

GUI

ЛЕКЦИОНЕН КУРС “ПРОГРАМИРАНЕ НА JAVA”



СТРУКТУРА НА ЛЕКЦИЯТА

- Въведение
- Събитиен модел
- Графични компоненти
- Примери

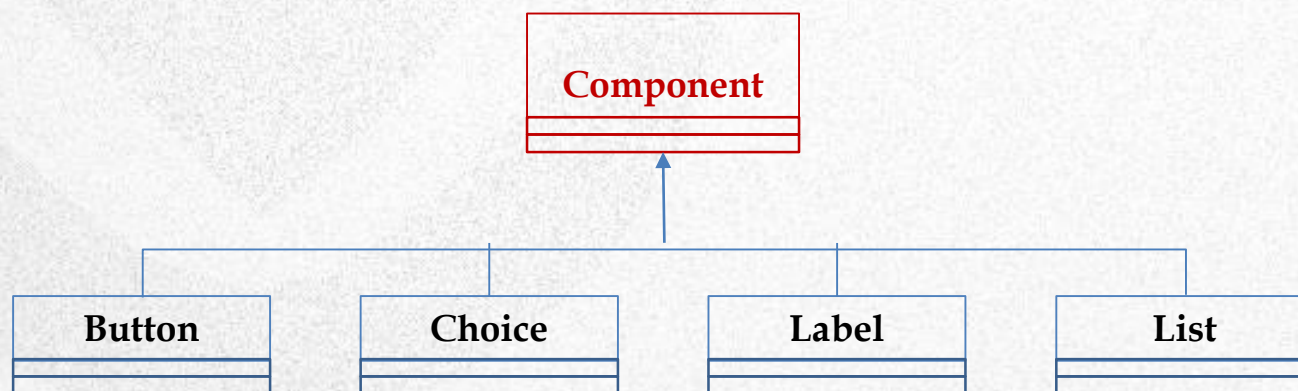
ИСТОРИЯ

- Основен принцип при проектирането: „Простите неща да се правят лесни, трудните - възможни“
- Цел на библиотеката за графичен потребителски интерфейс (GUI)
 - Изграждане на GUI, изглеждащ добре на всички платформи
- Първоначално не е постигната
 - AWT (Abstract Window Toolkit), Java 1.0 – еднакво посредствено за всички системи
 - С много ограничения – само 4 шрифта, не предоставя по-сложни GUI елементи
- Подобрена версия Java 1.1 AWT
 - По-ясен, ОО подход
 - Добавяне на JavaBeans – компонентен програмен модел
- Java 2 – окончателно завършена трансформация на стария AWT
 - GUI частта се нарича Swing
 - Голяма група лесни за употреба и за усвояване компоненти (JavaBeans)

ПРЕГЛЕД

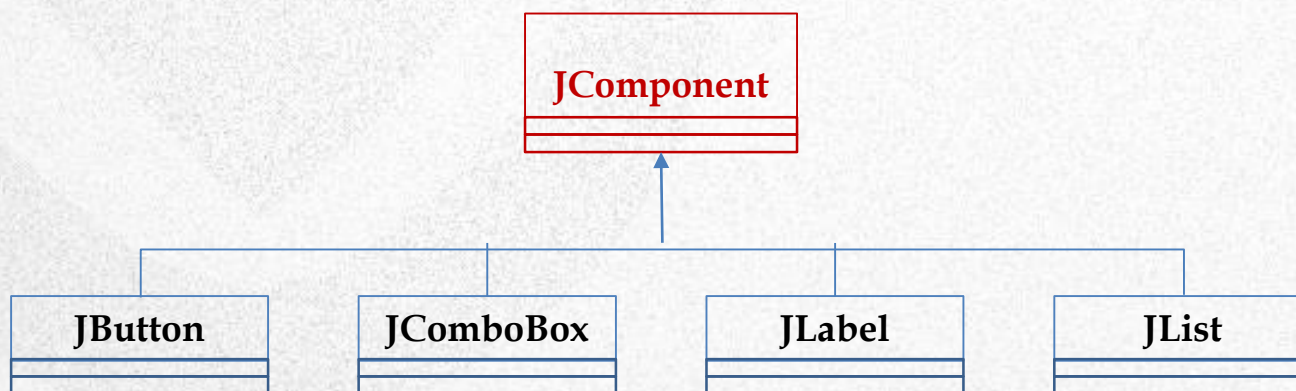
- AWT – пакет “java.awt”
 - Използва предимно компоненти, които са присъщи за прилежащата операционна система
 - Поради изискването за платформена независимост, обхваща само такива компоненти, които са общи за поддържаните от JVM операционни системи
 - Обикновено класовете: Button, Label, List, Textfield
- Swing библиотека
 - Разширение на AWT
 - Компонентите са напълно реализирани на Java
 - По-голяма независимост
 - Използват Model-View-Controller (MVC) архитектура
 - Един компонент се декомпозира на модел данни, представяне и контролна единица
- Java 2D
 - За създаване на графични потребителски интерфейси е съществено компонентите да бъдат интересно проектирани
 - Предоставя различни комплексни графични обекти
 - Линии с различна дебелина, канали, промяна на цветове, ротации, ...

КОМПОНЕНТИ: JAVA.AWT



Компонентите се моделират посредством класове, наследници на `java.awt.Component`

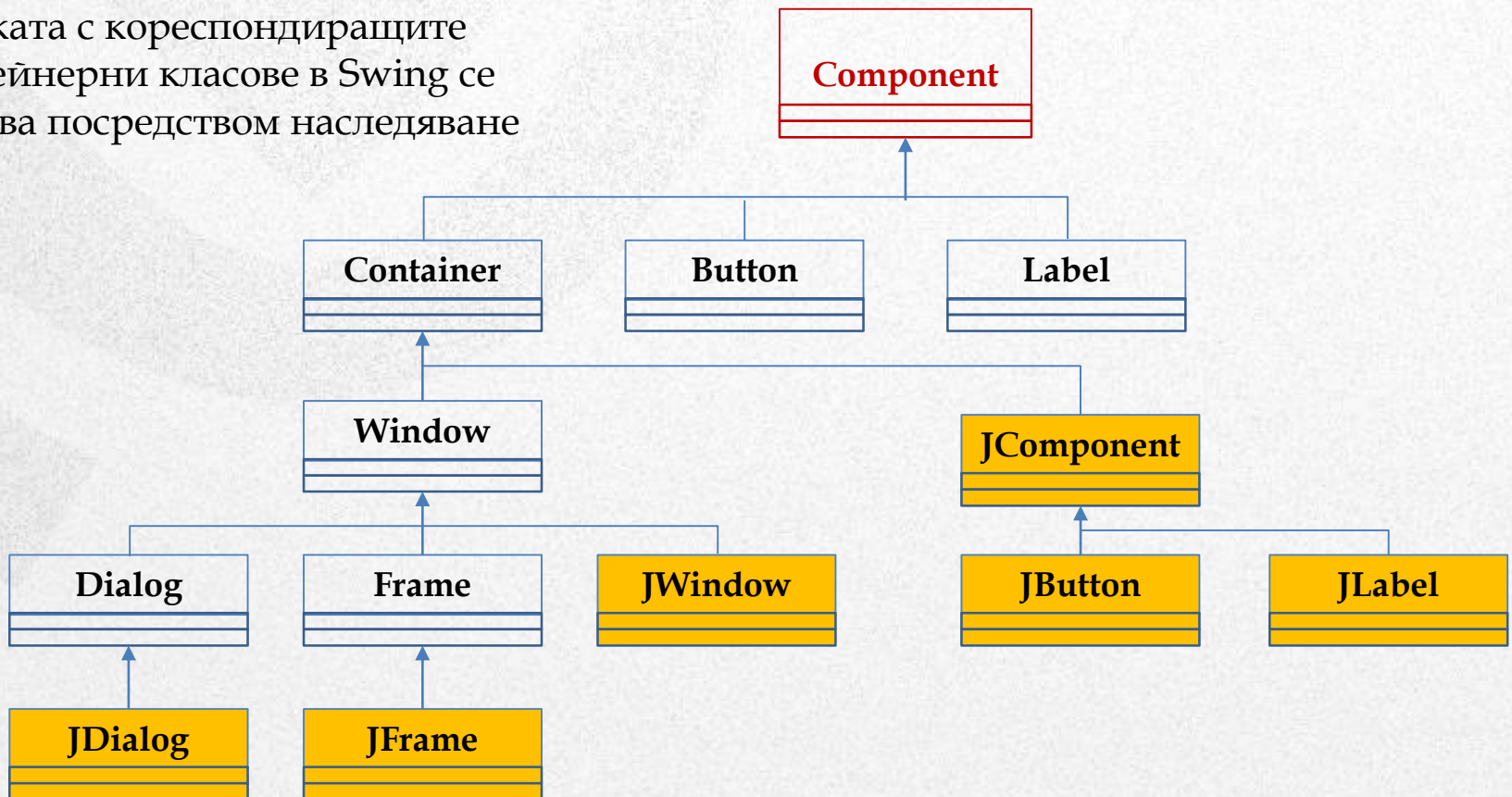
КОМПОНЕНТИ: JAVAX.SWING



- Компонентите се моделират посредством класове, наследници на `javax.swing.JComponent`
- В сравнение с AWT, префикс “J”
- Не са наследници на AWT компонентите

КОНТЕЙНЕРИ

- Контейнерите в AWT следва строга йерархия
- Връзката с кореспондиращите контейнерни класове в Swing се решава посредством наследяване



ПРИМЕР

```
final JLabel label = new JLabel("Label:");  
label.add(new JButton("First Button");  
label.add(new JButton("Second Button");  
  
System.out.println(label.getComponentCount());
```

2

УПРАВЛЕНИЕ НА ОФОРМЛЕНИЕТО

- Разполагането на компонентите в контейнерите
- Става автоматично, в зависимост от избраната схема, като се използва ефективно наличното място
- Компонентите могат също да се разполагат „ръчно“ посредством задаване на абсолютни координати

ОБРАБОТКА НА СЪБИТИЯ

- Всички, разгледани дотук градивни елементи на един GUI се занимават с „оптиката“, т.е. как ще изглежда един интерфейс
- За това, нещо да става когато се натисне един бутон напр., е отговорна обработката на събития
- С помощта на собствени реализации на предварително дефинирани интерфейси и използване на налични класове може да се реагира целево на различни събития

ПРИМЕР

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
public class FirstSwingExample {
    public static void main(final String[] args) {

        final JFrame frame = new JFrame("First Swing");

        frame.add(new JLabel("Text-Label:"));
        frame.add(new JButton("Button -- Press Me!"));

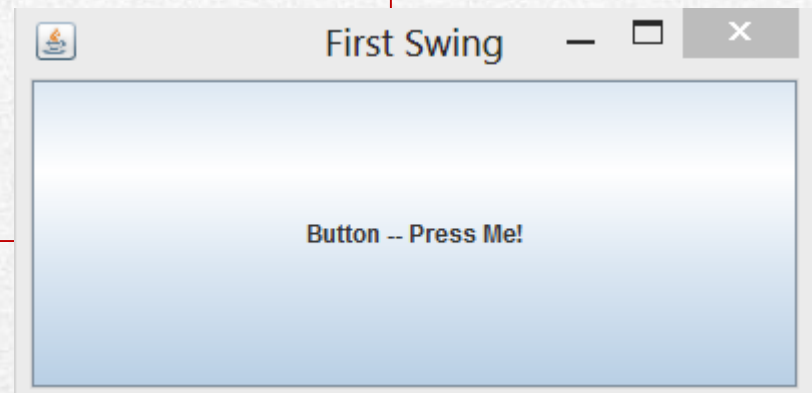
        frame.setSize(400, 200);
        frame.setVisible(true);
    }
}
```

setSize(): задава размера на JFrame

setVisible(): фреймът става видим върху екрана

ПРИМЕР

```
import javax.swing.JButton;  
import javax.swing.JFrame;  
import javax.swing.JLabel;  
public class FirstSwingExample {  
    public static void main(final String[] args) {  
  
        final JFrame frame = new JFrame("First Swing");  
  
        frame.add(new JLabel("Text-Label:"));  
        frame.add(new JButton("Button -- Press Me!"));  
  
        frame.setSize(400, 200);  
        frame.setVisible(true);  
    }  
}
```



АПЛЕТИ

- Аплет
 - Малка програма, работеща в рамките на уеб браузър
- Предназначение
 - Разширява функционалността на уеб страниците в браузъра
 - Мощно средство за програмира от страна на клиента, което е съществена част от уеб
- По тази причина аpletите трябва да бъдат сигурни
 - Те са ограничени по отношение на нещата, които могат да правят

ОГРАНИЧЕНИЯ НА АПЛЕТИТЕ

- Програмирането с аплети е значително ограничено
 - Често се нарича „пясъчна кутия“
 - Винаги има някой, който наблюдава
 - Това е run-time системата за сигурност на Java
- Приложенията са извън „пясъчната кутия“
 - Swing може да се използва също за GUI за обикновени приложения
- Ограничения:
 - Няма достъп (четене и писане) до локалния диск – не бихме искали един аplet да може да чете и изпраща лична информация по Интернет без наше разрешение
 - Изискват значително време – всеки път е необходимо да се сваля целия код, включително обръщенията към сървъра за всеки клас
 - По тази причина трябва винаги да са пакетирани в .jar файлове
 - Комбинира всички компоненти на аплета (вкл. .class файловете)

ПРЕДИМСТВО НА АПЛЕТИТЕ

- Не изискват инсталиране
 - Притежават истинска платформена независимост
 - Не се налага да променяме кода за различни платформи
 - Не се налага да се правят инсталации
- Инсталирането е автоматично - всеки път, когато се зарежда съдържащата го уеб страница
 - Не могат увреждат някоя система
 - Поради системата за сигурност на Java и структурата на аpletите
- Това прави Java предпочитан за Интернет клиент/сървър програмиране

РАМКИ ЗА ПРИЛОЖЕНИЯ

- Обикновено библиотеките се групират по функционалност
- Рамки за приложения
 - Библиотеки, осигуряващи клас или класове, които създават основното поведение на всяко приложение от определен тип
 - За да се адаптира рамката за решаване на конкретен проблем наследяваме от тези класове и предефинираме интересующите ни методи
 - Контролният механизъм на рамката ще извиква предефинираните методи при необходимост
- Рамките са добър пример за принципа „разделяне на нещата, които се променят, от тези, които остават постоянни“
- Аплетите се изграждат като се използва рамка за приложения
 - Наследяваме от класа JApplet и предефинираме подходящите методи

ОСНОВНИ МЕТОДИ НА JARPLET

Метод	Операция
init()	<ul style="list-style-type: none">Начална инициализация, вкл. разположението на компоненти на аплетаВинаги се предефинира
start()	<ul style="list-style-type: none">Извиква се всеки път когато аpletът влезе в зоната на видимост на брауъраПозволява на аплета да започне нормалното си опериране
stop()	<ul style="list-style-type: none">Извиква се всеки път когато аpletът излезе от зоната на видимост на брауъраИзвиква се преди destroy()
destroy()	<ul style="list-style-type: none">Извиква се когато аpletът се премахва от страницатаОкончателно се освобождават ресурсите, когато аpletът не се използва повече

ПРИМЕР

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

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

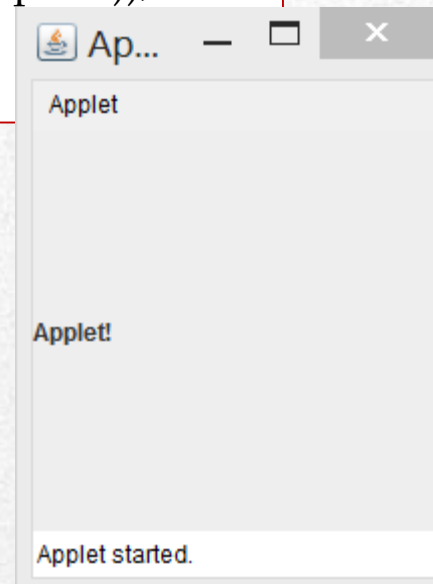
- Не се изисква main() - стартовият код се поставя в init()
- JLabel – поставя текстов етикет в аплета
- “J” – за Swing компонентите
- За примера Label за старата библиотека AWT
- add() се използва за поставяне на компонентите върху формата (в Swing е включен в init(), в AWT се използва самостоятелно)
- Swing изисква всички компоненти да се добавят към “content pane”

ПРИМЕР

1 Какъв резултат?

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

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



РАМКА ЗА ПОКАЗВАНЕ

```
import javax.swing.*;
import java.awt.event.*;
public class Console {
    public static String title(Object o) {
        String t = o.getClass().toString();
        if(t.indexOf("class") != -1)
            t = t.substring(6);
        return t;
    }
    public static void setupClosing(JFrame frame) {
        frame.addWindowListener(new WindowListener() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }
    public static void
    run(JFrame frame, int width, int height) {
        setupClosing(frame);
        frame.setSize(width, height);
        frame.setVisible(true);
    }
}
```

```
public static void
run(JApplet applet, int width, int height) {
    JFrame frame = new JFrame(title(applet));
```

- Класът Console е съставен изцяло от static методи
- Методът title() се използва за извличане на името на класа (използва RTTI) от всеки обект
- Премахва думата "class" – използва indexOf() за определяне, дали тази дума е налична и substring() за създаване на новия низ без "class" и интервали

```
run(JPanel panel, int width, int height) {
    JFrame frame = new JFrame(title(panel));
    setupClosing(frame);
    frame.getContentPane().add(panel);
    frame.setSize(width, height);
    frame.setVisible(true);
}
}
```


РАМКА ЗА ПОКАЗВАНЕ

```
import javax.swing.*;
import java.awt.event.*;
public class Console {
1 public static String title(Object o) {
    String t = o.getClass().toString();
    if(t.indexOf("class") != -1)
        t = t.substring(6);
    return t;
}
2 public static void setupClosing(JFrame frame) {
    frame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            System.exit(0);
        }
    });
}
public static void
run(JFrame frame, int width, int height) {
    setupClosing(frame);
    frame.setSize(width, height);
    frame.setVisible(true);
}
```

```
public static void
run(JApplet applet, int width, int height) {
    JFrame frame = new JFrame(title(applet));
    setupClosing(frame);
    frame.getContentPane().add(applet);
    frame.setSize(width, height);
    applet.init();
    applet.start();
    frame.setVisible(true);
}
public static void
run(JPanel panel, int width, int height) {
    JFrame frame = new JFrame(title(panel));
    setupClosing(frame);
    frame.add(panel);
    frame.setSize(width, height);
    frame.setVisible(true);
}
```

- Методът `setupClosing()` се използва за скриване на кода, който причинява даден `JFrame` да излиза от програмата, когато този `JFrame` бъде затворен
- Поведението по подразбиране е да не се прави нищо, така че ако не извикаме този метод приложението няма да се затвори

РАМКА ЗА ПОКАЗВАНЕ

```
import javax.swing.*;
import java.awt.event.*;
public class Console {
1 public static String title(Object o) {
    String t = o.getClass().toString();
    if(t.indexOf("class") != -1)
        t = t.substring(6);
    return t;
}
2 public static void setupClosing(JFrame frame) {
    frame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            System.exit(0);
        }
    });
}
3 public static void
run(JFrame frame, int width, int height) {
    setupClosing(frame);
    frame.setSize(width, height);
    frame.setVisible(true);
}
```

```
3 public static void
run(JApplet applet, int width, int height) {
    JFrame frame = new JFrame(title(applet));
    setupClosing(frame);
    frame.getContentPane().add(applet);
    frame.setSize(width, height);
    applet.init();
    applet.start();
    frame.setVisible(true);
}
3 public static void
run(JPanel panel, int width, int height) {
    JFrame frame = new JFrame(title(panel));
    setupClosing(frame);
    frame.getContentPane().add(panel);
    frame.setSize(width, height);
    frame.setVisible(true);
}
}
```

- Методът run() е преопределен да работи с JApplet, JPanel и JFrame
- init() и start() се използват само при JApplet

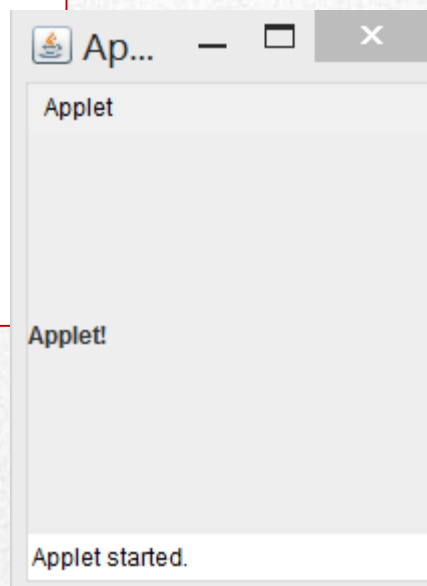
ПРИМЕР

1 Какъв резултат?

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

public class Applet1d extends JApplet {
    public void init() {
        getContentPane().add(new JLabel("Applet!"));
    }
    public static void main(String[] args) {
        Console.run(new Applet1d(), 100, 50);
    }
}
```

- Същият резултат
- Извикване през конзолата



ПРИМЕР: БУТОНИ

```
import javax.swing.*;
import java.awt.*;
import com.bruceeckel.swing.*;

public class Button1 extends JApplet {
    JButton
        b1 = new JButton("Button 1"),
        b2 = new JButton("Button 2");
    public void init() {
        Container cp = getContentPane();
        cp.setLayout(new FlowLayout());
        cp.add(b1);
        cp.add(b2);
    }
    public static void main(String[] args) {
        Console.run(new Button1(), 200, 50);
    }
}
```

- Създаването на един бутон е много просто – извиква се само конструкторът JButton с етикета, който искаме да поставим на бутона (могат да се задават също графични изображения)
- Поставяне на бутона във формата се прави в init()
- Използваме също „мениджър на разполагането“ (setLayout()) – типът FlowLayout позволява контролите да се подреждат равномерно върху формата

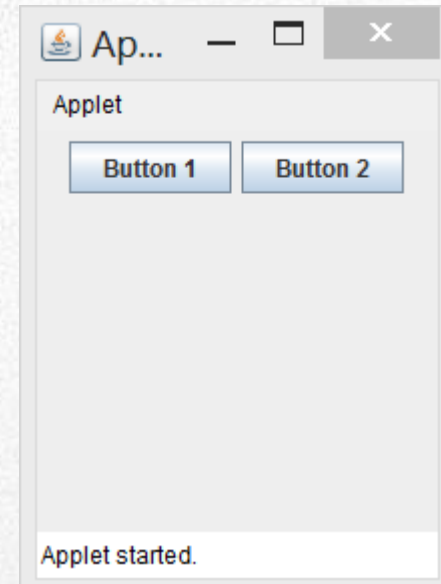
ПРИМЕР

1

Какъв резултат?

```
import javax.swing.*;
import java.awt.*;
import com.bruceeckel.swing.*;

public class Button1 extends JApplet {
    JButton
        b1 = new JButton("Button 1"),
        b2 = new JButton("Button 2");
    public void init() {
        Container cp = getContentPane();
        cp.setLayout(new FlowLayout());
        cp.add(b1);
        cp.add(b2);
    }
    public static void main(String[] args) {
        Console.run(new Button1(), 200, 50);
    }
}
```



При натискане на бутоните
няма никакво действие

КОНТРОЛ НА РАЗПОЛОЖЕНИЕТО

- Начинът, по който се разполагат компонентите върху формата при Java, се различава от други GUI системи
 - Всичко е код
 - Няма никакви „ресурси“
 - Не се използва „абсолютно позициониране“
 - Разполагането се контролира от „мениджър на разположението“ (layout manager)
 - Той решава как да се разположат компонентите, според начина, по който ги добавяме с `add()`
 - Размерът, формата и разположението на компонентите ще бъдат различни при смяна на един мениджър с друг
 - Освен това, мениджърите се адаптират към размерите на използвания аplet

РАБОТА С МЕНИДЖЪРИТЕ

- Компонентите JApplet, JFrame, JWindow и JDialog
 - Създават Container с помощта на getContentPane(), който може да съдържа и показва компоненти
 - В Container има метод setLayout(), позволяващ избирането на различен мениджър на разположението
- Класът JPanel съдържа и показва компонентите директно
 - Директно се установява мениджърът на разположението без да се използва панела със съдържанието

ПРИМЕР: BORDERLAYOUT

1 Какъв резултат?

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

```
public class BorderLayoutExample extends JApplet {  
  
    public void init() {  
        Container c = getContentPane();  
        c.setLayout(new BorderLayout());  
        c.add(new JButton("One"), BorderLayout.NORTH);  
        c.add(new JButton("Two"), BorderLayout.WEST);  
        c.add(new JButton("Three"), BorderLayout.CENTER);  
        c.add(new JButton("Four"), BorderLayout.EAST);  
        c.add(new JButton("Five"), BorderLayout.SOUTH);  
        c.add(new JButton("Six"), BorderLayout.SOUTH);  
    }  
}
```

- Поддържа 4 гранични и 1 централна области
- Когато искаме да добавим компонент към един панел използваме предефинирания метод add()
- По подразбиране CENTER



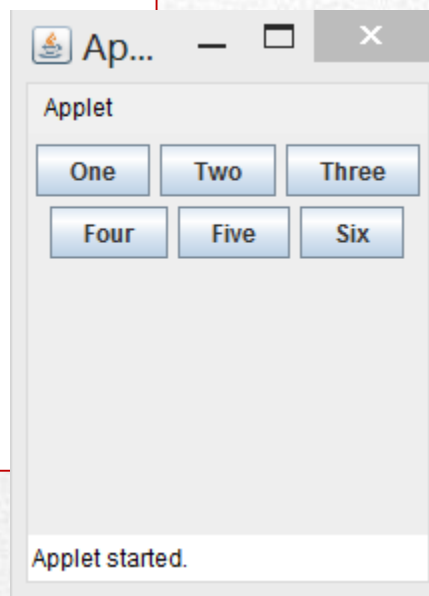
ПРИМЕР: FLOWLAYOUT

1 Какъв резултат?

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

```
public class FlowLayoutExample extends JApplet {  
  
    public void init () {  
        getContentPane().setLayout(new FlowLayout ());  
        getContentPane().add(new JButton("One"));  
        getContentPane().add(new JButton("Two"));  
        getContentPane().add(new JButton("Three"));  
        getContentPane().add(new JButton("Four"));  
        getContentPane().add(new JButton("Five"));  
        getContentPane().add(new JButton("Six"));  
    }  
}
```

- Компонентите се нареждат един след друг върху формата, отляво надясно, докато се запълни горното пространство
- След това се преминава към втория ред и т.н.
- Компонентите приемат „естествения“ си размер, т.е. сбити са до най-малките си размери



ПРИМЕР: GRIDLAYOUT

1 Какъв резултат?

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

```
public class GridLayoutExample extends JApplet {
```

```
    public void init() {  
        Container c = getContentPane();  
        c.setLayout(new GridLayout(2, 4));  
        c.add(new JButton("One"));  
        c.add(new JButton("Two"));  
        c.add(new JButton("Three"));  
        c.add(new JButton("Four"));  
        c.add(new JButton("Five"));  
    }  
}
```

- Позволява създаване таблица от компоненти
- Разполагат се отляво надясно и отгоре надолу
- В конструктора се задава броя на редовете и колоните



ПРИМЕР: BOXLAYOUT

1 Какъв резултат?

```
import javax.swing.*;
import java.awt.*;
public class BoxLayout1 extends JPanel {
    public void init() {
        JPanel jpv = new JPanel();
        jpv.setLayout(new BoxLayout(jpv, BoxLayout.Y_AXIS));
        for(int i = 0; i < 5; i++)
            jpv.add(new JButton(" " + i));
        JPanel jph = new JPanel();
        jph.setLayout(new BoxLayout(jph, BoxLayout.X_AXIS));
        for(int i = 0; i < 5; i++)
            jph.add(new JButton(" " + i));
        Container cp = getContentPane();
        cp.add(BorderLayout.EAST, jpv);
        cp.add(BorderLayout.SOUTH, jph);
    }
    public static void main(String[] args) {
        Console.run(new BoxLayout1(), 450, 200);
    }
}
```

- Малко по-различен от другите мениджъри на разположение
- Първият аргумент е формата, която трябва да бъде контролирана
- Вторият аргумент задава посоката на разполагането



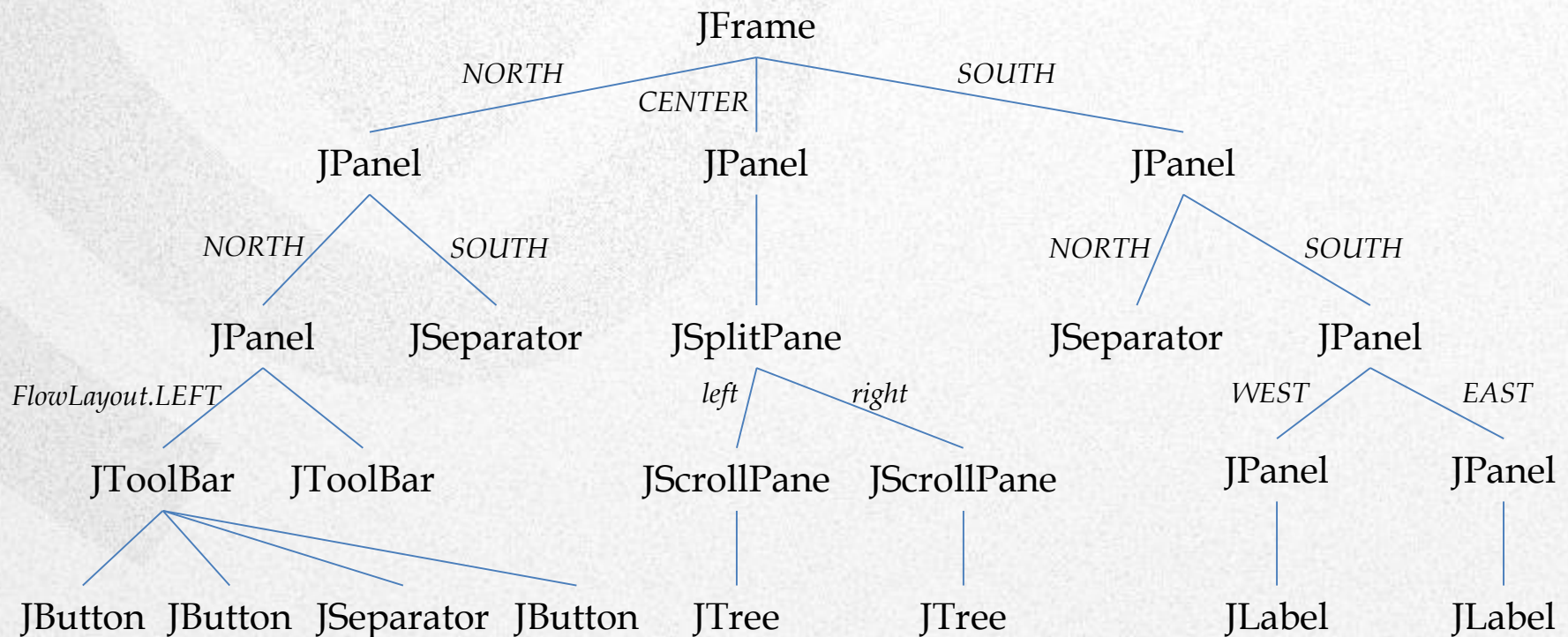
ПРИМЕР: КОМБИНИРАНЕ НА КОНТЕЙНЕРИ

The screenshot shows a Java Swing window titled "LayoutManagerCombinationExample". The window contains a JTree on the left and a table on the right. The JTree has a root node "JTree" with three children: "colors", "sports", and "food". The table has three columns: "Име", "Фамилия", and "Възраст". The table contains the following data:

Име	Фамилия	Възраст
Иван	Стоянов	34
Георги	Петров	28
Мария	Димитрова	29
Петър	Иванов	30
Стефан	Василев	31
Ива	Станчева	32

Below the table, there are two labels: "Left aligned info" and "Right aligned info".

ИЕРАРХИЧНА СТРУКТУРА НА КОНТЕЙНЕРА



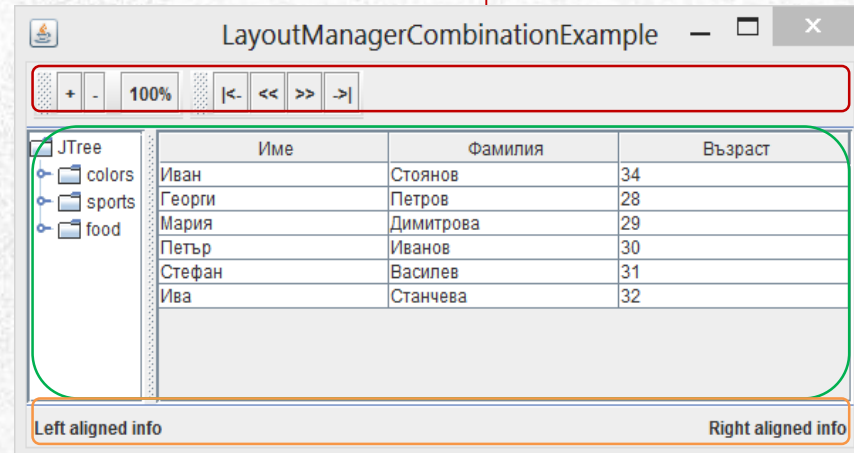
ПРИМЕР: КОМБИНИРАНЕ НА КОНТЕЙНЕРИ

```
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JSeparator;
import javax.swing.JSplitPane;
import javax.swing.JTable;
import javax.swing.JToolBar;
import javax.swing.JTree;
```

```
public class LayoutManagerCombinationExample {
    public static void main(final String[] args) {
        final JFrame frame = new JFrame("LayoutManagerCombinationExample");
```

```
        frame.add(createToolBarPanel(), BorderLayout.NORTH);
        frame.add(createCenterPanel(), BorderLayout.CENTER);
        frame.add(createStatusBarPanel(), BorderLayout.SOUTH);
```

```
        frame.setSize(600, 300);
        frame.setVisible(true);
```



- Основният прозорец е разделен на три области
- Разполагането на компонентите се управлява в режим BorderLayout

ПРИМЕР: КОМБИНИРАНЕ НА КОНТЕЙНЕРИ

```
private static JPanel createToolBarPanel() {  
  
    final JToolBar zoomToolBar = new JToolBar();  
    zoomToolBar.add(new JButton("+"));  
    zoomToolBar.add(new JButton("-"));  
    zoomToolBar.addSeparator();  
    zoomToolBar.add(new JButton("100%"));
```

```
    final JToolBar skipToolBar = new JToolBar();  
    skipToolBar.add(new JButton("| <-"));  
    skipToolBar.add(new JButton("<<"));  
    skipToolBar.add(new JButton(">>"));  
    skipToolBar.add(new JButton("-> |"));
```

```
    final JPanel toolbarPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));  
    toolbarPanel.add(zoomToolBar);  
    toolbarPanel.add(skipToolBar);
```

```
    final JPanel compoundPanel = new JPanel(new BorderLayout());  
    compoundPanel.add(toolbarPanel, BorderLayout.NORTH);  
    compoundPanel.add(new JSeparator(), BorderLayout.SOUTH);
```

```
    return compoundPanel;
```

Създават се два ToolBars

Ориентирането наляво е важно, понеже ToolBars ще бъдат разположени централно

Общ панел за Toolbar панела и JSeparator

ИЗПОЛЗВАНИ КОНТЕЙНЕРИ

JPanel:

- Използва се вграждане на контейнери, инстанциите на които съдържат различни компоненти
- Изграждане на йерархични композиции от контейнери

JSplitPane:

- Контейнер за различни компоненти (вкл. други контейнери)
- Разделя областта на две динамично променящи се подобласти

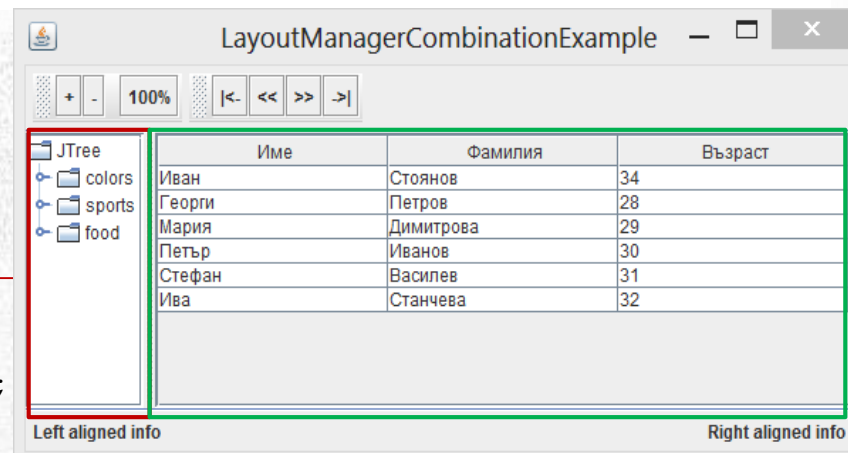
JScrollPane:

- Контейнер за различни компоненти (вкл. други контейнери)
- Позволява навигация върху цялата област на интегрирания компонент
- Особено полезен за интегриране на комплексни структури като напр., списъци, таблици, дървовидни структури

ПРИМЕР: КОМБИНИРАНЕ НА КОНТЕЙНЕРИ

1 Защо използваме JSplitPane?

```
private static JComponent createCenterPanel() {  
    final JSplitPane splitPane = new JSplitPane();  
    splitPane.setLeftComponent(new JScrollPane(new JTree()));  
  
    final String[] headers = { "Име", "Фамилия", "Възраст" };  
    final String[][] values = { { "Иван", "Стоянов", "34" }, { "Георги", "Петров", "28" },  
        { "Мария", "Димитрова", "29" }, { "Петър", "Иванов", "30" },  
        { "Стефан", "Василев", "31" }, { "Ива", "Станчева", "32" } };  
    final JScrollPane scrollPane = new JScrollPane(new JTable(values, headers));  
    splitPane.setRightComponent(scrollPane);  
  
    return splitPane;  
}
```



- Дървовидната структура може да се разположи WEST или CENTER - тук сме избрали CENTER
- Понеже BorderLayout не позволява разполагането на повече от един компонент в област, решаваме проблема посредством JSplitPane
- left, right – двете подобласти

ПРИМЕР: КОМБИНИРАНЕ НА КОНТЕЙНЕРИ

```
private static JPanel createStatusBarPanel() {  
    final JPanel statusBarPanel = new JPanel();  
    statusBarPanel.setLayout(new BorderLayout());  
  
    final JPanel leftPanel = new JPanel();  
    final JPanel rightPanel = new JPanel();  
  
    final JLabel info1 = new JLabel("Left aligned info");  
    final JLabel info2 = new JLabel("Right aligned info");  
    leftPanel.add(info1);  
    rightPanel.add(info2);  
  
    statusBarPanel.add(new JSeparator(), BorderLayout.NORTH);  
    statusBarPanel.add(leftPanel, BorderLayout.WEST);  
    statusBarPanel.add(rightPanel, BorderLayout.EAST);  
  
    return statusBarPanel;  
}
```

Ориентирането наляво е важно, понеже
ToolBars ще бъдат разположени централно

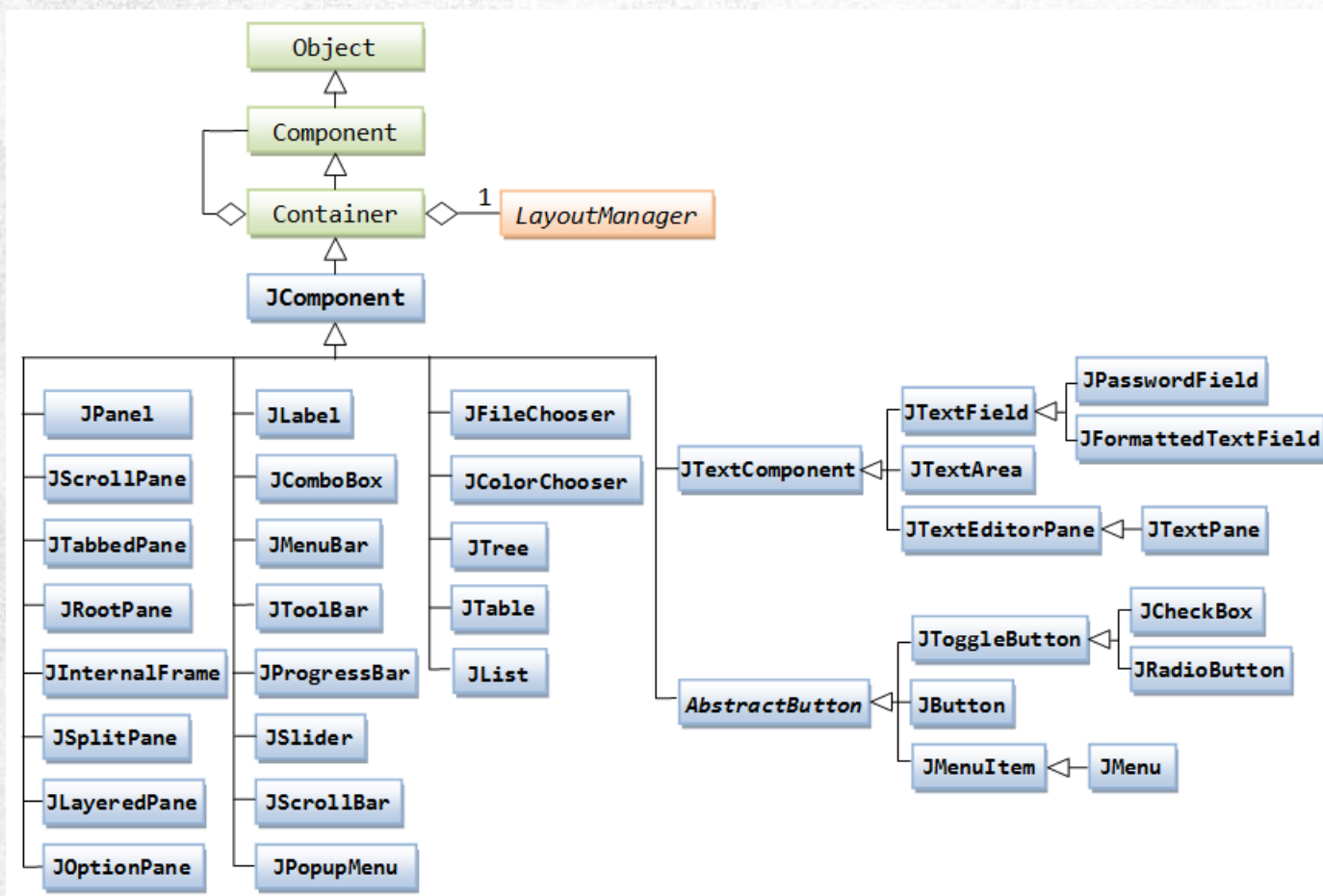
ОБЩА ХАРАКТЕРИСТИКА НА SWING

- Съдържа всички компоненти, които очакваме да видим в един съвременно разработен графичен потребителски интерфейс
 - Бутони, графика, дървовидни и таблични структури, ...
- Много голяма библиотека, но не налага писане на много код
- Ортогоналност на употребата
 - Щом веднъж разберем общите идеи на библиотеката, можем да ги прилагаме за всеки вид компонент
- За бързодействие всички компоненти са „олекотени“
- С цел преносимост библиотеката е написана напълно на Java

ОБЩА ХАРАКТЕРИСТИКА НА SWING

- Навигацията на клавишите е автоматична
 - Можем да стартираме едно Swing приложение без използване на мишката
 - Не изисква също никакво допълнително програмиране
- Поддръжката на скролирането не изисква никакви усилия
 - Просто обвиваме въпросния компонент в JScrollPane, когато го добавяме към желаната форма
- Поддържа “Pluggable look and feel”
 - Твърде радикална възможност
 - Външният вид на потребителския интерфейс може да бъде променян динамично, така че да отговаря на очакванията на потребителите, работещи под различни платформи и ОС

ИЕРАРХИЯ НА SWING



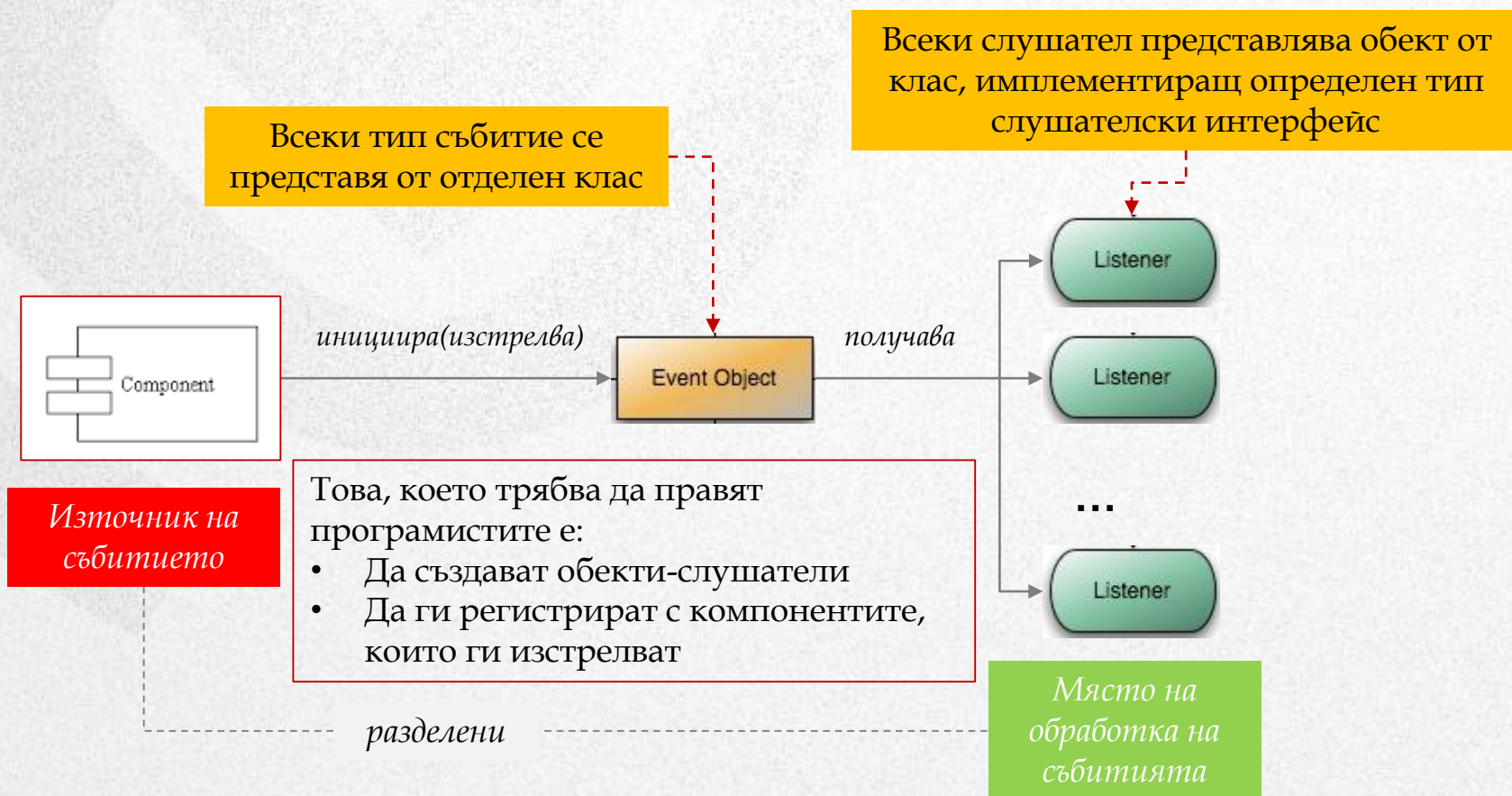
СЪБИТИЕН МОДЕЛ НА SWING

- Всеки тип събитие се представя от отделен клас
- Един компонент може да инициира („изстреля“) събитие
 - Когато се инициира едно събитие, то се получава от един или повече „слушатели“ (listeners)
 - Те оперират съобразно събитието
 - Всеки слушател на събитие представлява обект от клас, имплементиращ определен тип слушателски интерфейс
 - Цялата логика за обработка на събития се разполага в класа-слушател
 - Единственото ограничение е, че той трябва да имплементира съответния интерфейс
 - Така, източниците и получателите на събития могат да бъдат разделени

СЪБИТИЕН МОДЕЛ НА SWING

- Това, което е необходимо да правим:
 - Създаваме обект-слушател
 - Регистрираме го в компонента, инициращ събитието
 - Извършва се посредством `addXXXListener()` в компонента
 - “XXX” представлява типа на прослушващото събитие
- Можем да създадем един глобален клас-слушател, където вътрешните класове могат да бъдат много полезни
 - Позволяват логическо групиране на класовете-слушатели в потребителски интерфейс и класовете на бизнес логиката
 - Това, че вътрешните класове пазят референции към външните класове осигурява добър начин за извикване извън границите на класа и посистемата

СЪБИТИЕН МОДЕЛ



УСВОЯВАНЕ НА SWING

- Обикновено Swing библиотеката се изучава на две стъпки:
 - Разучаване и усвояване на мениджърите за разполагане и управление на събития
 - Използване на отделните Swing компоненти
 - Каталог на Swing компонентите

ТИПОВЕ СЪБИТИЯ И СЛУШАТЕЛИ

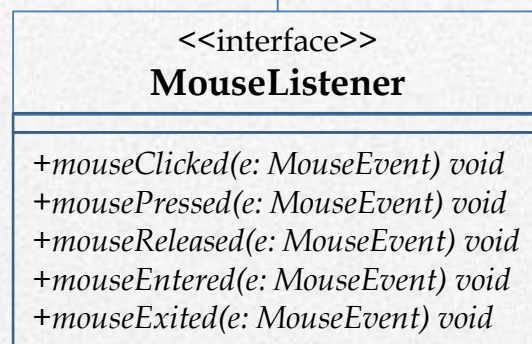
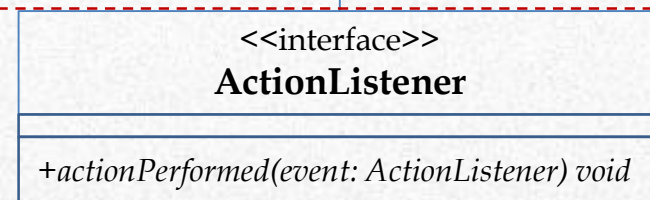
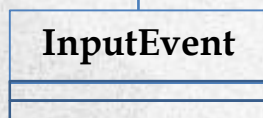
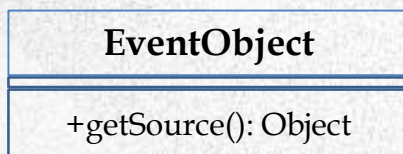
- Всеки Swing компонент включва методи за добавяне или премахване на необходими слушатели
 - addXXXListener
 - removeXXXListener
- Събитийният модел е проектиран така, че да бъде разширяем

ПРЕХВАЩАНЕ НА СЪБИТИЯ

- Ако стартираме предишния пример ще забележим, че при натискане на бутоните няма да има никаква реакция
- Трябва да се напише код, който да определи какво ще се случва при активиране на бутоните
- Основен механизъм
 - Управлявано от събития програмиране (event-driven programming)
 - GUI използва този механизъм
- В Swing - разделяне на:
 - Интерфейс - графичните компоненти
 - Имплементация – код, който се изпълнява, когато възникне определено събитие
- Всеки Swing компонент може да „докладва“ за всички или за отделно събитие, които могат да възникнат в него
 - Така програмистите могат да регистрират само интересуващите ги събития

СЪБИТИЙНА ЙЕРАРХИЯ

java.util



java.awt.event

ОБРАБОТКА НА СЪБИТИЯ

- Едно събитие остава необработено, докато не е регистриран интерес от тип `java.awt.event.ActionListener` към него
- Когато се регистрира такъв Listener, тогава се извиква метода `actionPerformed(ActionEvent)`, реализиран в потребителския клас

javax.swing



AbstractButton

```
+addActionListener(l: ActionListener) void
+removeActionListener(l: ActionListener) void
+getActionListener(l: ActionListener)
#fireActionPerformed(event: ActionEvent) void
```

JButton

register and get notified

java.awt

<<interface>>
ActionListener

```
+actionPerformed(event: ActionListener) void
```

MyActionListener

```
+actionPerformed(event: ActionListener) void
```

ПРИМЕР

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;

public final class ActionEventExample {
    public static void main(final String[] args) {
        final JFrame frame = new JFrame("ActionEventExample");
        final JButton button = new JButton("Button -- Press Me!");
        frame.add(button);

        // Event Listener register
        button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(final ActionEvent e) {
                button.setText(button.getText()+"!");
            }
        });

        frame.setSize(300, 200);
        frame.setVisible(true);
    }
}
```

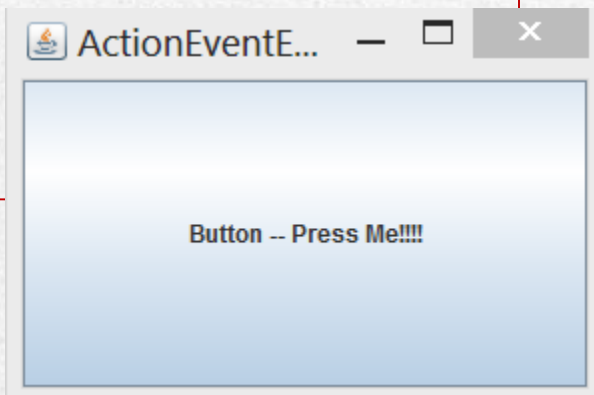
```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class FirstSwingExample {
    public static void main(final String[] args) {

        final JFrame frame = new JFrame("First Swing");

        frame.add(new JLabel("Text-Label:"));
        frame.add(new JButton("Button -- Press Me!"));

        frame.setSize(400, 200);
        frame.setVisible(true);
    }
}
```



ПРИМЕР: БУТОНИ

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

public class Button2 extends JPanel
    JButton
1   b1 = new JButton("Button 1"),
    b2 = new JButton("Button 2");
    JTextField txt = new JTextField(10);
2   class BL implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String name = ((JButton) e.getSource()).getText();
            txt.setText(name);
        }
    }
    BL al = new BL();
    public void init() {
3   b1.addActionListener(al);
    b2.addActionListener(al);
    Container cp = getContentPane();
    cp.setLayout(new FlowLayout());
    cp.add(b1);
    cp.add(b2);
    cp.add(txt);
    }
    public static void main(String[] args) {
        Console.run(new Button2(), 200, 75);
    }
}
```

- Фокусираме се върху главното събитие, което ни интересува за използваните компоненти – в случая JButton, това е, дали бутонът е натиснат
- За регистриране интереса ни към това, кога е натиснат бутона, използваме метода addActionListener() на JButton

За да прикачим код към даден бутон от тип JButton:

- Имплементираме интерфейса ActionListener в някакъв клас
- Регистрираме обект от този клас с JButton чрез метода addActionListener()

ПРИМЕР: БУТОНИ

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

public class Button2 extends JApplet {
    JButton
1   b1 = new JButton("Button 1"),
    b2 = new JButton("Button 2");
    JTextField txt = new JTextField(10);
2   class BL implements ActionListener {
        public void actionPerformed(ActionEvent e) {
3           String name = ((JButton) e.getSource()).getText();
            txt.setText(name);
        }
    }
    BL al = new BL();
    public void init() {
2       b1.addActionListener(al);
        b2.addActionListener(al);
        Container cp = getContentPane();
        cp.setLayout(new FlowLayout());
3       cp.add(b1);
        cp.add(b2);
        cp.add(txt);
    }
    public static void main(String[] args) {
        Console.run(new Button2(), 200, 75);
    }
}
```

- `JTextField` – място, където може да бъде въвеждан и променян текст
- Създаването на такова поле и поставянето му във формата се извършва по същия начин като другите компоненти
- В случая при натискане на бутона се чете текста върху него и се извежда в текстовото поле

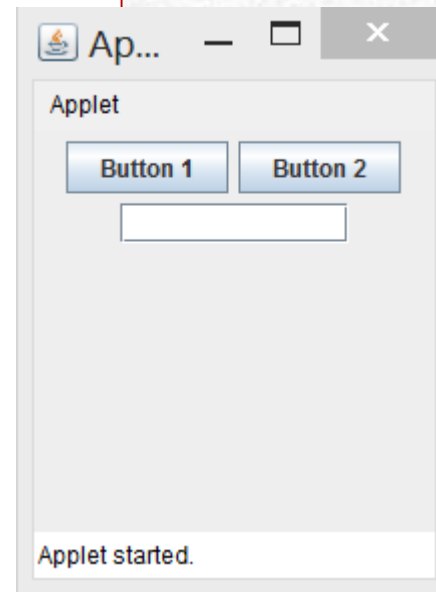
ПРИМЕР: БУТОНИ

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

public class Button2 extends JApplet {
    JButton
        b1 = new JButton("Button 1"),
        b2 = new JButton("Button 2");
    JTextField txt = new JTextField(10);
    class BL implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String name = ((JButton) e.getSource()).getText();
            txt.setText(name);
        }
    }
    BL al = new BL();
    public void init() {
        b1.addActionListener(al);
        b2.addActionListener(al);
        Container cp = getContentPane();
        cp.setLayout(new FlowLayout());
        cp.add(b1);
        cp.add(b2);
        cp.add(txt);
    }
    public static void main(String[] args) {
        Console.run(new Button2(), 200, 75);
    }
}
```



Какъв резултат?

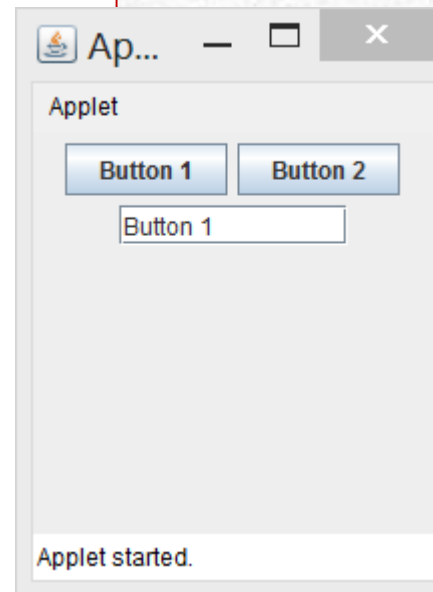


ПРИМЕР: БУТОНИ

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

public class Button2 extends JApplet {
    JButton
        b1 = new JButton("Button 1"),
        b2 = new JButton("Button 2");
    JTextField txt = new JTextField(10);
    class BL implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            String name = ((JButton) e.getSource()).getText();
            txt.setText(name);
        }
    }
    BL al = new BL();
    public void init() {
        b1.addActionListener(al);
        b2.addActionListener(al);
        Container cp = getContentPane();
        cp.setLayout(new FlowLayout());
        cp.add(b1);
        cp.add(b2);
        cp.add(txt);
    }
    public static void main(String[] args) {
        Console.run(new Button2(), 200, 75);
    }
}
```

При натискане на Button 1



ПРИМЕР: БУТОНИ

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

public class Button2b extends JApplet {
    JButton
        b1 = new JButton("Button 1"),
        b2 = new JButton("Button 2");
    JTextField txt = new JTextField(10);
    ActionListener al = new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            String name = ((JButton)e.getSource()).getText();
            txt.setText(name);
        }
    };
}
```

1

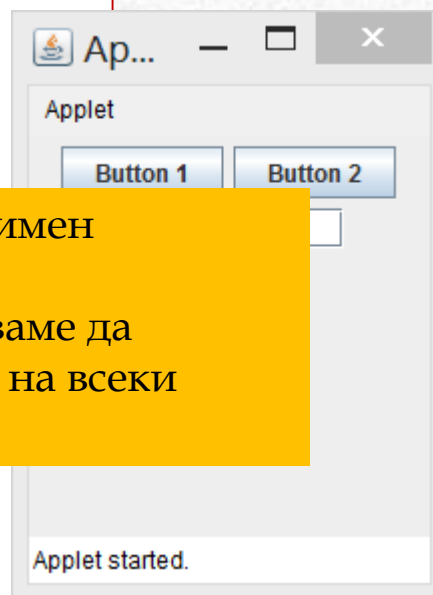
- ActionListener е реализиран като анонимен вътрешен клас
- По-удобно, особено когато възнамеряваме да използваме един единствен екземпляр на всеки клас-слушател

```
    private void createComponents() {
        getContentPane().add(b1);
        getContentPane().add(b2);
        getContentPane().add(txt);
    }

    public static void main(String[] args) {
        Console.run(new Button2b(), 200, 75);
    }
}
```

1

Разлика с предишния пример?



ПРИМЕР: SHOWADDLISTENERS

```
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
import java.lang.reflect.*;
import java.io.*;
import com.bruceeckel.swing.*;
import com.bruceeckel.util.*;
public class ShowAddListeners extends JApplet {
    Class cl;
    Method[] m;
    Constructor[] ctor;
    String[] n = new String[0];
    JTextField name = new JTextField(25);
    JTextArea results = new JTextArea(40,65);
```

- **name:** поле, където въвеждаме името на Swing класа
- **results:** в това поле ще се покажат резултатите

```
    if(nm.length() == 0) {
        results.setText("No match");
        n = new String[0];
        return;
    }
    try {
        cl = Class.forName("javax.swing." + nm);
    } catch(ClassNotFoundException ex) {
        results.setText("No match");
        return;
    }
}
```

```
    m = cl.getMethods();
    n = new String[m.length];
    for(int i = 0; i < m.length; i++)
        n[i] = m[i].toString();
    reDisplay();
}

void reDisplay() {
    String[] rs = new String[n.length];
    int j = 0;
    for(int i = 0; i < n.length; i++)
        if(n[i].indexOf("add") != -1 && n[i].indexOf("Listener") != -1)
            rs[j++] = n[i].substring(n[i].indexOf("add"));
    results.setText("");
    for(int i = 0; i < j; i++)
        results.append(stripQualifiers.strip(rs[i]) + "\n");

    name.addActionListener(new NameL());
    JPanel top = new JPanel();
    top.add(new JLabel ("Swing class name (press ENTER):"));
    top.add(name);
    Container cp = getContentPane();
    cp.add(BorderLayout.NORTH, top);
    cp.add(new JScrollPane(results));
}

public static void main(String[] args) {
    Console.run(new ShowAddListeners(), 500, 400);
}
}
```


ПРИМЕР: SHOWADDLISTENERS

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

```
m = cl.getMethods();
n = new String[m.length];
for(int i = 0; i < m.length; i++)
    n[i] = m[i].toString();
```

- Няма бутони, с които да стартираме търсенето – това е така, понеже полето се наблюдава от един ActionListener, който реагира на натискане на “enter” клавиша
- Ако полето за въвеждане на името не е празно, тогава съдържанието му се използва от Class.forName() за референция на съответния Class обект
- В противен случай се извежда в полето за резултата съобщение за неуспех

```
ring[n.length];
```

```
length; i++)
    if(n[i].indexOf("Listener") != -1)
        sb.append(n[i].indexOf("add"));
    sb.append("\n");
    i++;
}
return StripQualifiers.strip(rs[i]) + "\n";
```

```
String nm = name.getText().trim();
if(nm.length() == 0) {
    results.setText("No match");
    n = new String[0];
    return;
}
try {
    2 cl = Class.forName("javax.swing." + nm);
} catch(ClassNotFoundException ex) {
    results.setText("No match");
    return;
}
```

```
public void init() {
    name.addActionListener(new NameL());
    JPanel top = new JPanel();
    top.add(new JLabel("Swing class name (press ENTER):"));
    top.add(name);
    Container cp = getContentPane();
    cp.add(BorderLayout.NORTH, top);
    cp.add(new JScrollPane(results));
}
public static void main(String[] args) {
    Console.run(new ShowAddListeners(), 500, 400);
}
}
```

ПРИМЕР: SHOWADDLISTENERS

```
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
import java.lang.reflect.*;
import java.io.*;
```

```
3 m = cl.getMethods();
  n = new String[m.length];
  for(int i =0; i < m.length; i++)
    n[i] = m[i].toString();
  reDisplay();
}
```

- **getMethods:** връща масив от Method обекти
- Всеки тези обекти в масива се преобразува в String, т.е. пълната сигнатура на метода
- Записват се в n

```
1 String[] n = new String[0];
2 JTextField name = new JTextField(25);
JTextArea results = new JTextArea(40,65);
class NameL implements ActionListener {
  public void actionPerformed(ActionEvent e) {
    String nm = name.getText().trim();
    if(nm.length() == 0) {
      results.setText("No match");
      n = new String[0];
      return;
    }
    try {
      2 cl = Class.forName("javax.swing." + nm);
    } catch(ClassNotFoundException ex) {
      results.setText("No match");
      return;
    }
  }
}
```

```
String[] n = new String[n.length];
for(int i =0; i < n.length; i++)
  n[i] = m[i].toString();
results.setText("");
for(int i = 0; i < j; i++)
  results.append(StripQualifiers.strip(rs[i]) + "\n");
}
public void init() {
  name.addActionListener(new NameL());
  JPanel top = new JPanel();
  top.add(new JLabel ("Swing class name (press ENTER):"));
  top.add(name);
  Container cp = getContentPane();
  cp.add(BorderLayout.NORTH, top);
  cp.add(new JScrollPane(results));
}
public static void main(String[] args) {
  Console.run(new ShowAddListeners(), 500, 400);
}
}
```

ПРИМЕР: SHOWADDLISTENERS

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

```
3 m = cl.getMethods();
```

- **reDisplay():** създава резултатното множество, като от **n** се извличат само низовете, съдържащи "add" и "Listener"

```
length];  
length; i++)  
ring();
```

```
import com.bruceeckel.util.*;
```

```
public class ShowAddListeners extends JApplet {
```

```
    Class cl;
```

```
    Method[] m;
```

```
    Constructor[] ctor;
```

```
    String[] n = new String[0];
```

```
1 JTextField name = new JTextField(25);
```

```
2 JTextArea results = new JTextArea(40,65);
```

```
class NameL implements ActionListener {
```

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

```
        String nm = name.getText().trim();
```

```
        if(nm.length() == 0) {
```

```
            results.setText("No match");
```

```
            n = new String[0];
```

```
            return;
```

```
        }
```

```
        try {
```

```
2        cl = Class.forName("javax.swing." + nm);
```

```
        } catch(ClassNotFoundException ex) {
```

```
            results.setText("No match");
```

```
            return;
```

```
        }
```

```
void reDisplay() {
```

```
    String[] rs = new String[n.length];
```

```
    int j = 0;
```

```
    for(int i = 0; i < n.length; i++)
```

```
        if(n[i].indexOf("add") != -1 && n[i].indexOf("Listener") != -1)
```

```
            rs[j++] = n[i].substring(n[i].indexOf("add"));
```

```
            results.setText("");
```

```
            for(int i = 0; i < j; i++)
```

```
4                results.append(StripQualifiers.strip(rs[i]) + "\n");
```

```
    }
```

```
    public void init() {
```

```
        name.addActionListener(new NameL());
```

```
        JPanel top = new JPanel();
```

```
        top.add(new JLabel ("Swing class name (press ENTER):"));
```

```
        top.add(name);
```

```
        Container cp = getContentPane();
```

```
        cp.add(BorderLayout.NORTH, top);
```

```
        cp.add(new JScrollPane(results));
```

```
    }
```

```
    public static void main(String[] args) {
```

```
        Console.run(new ShowAddListeners(), 500, 400);
```

```
    }
```

```
}
```


1

Какъв резултат?

```
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
import java.lang.reflect.*;
import java.io.*;
```

```
import com.bruceeckel.s
import com.bruceeckel.u
```

```
public class ShowAddLi
```

```
Class cl;
```

```
Method[] m;
```

```
Constructor[] ctor;
```

```
String[] n = new String
```

```
TextField name = new
```

```
TextArea results = ne
```

```
class NameL impleme
```

```
public void actionPe
```

```
String nm = name
```

```
if(nm.length() == 0
```

```
results.setText("No
```

```
n = new String[0];
```

```
return;
```

```
}
```

```
try {
```

```
2 cl = Class.forName
```

```
} catch(ClassNotFoundException ex) {
```

```
results.setText("No match");
```

```
return;
```

```
}
```

3

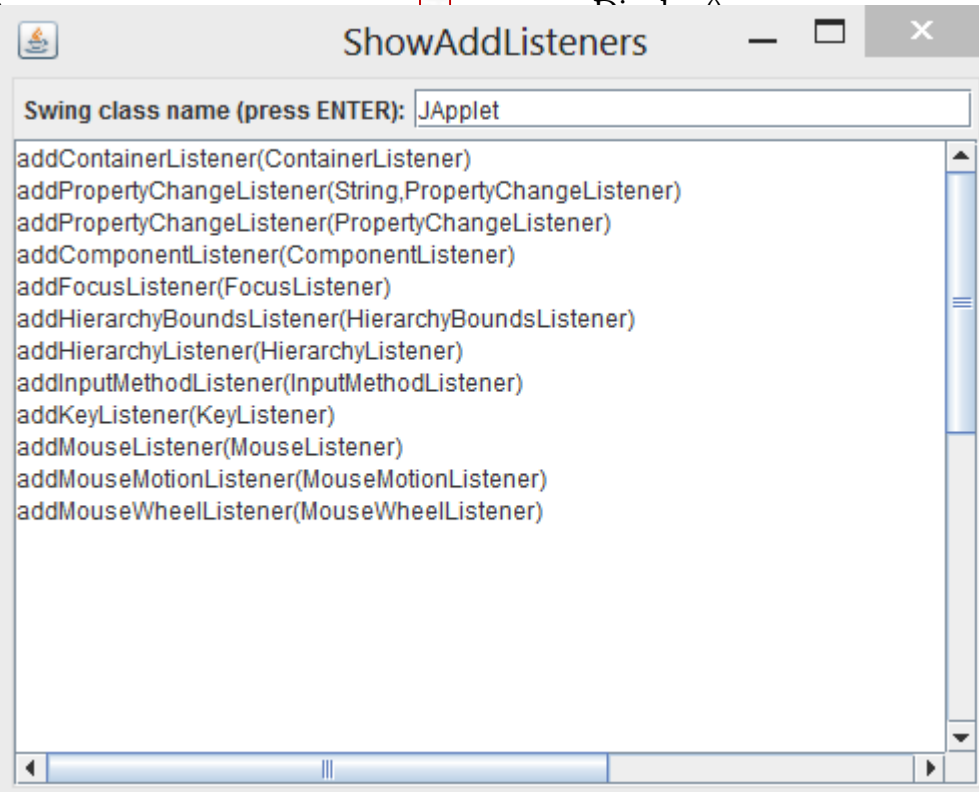
```
m = cl.getMethods();
```

```
n = new String[m.length];
```

```
for(int i = 0; i < m.length; i++)
```

```
n[i] = m[i].toString();
```

```
Display n[i]^
```



```
];
```

```
n[i].indexOf("Listener") != -1)
indexOf("add");
```

```
rs.strip(rs[i]) + "\n");
```

```
ameL());
```

```
s name (press ENTER:));
```

```
);
```

```
op);
```

```
);
```

```
}
```

```
public static void main(String[] args) {
```

```
Console.run(new ShowAddListeners(), 500, 400);
```

```
}
```

```
}
```

ПРОСЛЕДЯВАНЕ НА СЪБИТИЯ

- Искаме да създадем аplet, който да проследява поведението на един JButton
- Искаме също да наследяваме свой собствен бутон
- Той ще се използва като цел на всички интересующи ни събития

ПРИМЕР

- Класът `TrackEvent` съдържа хеш-масив, където се съхраняват низове, представящи типа на събитията и `TextField`, където се съхранява информация за събитията
- В него се създават два бутона

```
String[] event = {
    "focusGained", "focusLost", "keyPressed",
    "keyReleased", "keyTyped", "mouseClicked",
    "mouseEntered", "mouseExited", "mousePressed",
    "mouseReleased", "mouseDragged", "mouseMoved"
};
MyButton b1 = new MyButton(Color.blue, "test1"),
           b2 = new MyButton(Color.red, "test2");
class MyButton extends JButton {
    void report(String field, String msg) {
        ((TextField)h.get(field)).setText(msg);
    }
    FocusListener fl = new FocusListener() {
        public void focusGained(FocusEvent e) {
            report("focusGained", e paramString());
        }
        public void focusLost(FocusEvent e) {
            report("focusLost", e paramString());
        }
    };
};
```

```
new KeyListener() {
    public void keyPressed(KeyEvent e) {
        report("keyPressed", e.paramString());
    }
    public void keyReleased(KeyEvent e) {
        report("keyReleased", e.paramString());
    }
    public void keyTyped(KeyEvent e) {
        report("keyTyped", e.paramString());
    }
};
MouseListener ml = new MouseListener() {
    public void mouseClicked(MouseEvent e) {
        report("mouseClicked", e.paramString());
    }
    public void mouseEntered(MouseEvent e) {
        report("mouseEntered", e.paramString());
    }
    public void mouseExited(MouseEvent e) {
        report("mouseExited", e.paramString());
    }
    public void mousePressed(MouseEvent e) {
        report("mousePressed", e.paramString());
    }
    public void mouseReleased(MouseEvent e) {
        report("mouseReleased", e.paramString());
    }
};
```


ПРИМЕР

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

public class TrackEvent extends JApplet {
    1 HashMap h = new HashMap();
    String[] event = {
        "focusGained", "focusLost", "keyPressed",
        "keyReleased", "keyTyped", "mouseClicked",
        "mouseEntered", "mouseExited", "mousePressed",
        "mouseReleased", "mouseDragged", "mouseMoved"
    };
    MyButton b1 = new MyButton(Color.blue, "test1"),
              b2 = new MyButton(Color.red, "test2");
    2 class MyButton extends JButton {
        void report(String field, String msg) {
            ((JTextField)h.get(field)).setText(msg);
        }
        From JListner f1=new From JListner() {
```

- Класът MyButton е вътрешен за класа TrackEvent
- Той има достъп до външния прозорец и да манипулира неговите полета

```
        report("focusLost", e paramString());
    }
};
```

```
KeyListener kl = new KeyListener() {
    public void keyPressed(KeyEvent e) {
        report("keyPressed", e.paramString());
    }
    public void keyReleased(KeyEvent e) {
        report("keyReleased", e.paramString());
    }
    public void keyTyped(KeyEvent e) {
        report("keyTyped", e.paramString());
    }
};

MouseListener ml = new MouseListener() {
    public void mouseClicked(MouseEvent e) {
        report("mouseClicked", e.paramString());
    }
    public void mouseEntered(MouseEvent e) {
        report("mouseEntered", e.paramString());
    }
    public void mouseExited(MouseEvent e) {
        report("mouseExited", e.paramString());
    }
    public void mousePressed(MouseEvent e) {
        report("mousePressed", e.paramString());
    }
    public void mouseReleased(MouseEvent e) {
        report("mouseReleased", e.paramString());
    }
};
```

ПРИМЕР

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

public class TrackEvent extends JApplet {
    1 HashMap h = new HashMap();
    String[] event = {
        "focusGained", "focusLost", "keyPressed",
        "keyReleased", "keyTyped", "mouseClicked",
        "mouseEntered", "mouseExited", "mousePressed", "mouseReleased", "mouseMoved", "mouseDragged"
    };
}
```

report(): предава му се името и параметрите на съответното събитие

```
    MyButton b1 = new MyButton(Color.blue, "test1");
    MyButton b2 = new MyButton(Color.red, "test2");

    2 class MyButton extends JButton {
        void report(String field, String msg) {
            3 ((JTextField)h.get(field)).setText(msg);
        }
        FocusListener fl = new FocusListener() {
            public void focusGained(FocusEvent e) {
                report("focusGained", e paramString());
            }
            public void focusLost(FocusEvent e) {
                report("focusLost", e paramString());
            }
        };
    }
```

```
KeyListener kl = new KeyListener() {
    public void keyPressed(KeyEvent e) {
        report("keyPressed", e.paramString());
    }
    public void keyReleased(KeyEvent e) {
        report("keyReleased", e.paramString());
    }
    public void keyTyped(KeyEvent e) {
        report("keyTyped", e.paramString());
    }
};

MouseListener ml = new MouseListener() {
    public void mouseClicked(MouseEvent e) {
        report("mouseClicked", e.paramString());
    }
    public void mouseEntered(MouseEvent e) {
        report("mouseEntered", e.paramString());
    }
    public void mouseExited(MouseEvent e) {
        report("mouseExited", e.paramString());
    }
    public void mousePressed(MouseEvent e) {
        report("mousePressed", e.paramString());
    }
    public void mouseReleased(MouseEvent e) {
        report("mouseReleased", e.paramString());
    }
};
```

ПРИМЕР

```
MouseMotionListener mml = new MouseMotionListener() {
    public void mouseDragged(MouseEvent e) {
        report("mouseDragged", e paramString());
    }
    public void mouseMoved(MouseEvent e) {
        report("mouseMoved", e paramString());
    }
};

public MyButton(Color color, String label) {
    super(label);
    setBackground(color);
    addFocusListener(fl);
    addKeyListener(kl);
    addMouseListener(ml);
    addMouseMotionListener(mml);
}

public void init() {
    Container c = getContentPane();
    c.setLayout(new GridLayout(event.length+1,2));
    for(int i = 0; i < event.length; i++) {
        JTextField t = new JTextField();
        t.setEditable(false);
        c.add(new JLabel(event[i], JLabel.RIGHT));
        c.add(t);
        h.put(event[i],t);
    }
    c.add(b1);
    c.add(b2);
}
```

```
public static void main(String[] args) {
    Console.run(new TrackEvent(), 700, 500);
}
```


1 Какъв резултат?

ПРИМЕР

```
MouseMotionListener mml = new MouseMotionListener() {  
    public void mouseDragged(MouseEvent e) {  
        report("mouseDragged", e.paramString());  
    }  
    public void mouseMoved(MouseEvent e) {  
        report("mouseMoved", e.paramString());  
    }  
};  
public MyButton extends JButton {  
    super(label);  
    setBackground(Color.BLUE);  
    addFocusListener(this);  
    addKeyListener(this);  
    addMouseListener(this);  
    addMouseMotionListener(mml);  
}  
public void init(Container c) {  
    c.setLayout(new BorderLayout());  
    for(int i = 0; i < 2; i++) {  
        JTextField t = new JTextField(20);  
        t.setEditable(false);  
        c.add(new JLabel("test" + (i+1)), BorderLayout.CENTER);  
        c.add(t);  
        h.put(i, t);  
    }  
    c.add(b1);  
    c.add(b2);  
}
```

```
public static void main(String[] args) {  
    Console.run(new TrackEvent(), 700, 500);  
}
```

TrackEvent

focusGained	FOCUS_GAINED,permanent,opposite=null
focusLost	FOCUS_LOST,temporary,opposite=null
keyPressed	primaryLevelUnicode=112,scanCode=25,extendedKeyCode=0x50
keyReleased	primaryLevelUnicode=112,scanCode=25,extendedKeyCode=0x50
keyTyped	primaryLevelUnicode=0,scanCode=0,extendedKeyCode=0x00
mouseClicked	
mouseEntered	_ENTERED,(248,6),absolute(598,457),button=0,clickCount=0
mouseExited	E_EXITED,(296,40),absolute(646,491),button=0,clickCount=0
mousePressed	button=1,modifiers=Button1,extModifiers=Button1,clickCount=1
mouseReleased	button=1,absolute(304,476),button=1,modifiers=Button1,clickCount=1
mouseDragged	button=1,absolute(304,476),modifiers=Button1,extModifiers=Button1,clickCount=0
mouseMoved	MOUSE_MOVED,(273,29),absolute(623,480),clickCount=0

test1

test2

ИЗБРАНИ КОМПОНЕНТИ

Радиобутони:

- Концепцията за радио-бутон идва от радио-апаратите на автомобилите преди електрониката – когато се натисне един бутон всички други, които са били натиснати, изскачат навън
- Т.е., в един момент е включен само един бутон от група бутони

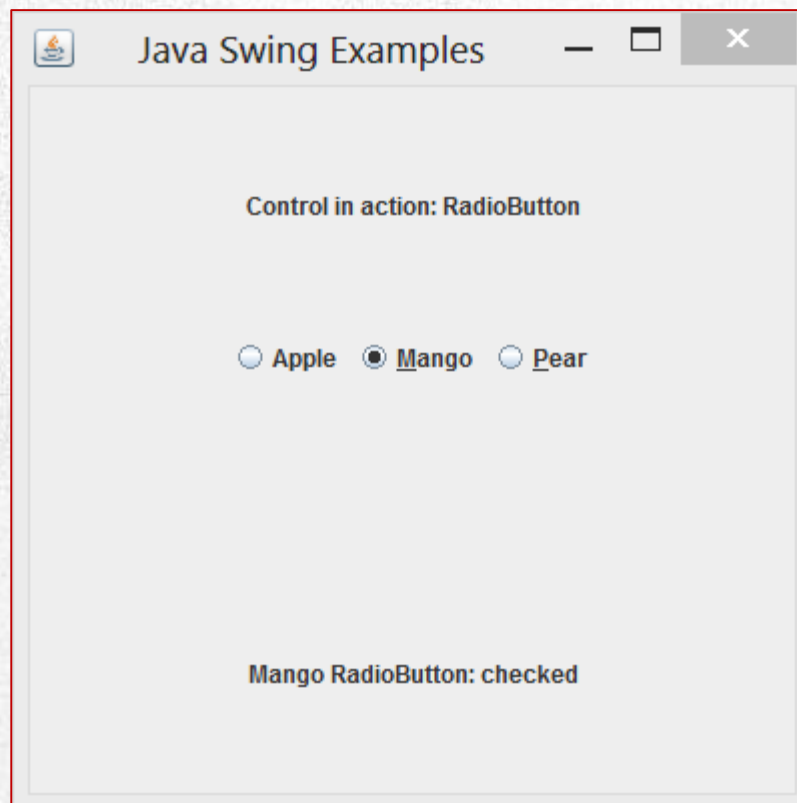
Комбобокс:

- Подобно на радио-бутоните, но с падащ списък, който е начин да се накара потребителят да избере само един елемент от група възможности
- Това е един по-компактен начин за изпълнение на тази задача
- Промяната на елементите от списъка е по-лесно

Диалогови страници:

- Позволяват създаване на „страници диалог“
- Съществуват етикети, разположени по края
- За придвижване към друга страница от диалога, кликваме върху етикета

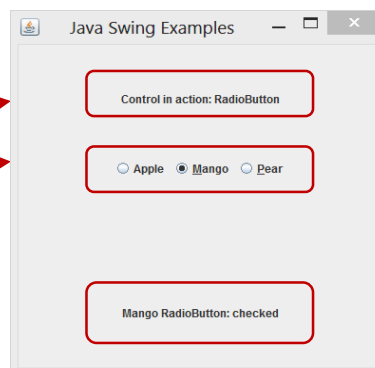
ПРИМЕР: РАДИО-БУТОН



ПРИМЕР: РАДИО-БУТОНИ

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

public class RadioButtonDemo {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel controlPanel;
    private JPanel statusLabel;
    public SwingControlDemo2() {
        prepareGUI();
    }
    public static void main(String[] args) {
        SwingControlDemo2 swingControlDemo = new SwingControlDemo2();
        swingControlDemo.showRadioButtonDemo();
    }
}
```



ПРИМЕР: РАДИО-БУТОНИ

```
private void prepareGUI() {  
  
    mainFrame = new JFrame("Java Swing Examples");  
    mainFrame.setSize(400,400);  
    mainFrame.setLayout(new GridLayout(3, 1));  
  
    mainFrame.addWindowListener(new WindowAdapter() {  
        public void windowClosing(WindowEvent windowEvent) {  
            System.exit(0);  
        }  
    });  
  
    headerLabel = new JLabel("", JLabel.CENTER);  
    statusLabel = new JLabel("",JLabel.CENTER);  
    statusLabel.setSize(350,100);  
  
    controlPanel = new JPanel();  
    controlPanel.setLayout(new FlowLayout());  
  
    mainFrame.add(headerLabel);  
    mainFrame.add(controlPanel);  
    mainFrame.add(statusLabel);  
  
    mainFrame.setVisible(true);  
}
```

Подготовка на фрейма, където ще се разполагат радио-бутоните

АДАПТОРИ

Използване на слушателски адаптери:

- Някои слушателски интерфейси имат само един метод – те са лесни за имплементиране, тъй като ги имплементираме само тогава, когато искаме да създадем определен метод
- Слушателите, които имат повече методи могат да бъдат не толкова приятни за употреба
- Напр., нещо, което трябва да правим винаги, когато създаваме едно приложение, е да осигурим WindowListener към JFrame , така че когато получим събитието windowClosing() да можем да извикаме System.exit() за да излезем от приложението
- Понеже WindowListener е интерфейс трябва да имплементираме всички останали методи, дори когато не правят нищо
- За да се реши този проблем, някои слушателски интерфейси, притежаващи повече от един метод, бяха снабдени с адаптори
- Един адаптер предоставя подразбиращи се празни методи за всеки от интерфейсите методи
- Предефинираме само методите, които искаме да променим

ПРИМЕР: РАДИО-БУТОНИ

```
private void showRadioButtonDemo() {
    headerLabel.setText("Control in action: RadioButton");

    final JRadioButton radApple = new JRadioButton("Apple", true);
    final JRadioButton radMango = new JRadioButton("Mango");
    final JRadioButton radPear = new JRadioButton("Pear");

    radApple.setMnemonic(KeyEvent.VK_C);
    radMango.setMnemonic(KeyEvent.VK_M);
    radPear.setMnemonic(KeyEvent.VK_P);

    radApple.addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e) {
            statusLabel.setText("Apple RadioButton: "
                + (e.getStateChange()==1?"checked":"unchecked"));
        }
    });
    radMango.addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e) {
            statusLabel.setText("Mango RadioButton: "
                + (e.getStateChange()==1?"checked":"unchecked"));
        }
    });
    radPear.addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e) {
            statusLabel.setText("Pear RadioButton: "
                + (e.getStateChange()==1?"checked":"unchecked"));
        }
    });
}
```

- Създаване на радио-бутоните
- “Apple”: включен (дадено във втория параметър като true)
- Ако зададем повече такива бутони само последният от тях ще бъде активиран

setMnemonic: алтернатива на натискане на бутон, като код (цяло число)

Ако се промени състоянието на бутон, се отбелязва в statusLabel

- public int getStateChange (1 – selected)



ПРИМЕР: РАДИО-БУТОНИ

```
ButtonGroup group = new ButtonGroup();
```

```
group.add(radApple);  
group.add(radMango);  
group.add(radPear);
```

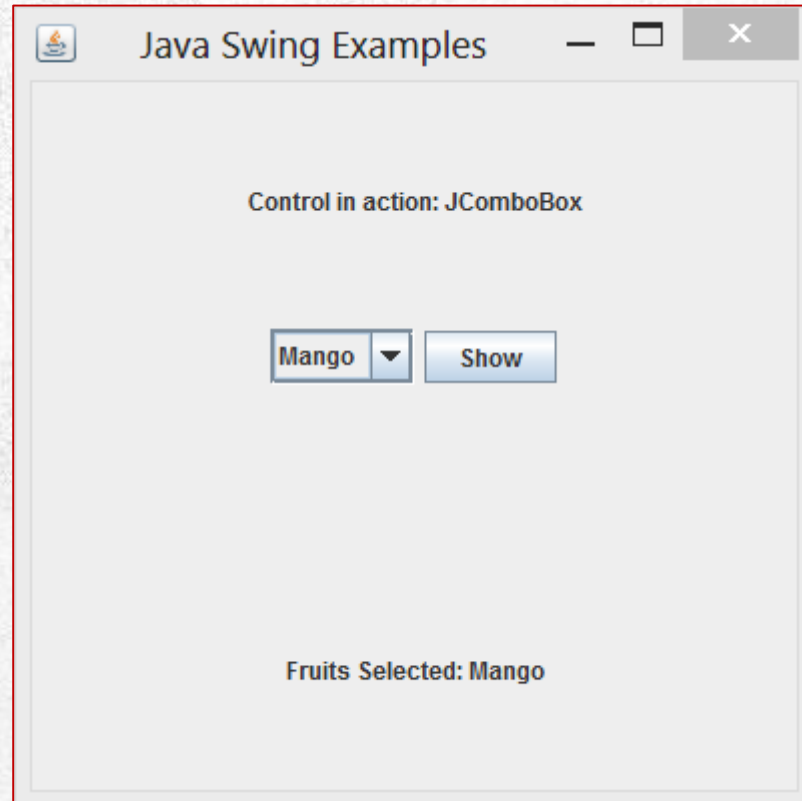
```
controlPanel.add(radApple);  
controlPanel.add(radMango);  
controlPanel.add(radPear);
```

```
mainFrame.setVisible(true);
```

```
}  
}
```

- Трябва да организираме бутоните като група
- Добавяме бутоните към групата

ПРИМЕР: JCOMBOBOX



ПРИМЕР: JCOMBOBOX

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class SwingControlDemo {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;
    public SwingControlDemo() {
        prepareGUI();
    }
    public static void main(String[] args) {
        SwingControlDemo swingControlDemo = new SwingControlDemo();
        swingControlDemo.showComboboxDemo();
    }
}
```

ПРИМЕР: JCOMBOBOX

```
private void prepareGUI() {  
    mainFrame = new JFrame("Java Swing Examples");  
    mainFrame.setSize(400,400);  
    mainFrame.setLayout(new GridLayout(3, 1));  
    mainFrame.addWindowListener(new WindowAdapter() {  
        public void windowClosing(WindowEvent windowEvent) {  
            System.exit(0);  
        }  
    });  
  
    headerLabel = new JLabel("", JLabel.CENTER);  
    statusLabel = new JLabel("",JLabel.CENTER);  
    statusLabel.setSize(350,100);  
  
    controlPanel = new JPanel();  
    controlPanel.setLayout(new FlowLayout());  
  
    mainFrame.add(headerLabel);  
    mainFrame.add(controlPanel);  
    mainFrame.add(statusLabel);  
  
    mainFrame.setVisible(true);  
}
```

ПРИМЕР: JCOMBOBOX

```
private void showComboboxDemo(){
    headerLabel.setText("Control in action: JComboBox");
    final DefaultComboBoxModel fruitsName = new DefaultComboBoxModel();
    fruitsName.addElement("Apple");
    fruitsName.addElement("Grapes");
    fruitsName.addElement("Mango");
    fruitsName.addElement("Pear");
    final JComboBox fruitCombo = new JComboBox(fruitsName);

    fruitCombo.setSelectedIndex(0);
    JScrollPane fruitListScrollPane = new JScrollPane(fruitCombo);
    JButton showButton = new JButton("Show");
    showButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            String data = "";
            if (fruitCombo.getSelectedIndex() != -1) {
                data = "Fruits Selected: "
                    + fruitCombo.getItemAt
                        (fruitCombo.getSelectedIndex());
            }
            statusLabel.setText(data);
        }
    });
    controlPanel.add(fruitListScrollPane);
    controlPanel.add(showButton);
    mainFrame.setVisible(true);
}
```

ПРИМЕР: JCOMBOBOX

```
private void showComboboxDemo(){
    headerLabel.setText("Control in action: JComboBox");
    final DefaultComboBoxModel fruitsName = new DefaultComboBoxModel();
    fruitsName.addElement("Apple");
    fruitsName.addElement("Grapes");
    fruitsName.addElement("Mango");
    fruitsName.addElement("Pear");
    final JComboBox fruitCombo = new JComboBox(fruitsName);

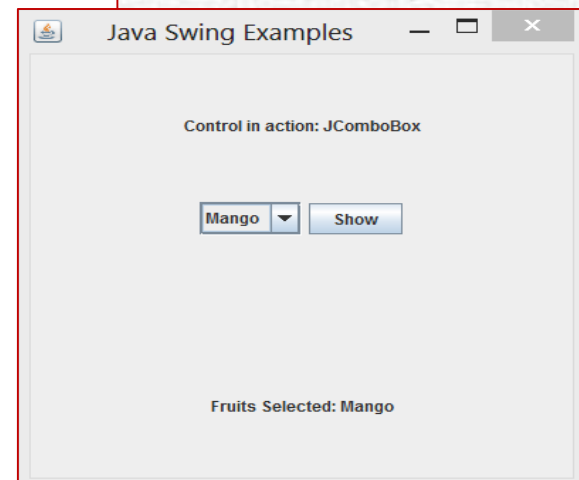
    fruitCombo.setSelectedIndex(0);
    JScrollPane fruitListScrollPane = new JScrollPane(fruitCombo);
    JButton showButton = new JButton("Show");
    showButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            String data = "";
            if (fruitCombo.getSelectedIndex() != -1) {
                data = "Fruits Selected: "
                    + fruitCombo.getItemAt
                        (fruitCombo.getSelectedIndex());
            }
            statusLabel.setText(data);
        }
    });
    controlPanel.add(fruitListScrollPane);
    controlPanel.add(showButton);
    mainFrame.setVisible(true);
}
```

DefaultComboBoxModel fruits: съдържа данните, които ще се показват в списъка

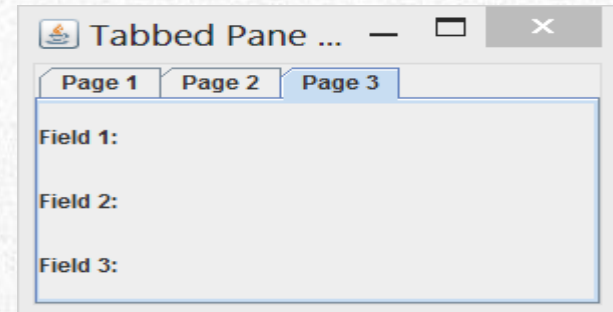
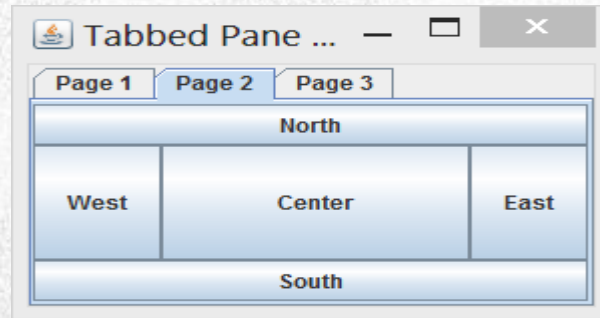
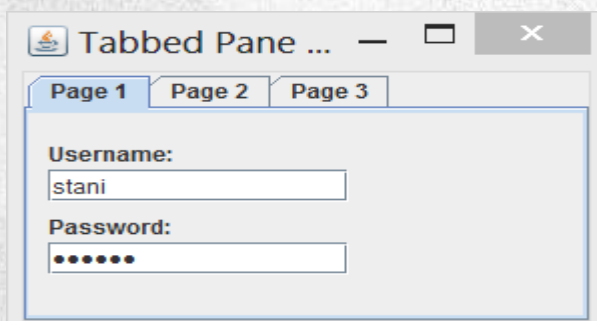
setSelectedIndex: позиционира в указаната опция в списъка

getItemAt: връща съдържанието на избраната опция от списъка

При активиране на Show-бутона, избраната опция се извежда в statusLabel



ПРИМЕР: JTABBEDPANE



ПРИМЕР: JTABBEDPANE

```
import java.awt.*;
import javax.swing.*;
class TabbedPaneExample extends JFrame {
    private JTabbedPane tabbedPane;
    private JPanel panel1;
    private JPanel panel2;
    private JPanel panel3;

    public TabbedPaneExample() {
        setTitle( "Tabbed Pane Application" );
        setSize( 300, 200 );
        setBackground( Color.gray );

        JPanel topPanel = new JPanel();
        topPanel.setLayout( new BorderLayout() );
        getContentPane().add( topPanel );

        createPage1();
        createPage2();
        createPage3();

        tabbedPane = new JTabbedPane();
        tabbedPane.addTab( "Page 1", panel1 );
        tabbedPane.addTab( "Page 2", panel2 );
        tabbedPane.addTab( "Page 3", panel3 );
        topPanel.add( tabbedPane, BorderLayout.CENTER
    );
}
```

ПРИМЕР: JTABBEDPANE

```
public void createPage1() {  
    panel1 = new JPanel();  
    panel1.setLayout( null );  
    JLabel label1 = new JLabel( "Username:" );  
    label1.setBounds( 10, 15, 150, 20 );  
    panel1.add( label1 );  
    JTextField field = new JTextField();  
    field.setBounds( 10, 35, 150, 20 );  
    panel1.add( field );  
    JLabel label2 = new JLabel( "Password:" );  
    label2.setBounds( 10, 60, 150, 20 );  
    panel1.add( label2 );  
    JPasswordField fieldPass = new JPasswordField();  
    fieldPass.setBounds( 10, 80, 150, 20 );  
    panel1.add( fieldPass );  
}
```

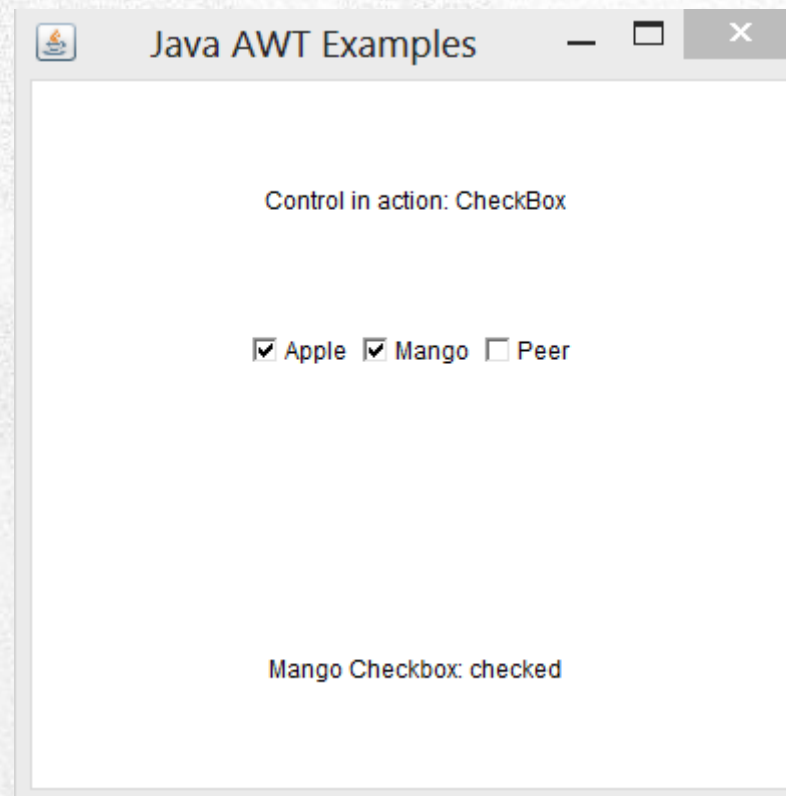
ПРИМЕР: JTABBEDPANE

```
public void createPage2() {  
    panel2 = new JPanel();  
    panel2.setLayout( new BorderLayout() );  
    panel2.add( new JButton( "North" ), BorderLayout.NORTH );  
    panel2.add( new JButton( "South" ), BorderLayout.SOUTH );  
    panel2.add( new JButton( "East" ), BorderLayout.EAST );  
    panel2.add( new JButton( "West" ), BorderLayout.WEST );  
    panel2.add( new JButton( "Center" ), BorderLayout.CENTER );  
}
```

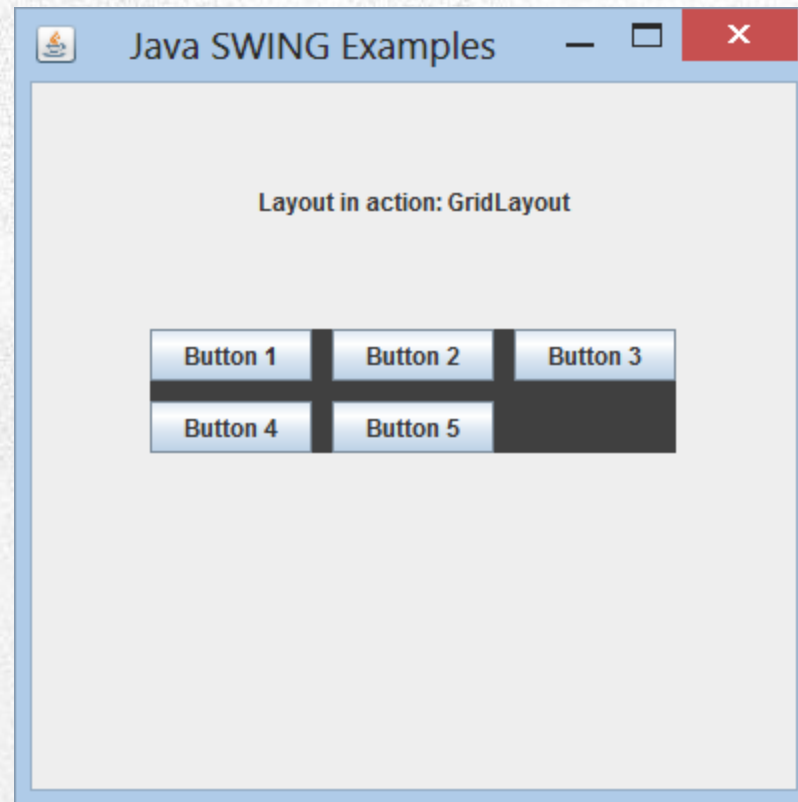

ПРИМЕР: JTABBEDPANE

```
public void createPage3() {  
    panel3 = new JPanel();  
    panel3.setLayout( new GridLayout( 3, 2 ) );  
    panel3.add( new JLabel( "Field 1:" ) );  
    panel3.add( new TextArea() );  
    panel3.add( new JLabel( "Field 2:" ) );  
    panel3.add( new TextArea() );  
    panel3.add( new JLabel( "Field 3:" ) );  
    panel3.add( new TextArea() );  
}  
public static void main( String args[] ) {  
    TabbedPaneExample mainframe = new TabbedPaneExample();  
    mainFrame.setVisible( true );  
}  
}
```

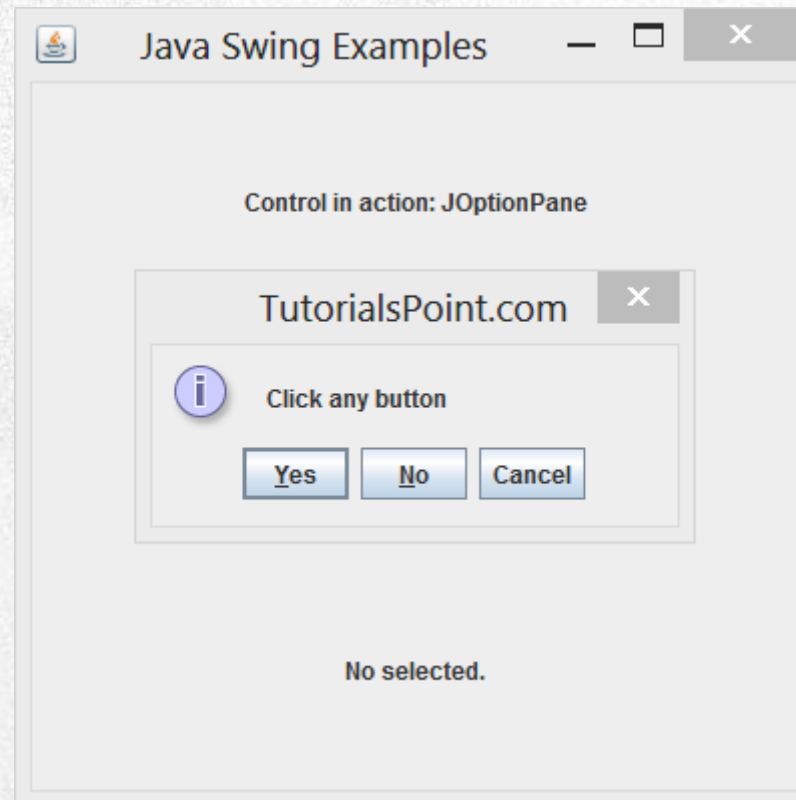
JCHECKBOX



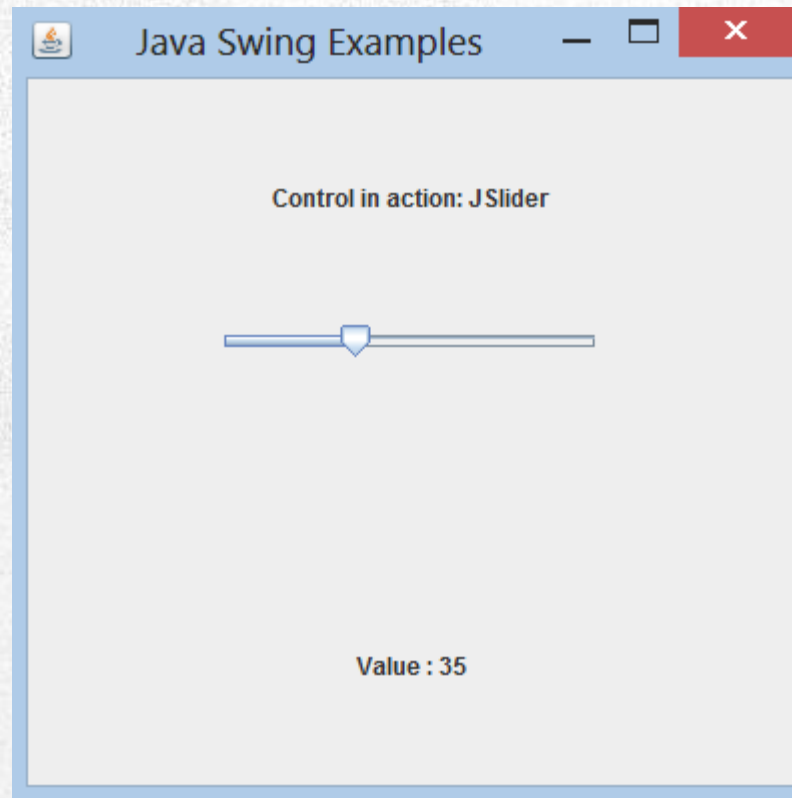
GRIDLAYOUT



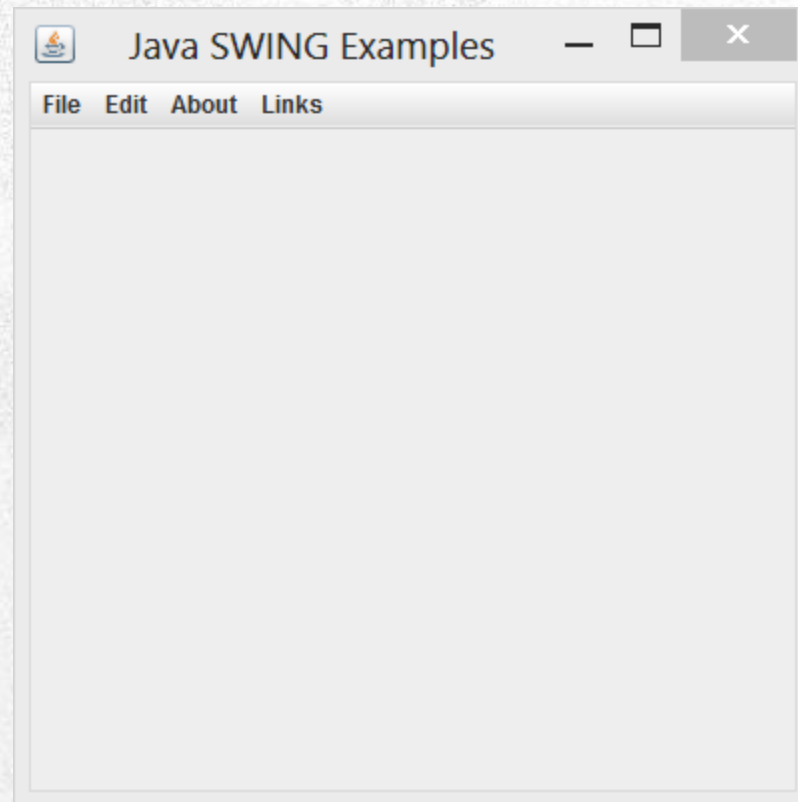
JOPTIONPANE



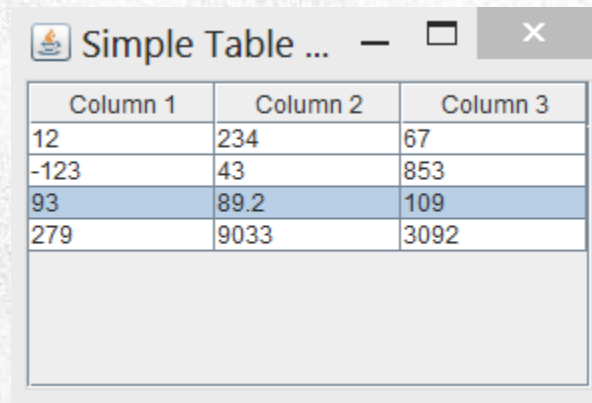
JSLIDER



JMENU

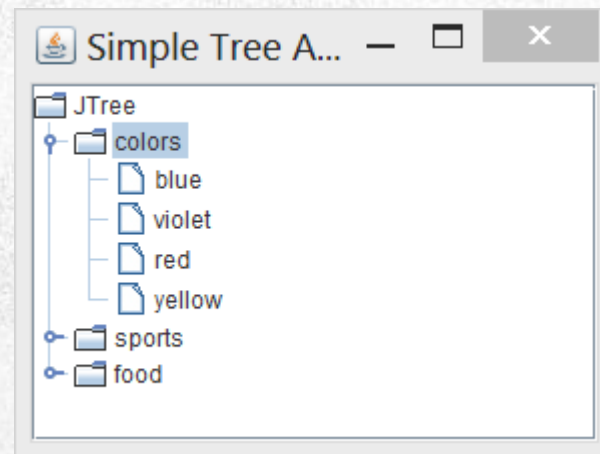


JTABLE



Column 1	Column 2	Column 3
12	234	67
-123	43	853
93	89.2	109
279	9033	3092

JTREE



LOOK AND FEEL (LAF)

- Един от интересните аспекти на Swing е възможността за избор на “look and feel” (изглед и усещане)
 - look се отнася към вида GUI елементите
 - feel се отнася към тяхното поведение
- Позволява в нашата програма да подражаваме LAF на различни операционни среди
- Можем динамично да променяме LAF по време на изпълнение на програмата
- Подразбиращ се LAF
 - Междуплатформен (“metal”)
- LAF на актуалната операционна среда
 - “system”

ПРИМЕР: LAF

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

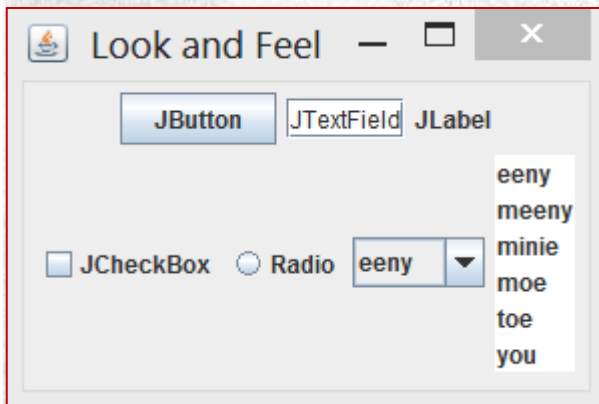
public class LookAndFeel extends JFrame {
    String[] choices = { "eeny", "meeny", "minie", "moe", "toe", "you" };
    Component[] samples = { new JButton("JButton"), new JTextField("JTextField"), new JLabel("JLabel"),
        new JCheckBox("JCheckBox"), new JRadioButton("Radio"), new JComboBox(choices), new JList(choices), };
    public LookAndFeel() {
        super("Look and Feel");
        Container cp = getContentPane();
        cp.setLayout(new FlowLayout());
        for(int i = 0; i < samples.length; i++)
            cp.add(samples[i]);
    }
    private static void usageError() {
        System.out.println(" Usage: LookAndFeel [cross | system | motif]");
        System.exit(1);
    }
}
```

ПРИМЕР: LAF

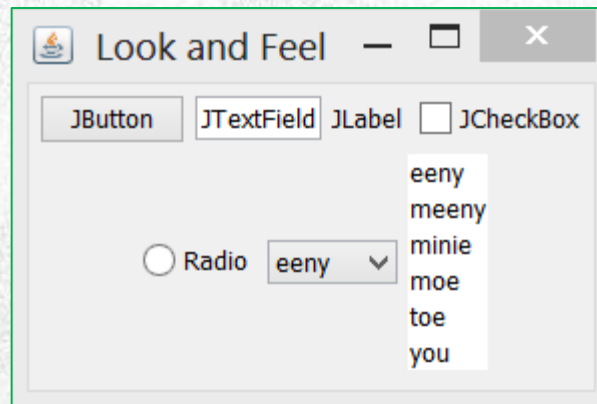
```
public static void main(String[] args) {
    if(args.length == 0) usageError();
    if(args[0].equals("cross") ) {
        try {
            UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());
        } catch(Exception e) {
            e.printStackTrace(System.err);
        }
    } else if(args[0].equals("system") ) {
        try {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        } catch(Exception e) {
            e.printStackTrace(System.err);
        }
    } else if(args[0].equals("motif") ) {
        try {
            UIManager.setLookAndFeel("com.sun.java." + "swing.plaf.motif.MotifLookAndFeel");
        } catch(Exception e) {
            e.printStackTrace(System.err);
        }
    } else usageError();
    Console.run(new LookAndFeel(), 300, 200);
}
```

ПРИМЕР: LAF

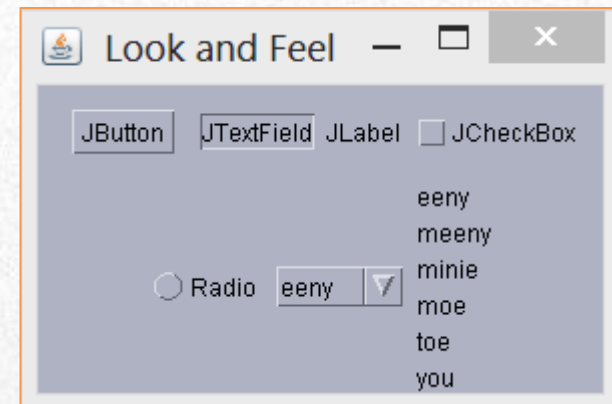
cross



system



motif



БЛАГОДАРЯ ЗА ВНИМАНИЕТО!

КРАЙ “GUI”

