

# Swing

© Составление, Будаев Д.С., Гаврилов А.В., 2013

**Лекция 10**

**NetCracker®**

**УНЦ «Инфоком»  
Самара  
2013**

# План лекции

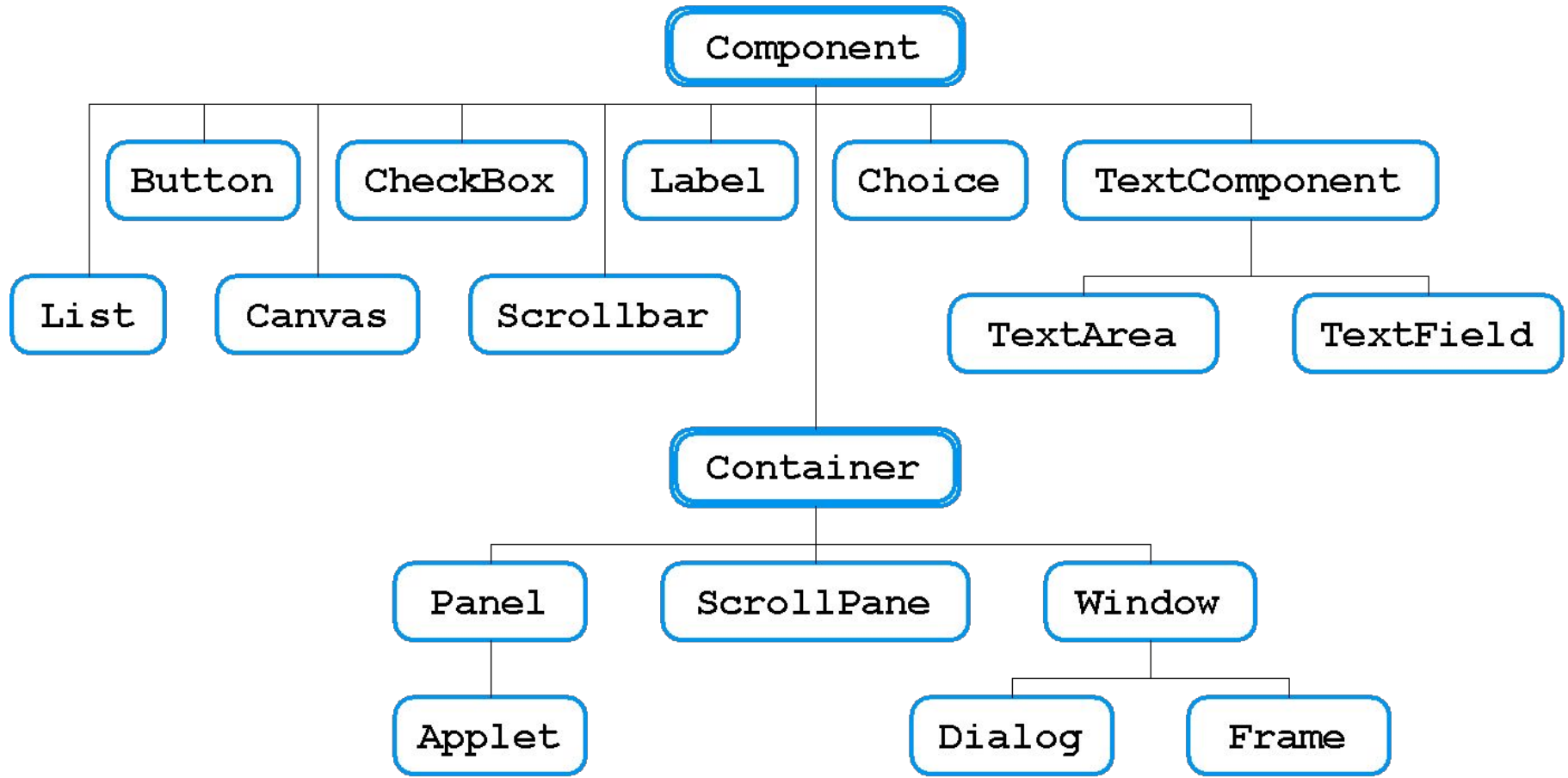
---

- Технология AWT и ее особенности
- Технология Swing и ее особенности
- Иерархия компонентов Swing
- Обработка событий в Swing
- Апплеты

# Графические приложения на Java

- Кроссплатформенное графическое приложение?..
- Abstract Window Toolkit (AWT)
  - Компоненты являются компонентами ОС
  - Большое количество native-кода
  - Отображение изменяется при смене ОС
  - Класс `java.awt.Component` определяет базовую функциональность всех компонентов

# Иерархия классов AWT



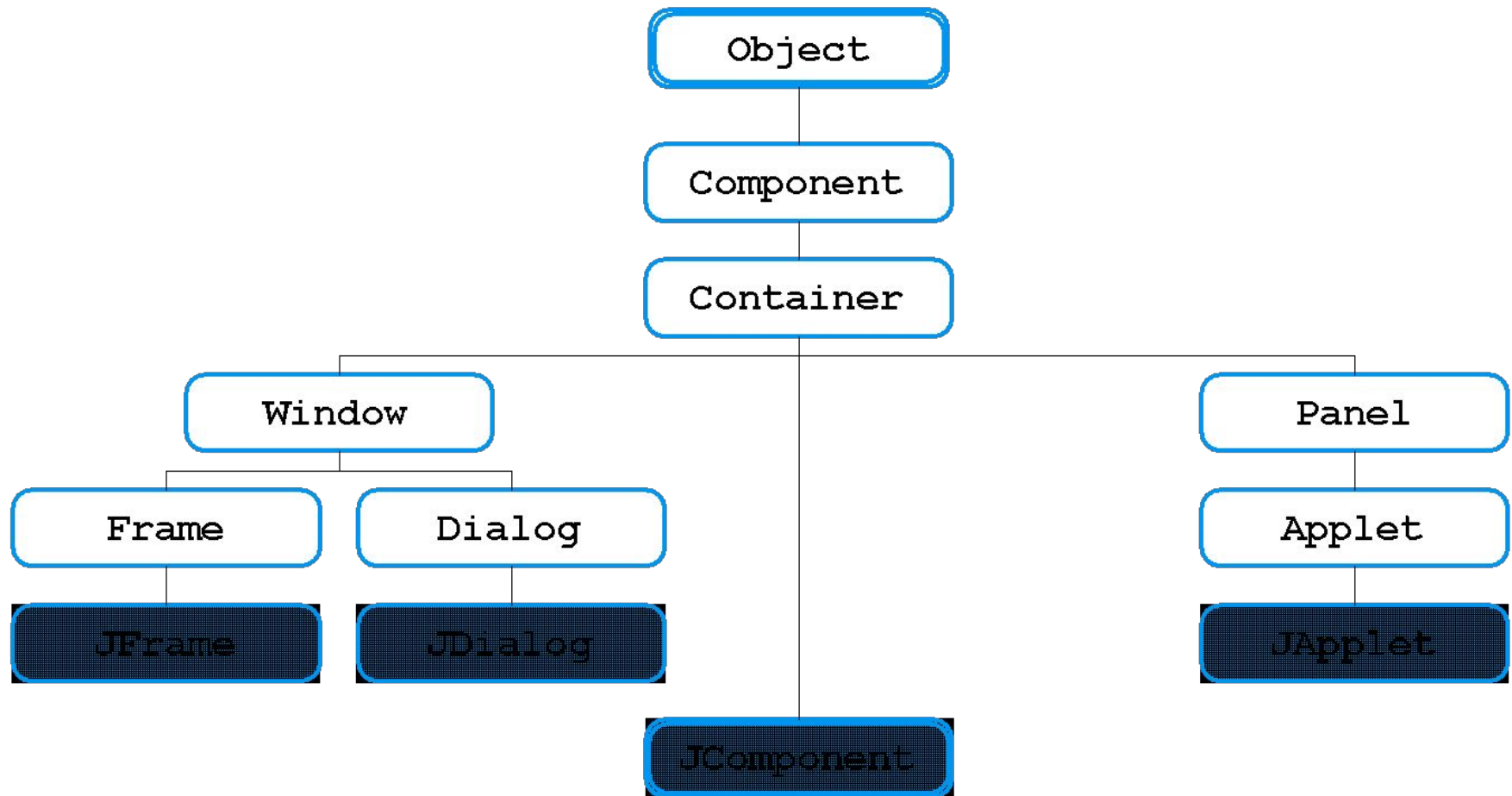
# Проблемы AWT

- Сходные элементы в различных ОС могут иметь некоторые различия
- ОС часто имеют элементы GUI, отсутствующие в других ОС
- Использование native-методов в AWT приводит к возникновению ошибок на конкретных платформах

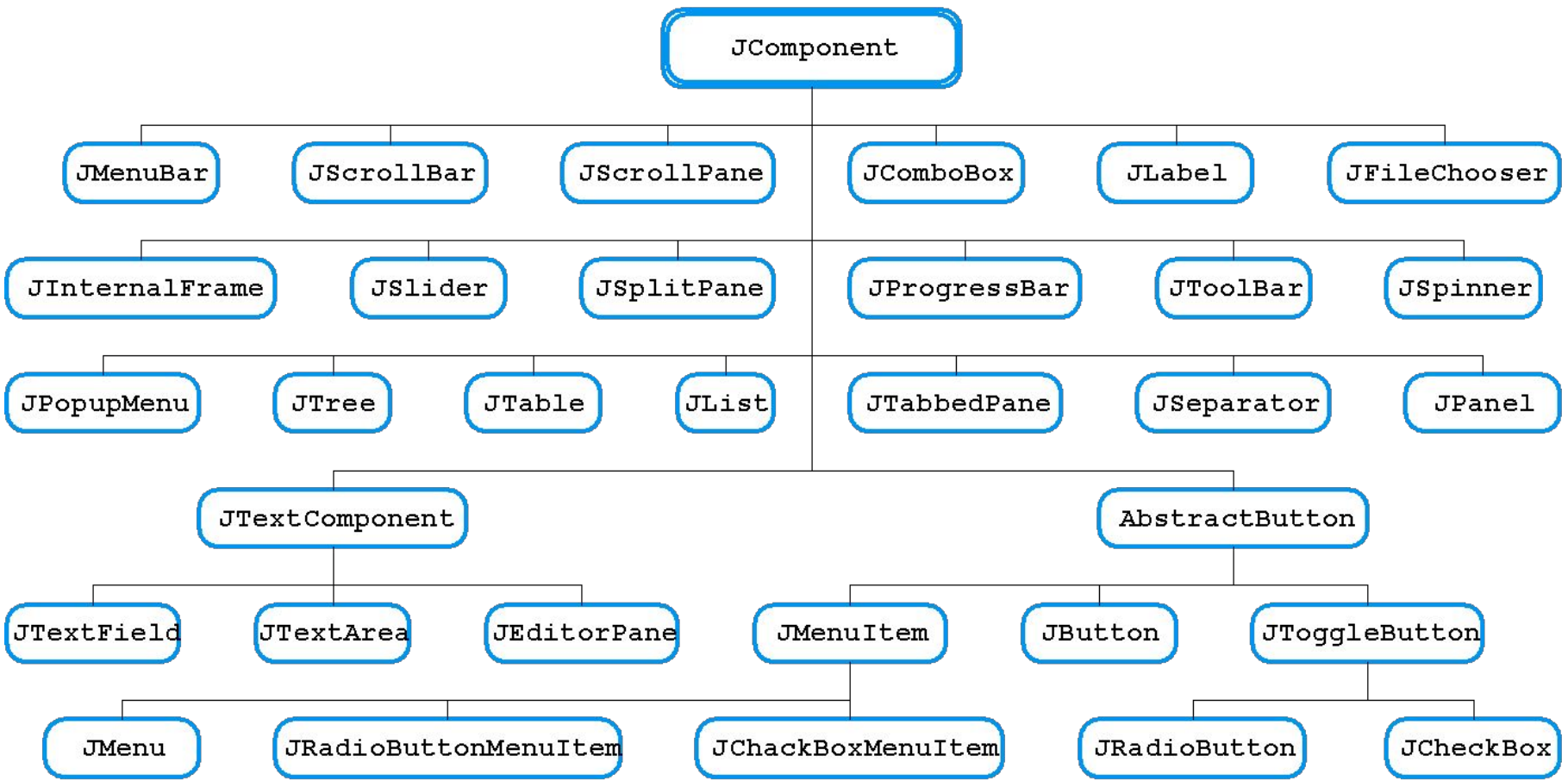
# Технология Swing

- Элементы GUI **отрисовываются** в пустых окнах
- «Нативные» функции используются только для вывода окна, отрисовки и получения информации о действиях пользователя
- Набор элементов GUI более широк, чем в AWT, и может быть еще расширен
- Сильная привязка к «нативным» методам отсутствует, что снижает вероятность возникновения ошибок
- Отображение на различных платформах единообразно

# Иерархия классов



# Иерархия классов. Пакет javax.swing





# Отрисовка КОМПОНЕНТОВ

- Отрисовка производится в методе `paintComponent()`
- Запрос на перерисовку
  - `public void repaint()`
  - `public void repaint(long tm)`
  - `public void repaint(int x, int y, int width, int height)`
  - `public void repaint(long tm, int x, int y, int width, int height)`

```
class NotHelloWorldPanel extends JPanel {
    public static final int MESSAGE_X = 75;
    public static final int MESSAGE_Y = 100;

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawString("Not a Hello, World program", MESSAGE_X, MESSAGE_Y);
    }
}
```

# Работа с графикой

## ■ `java.awt.Graphics`

- Базовый класс, предназначенный для рисования в контекстах компонентов, изображениях в памяти и т.д.
- Предлагает простые средства рисования:  
`void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)`  
`void drawString(String str, int x, int y)` и т.д.

## ■ `java.awt.Graphics2D`

- Наследует `java.awt.Graphics`, обеспечивающий большую функциональность, в т.ч. работа с 2D-сценой
- `java.awt.geom` содержит набор классов работы с графическими примитивами

# Работа с цветом

- Класс `java.awt.Color`
- Константы `Color.BLUE`, `Color.RED`
- Покомпонентные конструкторы  
`Color(float r, float g, float b, float a)`  
`Color(ColorSpace cspace, float[] components, float alpha)`
- Методы получения параметров цвета  
`getRed()`, `getTransparency()`

# Работа со шрифтами

- Класс `java.awt.Font`
  - Константы
  - Конструкторы  
`Font(String name, int style, int size)`
  - Методы модификации и получения параметров шрифта
- Класс `java.awt.FontMetrics`
  - Содержит методы определения геометрических характеристик шрифтов

# Двойная буферизация

- В целях экономии времени на перерисовку логично запоминать однажды нарисованный статичный объект как рисунок в памяти
- При использовании Swing для этого не надо предпринимать дополнительных действий по выделению памяти и.д.
- Используется т.н. механизм «двойной буферизации», реализующий сохранение информации на уровне механизмов отрисовки
- Для одного участка «видимой области» приложения используется не более одного изображения-буфера

# Двойная буферизация

## Методы класса `JComponent`

- `setDoubleBuffered(boolean aFlag)`  
Устанавливает, буферизует ли объект свой вывод
- `boolean isDoubleBuffered()`  
Возвращает булевское значение, показывающее, используется ли двойная буферизация, или нет

# Pluggable Look And Feel

- Программе можно придать различный вид, изменив «стиль» отрисовки компонентов
  - Внешний вид программы может изменяться во время исполнения
  - Перерисовка должна вызываться принудительно
- Есть разработанные «стили» для базовых платформ
- Существует возможность создания собственных «стилей»

# Pluggable Look And Feel

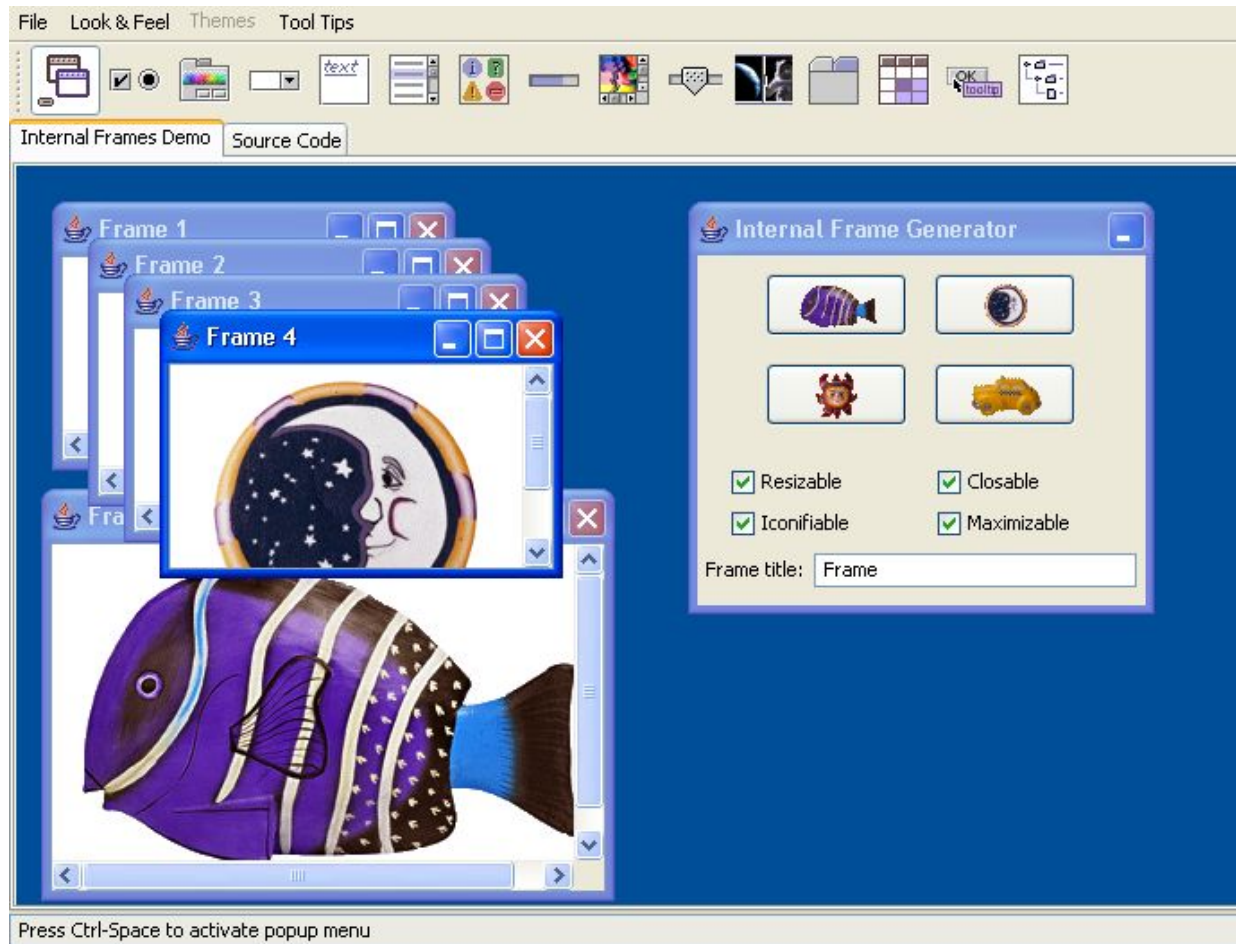
```
String s;

switch (value) {
    case 0: s = "javax.swing.plaf.metal.MetalLookAndFeel";
        break;
    case 1: s =
"com.sun.java.swing.plaf.motif.MotifLookAndFeel";
        break;
    case 2: s =
"com.sun.java.swing.plaf.windows.WindowsLookAndFeel";
        break;
    default: s = UIManager.getSystemLookAndFeelClassName();
}

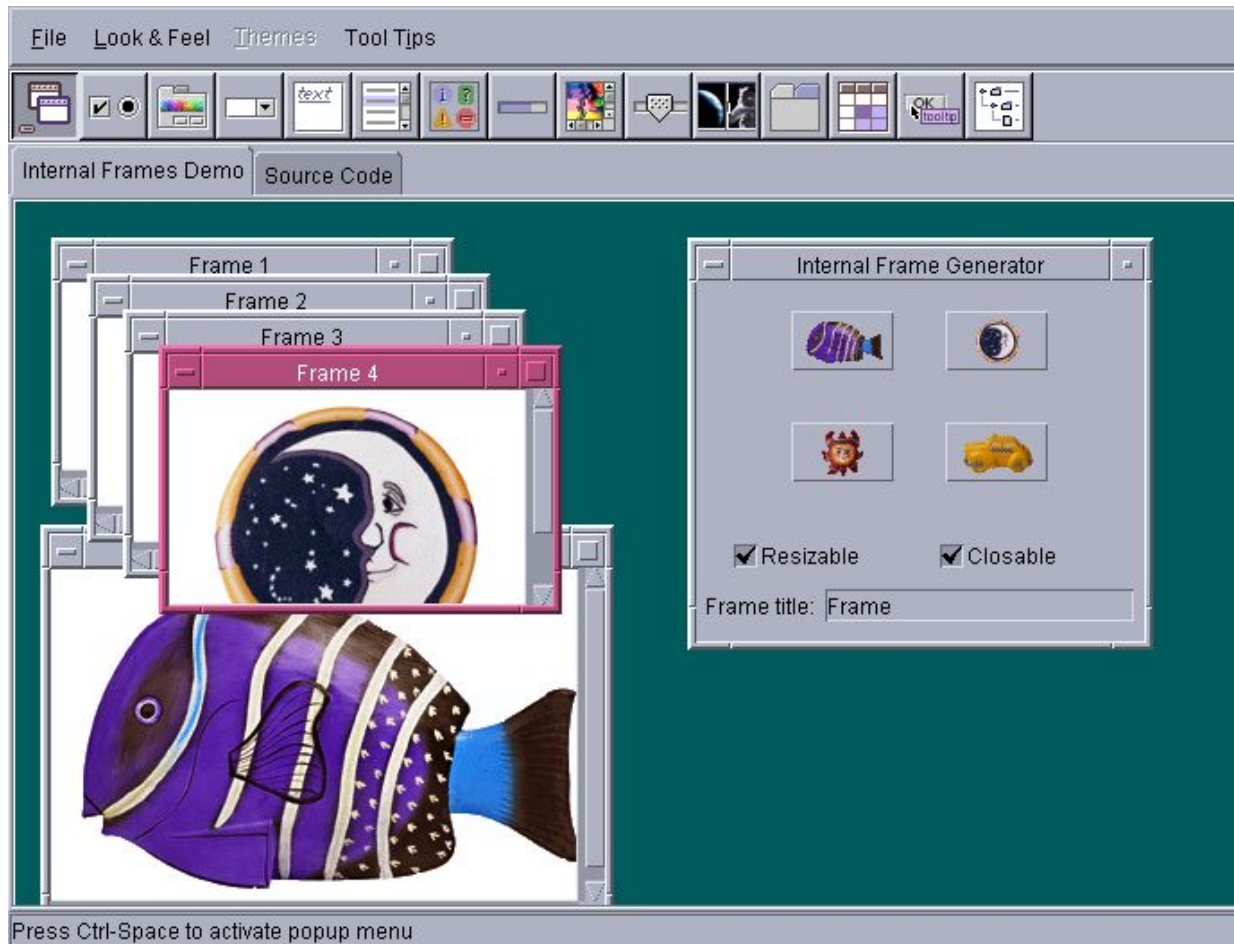
UIManager.setLookAndFeel(s);
SwingUtilities.updateComponentTreeUI(getContentPane());
```



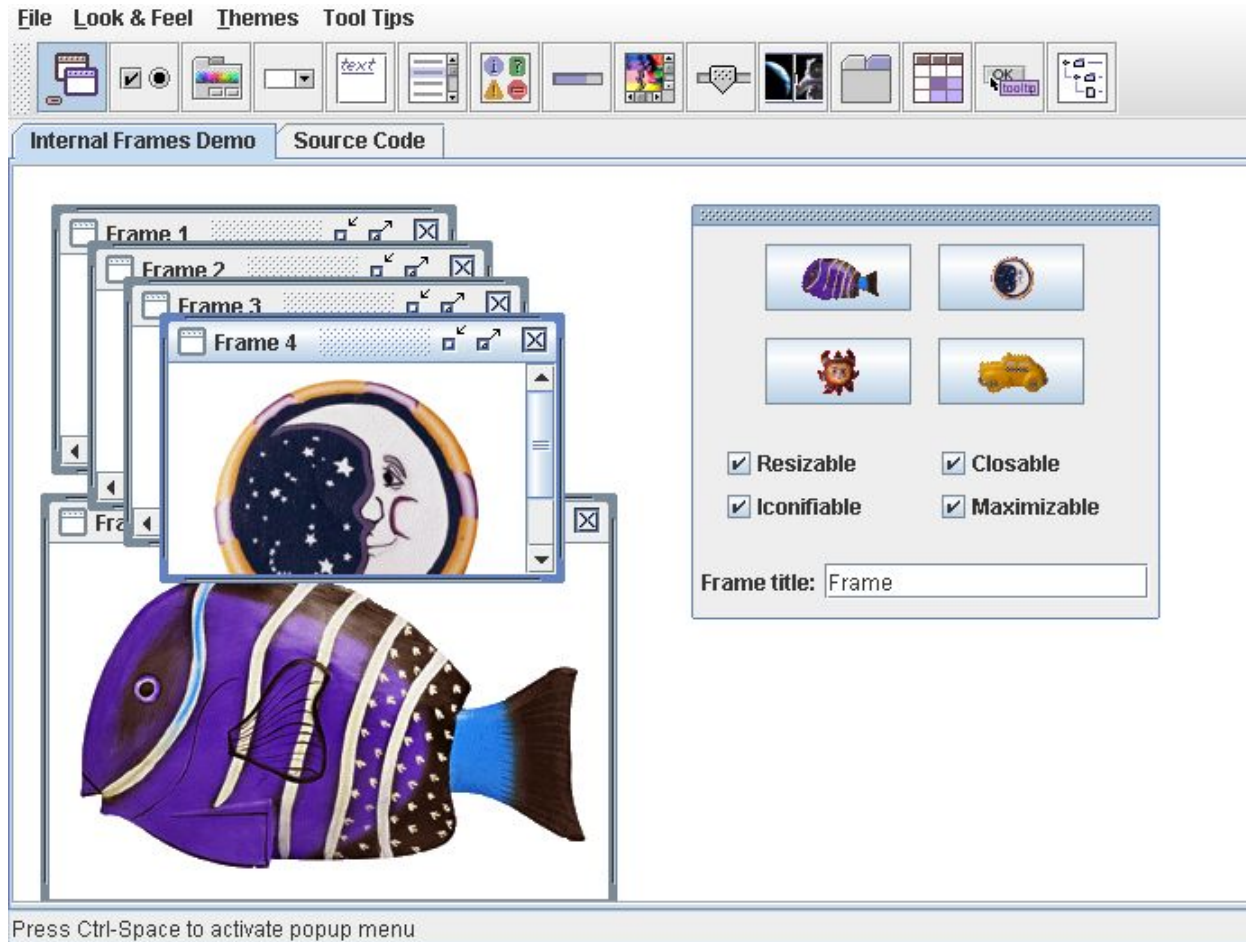
# Windows



# Motif



# Java (Metal)



# Оконные приложения в Swing

- Используется класс `javax.swing.JFrame`
- Содержимое окна находится на панели, ссылку на которую можно получить вызовом метода `getContentPane()`
- Параметрами окна можно управлять

# Пример оконного приложения

```
import java.awt.*;
import javax.swing.*;
public class CenteredFrameTest {
    public static void main(String[] args) {
        CenteredFrame frame = new CenteredFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
class CenteredFrame extends JFrame {
    public CenteredFrame() {
        Toolkit kit = Toolkit.getDefaultToolkit();
        Dimension screenSize = kit.getScreenSize();
        int screenHeight = screenSize.height;
        int screenWidth = screenSize.width;
        setSize(screenWidth / 2, screenHeight / 2);
        setLocation(screenWidth / 4, screenHeight / 4);
        Image img = kit.getImage("icon.gif");
        setIconImage(img);        setTitle("CenteredFrame");
    }
}
```

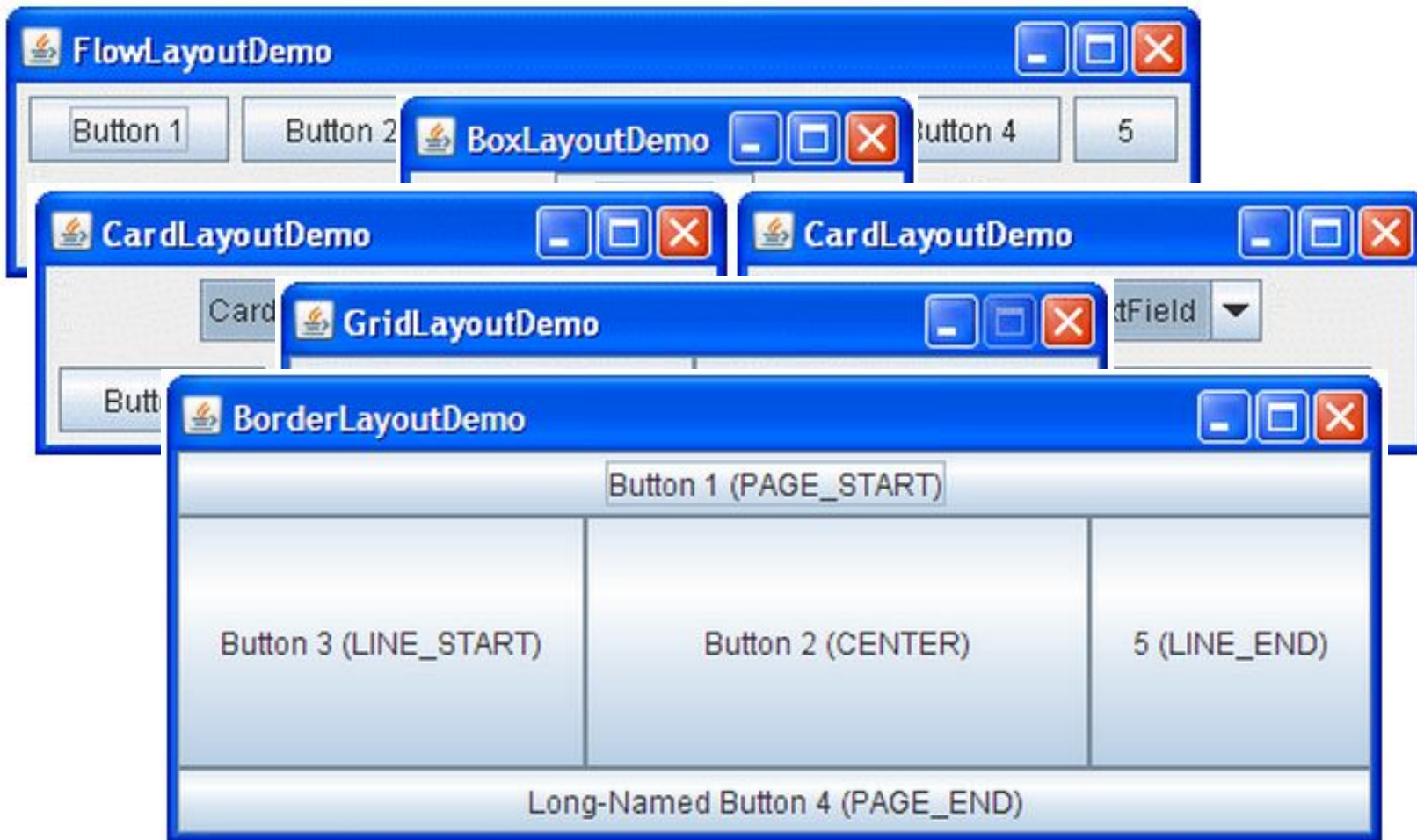
# Работа с меню

- `javax.swing.JMenuBar`  
Панель меню в верхней части окна
- `javax.swing.JMenu`  
Меню как таковое
- `javax.swing.JMenuItem`  
Единичный элемент меню
- `javax.swing.JCheckBoxMenuItem`  
Элемент меню «флажок»
- `javax.swing.JRadioButtonMenuItem`  
Элемент меню «радио-кнопка»
- `javax.swing.JPopupMenu`  
Всплывающее меню

# Менеджеры компоновки

- Управляют размещением компонентов в контейнере, учитывая параметры этих компонентов (например, предпочтительный размер)
- Реализуют интерфейс `java.awt.LayoutManager`
- Устанавливаются с помощью метода `setLayout()` контейнера
- Примеры простых компоновок:
  - Простая поточная компоновка `FlowLayout`
  - Поточная компоновка по вертикали/горизонтали `BoxLayout`
  - Компоновка в сетку `GridLayout`
  - Компоновка «Колода карт» `CardLayout`
  - Сетка с настраиваемыми размерами `GridBagLayout`
  - Граничная компоновка `BorderLayout`

# Примеры размещения компонентов





# Обработка событий

- Событие
  - Классы событий в пакетах
    - `java.awt.event`
    - `javax.swing.event`
- Источник
  - `public void addTypeListener (TypeListener el) throws java.util.TooManyListenersException`
  - `public void removeTypeListener (TypeListener el)`
- Слушатель
  - Должен реализовывать методы для приема и обработки уведомлений
  - Существует набор интерфейсов `TypeListener`, описывающих методы обработки событий
  - Класс слушателя должен реализовывать интерфейс, соответствующий событию

# Некоторые типы и порождаемые события

| Компонент | Событие  | Значение  |
|-----------|--|---|
| Component | <code>ComponentEvent</code>                          | Элемент либо перемещен, либо он стал скрытым, либо видимым  |
|           | <code>FocusEvent</code>                              | Элемент получил или потерял фокус ввода   |
|           | <code>KeyEvent</code>                                | Пользователь нажал или отпустил клавишу   |
|           | <code>MouseEvent</code> ,<br><code>MouseEvent</code> | Пользователь нажал или отпустил кнопку мыши, либо курсор мыши вошел или покинул область, занимаемую элементом, либо пользователь просто переместил мышь или переместил мышь при нажатой кнопке мыши |

# Некоторые типы и порождаемые события

| Компонент                      | Событие                      | Значение  |
|--------------------------------|------------------------------|---|
| <code>Container</code>         | <code>ContainerEvent</code>  | Элемент добавлен в контейнер или удален из него   |
| <code>Window</code>            | <code>WindowEvent</code>     | Окно было открыто, закрыто, представлено в виде пиктограммы, восстановлено или требует восстановления |
| <code>JButton</code>           | <code>ActionEvent</code>     | Пользователь нажал кнопку   |
| <code>JScrollBar</code>        | <code>AdjustmentEvent</code> | Пользователь осуществил прокрутку   |
| <code>JCheckBoxMenuItem</code> | <code>ItemEvent</code>       | Пользователь поставил или снял флажок   |
|                                | <code>ActionEvent</code>     | Пользователь выбрал пункт меню  |

# Пример приложения

## Часть 1

```
package swingdemo;

import javax.swing.JOptionPane;

public class DemoFrame extends javax.swing.JFrame {
    private javax.swing.JButton clearButton;
    private javax.swing.JScrollPane scrollPane;
    private javax.swing.JTextArea textArea;

    public DemoFrame() {
        initComponents();
    }

    private void initComponents() {
        scrollPane = new javax.swing.JScrollPane();
        textArea = new javax.swing.JTextArea();
        clearButton = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
```

# Пример приложения

## Часть 2

```
textArea.setColumns(20);
textArea.setRows(5);
scrollPane.setViewportView(textArea);

clearButton.setText("Clear");
clearButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        clearButtonActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
```

# Пример приложения

## Часть 3

```
layout.setHorizontalGroup(  
  
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
    .addGroup(layout.createSequentialGroup()  
        .addContainerGap()  
  
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment  
.LEADING)  
    .addComponent(scrollPane, javax.swing.GroupLayout.DEFAULT_SIZE,  
                    380, Short.MAX_VALUE)  
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,  
                layout.createSequentialGroup()  
    .addGap(0, 0, Short.MAX_VALUE)  
    .addComponent(clearButton))  
    .addContainerGap())));
```

# Пример приложения

## Часть 4

```
layout.setVerticalGroup(  
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
        .addGroup(layout.createSequentialGroup()  
            .addContainerGap()  
            .addComponent(scrollPane, javax.swing.GroupLayout.PREFERRED_SIZE,  
                251, javax.swing.GroupLayout.PREFERRED_SIZE)  
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
            .addComponent(clearButton)  
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,  
                Short.MAX_VALUE)  
        )  
    );
```

# Пример приложения

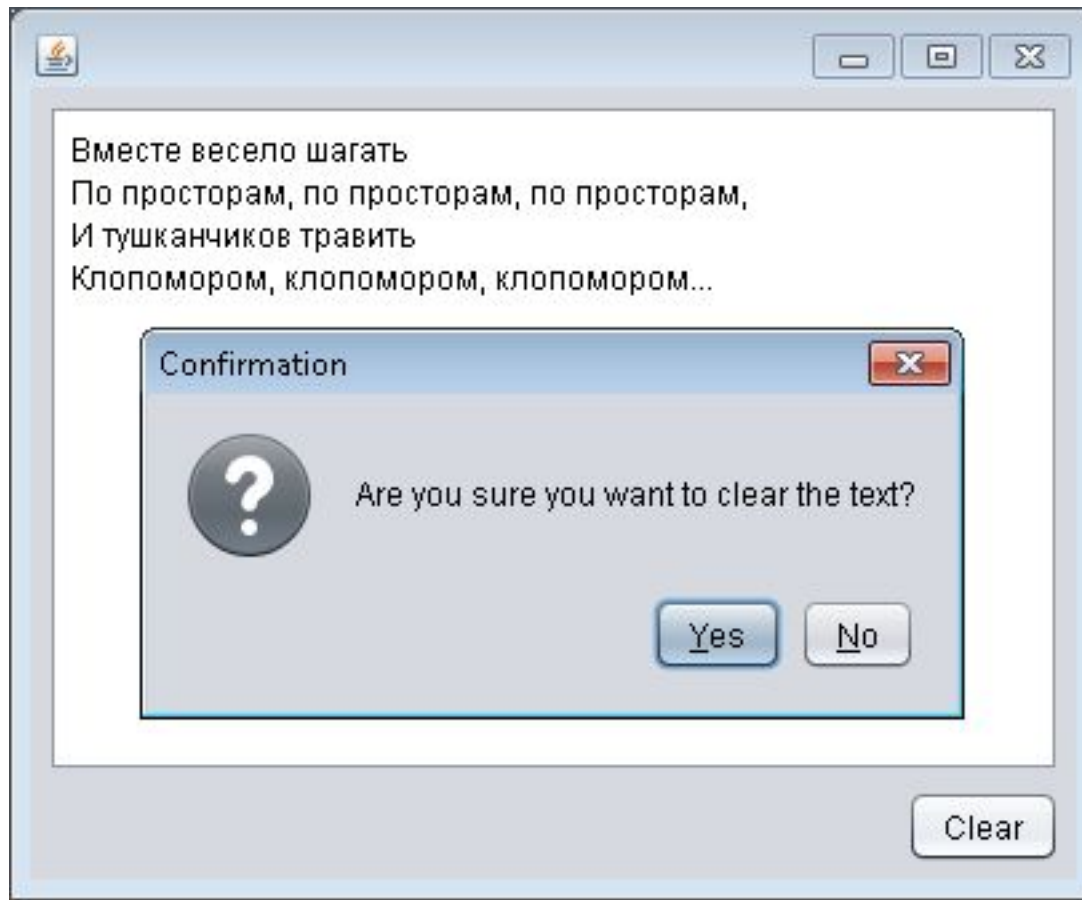
## Часть 5

```
    pack();
}

private void clearButtonActionPerformed(java.awt.event.ActionEvent evt) {
    if (JOptionPane.showConfirmDialog(
        rootPane,
        "Are you sure you want to clear the text?",
        "Confirmation",
        JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION) {
        textArea.setText("");
    }
}
}
```



# Общий вид окна программы-примера



# События, слушатели и методы

| Класс события                | Интерфейс слушателя             | Методы слушателя  |
|------------------------------|---------------------------------|---|
| <code>ActionEvent</code>     | <code>ActionListener</code>     | <code>actionPerformed()</code>  |
| <code>AdjustmentEvent</code> | <code>AdjustmentListener</code> | <code>adjustmentValueChanged()</code>   |
| <code>ComponentEvent</code>  | <code>ComponentListener</code>  | <code>componentHidden()</code><br><code>componentMoved()</code><br><code>componentResized()</code><br><code>componentShown()</code> |
| <code>ContainerEvent</code>  | <code>ContainerListener</code>  | <code>componentAdded()</code><br><code>componentRemoved()</code>  |
| <code>FocusEvent</code>      | <code>FocusListener</code>      | <code>focusGained()</code><br><code>focusLost()</code>  |
| <code>ItemEvent</code>       | <code>ItemListener</code>       | <code>itemStateChanged()</code>   |

# События, слушатели и методы

| Класс события           | Интерфейс слушателя              | Методы слушателя  |
|-------------------------|----------------------------------|---|
| <code>KeyEvent</code>   | <code>KeyListener</code>         | <code>keyPressed()</code><br><code>keyReleased()</code><br><code>keyTyped()</code>  |
| <code>MouseEvent</code> | <code>MouseListener</code>       | <code>mouseClicked()</code><br><code>mouseEntered()</code><br><code>mouseExited()</code><br><code>mousePressed()</code><br><code>mouseReleased()</code> |
| <code>MouseEvent</code> | <code>MouseMotionListener</code> | <code>mouseDragged()</code><br><code>mouseMoved()</code>  |

# События, слушатели и методы

| Класс события            | Интерфейс слушателя         | Методы слушателя   |
|--------------------------|-----------------------------|--|
| <code>WindowEvent</code> | <code>WindowListener</code> | <code>windowActivated()</code><br><code>windowClosed()</code><br><code>windowClosing()</code><br><code>windowDeactivated()</code><br><code>windowDeiconified()</code><br><code>windowIconified()</code><br><code>windowOpened()</code> |

# Классы-адаптеры

- Находятся в пакете `java.awt.event`
- Определены для интерфейсов слушателей того же пакета, содержащих более одного метода
- Являются пустыми реализациями соответствующего интерфейса
- Наследники классов-адаптеров переопределяют необходимые методы

# Нерассмотренные возможности

- Компоненты и виды порождаемых событий
- Создание своих «стилей» отображения  
`javax.swing.plaf`
- Вспомогательные классы  
`JOptionPane`, `JFileChooser`, `JColorChooser` и т.д.
- «Высокоинтеллектуальные» компоненты  
`JTree`, `JTable` и т.д.
- Drag&Drop  
`java.awt.dnd`
- Вывод на печать  
`java.awt.print`

# Понятие апплета

- **Апплет (applet)** – небольшое приложение, доступное на Интернет-сервере, транспортирующееся по Интернет, автоматически устанавливающееся и выполняемое как часть Web-документа
- В HTML-документ апплет встраивается с помощью тегов `<applet>` и `<object>`
- После доставки к клиенту апплет имеет ограниченный доступ к ресурсам системы

# Простейший апплет

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

/*
  <applet code="HelloWorldApplet" width=200 height=40>
  </applet>
*/

public class HelloWorldApplet extends Applet {
    public void paint(Graphics g) {
        g.drawString("Hello World!", 20, 20);
    }
}
```



# Тэг <applet>

```
<APPLET
  CODE = appletFile.class
  OBJECT = appletSerialFile
  WIDTH = pixels
  HEIGHT = pixels
  [ARCHIVE = jarFiles]
  [CODEBASE = codebaseURL]
  [ALT = alternateText]
  [NAME = appletInstanceName]
  [ALIGN = alignment]
  [VSPACE = pixels]
  [HSPACE = pixels]
>
[< PARAM NAME = AttributeName1 VALUE = AttributeValue1 >]
[< PARAM NAME = AttributeName2 VALUE = AttributeValue2 >]
[HTML-code if there is no Java support]
</APPLET>
```

# Тэг <applet>

```
<html>
  <head>
    <title>Simple Graph</title>
  </head>
  <body>
    <h1>Simple Graph</h1>
    <applet code="GraphApplet.class" width="300" height="120">
      alt="Browser understands tag but isn't running the applet"
      Browser is completely ignoring the tag!
    </applet>
  </body>
</html>
```

# Tэг <object>

```
<html>
  <head>
    <title>Simple Graph</title>
  </head>
  <body>
    <h1>Simple Graph</h1>
    <!--[if !IE]>-->
    <object classid="java:GraphApplet.class"
            type="application/x-java-applet"
            height="300" width="550" >
    <!--<![endif]-->
    <object classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
            height="300" width="550" >
      <param name="code" value="GraphApplet" />
    </object>
    <!--[if !IE]>-->
    </object>
    <!--<![endif]-->
  </body>
</html>
```

# Отладочная печать

- Может выводиться на консоль и в статусную строку программы просмотра апплетов
- В браузере можно получить доступ к консоли:  
Netscape: Options>Show Java Console  
IE: Tools>Sun Java Console
- А можно и не получить...

# Класс Applet

- `java.applet.Applet`
- Является классом-предком для любого апплета, включаемого в web-страницу или просматриваемого в Java Applet Viewer
- При наследовании обычно переопределяется ряд методов

# Скелетная структура апплета

- `void init()`  
Вызывается один раз при инициализации
- `void start()`  
Вызывается каждый раз при выводе документа, содержащего апплет, на экран
- `void stop()`  
Вызывается каждый раз, когда браузер покидает документ, содержащий апплет
- `void destroy()`  
Вызывается один раз, когда выполнение апплета заканчивается

# Апплеты в Swing

- Используется класс `javax.swing.JApplet`
- Содержимое апплета находится на панели, ссылку на которую можно получить вызовом метода `getContentPane()`
- Для добавление элементов используется её метод `add()`
- Апплет может являться полноценным Swing-приложением

# Пример апплета

## с использованием КОМПОНЕНТОВ

```
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.applet.*;
import javax.swing.*;
public class SwingApplet extends JApplet {
    JButton button;
    public void init() {
        String laf = UIManager.getSystemLookAndFeelClassName();
        try {
            UIManager.setLookAndFeel(laf);
        } catch (UnsupportedLookAndFeelException exc) {
            System.err.println("Warning: UnsupportedLookAndFeel: " + laf);
        } catch (Exception exc) {
            System.err.println("Error loading " + laf + ": " + exc);
        }
        getContentPane().setLayout(new FlowLayout());
        button = new JButton("Hello, I'm a Swing Button!");
        getContentPane().add(button);
    }
}
```



# Пример апплета с обработкой событий

```
/* <applet code = "Scribble2" width=640 height=480> </applet> */
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class Scribble2 extends JApplet implements
    MouseListener, MouseMotionListener {

    private int lastX, lastY;
    public void init() {
        this.addMouseListener(this) ;
        this.addMouseMotionListener(this) ;
    }

    public void mousePressed(MouseEvent e) {
        lastX = e.getX();
        lastY = e.getY();
    }
}
```

# Пример апплета с обработкой событий

```
public void mouseDragged(MouseEvent e) {
    Graphics g = this.getGraphics();
    int x = e.getX(), y = e.getY();
    g.drawLine(lastX, lastY, x, y);
    lastX = x; lastY = y;
}

public void mouseReleased(MouseEvent e) {}
public void mouseClicked(MouseEvent e) {}
public void mouseEntered(MouseEvent e) {}
public void mouseExited(MouseEvent e) {}
public void mouseMoved(MouseEvent e) {}
}
```

# Результат работы программы



Спасибо за внимание!

# Дополнительные источники

- Арнолд, К. Язык программирования Java [Текст] / Кен Арнолд, Джеймс Гослинг, Дэвид Холмс. – М. : Издательский дом «Вильямс», 2001. – 624 с.
- Вязовик, Н.А. Программирование на Java. Курс лекций [Текст] / Н.А. Вязовик. – М. : Интернет-университет информационных технологий, 2003. – 592 с.
- Хорстманн, К. Java 2. Библиотека профессионала. Том 1. Основы [Текст] / Кей Хорстманн, Гари Корнелл. – М. : Издательский дом «Вильямс», 2010 г. – 816 с.
- Хорстманн, К. Java 2. Библиотека профессионала. Том 2. Тонкости программирования [Текст] / Кей Хорстманн, Гари Корнелл. – М. : Издательский дом «Вильямс», 2010 г. – 992 с.
- JavaSE APIs & Documentation [Электронный ресурс]. – Режим доступа: <http://www.oracle.com/technetwork/java/javase/documentation/api-jsp-136079.html>, дата доступа: 21.10.2011.
- Java Media APIs [Электронный ресурс]. – Режим доступа: <http://java.sun.com/javase/technologies/desktop/media/>, дата доступа: 21.10.2011.