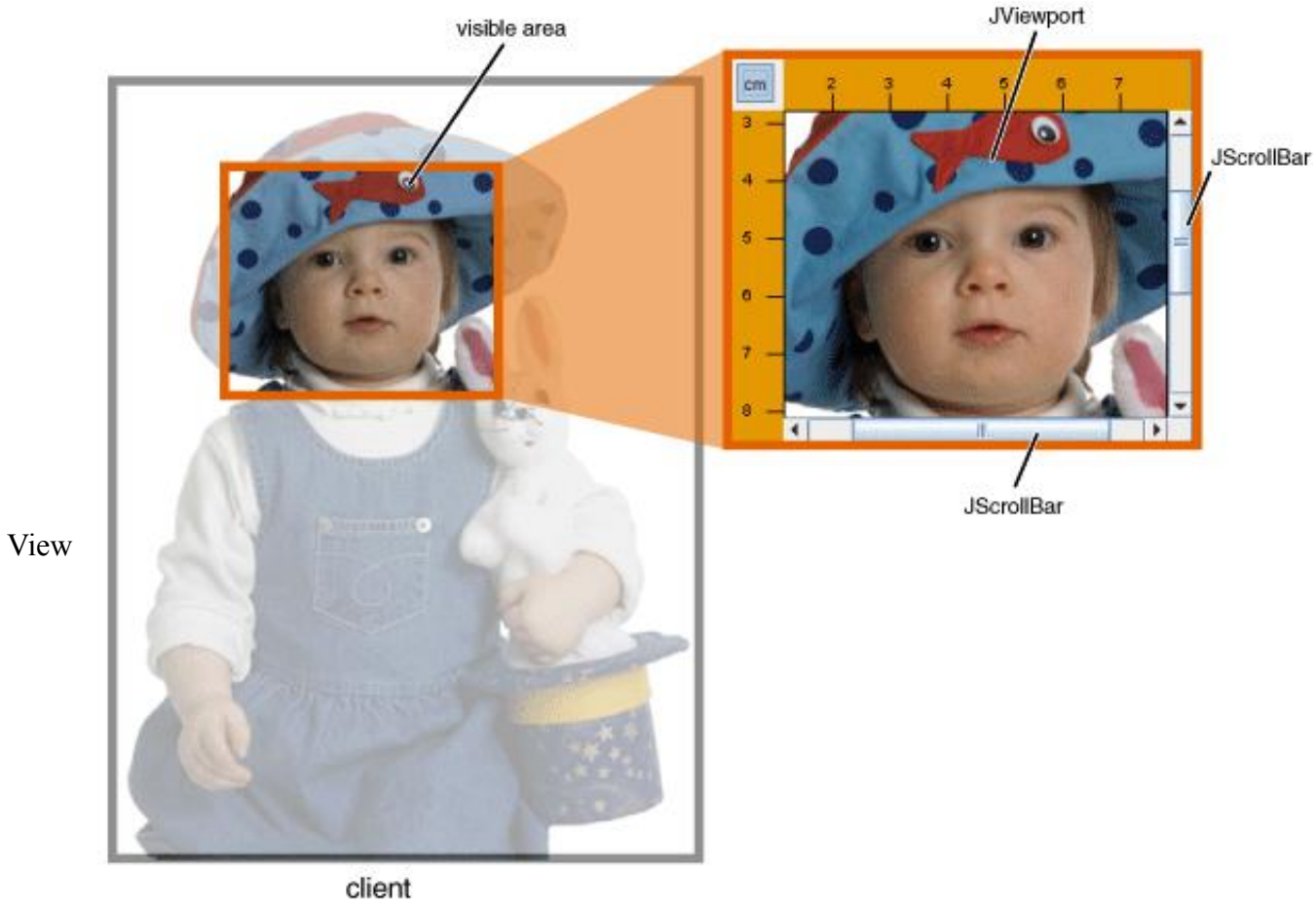
JSplitPane

- To Split One Container into Two
- Constructor
  - JSplitPane()
  - JSplitPane(int newOrientation)
    - JSplitPane.HORIZONTAL_SPLIT
    - JSplitPane.VERTICAL_SPLIT
  - JSplitPane(int newOrientation, Component newLeftComponent, Component newRightComponent)

```
JFrame frame = new JFrame("JSplitPanel example");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
Container cp = frame.getContentPane();
String[] stringList = {"one", "two", "three","four","five"};
JList<String> list = new JList<String>(stringList);
JSplitPane splitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
        list, new JTextArea());
splitPane.setDividerLocation(80);
cp.add(splitPane);
frame.setSize(200, 200);
frame.setVisible(true);
```



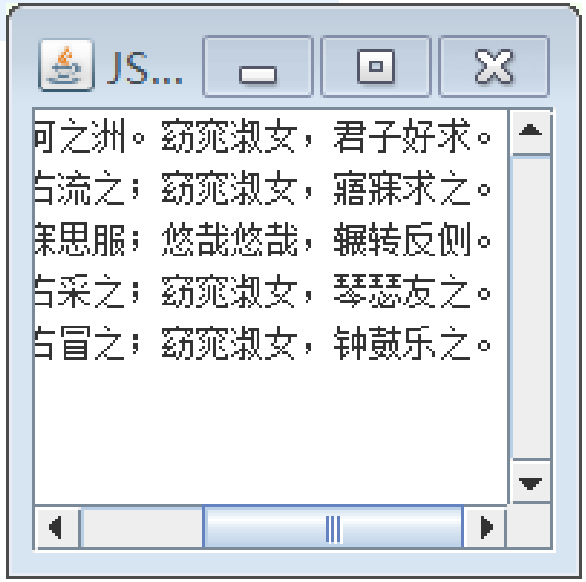
JScrollPane



JScrollPane

- To Show Horizontal or Vertical Scroll Bar When Content is Out Of Range
- Constructor
  - JScrollPane()
  - JScrollPane(Component view)
  - JScrollPane(Component view, int vsbPolicy, int hsbPolicy)
  - JScrollPane(int vsbPolicy, int hsbPolicy)

```
JFrame f = new JFrame("JSplitPanel example");
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
Container cp = f.getContentPane();
JScrollPane sp = new JScrollPane(new JTextArea());
sp.setVerticalScrollBarPolicy(
    JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
sp.setHorizontalScrollBarPolicy(
    JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
cp.add(sp);
f.setSize(200, 200);
f.setVisible(true);
```



```
JFrame frame = new JFrame("JSplitPanel example");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

ImageIcon icon = new ImageIcon("d:/1.png");
icon.setImage(icon.getImage().getScaledInstance(icon.getIconWidth(),
        icon.getIconHeight(), Image.SCALE_DEFAULT));
JLabel label = new JLabel();
label.setHorizontalAlignment(0);
label.setIcon(icon);

JScrollPane sp = new JScrollPane(label);
sp.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
sp.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

frame.setSize(500, 500);
frame.add(sp);
frame.setVisible(true);
```

JComponent

- A Basic Class for All Swing Component Except For Top-level Container
- Light-weighted

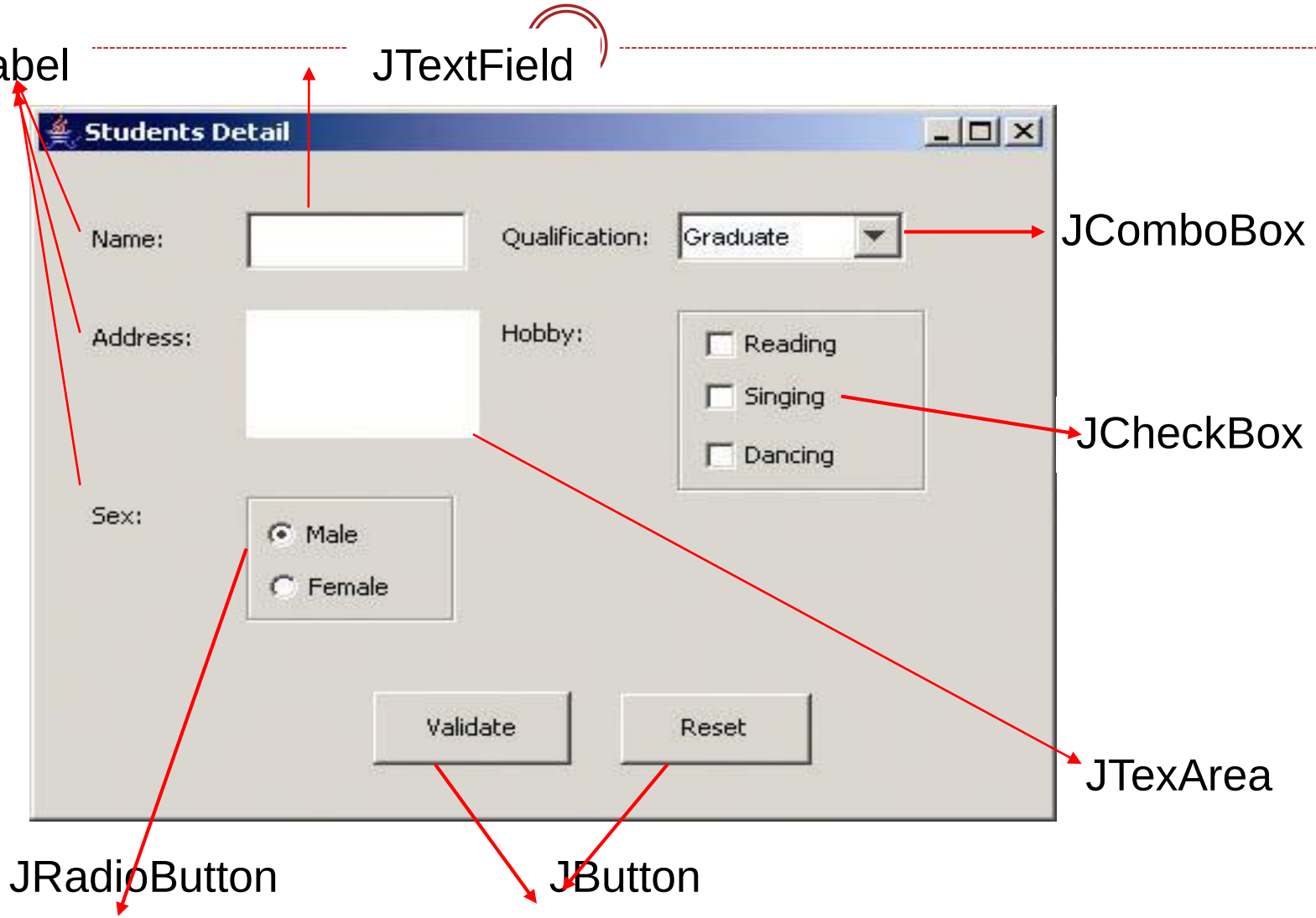
```
public abstract class JComponent extends Container
```

JComponent

- Methods

```
Graphics getGraphics();
int getX(); int getY(); int getWidth(); int getHeight()
void setVisible(boolean aFlag)
void setEnabled(boolean b)
void setFocusable(boolean focusable)
Font getFont(); void setFont(Font f)
Color getBackground(); void setBackground(Color c)
Cursor getCursor(); void setCursor(Cursor cursor)
Rectangle getBounds();
void setBounds(Rectangle r);
void setBounds(int x, int y, int width, int height)
String getToolTipText()、void setToolTipText(String text)
```

JComponents



JLabel

Name:

- Constructor
  - JLabel()
  - JLabel(String text)
  - JLabel(Icon image)
- Methods
  - String getText()、 void setText(String text)
  - void setIcon(Icon icon)

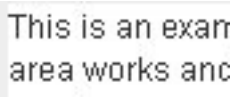
JTextField

public class JTextField extends JTextComponent

- Constructor
  - JTextField()
  - JTextField(String text)
- Methods

```
boolean isEditable(); void setEditable(boolean b)
int getColumns(); void setColumns(int columns)
int getHorizontalAlignment; void setHorizontalAlignment(int value)
String getSelectedText()
void setSelectionEnd(int selectionEnd)
void setSelectionStart(int selectionStart)
```

JTextArea



public class JTextArea extends JTextComponent

- Constructor
  - JTextArea(); JTextArea(int rows, int columns)
  - JTextArea(String text); JTextArea(String text, int rows, int columns)
- Methods

```
int getRows(); void setRows(int rows)
int getColumns(); void setColumns(int columns)
void insert(String str, int pos); void append(String str)
void replaceRange(String str, int start, int end)
```

JButton

public class JButton extends AbstractButton

- Constructor
  - JButton(); JButton(Icon icon)
  - JButton(String text); JButton(String text, Icon icon)
- Methods boolean isDefaultButton()
  String getText(); void setText(String text)
  String getActionCommand()
  void setActionCommand(String actionCommand)
  public ActionListener[] getActionListeners()
  public void addActionListener(ActionListener l)
  void removeActionListener(ActionListener l)

Example

- Create Following GUI
  - Create a JTextArea, where users can type text;
  - Create an noneditable JTextField ;
  - Create a Jbutton. When the button is clicked, the selected text in the JTextArea will be copied into the JTextField.

three steps to create a complicated UI

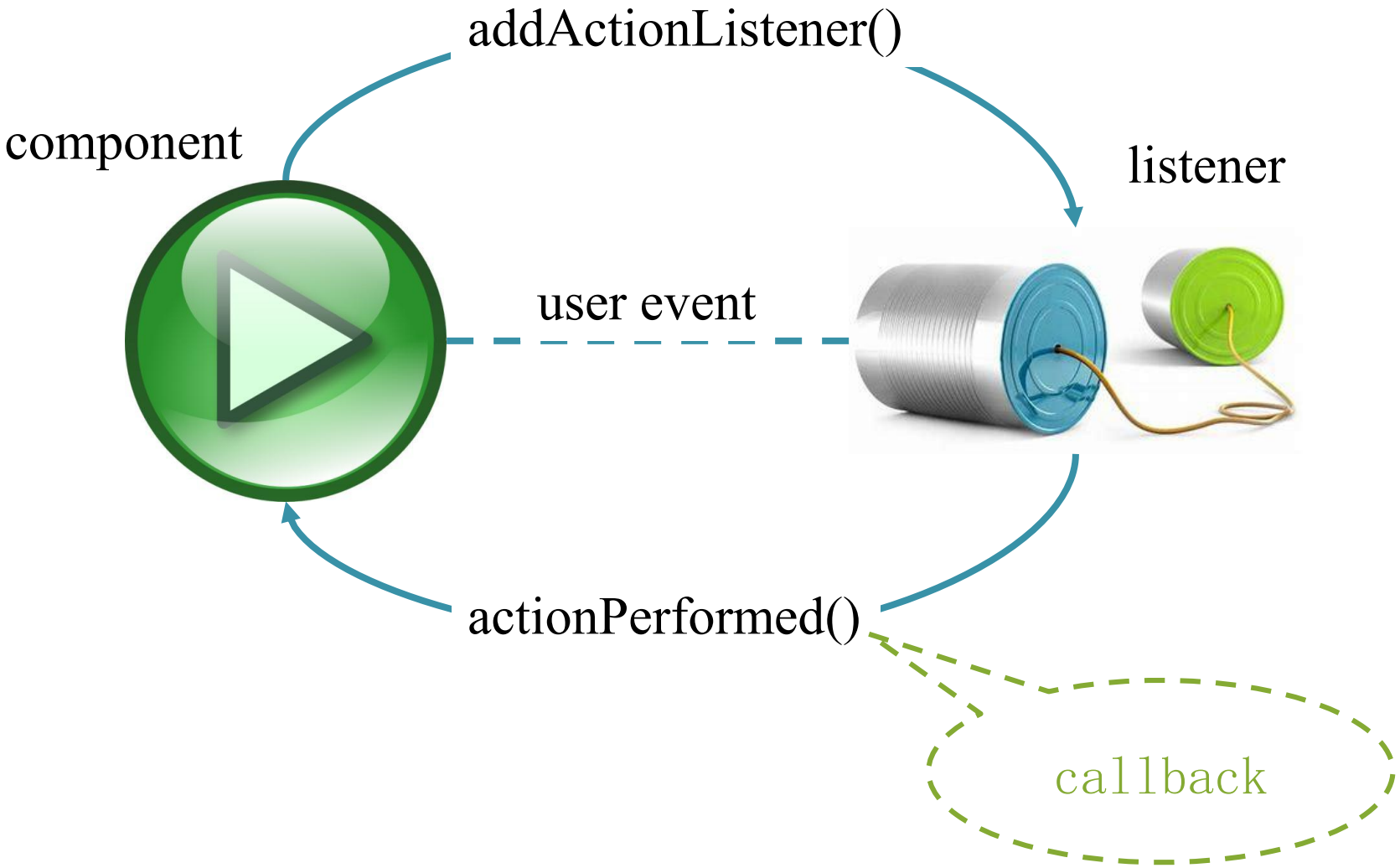
1. To create a JFrame	2. To create one or more JPanels	3. To specify the behavior
To give your UI a basic framework	To give your UI details	To make your UI interactive
To build a subclass of JFrame	To build one or more subclasses of JPanel	To use addAction Listener()

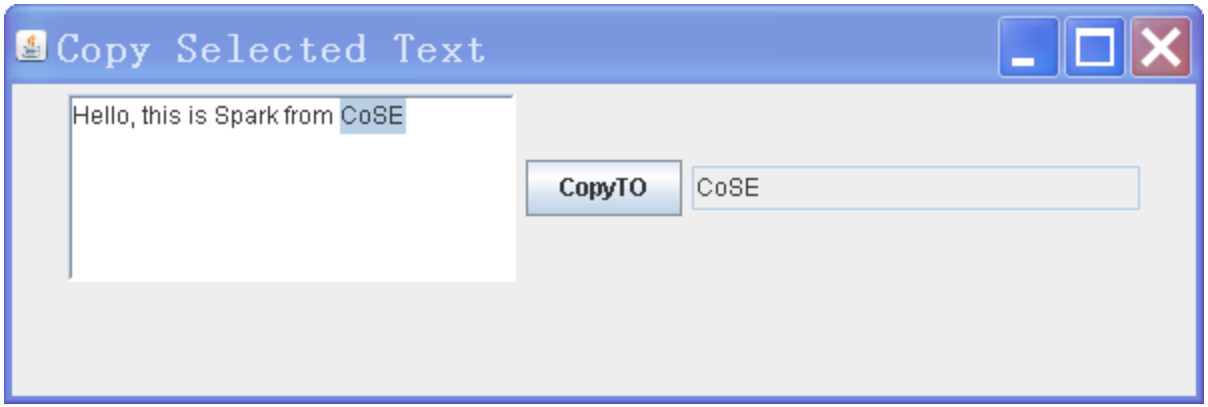
```
public class TextSelectionFrame extends JFrame{

    public TextSelectionFrame(){
        TextSelectionPanel panel = new TextSelectionPanel();
        this.setTitle("Copy Selected Text");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.add(panel);
        this.setSize(600,200);
    }

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

```
public class TextSelectionPanel extends JPanel{

    JTextArea textArea; //源输入框
    JTextField textField; //目标输出框
    JButton copyToButton; //拷贝按钮

    public TextSelectionPanel(){
        this.setLayout(new FlowLayout());
        this.setName("inner panel");
        textArea = new JTextArea(5,20);
        textArea.setBorder(BasicBorders.getTextFieldBorder());
        textField = new JTextField(20);
        textField.setEditable(false);
```

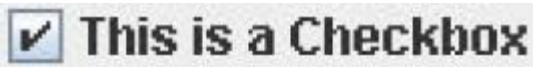
```
        copyToButton = new JButton("CopyTO");
        copyToButton.addActionListener(new CopyActionListener());
        this.add(textArea);
        this.add(copyToButton);
        this.add(textField);
    }

    private class CopyActionListener implements ActionListener{
        public void actionPerformed(ActionEvent event){
            textField.setText("");
            String selected = textArea.getSelectedText();
            textField.setText(selected);
        }
    }
}
```

This is an inner class, can you modify it to an anonymous class?


addActionListener()
component
listener
user event
actionPerformed()
callback


Copy Selected Text
Hello, this is Spark from CoSE
CopyTO
CoSE

JCheckBox



public class JCheckBox extends JToggleButton

- Constructor
  - JCheckBox(Icon icon)、JCheckBox(Icon icon, boolean selected)
  - JCheckBox(String text)、JCheckBox(String text, boolean selected)
  - JCheckBox(String text, Icon icon)、JCheckBox(String text, Icon icon, boolean selected)
- Methods
  - boolean isSelected() 、void setSelected(boolean b)
  - public ActionListener[] getActionListeners()、public void addActionListener(ActionListener l)、void removeActionListener(ActionListener l)

JRadioButton

public class JRadioButton extends JToggleButton

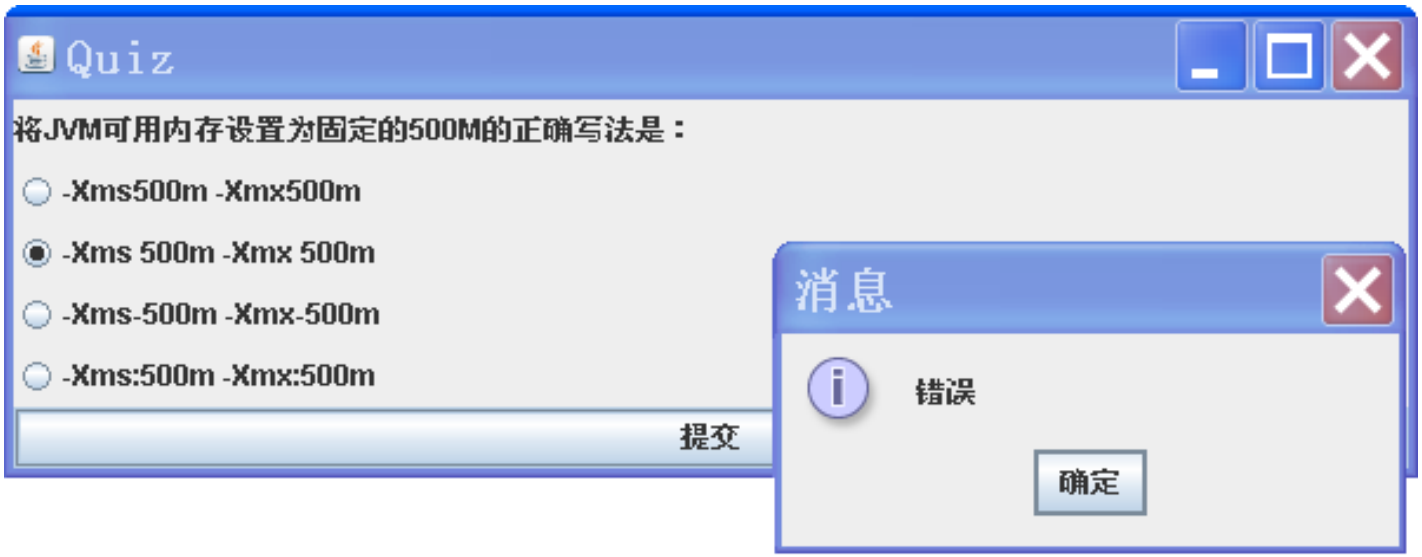
- Constructor
  - JRadioButton(Icon icon)、JRadioButton(Icon icon, boolean selected)
  - JRadioButton(String text)、JRadioButton(String text, boolean selected)
  - JRadioButton(String text, Icon icon)、JRadioButton(String text, Icon icon, boolean selected)
- Methods
  - boolean isSelected() 、void setSelected(boolean b)
  - public ActionListener[] getActionListeners()、public void addActionListener(ActionListener l)、void removeActionListener(ActionListener l)

ButtonGroup

- To Group JRadioButton
- Constructor
  - ButtonGroup()
- Methods
  - int getButtonCount()
  - void add(AbstractButton b)
  - void remove(AbstractButton b)

Example

- Quiz
- Write the Following GUI:
  - A question
  - Four options: A B C D
  - A Submit button
  - If correct, pop up “Correct”, or “Wrong” otherwise


Quiz
将JVM可用内存设置为固定的500M的正确写法是：
-Xms500m -Xmx500m
-Xms 500m -Xmx 500m
-Xms-500m -Xmx-500m
-Xms:500m -Xmx:500m
提交
消息
错误
确定

```
public class QuizFrame extends JFrame{

    public QuizFrame(){
        QuizPanel panel = new QuizPanel(this);
        this.setTitle("Quiz");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.add(panel);
        this.setSize(600, 200);
    }

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

```
public class QuizPanel extends JPanel{
    JFrame quizFrame;
    JLabel question;
    JRadioButton a,b,c,d;
    ButtonGroup quizGroup;
    JButton submit;
    public QuizPanel(JFrame frame){
        this.quizFrame = frame;
        question = new JLabel("将JVM可用内存设置为固定的500M的正确写法是：");
        a = new JRadioButton("-Xms500m -Xmx500m");
        b = new JRadioButton("-Xms 500m -Xmx 500m");
        c = new JRadioButton("-Xms-500m -Xmx-500m");
        d = new JRadioButton("-Xms:500m -Xmx:500m");
        quizGroup = new ButtonGroup();
        quizGroup.add(a);quizGroup.add(b);
        quizGroup.add(c);quizGroup.add(d);
        submit = new JButton("提交");
        submit.setSize(50, 50);
        submit.addActionListener(new SubmitListener());
```

```
        this.setLayout(new GridLayout(6,1));
        this.add(question);
        this.add(a);this.add(b);this.add(c);this.add(d);
        this.add(submit);
    }
    private class SubmitListener implements ActionListener{
        public void actionPerformed(ActionEvent event){
            if(a.isSelected()){
                JOptionPane.showMessageDialog(submit, "正确");
            }else{
                JOptionPane.showMessageDialog(submit, "错误");
            }
        }
    }
}
```

Other JComponents

- JComboBox
  - addItem()
  - get/setSelectedIndex
  - get/setSelectedItem
  - removeAllItems

Other JComponents

- JPasswordField
  - get/setEchoChar()
  - getPassword()

Other JComponents

- JSlider
  - get/setMinimum()
  - get/setMaximum()
  - get/setOrientation()

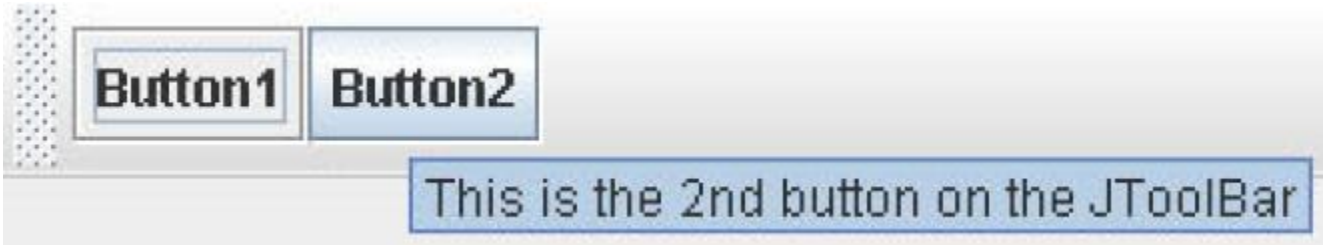
Other JComponents

- Jspinner
  - get/setValue()
  - getNextValue()
  - getPreviousValue()

Other JComponents

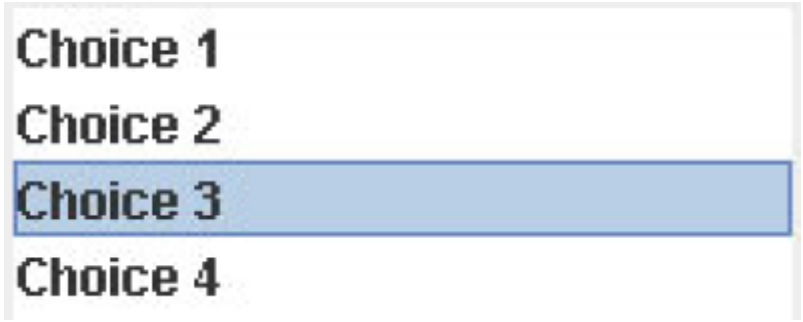
- JToolBar

- JToolTip



Other JComponents

- JList



- JTable

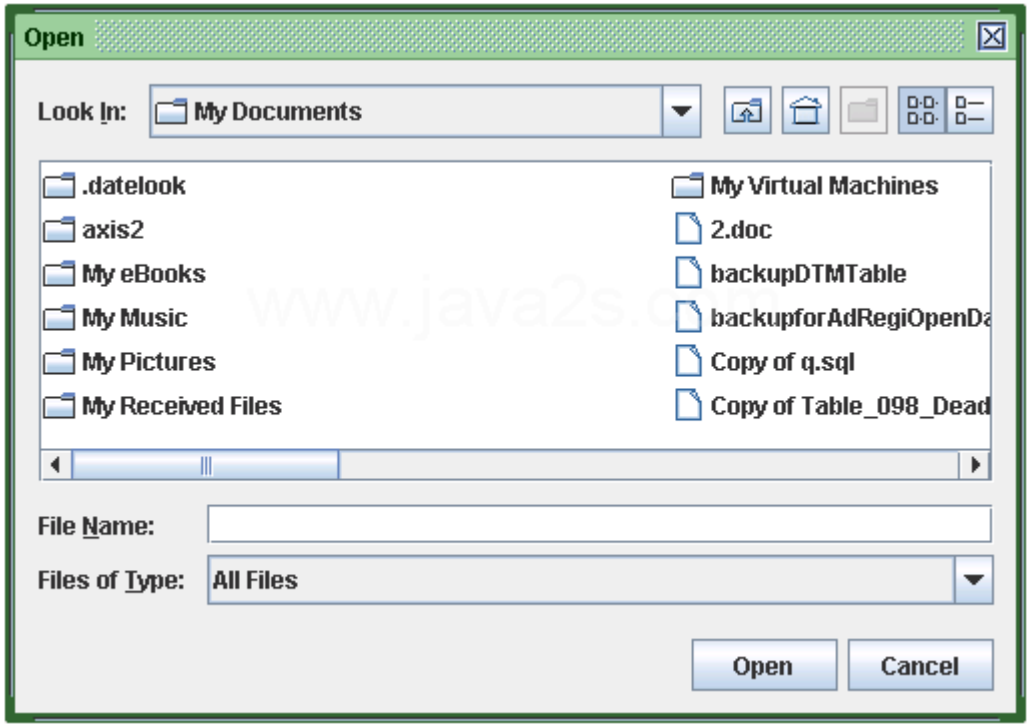
A	B	C	D	E
0x0	0x1	0x2	0x3	0x4
1x0	1x1	1x2	1x3	1x4
2x0	2x1	2x2	2x3	2x4
3x0	3x1	3x2	3x3	3x4
4x0	4x1	4x2	4x3	4x4

Other JComponents

- JTree

Other JComponents

- JFileChooser

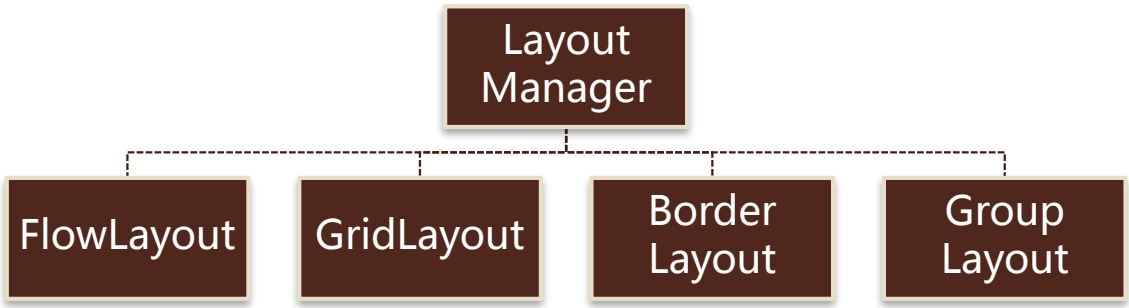


Layout Manager

- Motivation
  - Portability
  - Dynamic Layout
- Function
  - Assemble the components in an ordered way
  - Set the size, position of components
  - Know how to adapt when frame is moved or resized
- Different LM Uses Different Algorithm and Policy
- Each Container has a default Layout Manager

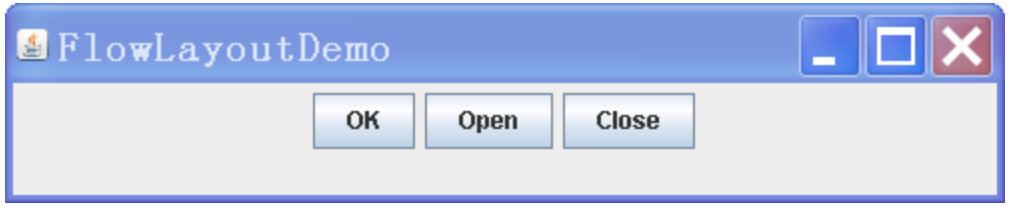
Layout Manager

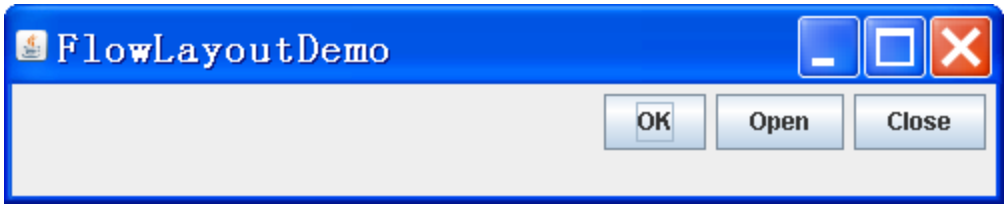
- Only Container and Subclasses Can Set Layout
- Setting Layout: setLayout(new xxxLayout())
- Common Layout Manager
  - FlowLayout
  - BorderLayout
  - GridLayout
  - GroupLayout



FlowLayout

- Default LM for JPanel
- Constructor
  - FlowLayout()
  - FlowLayout(int align)
  - FlowLayout(int align, int hgap, int vgap)
- Methods
  - int getAlignment()、void setAlignment(int align)
  - int getHgap()、void setHgap(int hgap)
  - int getVgap()、void setVgap(int vgap)






```
JFrame frame = new JFrame("FlowLayoutDemo");
frame.getContentPane().setLayout(
        new FlowLayout(FlowLayout.RIGHT));
JButton button1 = new JButton("OK");
JButton button2 = new JButton("Open");
JButton button3 = new JButton("Close");
frame.add(button1);
frame.add(button2);
frame.add(button3);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(500, 100);
frame.setVisible(true);
```

GridLayout

- Grid-style Layout
- Occupy the space of container evenly
- Left-to-Right; Top-to-Bottom
- Each component is the same size in all cases

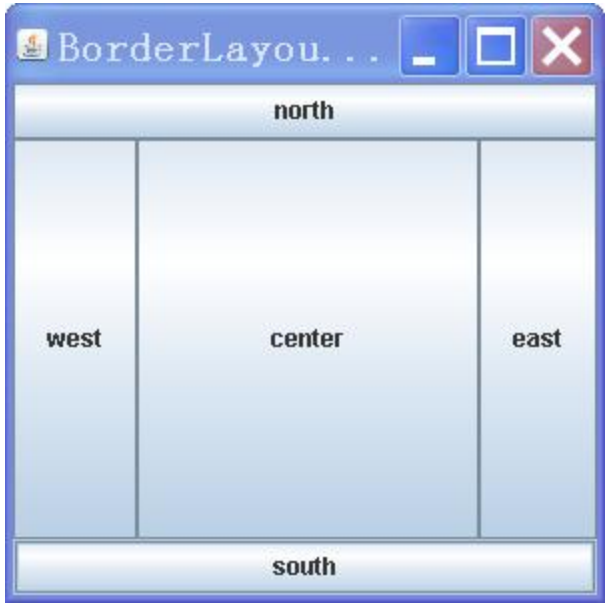
- Constructor
  - GridBagLayout()
  - GridBagLayout(int rows, int columns)
  - GridBagLayout(int rows, int cols, int hgap, int vgap)
- Methods
  - int getColumns()、void setRows(int rows)
  - int getRows()、void setColumns(int cols)
  - int getHgap()、void setHgap(int hgap)
  - int getVgap()、void setVgap(int vgap)

```
JFrame frame = new JFrame("GridLayoutDemo");
frame.getContentPane().setLayout(
          new GridLayout(3,3));
JButton button1 = new JButton("1");
JButton button2 = new JButton("2");
// 省略
JButton button9 = new JButton("9");
frame.add(button1);
frame.add(button2);
//省略
frame.add(button9);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(200, 200);
frame.setVisible(true);
```

BorderLayout

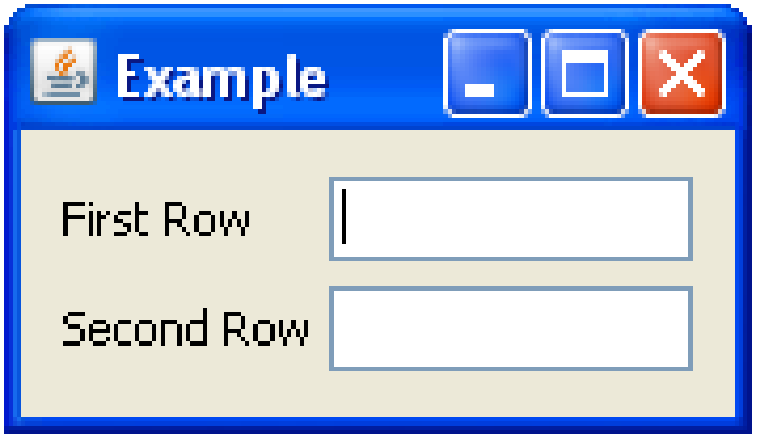
- Divide the Container into 5 Zones: North/South/East/West/Center
- Default LM for JFrame
- Constructor
  - BorderLayout()
  - BorderLayout(int hgap, int vgap)
- Methods
  - int getAlignment()、void setAlignment(int align)
  - int getHgap()、void setHgap(int hgap)
  - int getVgap()、void setVgap(int vgap)

```
JFrame frame = new JFrame("BorderLayoutDemo");
frame.getContentPane().setLayout(
        new BorderLayout());
JButton button1 = new JButton("center");
JButton button2 = new JButton("east");
...
frame.add(button1, BorderLayout.CENTER);
frame.add(button2,BorderLayout.EAST);
...
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(300, 300);
frame.setVisible(true);
```



Self-study

- GroupLayout



Other Layouts

- http://blog.sina.com.cn/s/blog_6f116c940101alna.html
  - GridBagLayout
  - CardLayout
  - BoxLayout

Nested Layout


```
private Component initLeft() {
    JPanel panelLeft = new JPanel();
    panelLeft.setLayout(new FlowLayout(FlowLayout.CENTER));
    panelLeft.add(new JButton("open"));
    panelLeft.add(new JButton("save"));
    panelLeft.add(new JButton("close"));
    panelLeft.add(new JButton("exit"));
    return panelLeft;
}
```

Nested Layout


```
private Component initRight() {
    JPanel panelRight = new JPanel();
    panelRight.setLayout(new GridLayout(5, 1));
    JPanel panelTemp = null;
    for (int i = 0; i < 5; i++) {
        panelTemp = new JPanel();
        panelTemp.setLayout(new  FlowLayout(FlowLayout.LEFT));
        panelTemp.add(new JLabel("Label " + i));
        panelTemp.add(new JTextField("TextField " + i));
        panelRight.add(panelTemp);
    }
    return panelRight;
}
```

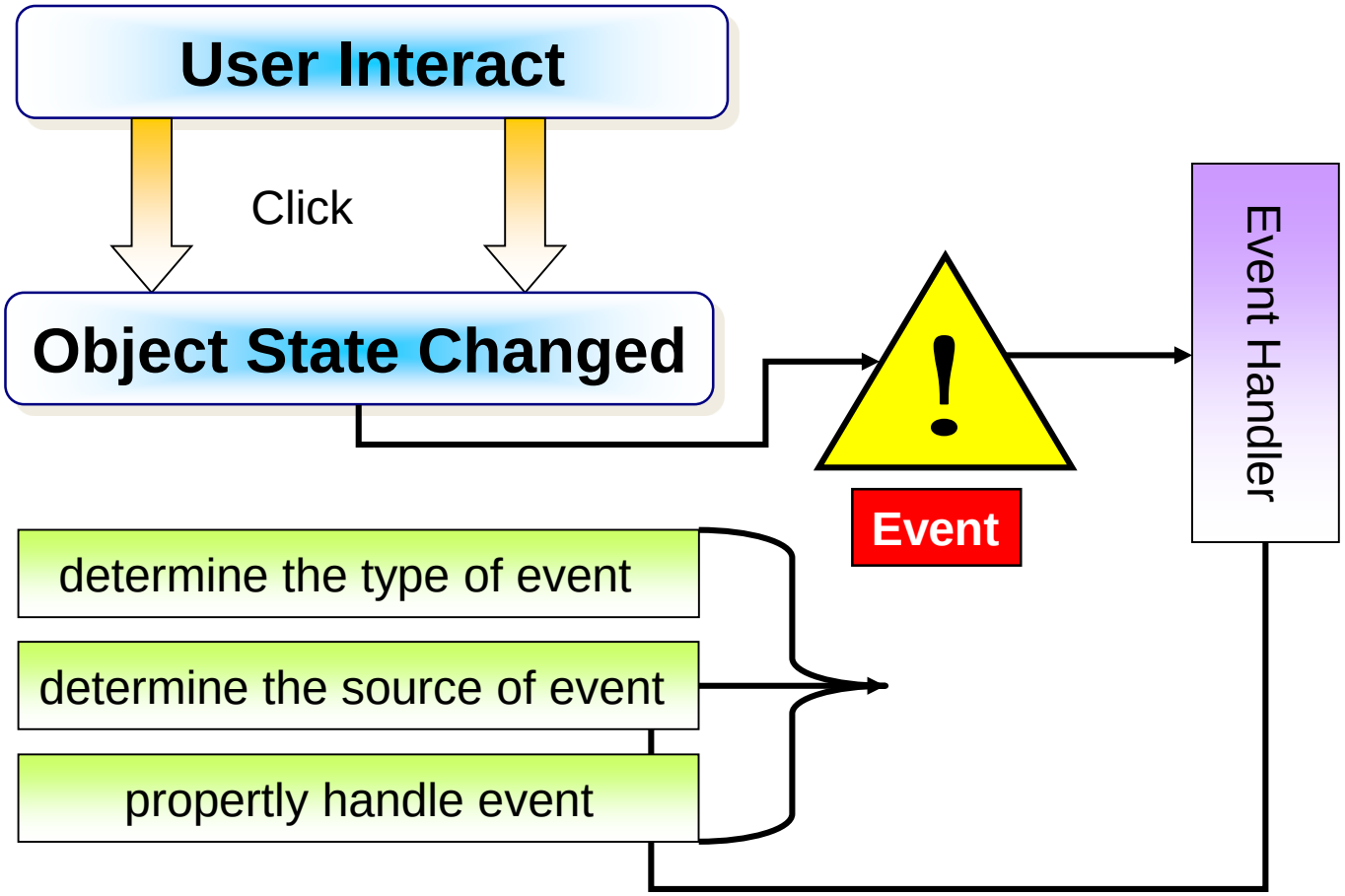
Nested Layout


```
public static void main(String[] args) {
        JFrame f = new JFrame("Composed Layout example");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        CCompostExample compost = new CCompostExample();
        JSplitPane splitPanel = new JSplitPane();
        splitPanel.setLeftComponent(compost.initLeft());
        splitPanel.setRightComponent(compost.initRight());
        splitPanel.setDividerLocation(80);
        f.add(splitPanel);
        f.setSize(400, 200);
        f.setVisible(true);
}
```

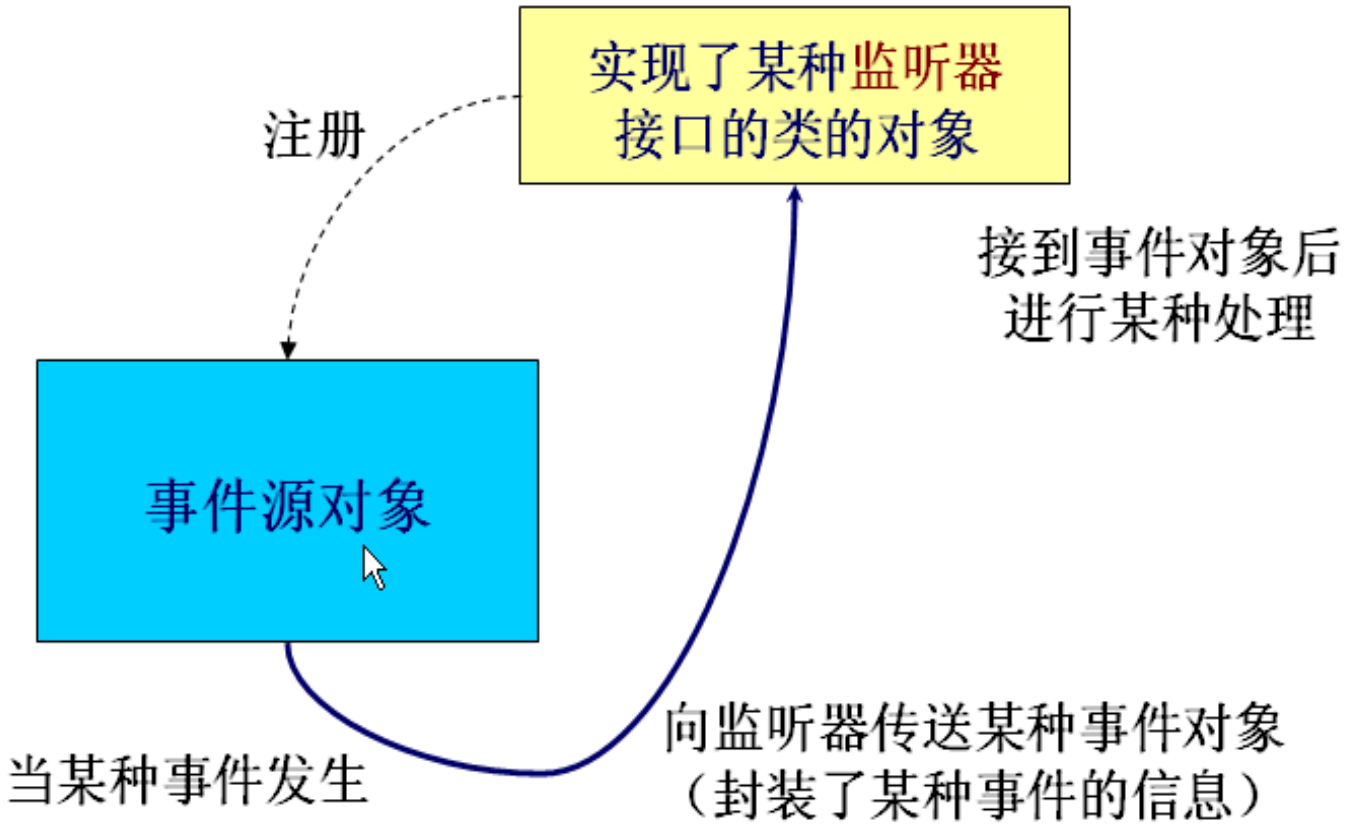
Event Handling Model

- Event
  - Something Happened
  - In a form of Java Class, representing user operation in GUI
- Event Source
  - The source of an event
  - Components, such as button, menu…
- Event Handler
  - The handler of events
  - An object receiving events and process them

Event Handling Model



Event Handling Model



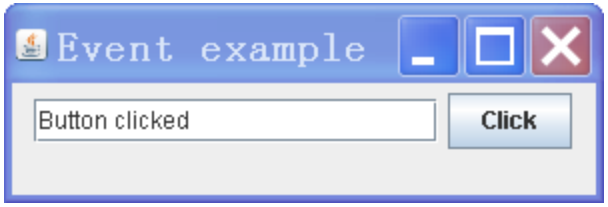
Event Handling Model

- Java defined most common events (XXXEvent)
- If one type of event need to be handled, a Class should be written implementing corresponding interface XXXListener
- The event source should add a listener using addXXXListener()

```
public class ActionDemo{

    private JFrame frame;
    private JTextField textField;

    private class ButtonListener implements ActionListener{
        public void actionPerformed(ActionEvent e){
        textField.setText("Button clicked");
        }
    }
```

```
public ActionDemo(){
    JFrame frame = new JFrame("Event example");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().setLayout(new FlowLayout());
    textField = new JTextField();
    textField.setColumns(18);
    frame.add(textField);
    JButton btn = new JButton("Click");
    frame.add(btn);
    btn.addActionListener(new ButtonListener());
    frame.setSize(300, 100);
    frame.setVisible(true);
}
```

```
    public static void main(String[] args) {
        ActionDemo demo = new ActionDemo();
    }
}
```



Event Class

- Package: java.awt.event
- Classification of Event Class
  - Low-level Event: event based on component or container
    - KeyEvent
    - MouseEvent
  - High-level Event: event based on semantics
    - ActionEvent
    - DocumentEvent

Listener Interface

- Package: java.awt.event
- Basic Interface: java.util.EventListener
- Each kind of event has a corresponding listener
- EventListener is an interface, the method invocation is determined by action
- Each listener listens to different event

Common Event and EventListener

Event Class	Listener Interface
MouseEvent	MouseListener
KeyEvent	KeyListener
FocusEvent	FocusListener
ComponentEvent	ComponentListener
WindowEvent	WindowListener
ContainerEvent	ContainerListener
ActionEvent	ActionListener
ItemEvent	ItemListener
DocumentEvent	DocumentListener

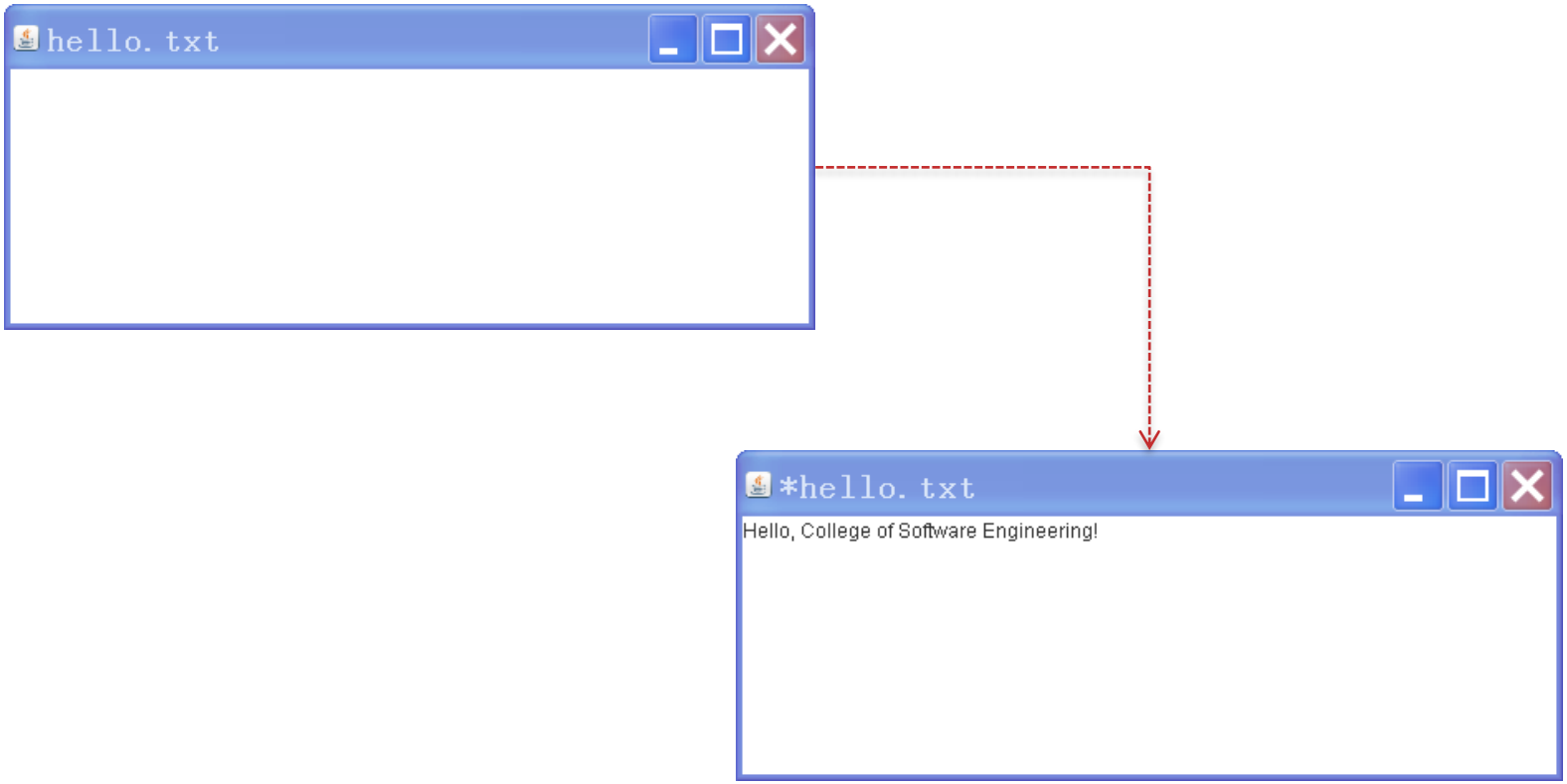
Example

- Create a editable text input area
- Once user modifies the hello.txt, set the title of frame to “*hello.txt”, indicating the content is not saved

```
public class TextDemo {

    JFrame frame;
    JTextArea text;

    public TextDemo(){
        frame = new JFrame();
        frame.setTitle("hello.txt");
        text = new JTextArea();
        text.getDocument().addDocumentListener(new TextChangeListener());
        frame.add(text);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(500, 200);
        frame.setVisible(true);
    }
```

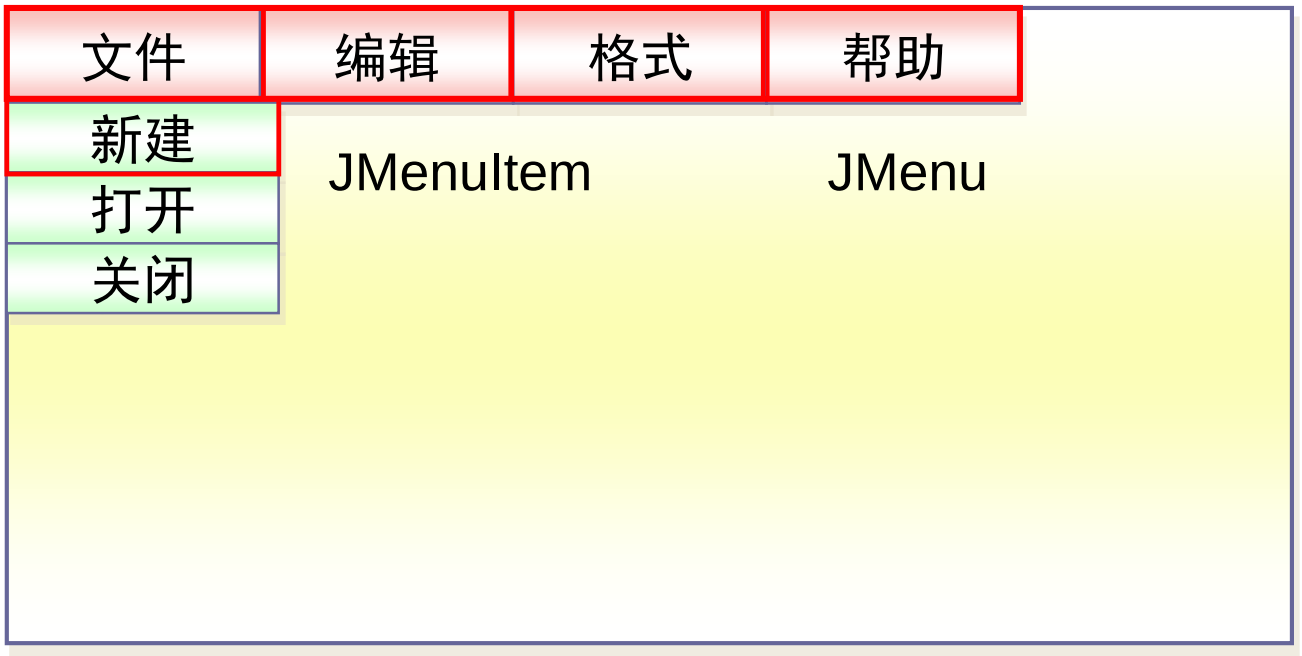
```
private class TextChangeListener implements DocumentListener·
    boolean changed = false;
    public void changedUpdate(DocumentEvent e) {
        if(!changed){
            frame.setTitle("*" + frame.getTitle());
            changed = true;
        }
    }
    public void insertUpdate(DocumentEvent e) {
        ...         // the same
    }
    public void removeUpdate(DocumentEvent e) {
        ...         // the same
    }
}
```

```
    public static void main(String[] args){
        TextDemo demo = new TextDemo();
    }
}
```

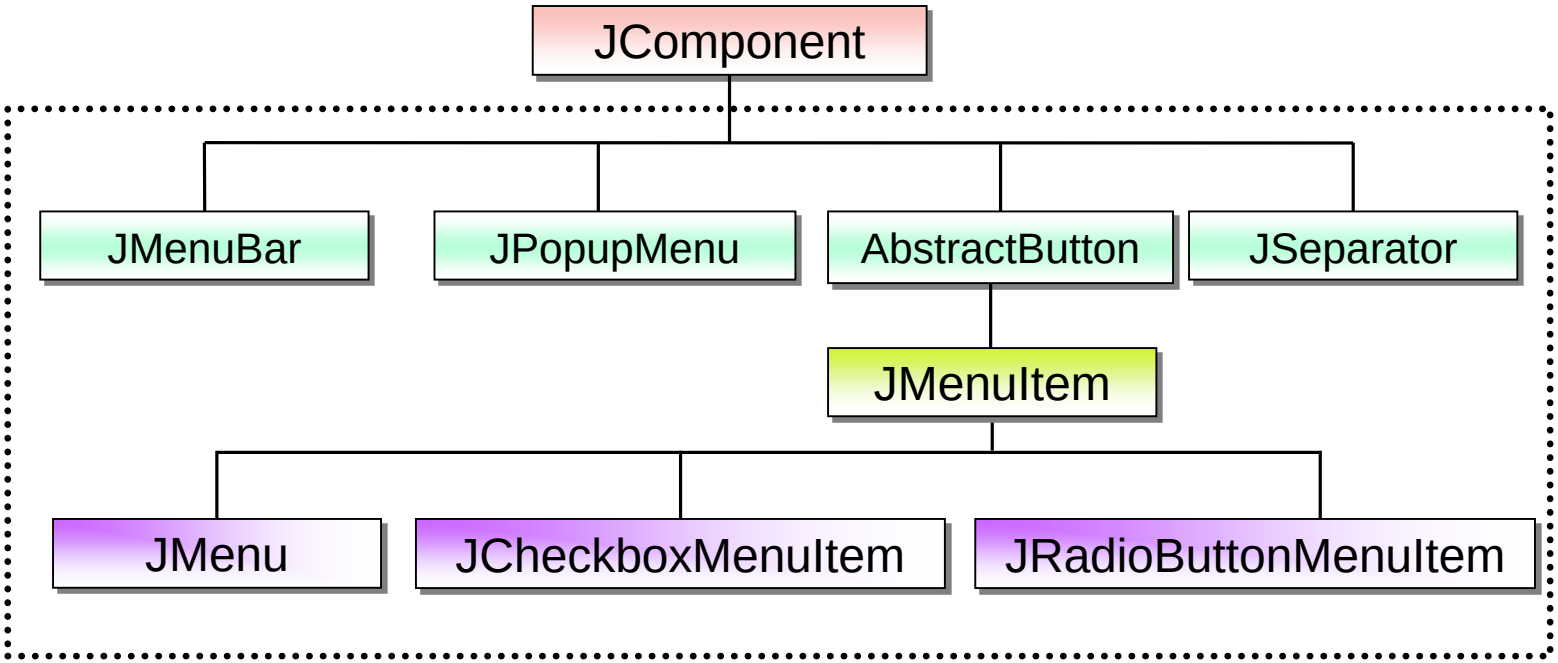


Menu

- An item list, showing all possible operation



The Hierarchy of Menu Classes



Self-study

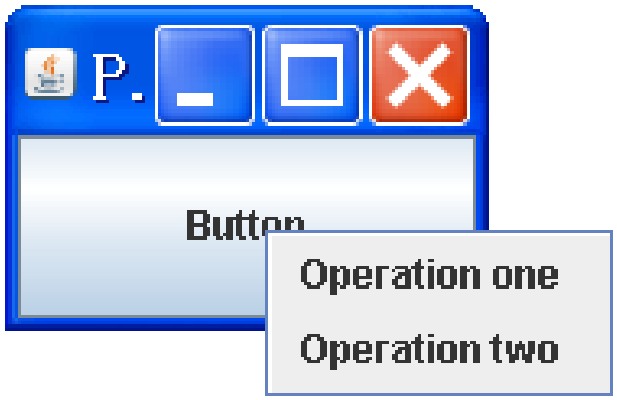
- Create a MenuBar, including File and Option menu
- File menu includes Open/Save/Close items
- Option includes
  - Red / Green / Blue three RadioButton item
  - A split line
  - Red / Green / Blue three CheckBox item
- Source code: ftp　源码/Menu Demo/MenuDemo.java"

Pop-up Menu

- JPopupMenu
- Showing menu in pop-up style
- Can appear in any place in the frame
- Usually trigger by right-click
- The menu items depends on context

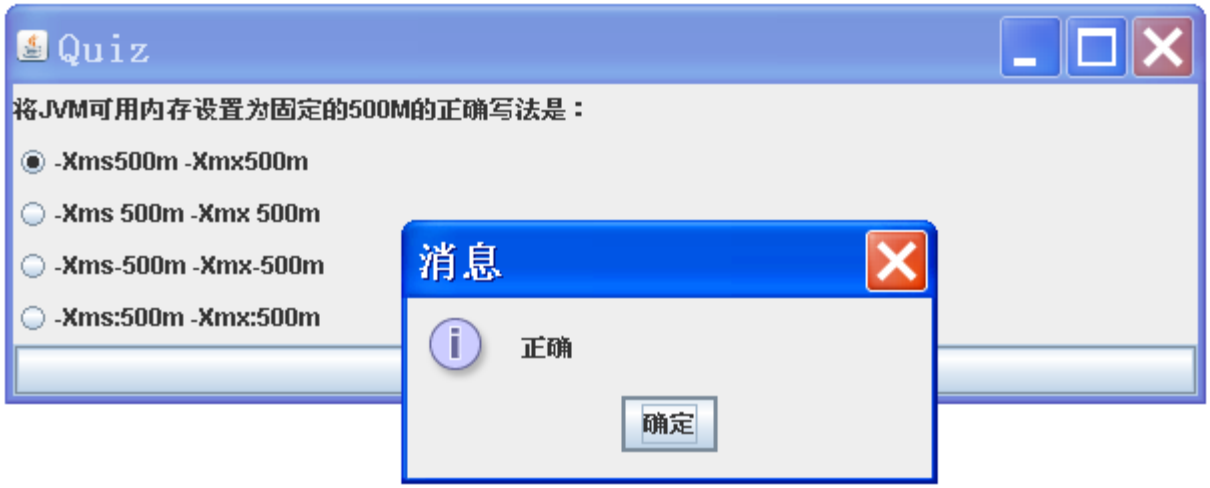
Self-study

- Create a JButton
- Add a pop-up menu for JButton
  - Including Operation 1 and Operation 2, two menu item
- Source code: 源码/MenuDemo/PopupDemo.java



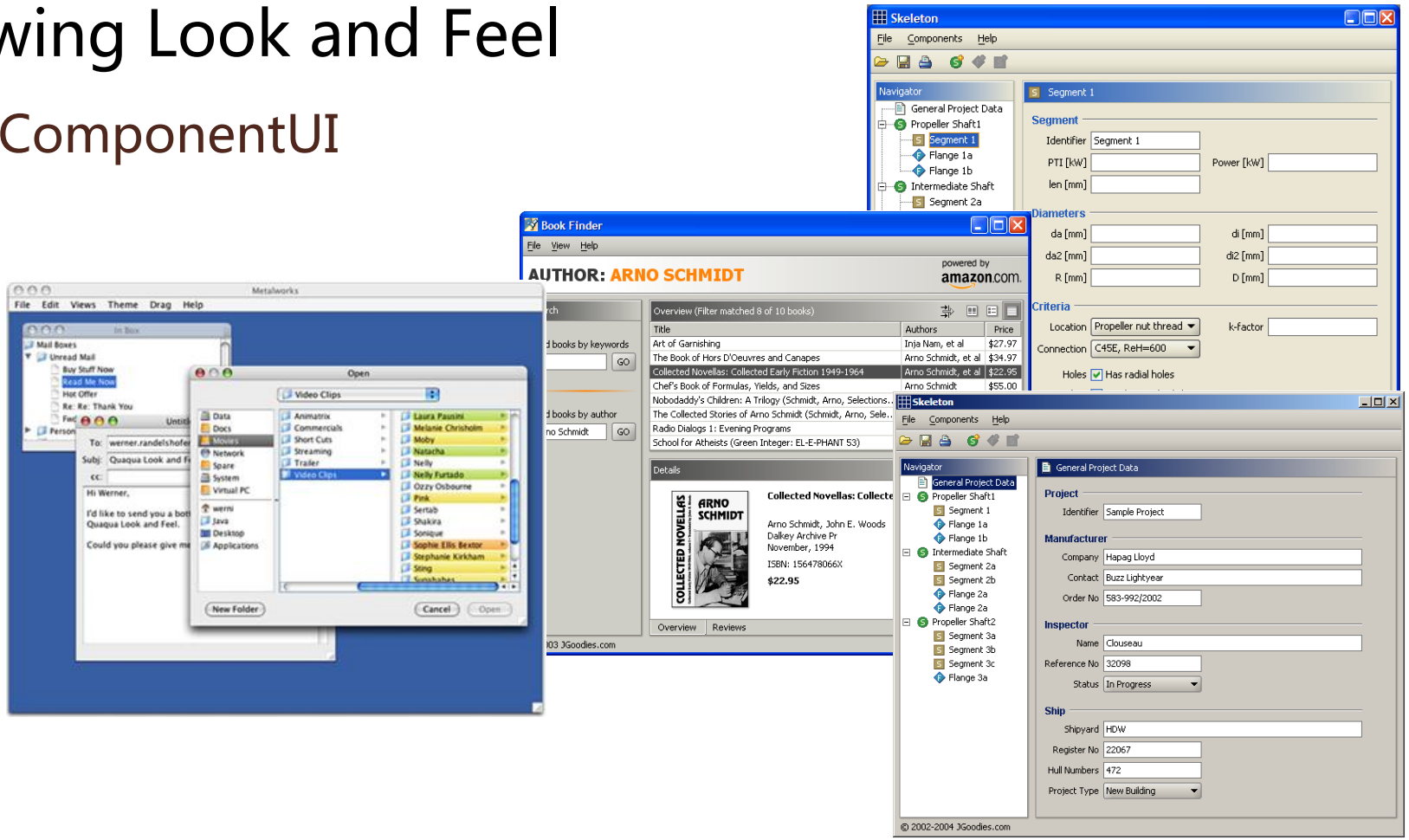
Self-study

- JDialog
- Create dialog using JOptionPane



Self-study

- Swing Look and Feel
  - ComponentUI



Forecast

- Notion of Thread
- Creation of Thread
- Scheduling of Thread
- Priority of Thread
- Synchronization of Thread