



# GUI программирование. Swing

## Лекция 5.2

# Цели



- *Описать структуру Swing*
- *Создать простое приложение*
- *Создать Swing GUI*
- *Описать классы*
  - ❖ *JComponent*
  - ❖ *JFrame*
  - ❖ *JPanel*
- *Описать компоненты*
  - ❖ *TextComponents*
  - ❖ *Checkboxes и RadioButtons*
  - ❖ *Lists и ComboBoxes*

# Введение

- В первых версиях языка это был единственный пакет, разработанный для этих целей. Этот пакет выполняет свои функции, создавая экземпляры парных классов (*peer*-классов), используемых операционной системой (так называемые *тяжелые* компоненты).
- То есть, при создании объекта, фактически создаются два: абстрактный объект Java и объект операционной системы, которые взаимодействуют между собой.
- В настоящее время компания Sun предоставила разработчику пакет *облегченных* компонентов (*Swing*-компонентов), которые не нуждаются в создании вспомогательных объектов.
- Библиотека **Swing** - самое важное нововведение в Java 2. Она является частью JFC (*Java Foundation Classes*). Общее количество компонентов библиотеки Swing почти вдвое превышает количество компонентов, входящих в библиотеку AWT, и эти компоненты обладают большими функциональными возможностями.

# Преимущества Swing

- Использование элементов: окон, кнопок т.д.
- Выбор произвольного стиля представления элементов (т.н. Looks & Feels) даже в процессе работы программы. В Swing существует пока три типа представления элементов: Windows, Metal, Motiff, но предполагается расширить их количество.
- Предоставляет некоторые возможности для людей с ограниченным зрением, в частности, при использовании специальных экранов со шрифтом Брайля.
- Предоставляет возможность разработчикам легко интегрировать высокого качества 2D графику, тексты, рисунки в апплеты и приложения.
- Предоставляет возможность использовать технологию "drag and drop" между Java приложением и другими приложениями на конкретной платформе.

# Компоненты Swing

- **Swing** – это набор классов в составе JFC.
- Предоставляет упрощённые визуальные компоненты и позволяет создавать привлекательный интерфейс GUI.
- Содержит компоненты, заменяющие визуальные компоненты AWT, а также комплексные сложные компоненты – деревья и таблицы.
- При проектировании GUI создаётся основное окно, в котором размещаются визуальные компоненты.
- Компоненты Swing содержатся в пакете `javax.swing`
- Имена всех классов компонентов Swing начинаются с буквы **J**.

# Компоненты Swing

Для того чтобы графические программы выполнялись методами библиотеки Swing, необходимо внести небольшие изменения:

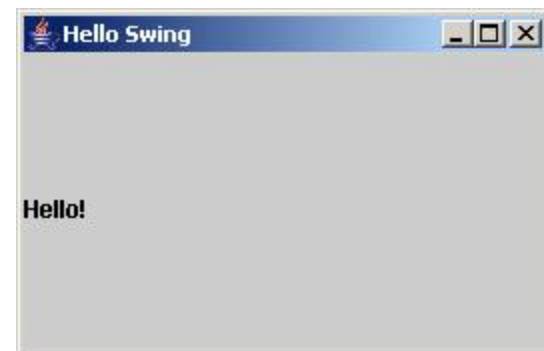
- добавить в заголовков строку `import javax.swing.*;`
- поменять `Frame` на `JFrame`, `Applet` на `JApplet`, `Component` на `JComponent`, `Panel` на `JPanel`;
- заменить компоненты AWT на близкие к ним компоненты Swing. Чаще всего надо просто приписать букву J: `JButton`, `JCheckBox`, `JDialog`, `JList`, `JMenu` и т. д. Закомментируйте временно строку `import java.awt.*;` и попробуйте откомпилировать программу. Компилятор покажет, какие компоненты требуют замены;
- включить в конструктор класса, расширяющего `JFrame`, строку `Container c = getContentPane();` и располагайте все компоненты в данном контейнере;
- в прямых подклассах класса `JPanel` замените метод `paint()` на `paintComponent()` и удалите метод `update()`. Класс `JPanel` автоматически производит двойную буферизацию и надобности в методе `update()` больше нет. В начало метода `paintComponent()` включите обращение `super.paintComponent(g);`
- при создании апплетов расширением класса `JApplet` не забывайте, что в классе `Applet` менеджером размещения по умолчанию служит класс `FlowLayout`, а в классе `JApplet` менеджер размещения по умолчанию `BorderLayout`.

# Простое Swing приложение

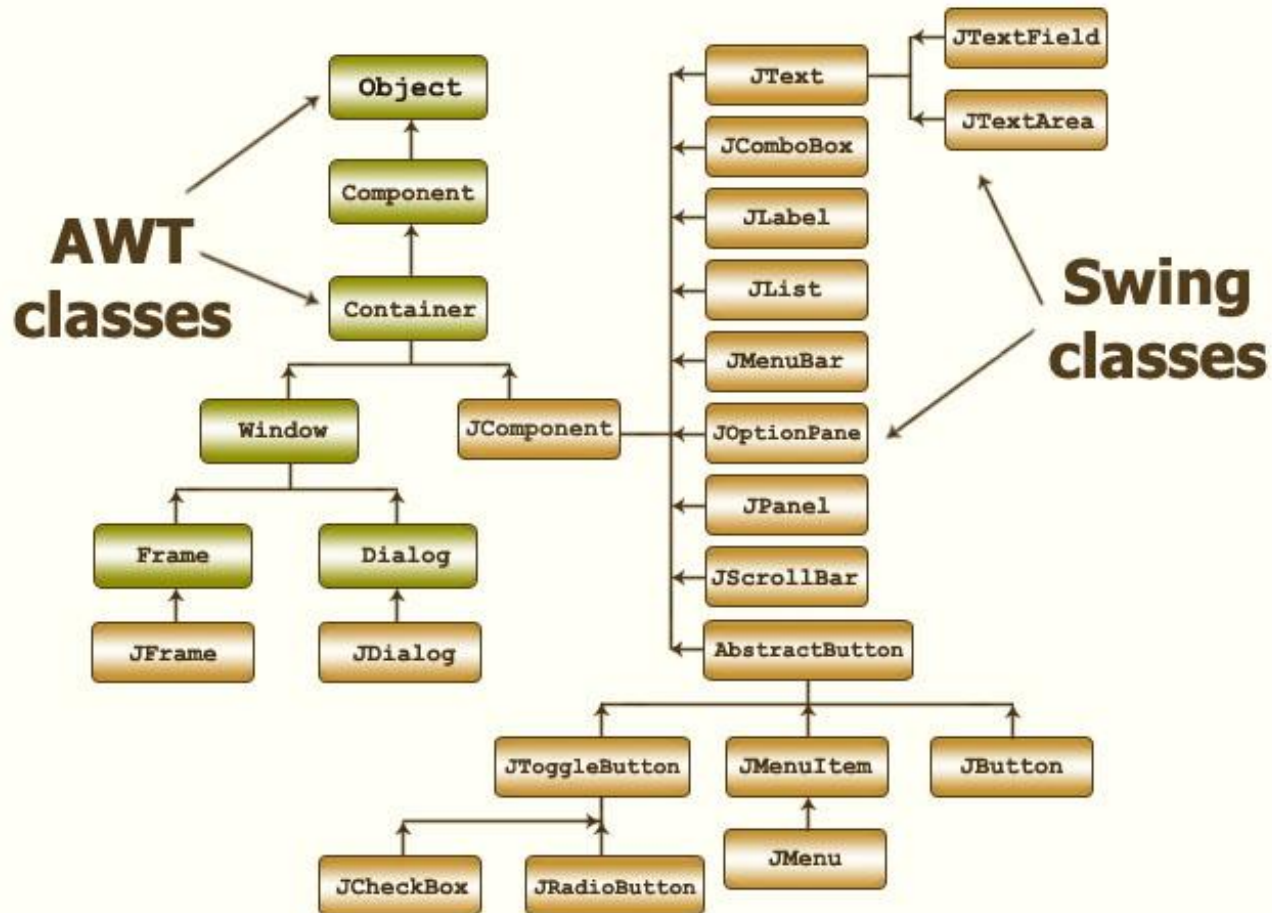
```
import javax.swing.*;

public class DemoSwing {

    public static void main(String[] args) {
        JFrame frame = new JFrame("Hello Swing");
        final JLabel label = new JLabel("Hello!");
        frame.getContentPane().add(label);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

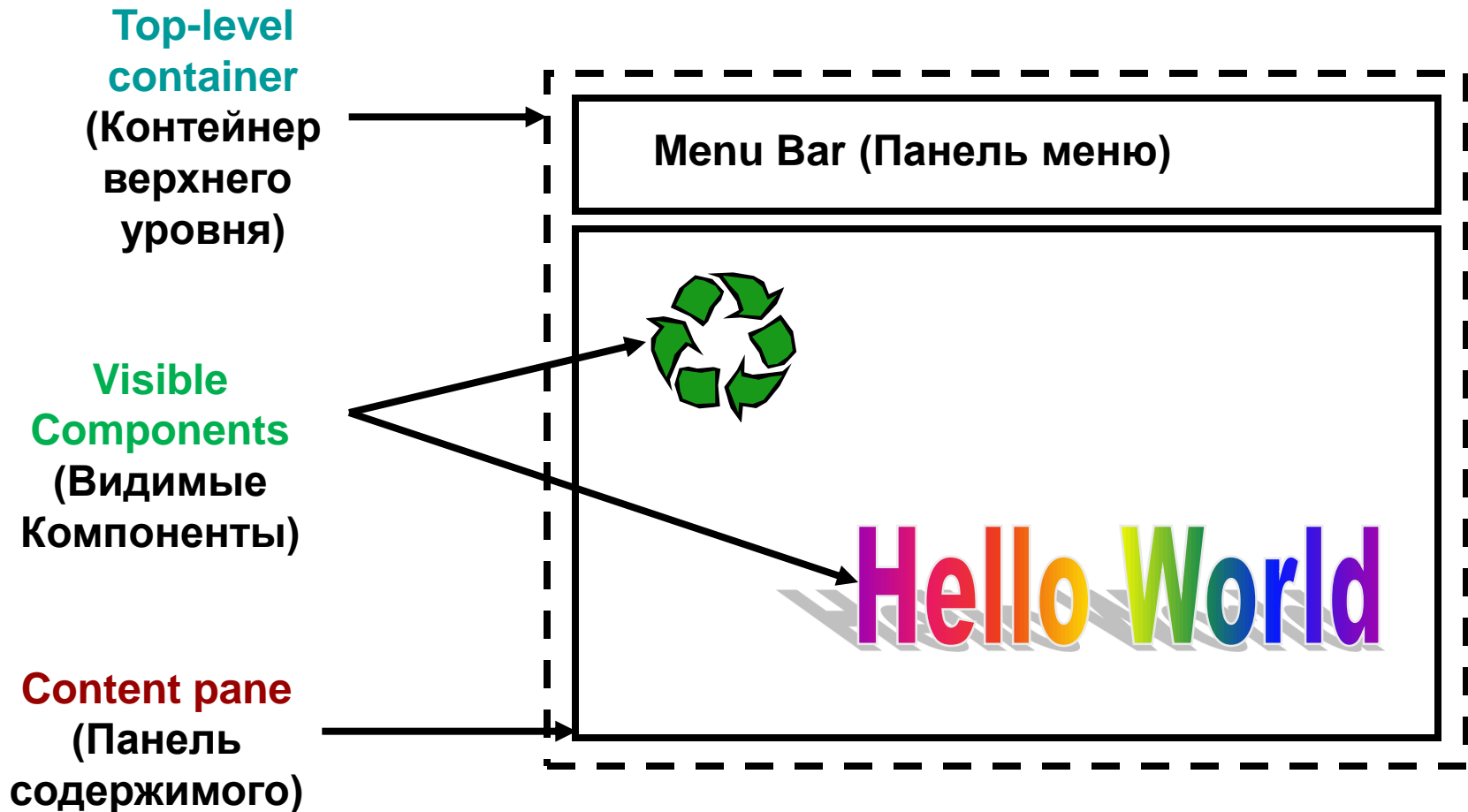


# Структура Swing





# Компоненты Swing



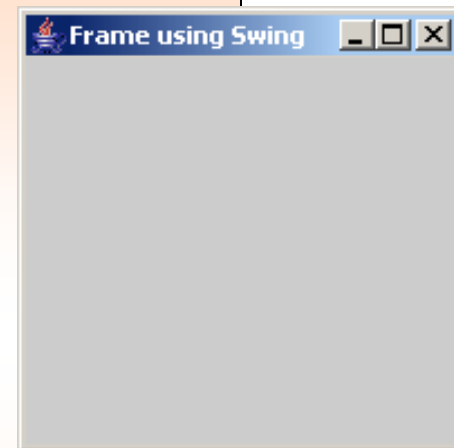
# JFrame

- Контейнер верхнего уровня или окно.
- Предоставляет место для других компонентов Swing.
- Компонент `JFrame` используется для создания окон в Swing-программе.
- Некоторые из его конструкторов:
  - ❖ `JFrame()`
  - ❖ `JFrame(String Title)`
- Компоненты должны добавляться в панель содержимого (`content pane`), а не непосредственно в объект `JFrame`
  - ❖ Пример: `frame.getContentPane().add(b);`

# Пример

```
import java.awt.*;
import javax.swing.*;
public class FrameDemo extends JFrame
{
    public FrameDemo(String title)
    {
        super(title);
        setVisible(true);
        setSize(200,200);
    }

    public static void main(String args[])
    {
        FrameDemo objFrameDemo = new FrameDemo("Frame using Swing");
    }
}
```



- Компонент `JPanel` является промежуточным контейнером.
- Используется для объединения небольших простых компонентов в группу.
- Объекты `JPanel` принимают `FlowLayout`, как макет размещения по умолчанию.
- Компонент `JPanel` содержит следующие конструкторы:
  - ❖ `JPanel()`
  - ❖ `JPanel(LayoutManager lm)`

# JApplet

- Компонент `javax.swing.JApplet` немного отличается от компонента `java.applet.Applet`.
- При добавлении компонента в `JApplet` становится обязательным его добавление в панель содержимого компонента `JApplet`
  - ❖ Пример: `getContentPane().add(component);`

# Пример

```
/*  
<applet code = SwingApplet width = 150 height =150>  
</applet>  
*/  
  
import java.awt.*;  
import javax.swing.*;  
public class SwingApplet extends JApplet  
{  
    public void init() {  
    }  
}
```



# Панель содержимого и апплеты

- Панель содержимого отличает Swing-апплеты от обычных апплетов по следующим пунктам:
  - ❖ Компоненты добавляются в панель содержимого апплетов `Swing`, а не напрямую в апплет.
  - ❖ Менеджер макета устанавливается в панель содержимого Swing-апплета, а не в сам апплет.
  - ❖ По умолчанию для панели содержимого Swing-апплета выбран менеджер макета `BorderLayout`, в то время как для обычных апплетов — `FlowLayout`.
  - ❖ Код метода `paint()` помещается в объект `JApplet` посредством включения метода `paintComponent()`.

# Основные компоненты GUI

- Форма (form) может использоваться для сбора информации.
- При создании интерфейсов GUI компонентом, который может быть использован для ввода данных, является текстовое поле (text field) или текстовая панель (text box).
- Для создания элемента в интерфейсе GUI необходимо выполнить следующие шаги:
  - ❖ Создать элемент
  - ❖ Установить его атрибуты (размер, цвет, шрифт)
  - ❖ Задать место его расположения
  - ❖ Добавить его на экран (вывести)



# JLabel

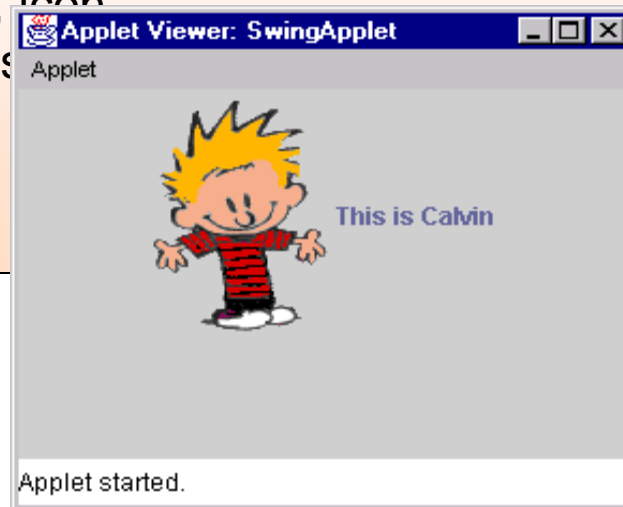
- Компонент JLabel — это просто строка текста, или строка с изображением, или с обоими сразу, оформленная как графический компонент для размещения в контейнере.
- Может выводить как текст, так и изображения.
- Текст можно поменять только методом доступа **setText(String text)**, но не вводом пользователя с клавиатуры или с помощью мыши.

Конструкторы:

- JLabel();
- JLabel (Icon image);
- JLabel (Icon image, int horizontalAlignment);
- JLabel (String text);
- JLabel (String text, Icon icon, int horizontalAlignment);
- JLabel (String text, int horizontalAlignment).

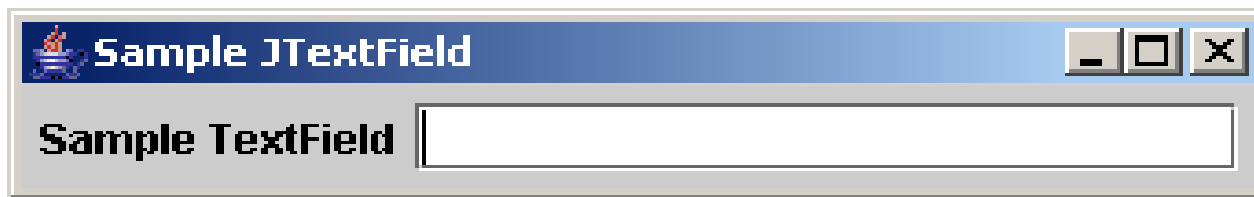
# Пример

```
/*  
<applet code = LabelDemo width = 200 height =200> </applet>  
*/  
import java.awt.*;  
import javax.swing.*;  
public class LabelDemo extends JApplet  
{  
    public void init()  
    {  
        getContentPane().setLayout(new FlowLayout());  
        ImageIcon icon = new ImageIcon("Calv.gif");  
        JLabel calvLabel = new JLabel("This is Calvin", icon,  
                                       SwingConstants.LEFT);  
        getContentPane().add(calvLabel);  
    }  
}
```



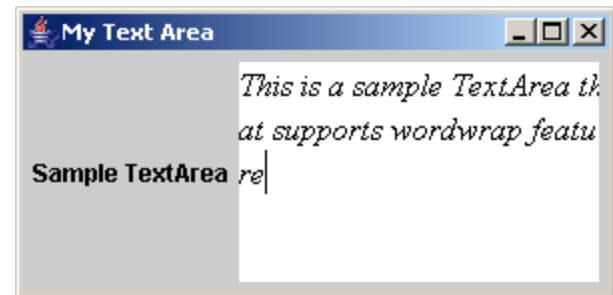
# JTextFieldComponent

- Компонент `JTextField` позволяет редактировать одну строку текста.
- Конструкторы класса `JTextField`:
  - ❖ `JTextField()`
  - ❖ `JTextField(int columns)`
  - ❖ `JTextField(String text)`
  - ❖ `JTextField(String text, int columns)`



# JTextArea

- Компонент `JTextArea` используется для приёма нескольких строк текста от пользователя.
- Применяет интерфейс скроллинга для активизации полос прокрутки (scrollbars).
- Компонент `JTextArea` может быть создан с использованием одного из следующих конструкторов:
  - ❖ `JTextArea()`
  - ❖ `JTextArea(int rows, cols)`
  - ❖ `JTextArea(String text)`
  - ❖ `JTextArea(String text, int rows, int cols)`



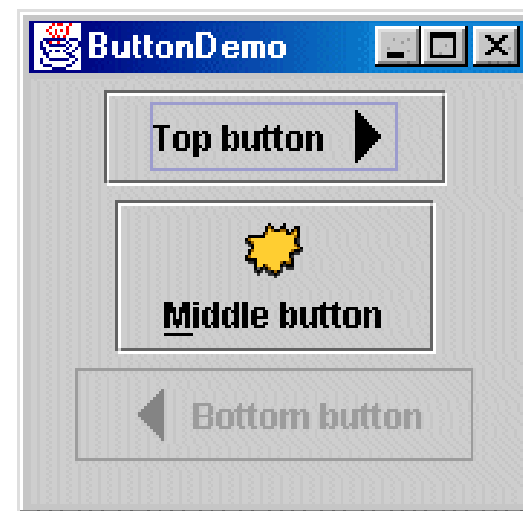
# JPasswordField

- Компонент **JPasswordField** — поле для ввода пароля.
- В таком поле вместо вводимых символов появляется какой-нибудь особый эхо-символ, чаще всего звездочка.
- Данное поле ввода получается выполнением метода **setEchoChar(char echo)**.
- Аргумент **echo** — это символ, который будет появляться в поле.
- Проверить, установлен ли эхо-символ, можно логическим методом **echoCharIsSet()**, получить эхо-символ — методом **getEchoChar()**.
- Чтобы вернуть поле ввода в обычное состояние, достаточно выполнить метод **setEchoChar(0)**.
- Конструкторы:
  - ❖ **JPasswordField();**
  - ❖ **JPasswordField(Document doc, String txt, int columns);**
  - ❖ **JPasswordField(int columns);**
  - ❖ **JPasswordField(String text);**
  - ❖ **JPasswordField(String text, int columns).**



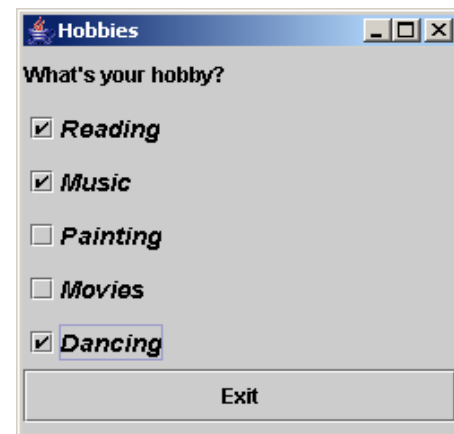
# JButton

- Класс `JButton` является производным от класса `javax.swing.AbstractButton`.
- Объект `JButton` состоит из текстовой надписи и/или значка-изображения, пустого пространства вокруг текста/значка и рамки (границы кнопки).
- `JButton` может быть создан с использованием:
  - ❖ `JButton()`
  - ❖ `JButton(Icon icon)`
  - ❖ `JButton(String text)`
  - ❖ `JButton(String text, Icon icon)`
  - ❖ `JButton(Action a)`



# JCheckbox

- Чек-бокс (Checkbox) используется для предоставления пользователю набора вариантов для выбора.
- Класс `JCheckBox` содержит следующие конструкторы:
  - ❖ `JCheckBox()`
  - ❖ `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)`
  - ❖ `JCheckBox(Action a)`



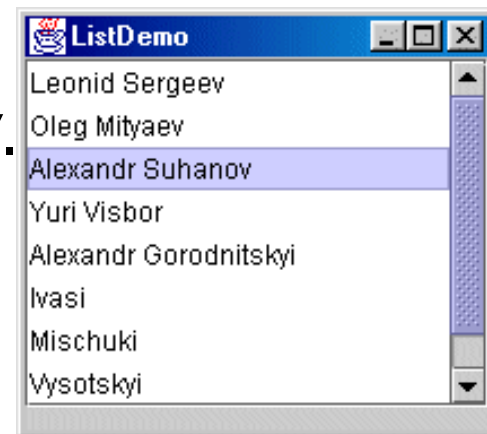
# JRadioButton

- Набор радиокнопок (radio buttons) выводит некоторое количество вариантов, из которых может быть выбран только один.
- `ButtonGroup` используется для создания группы кнопок в `Swing`.
- Объект `JRadioButton` может быть создан с использованием:
  - ❖ `JRadioButton()`
  - ❖ `JRadioButton(Icon icon)`
  - ❖ `JRadioButton(Icon, boolean selected)`
  - ❖ `JRadioButton(String text)`
  - ❖ `JRadioButton(String text, boolean selected)`
  - ❖ `JRadioButton(String text, Icon icon)`
  - ❖ `JRadioButton(String text, Icon icon, boolean selected)`
  - ❖ `JRadioButton(Action a)`



- Компонент **JList** — это список с полосой прокрутки, в котором можно выделить один или несколько пунктов.
- Количество видимых на экране пунктов определяется конструктором списка и размером компонента.
- Список значительно отличается от JComboBox и не только по внешнему виду. В то время как JComboBox выпадает вниз при активации, JList занимает определенное фиксированное число строк на экране все время и не изменяется.
- Компонент JList располагает элементы последовательно, один за другим, и эти элементы могут быть выбраны по отдельности или в группе.
- Класс **JList** может выводить как строки текста, так и значки.

- **JList** позволяет множественный выбор: если вы используете кнопку CTRL при щелчке мышью на более чем одном элементе (удерживайте кнопку CTRL при выполнении дополнительных щелчков мышью), начальный элемент остается подсвеченным, и вы можете выбрать столько элементов, сколько хотите.
- Конструкторы:
  - ❖ `public JList()` – создаёт JList с пустой моделью.
  - ❖ `public JList (ListModel dataModel)` – выводит элементы в определённой, ненулевой модели списка.
  - ❖ `public JList(Object [] listData)`
    - выводит элементы заданного массива "listData".



# JComboBox

- Компонент **JComboBox** - комбинированное поле, которое объединяет кнопку или доступное для редактирования поле и раскрывающийся список.
- Комбинация текстового поля и спускающегося списка.
- Конструкторы:
  - ❖ `public JComboBox();`
  - ❖ `public JComboBox(ComboBoxModel asModel)` – комбо-бокс, который берёт свои элементы из существующей модели `ComboBoxModel`.
  - ❖ `public JComboBox(Object [] items)` – комбо-бокс, который содержит элементы заданного массива.

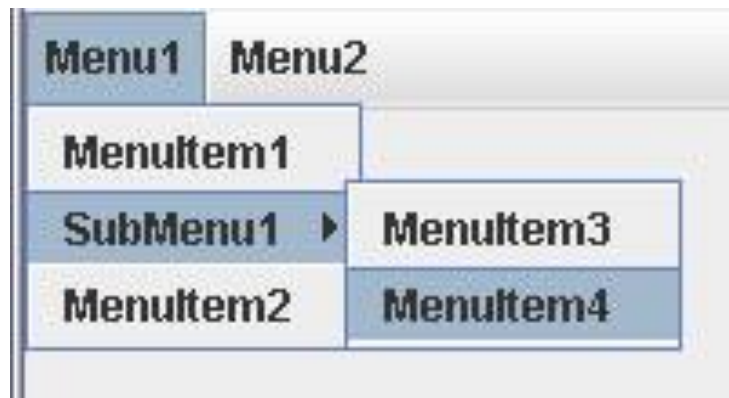


# Меню

- Меню выводит список элементов, которые обозначают различные задачи.
- В меню выводятся группы из нескольких элементов, которые объединены по категориям.
- После выбора или щелчка по пункту (группы) открывается другой список или подменю.
- Swing-меню состоит из панели меню (**menubar**), пунктов меню (**menu\_items**) и других меню (**menus**).
- Панель меню является корневым элементом для всех меню и пунктов меню.

# JMenuBar

- Компоненты **JMenu**, **JMenuItem** и **JMenuBar** являются главными строительными блоками для разработки системы меню в вашем JFrame.
- Основой любой системы меню является **JMenuBar**.
- Для присоединения **JMenuBar** к JFrame используется метод **setJMenuBar()**. После его закрепления в JFrame вы можете добавлять все меню, подменю и элементы меню, какие хотите.
- **JMenuBar** с компонентами **JMenu** и **JMenuItem** в Swing:



Важными методами, которые вам нужны в этих классах, являются:

- **JMenuItem и JMenu:**

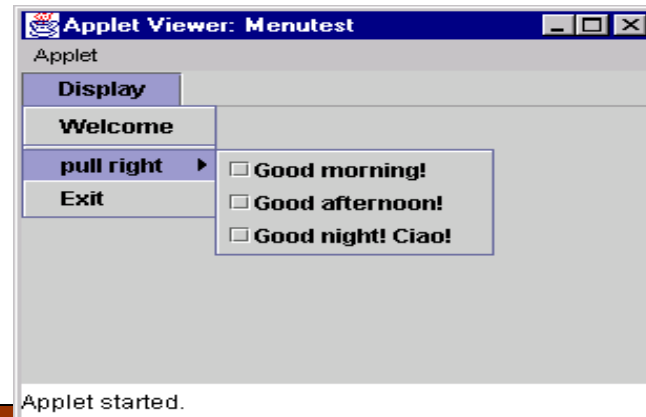
- ❖ **get/setText():** Получить/установить текст для меню.
- ❖ **get/setIcon():** Получить/установить изображение, используемое в меню.

- **Только JMenu:**

- ❖ **add():** Добавить еще один JMenu или JMenuItem к JMenu (создание вложенного меню).

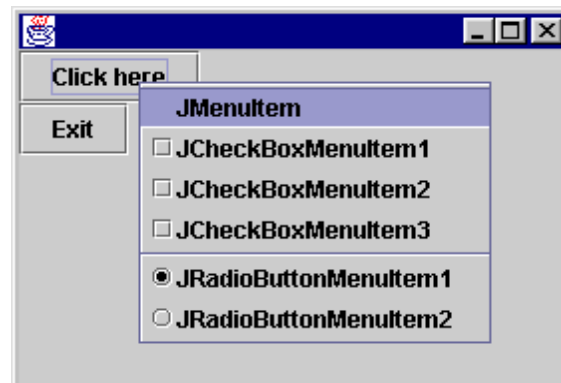
# JCheckBoxMenuItem

- Это подкласс класса `JMenuItem`.
- Содержит чек-боксы в качестве пунктов меню.
- Чек-боксы создаются с помощью класса `JCheckBox`.
- Информация о внешнем состоянии чек-бокса, такая как текст, значок и цвет фона может быть изменена.
- При щелчке по пункту `JCheckBoxMenuItem` и последующем отпускании кнопки мыши состояние пункта меню изменяется на выбранное (с меткой) или не выбранное.



# JRadioButtonMenuItem

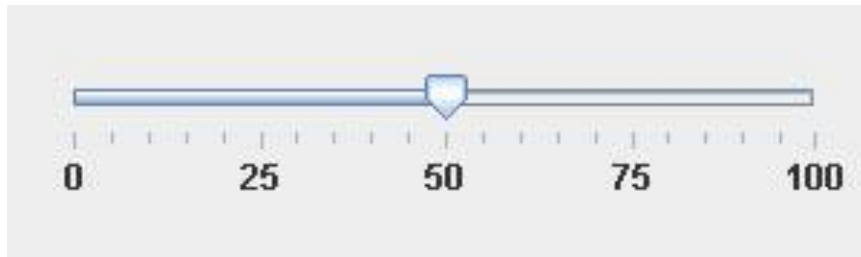
- Похожи на чек-боксы, за исключением того, что одновременно можно выбрать только одну радиокнопку.
- Могут содержать строку текста и/или значок.
- При выборе радиокнопки может возникнуть одна из двух ситуаций:
  - Щелчок по ранее выбранной радиокнопке не изменяет её состояние.
  - Щелчок по невыбранной радиокнопке отменяет выбор ранее выбранной радиокнопки и выбирает текущую кнопку.





# JSlider

- Компонент **JSlider** используется в приложении для изменения числового значения.
- Это быстрый и простой способ позволить пользователям визуально получить ответную реакцию не только на их текущий выбор, но увидеть диапазон допустимых значений.



Важными методами в JSlider являются:

- **get/setMinimum():** Получить/установить минимальное значение, которое вы можете выбрать.
- **get/setMaximum():** Получить/установить максимальное значение, которое вы можете выбрать.
- **get/setOrientation():** Получить/установить ориентацию JSlider (вверх/вниз или вправо/влево).
- **get/setValue():** Получить/установить начальное значение JSlider.

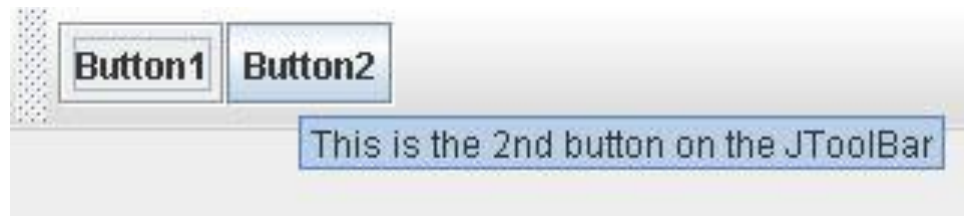
# JSpinner

- JSpinner намного более гибок и может быть использован для выбора из любой группы значений. Кроме выбора чисел он может быть использован для выбора дат, имен, цветов, чего-угодно.
- Важными методами являются:
  - ❖ **getValue()**: Получить/установить начальное значение JSpinner, которое в базовом случае должно быть целым числом.
  - ❖ **getNextValue()**: Получить следующее значение, которое будет выбрано после нажатия клавиши управления курсором "стрелка вверх".
  - ❖ **getPreviousValue()**: Получить предыдущее значение, которое будет выбрано после нажатия клавиши управления курсором "стрелка вниз".



# JToolTip

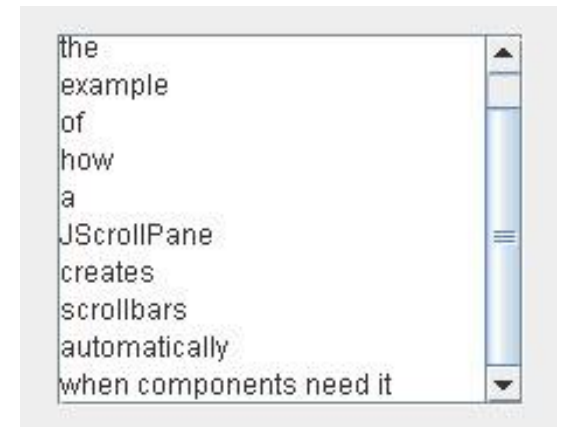
- Они активизируются в Swing, если оставить курсор мышки над компонентом на определенное количество времени.
- Метод **setToolTip()** является методом класса `JComponent`, то есть, каждый Swing-компонент может иметь всплывающую подсказку, связанную с НИМ.



# JScrollPane

Методами, которые вы должны использовать с JScrollPane, являются:

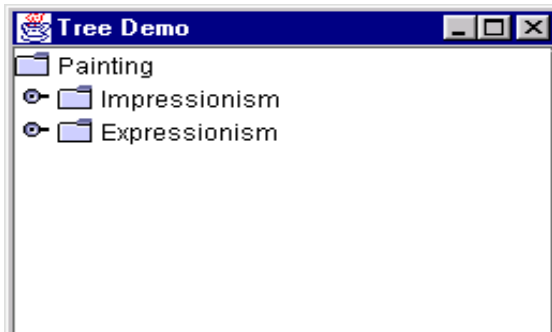
- ❖ **getHorizontalScrollBar():** Возвращает горизонтальный компонент JScrollBar.
- ❖ **getVerticalScrollBar():** Возвращает вертикальный компонент JScrollBar.
- ❖ **get/setHorizontalScrollBarPolicy():** Эта "политика" может принимать одно из следующих значений: Always (всегда), Never (никогда), или As Needed (по необходимости).
- ❖ **get/setVerticalScrollBarPolicy():** Аналогично горизонтальной функции.



# Деревья

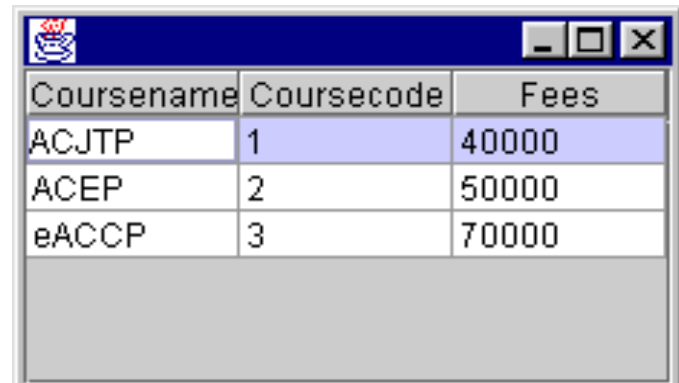
- **Дерево (Tree)** изображает информацию в иерархической форме с направлением чтения по вертикали.
- Проводник Windows (Explorer) использует древовидную структуру для изображения файлов и папок.
- Структуры Проводника Windows могут быть созданы на языке Java с использованием `JTree`.
- Каждая строка в иерархии определяется термином узел (node).
- По умолчанию дерево выводит свой корневой узел (root node).
- Узел, содержащий узлы-потомки (child nodes), называется узлом с ответвлениями или просто «ветвь» (branch node); в противном случае он называется конечным узлом или «листом» дерева (leaf node).

# Пример



# Таблицы

- С большими объёмами данных легче работать в табличном формате, чем в виде списка.
- Таблица удобна для хранения числовых данных.
- Некоторым компьютерным приложениям необходимы таблицы для вывода данных с предоставлением пользователю возможности редактировать их.
- Класс `JTable` в `Swing` позволяет создавать таблицы.
- `JTable` не хранит данные, он всего лишь создаёт их визуальное представление.



Course name	Course code	Fees
ACJTP	1	40000
ACEP	2	50000
eACCP	3	70000