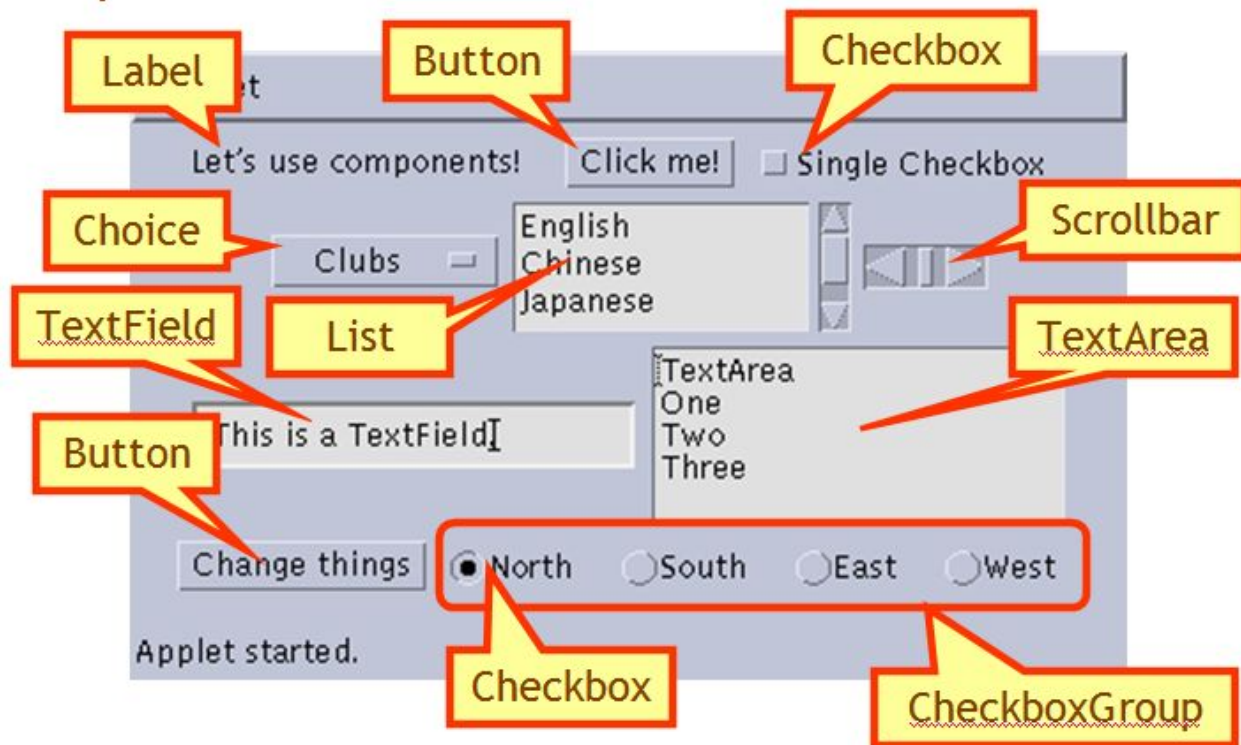


A background image showing a complex network of interconnected nodes and lines, resembling a web or a data network, set against a blue gradient.

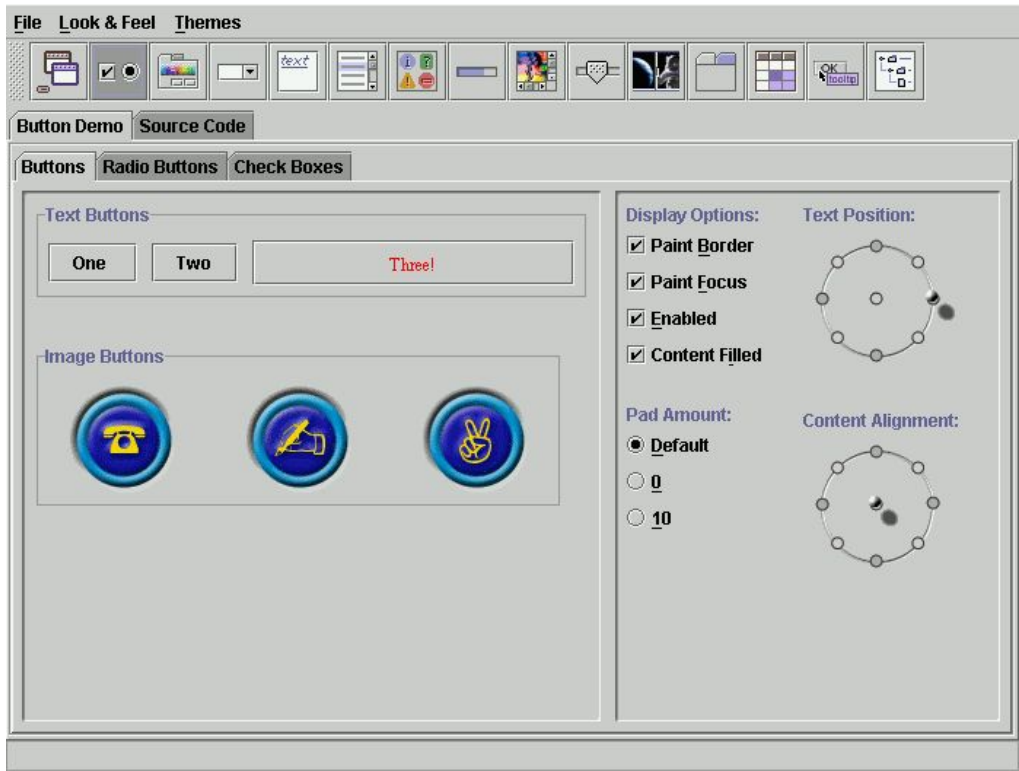
Лекция 14. Графический интерфейс пользователя

NetCracker®

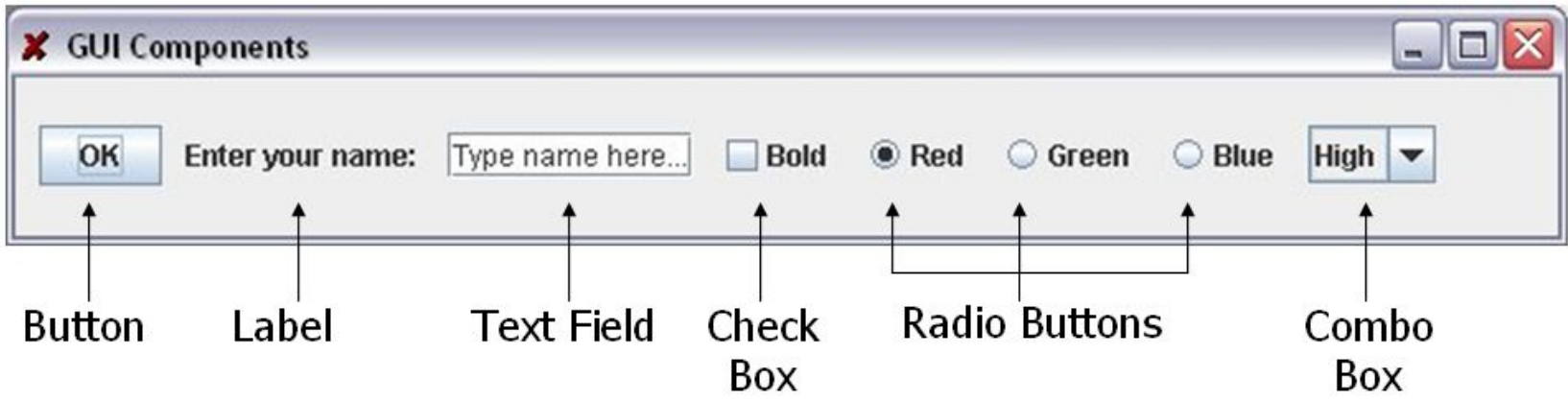
- AWT – платформозависимая, java.awt.*
- Swing – платформонезависимая, java.swing.*
- SWT – платформозависимая



Java имеет стандартные пакеты для создания интерфейсов пользователя (**Graphical User Interfaces**).



Основные компоненты интерфейса:



- Присутствует во всех реализациях Java
- Описанный в большинстве Java учебников
- Адекватная для многих приложений
- Использует элементы управления, определенные ОС
- Трудно построить понятный интерфейс

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

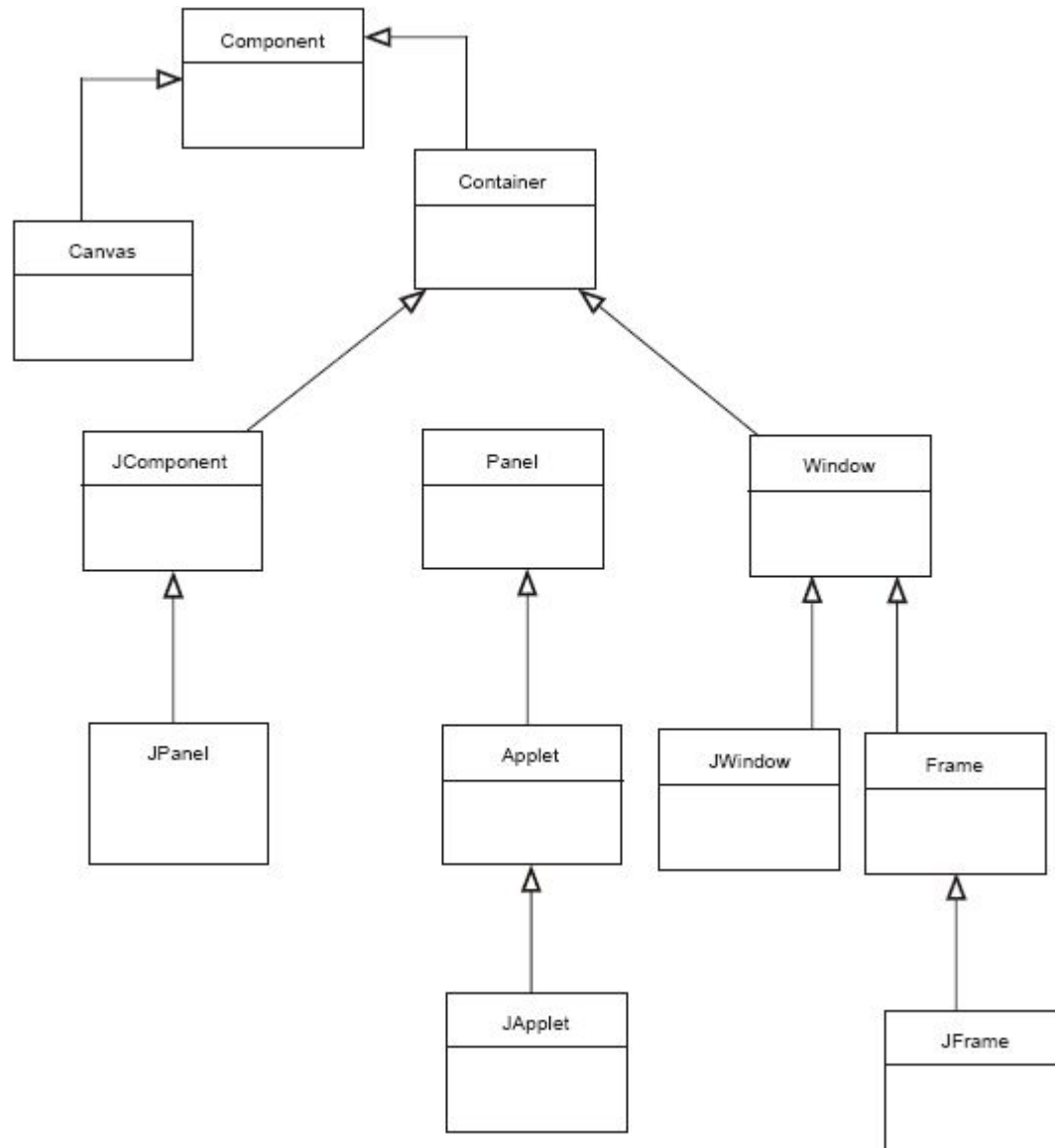
- Схожа с AWT
- Не работает в ранних версиях Java реализаций (Java 1.1 и выше)
- Намного больше элементов более гибких управления
- Некоторые элементы управления являются гораздо более сложными
- Гораздо проще построить понятный интерфейс

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

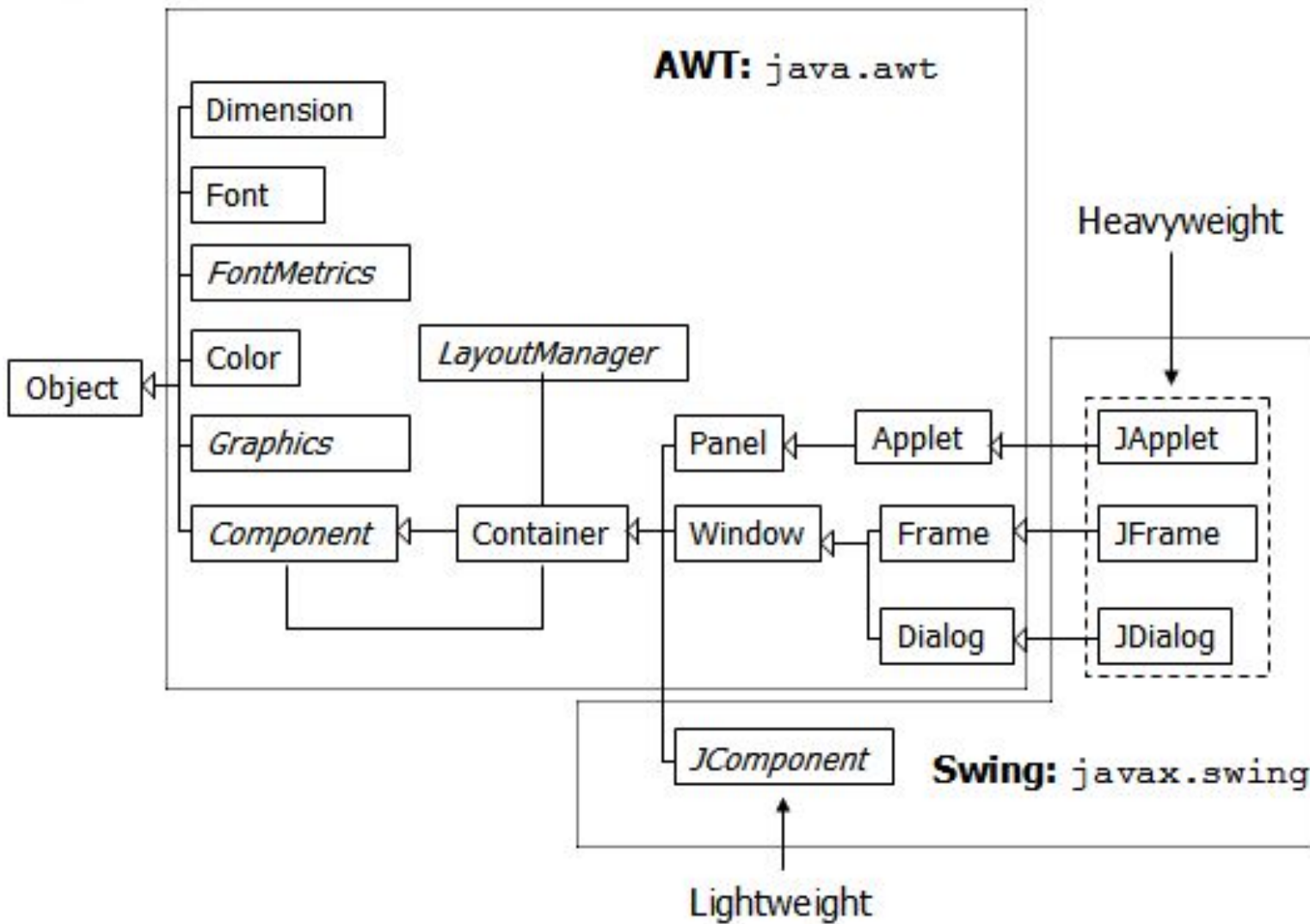
- Пакет Swing большой, работает медленнее, и сложнее, чем AWT
- Swing является более гибким и его элементы лучше выглядят
- Swing vs. AWT несовместимы - нужно использовать любой один пакет
- Изучение AWT является хорошим началом для Swing
- Многие из наиболее распространенных элементов управления похожи

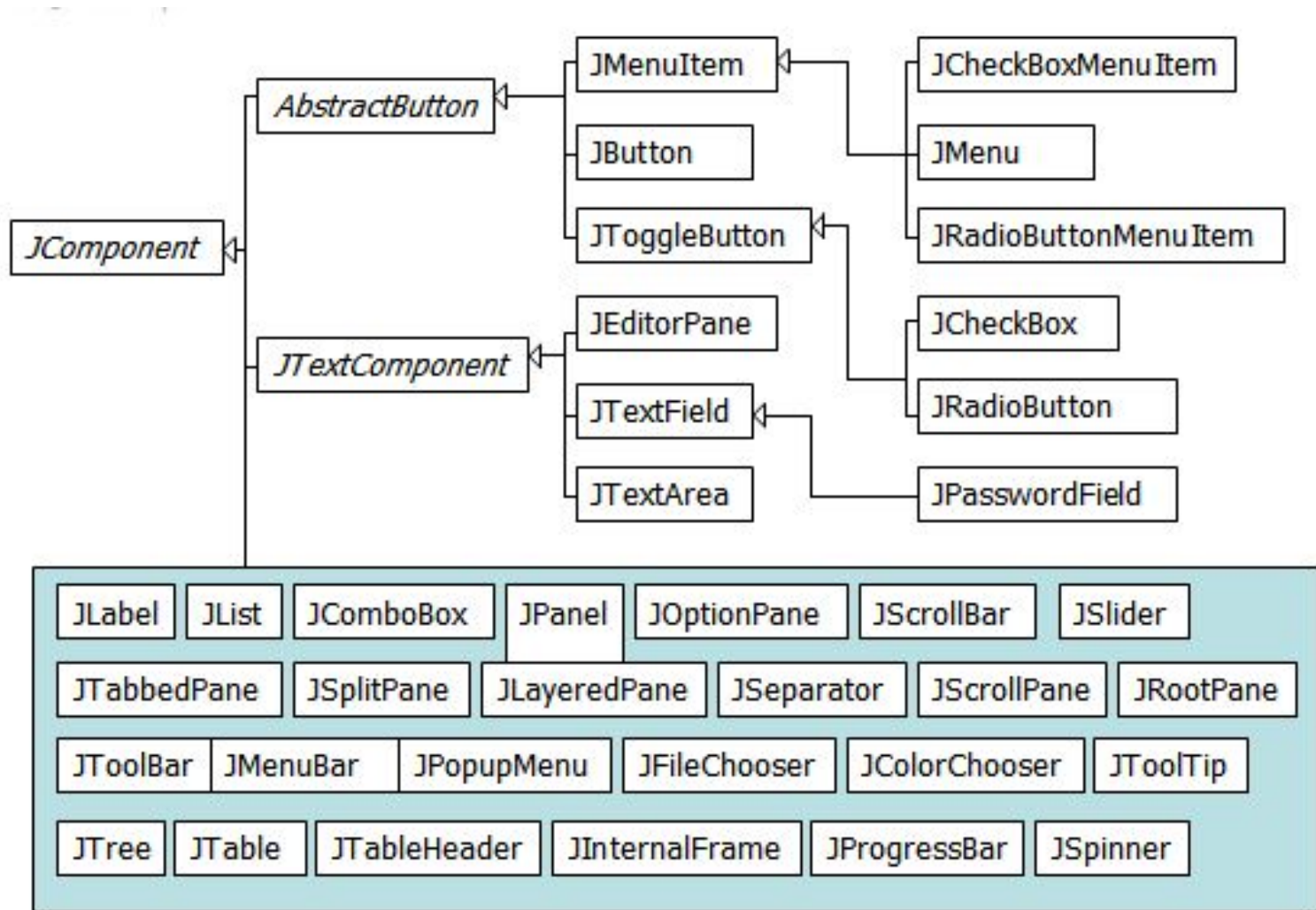
AWT: `Button b = new Button("OK");`

Swing: `JButton b = new JButton("OK");`



- Тяжеловесные (heavyweight) компоненты
 - Отрисовываются операционной системой
 - Большинство AWT-компонент
- Легковесные (lightweight) компоненты
 - Отрисовываются java-кодом
 - Все Swing-компоненты, кроме окон верхнего уровня
- Тяжеловесные компоненты всегда отрисовываются поверх легковесных





- Классы **Container** – это GUI-компоненты, которые используются как контейнеры для других GUI-компонентов
- Swing: Component, Container, JFrame, JDialog, JApplet, JPanel
 - **JFrame** - окно, не содержащее внешних окон
 - **JDialog** - временное всплывающее окно или сообщение
 - **JApplet** — апплет
 - **JPanel** - контейнер, содержащий UI-компонеты или графические элементы
- **Layout manager** используется для позиционирования компонентов

Части интерфейса пользователя, содержащие другие компоненты

- JPanel – панель
 - JFrame – окно приложения
 - JDialog – диалоговое окно
 - JScrollPane – область с полосой прокрутки
-
- `add(Component component)` — добавляет в контейнер элемент `component`;
 - `remove(Component component)` — удаляет из контейнера элемент `component`;
 - `removeAll()` — удаляет все элементы контейнера;
 - `getComponentCount()` — возвращает число элементов контейнера.

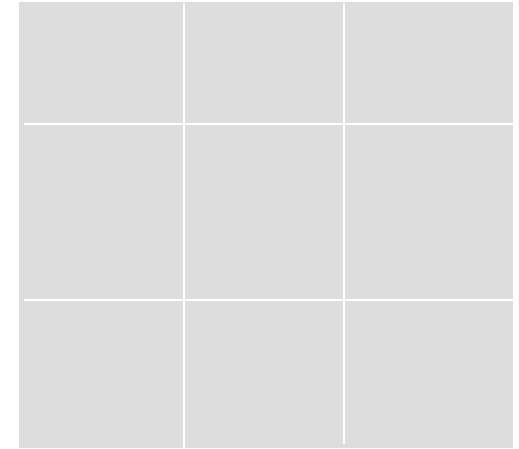
null

none,
programmer
sets x,y,w,h

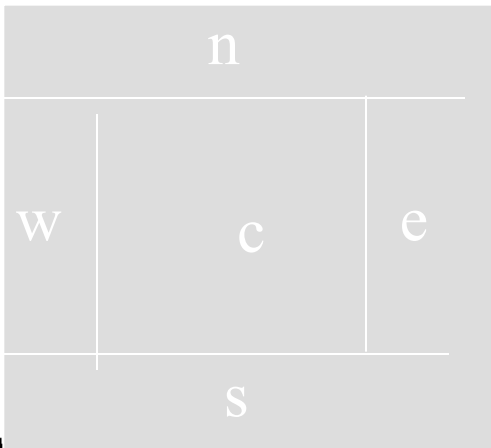
FlowLayout

Left to right,
Top to bottom

GridLayout



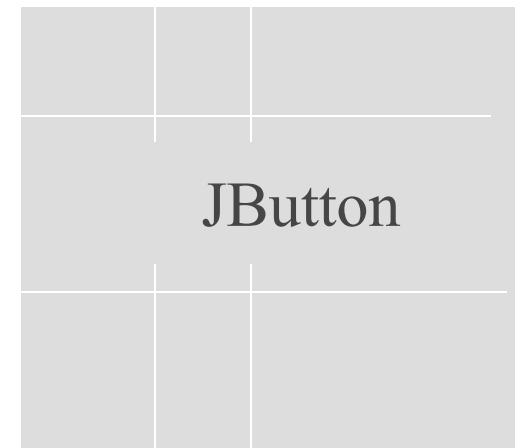
BorderLayout

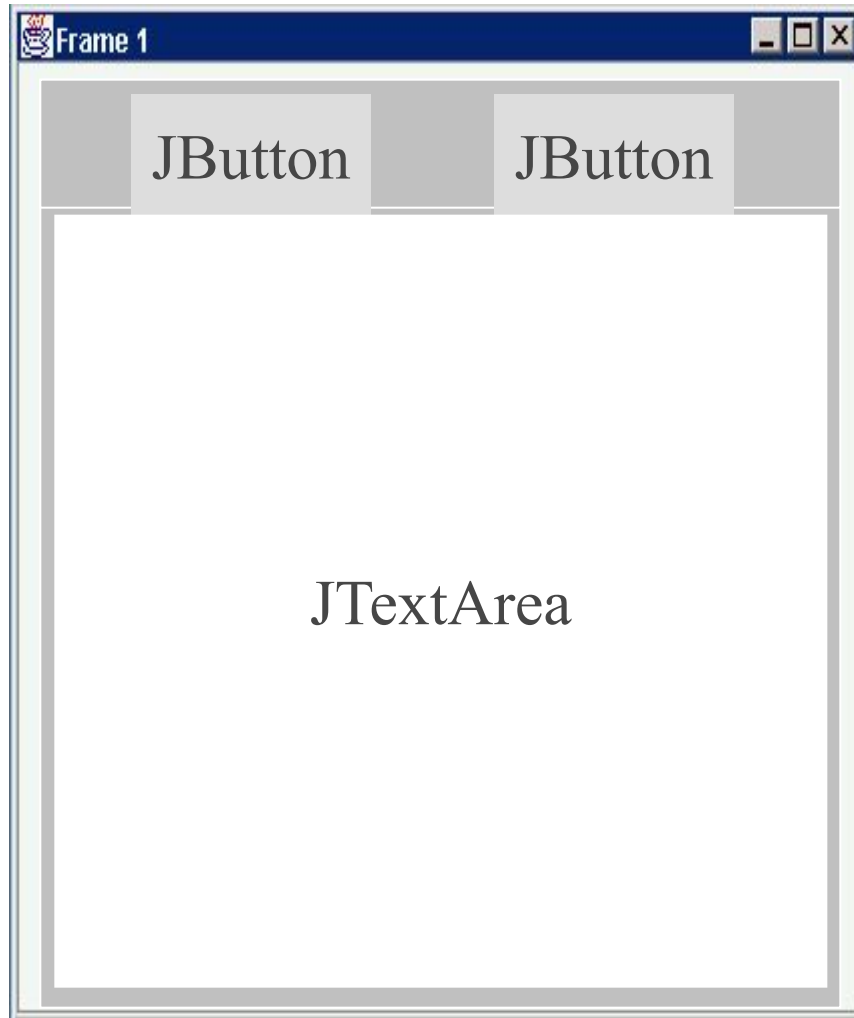


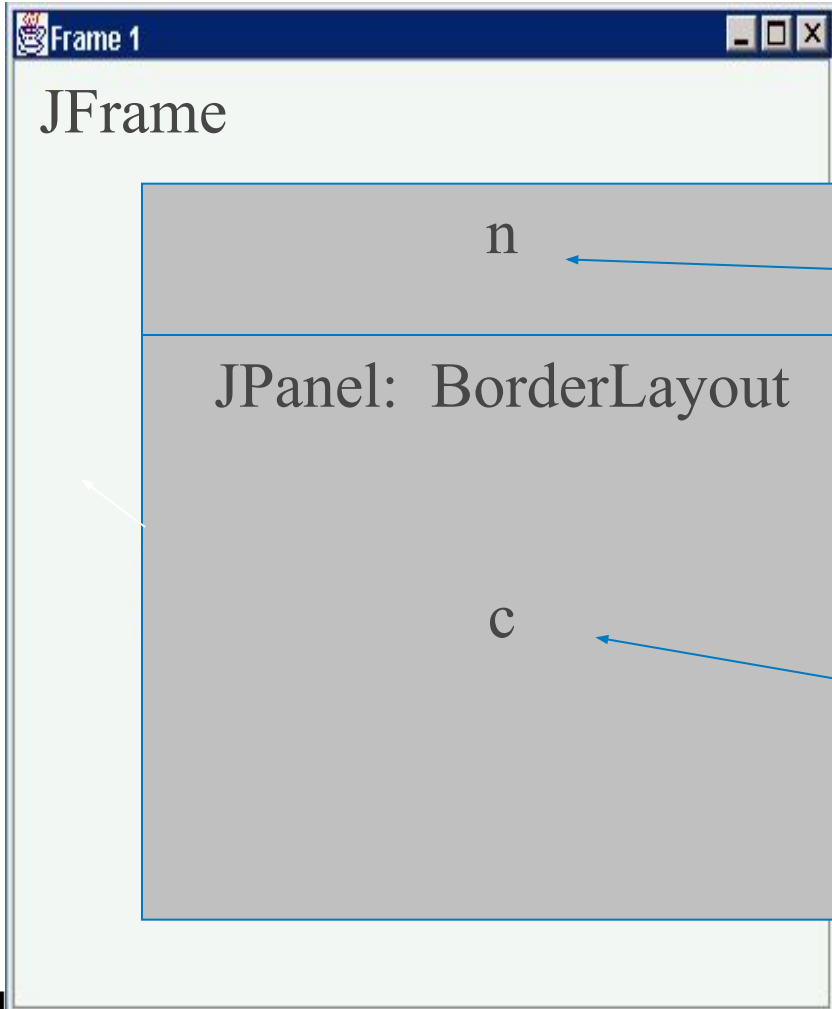
CardLayout

One at a time

GridBagLayout







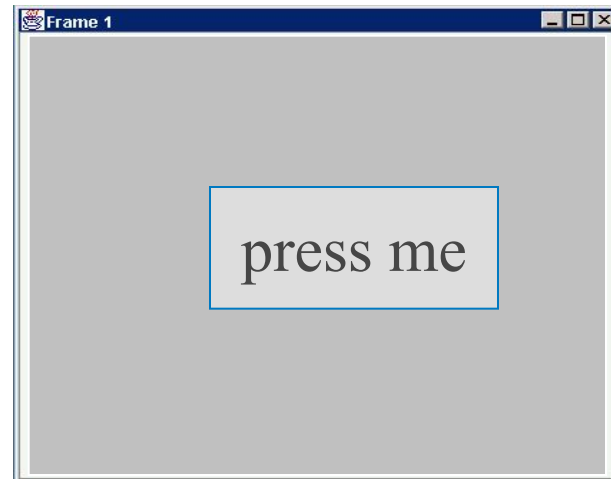
JButton

JButton

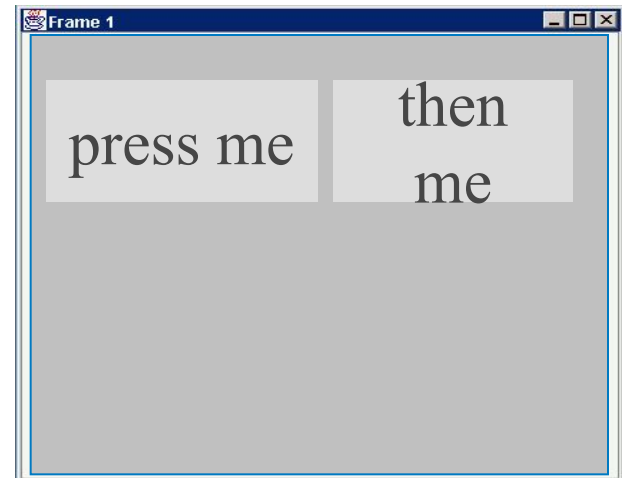
JPanel: FlowLayout

JTextArea

```
JFrame f = new JFrame("title");  
JPanel p = new JPanel( );  
JButton b = new JButton("press me");  
  
b.setBounds(new Rectangle(10,10, 100,50));  
p.setLayout(null);    // x,y layout  
p.add(b);  
f.setContentPane(p);
```




```
JFrame f = new JFrame("title");  
JPanel p = new JPanel( );  
FlowLayout L = new FlowLayout( );  
JButton b1 = new JButton("press me");  
JButton b2 = new JButton("then me");  
  
p.setLayout(L);  
p.add(b1);  
p.add(b2);  
f.setContentPane(p);
```



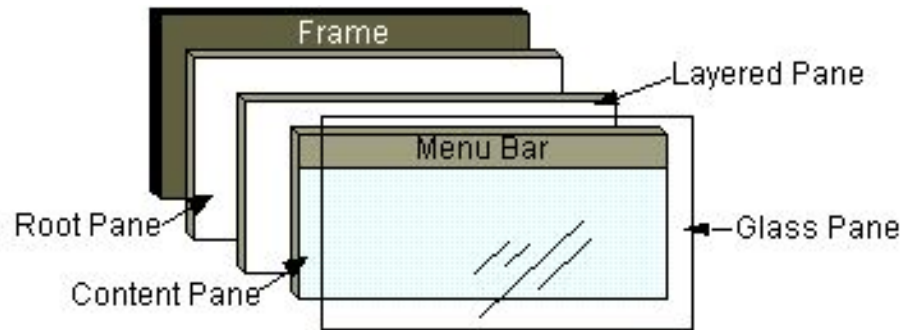
- Класс JFrame
- Конструкторы
 - JFrame(title)
- Свойства
 - title – заголовок
 - jMenuBar – меню
 - iconImage – иконка окна

- Метод
 - `setDefaultCloseOperation(operation)` – установить действие при закрытии окна
 - `DO_NOTHING_ON_CLOSE`
 - `HIDE_ON_CLOSE`
 - `DISPOSE_ON_CLOSE`
 - `EXIT_ON_CLOSE` (JFrame)

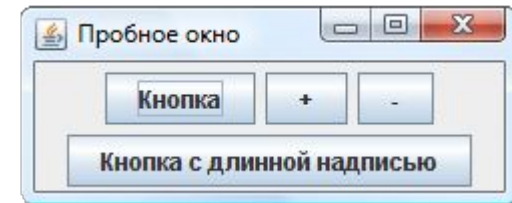
- Класс JOptionPane
- Методы
 - `showConfirmDialog(...)` – да/нет/отмена
 - `showInputDialog(...)` – ввод текста
 - `showMessageDialog(...)` – информация
 - `showOptionDialog(...)` – выбор из списка
- Параметры
 - `parentComponent` – родительская компонента
 - `message` – сообщение
 - `optionType` – набор кнопок
 - `messageType` – вид иконки

- Методы

- `getXXXPane()` – возвращает панель
- `setXXXPane()` – устанавливает панель
- `getContentPane()`, `setContentPane()`



```
JButton newButton = new JButton();  
getContentPane().add(newButton);
```



Части интерфейса пользователя, не содержащие других компонентов

- JLabel – метка
- JButton – кнопка
- JMenuItem – элемент меню
- JTextArea – редактор текста

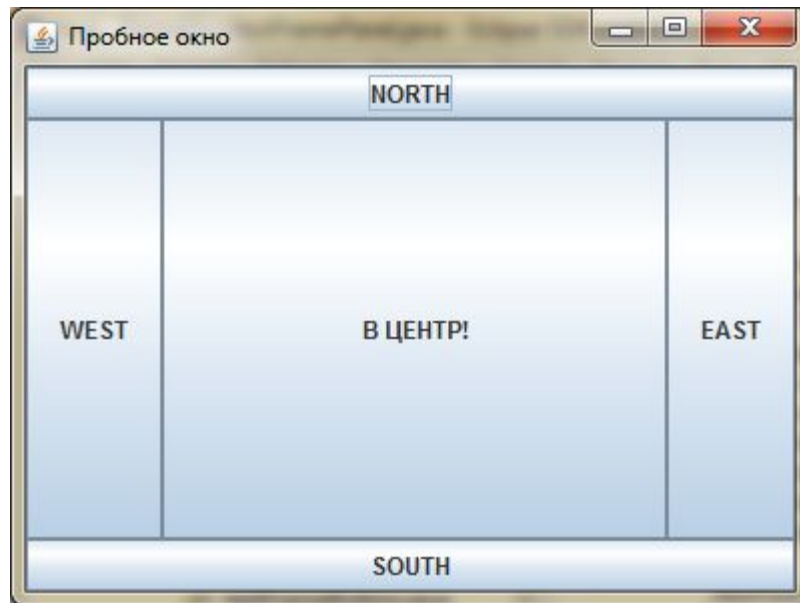
Возможности компонентов

- Генерация событий
- Обработка ввода пользователя
- Рамки
- Отрисовка “в ручную”
- Поддержка Drag & Drop
- компоновка
- ...

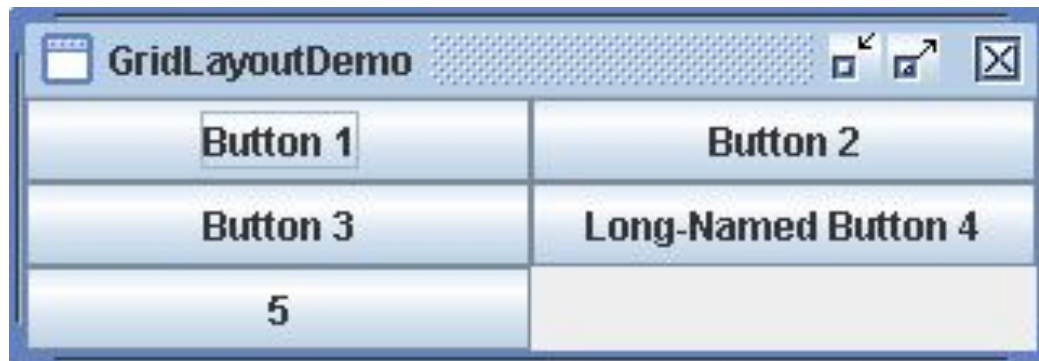
- Размещают компоненты внутри контейнера
- Интерфейс `java.awt.LayoutManager`
- `panel.setLayout(new FlowLayout());`
- Разместить компоненты так, что бы удовлетворялись рекомендации
- Рекомендации по размеру
 - `Dimension minimumSize` – минимальный
 - `Dimension preferredSize` – наилучший
 - `Dimension maximumSize` -- максимальный

- Компоненты выкладываются одна за другой, с переносом строк
- Свойства
 - **alignment** – выравнивание
 - LEADING, CENTER, TRAILING
 - **vgap** / **hgap** – расстояние по горизонтали / вертикали

- Компоненты располагаются по краям
- Свойства
 - **vgap** / **hgap** – расстояние по вертикали / горизонтали



- Компоненты располагаются в виде таблицы
- Свойства
 - **rows** / **columns** – количество строк / столбцов
 - **vgap** / **hgap** – расстояние по вертикали / горизонтали



- Выкладывает компоненты горизонтально / вертикально
- `Box.createHorizontalBox()`
- `Box.createVerticalBox()`

- `Box box = Box.createVerticalBox();`
- `box.add(new JButton("Кнопка"));`
- `box.add(Box.createVerticalStrut(10));`
- `box.add(Box.createVerticalGlue());`

- **CardLayout** – помещает компоненты друг за другом
- **GridBagLayout** – помещает компоненты в гибкую таблицу
- **SpringLayout** – очень гибкий компоновщик, используется при кодогенерации

Запуск компоновщика

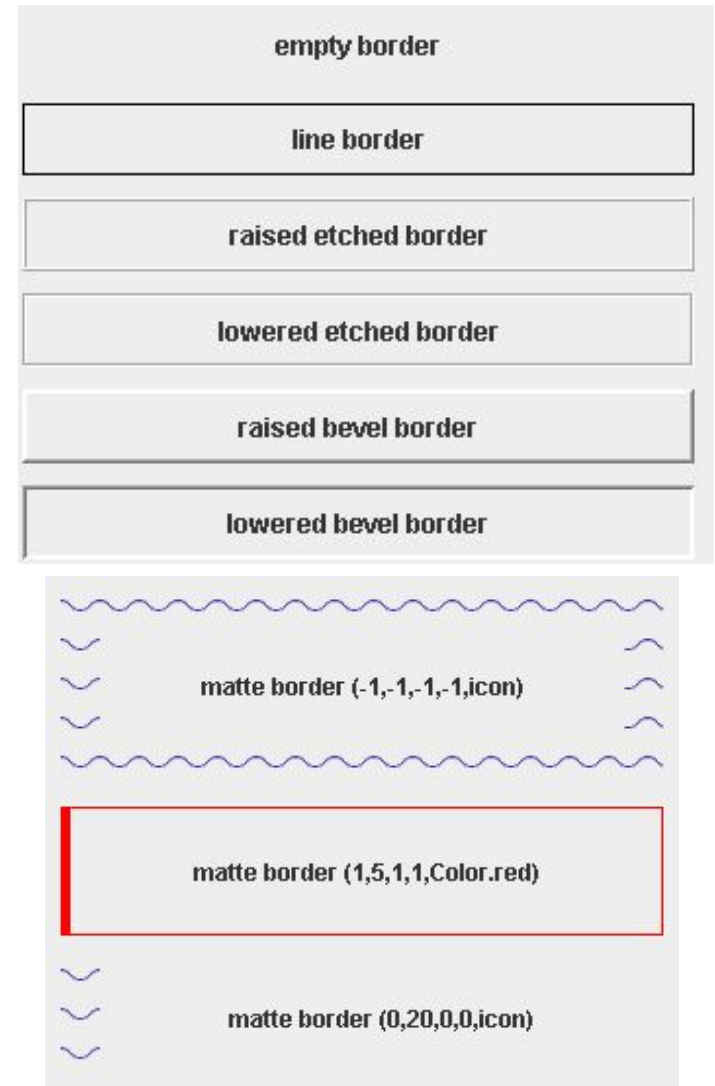
- Автоматически – при изменении размера контейнера
- Вручную
 - **invalidate()** – запросить перекomпоновку компоненты и всех ее предков
 - **revalidate()** – thread-safe invalidate()

Обрамление

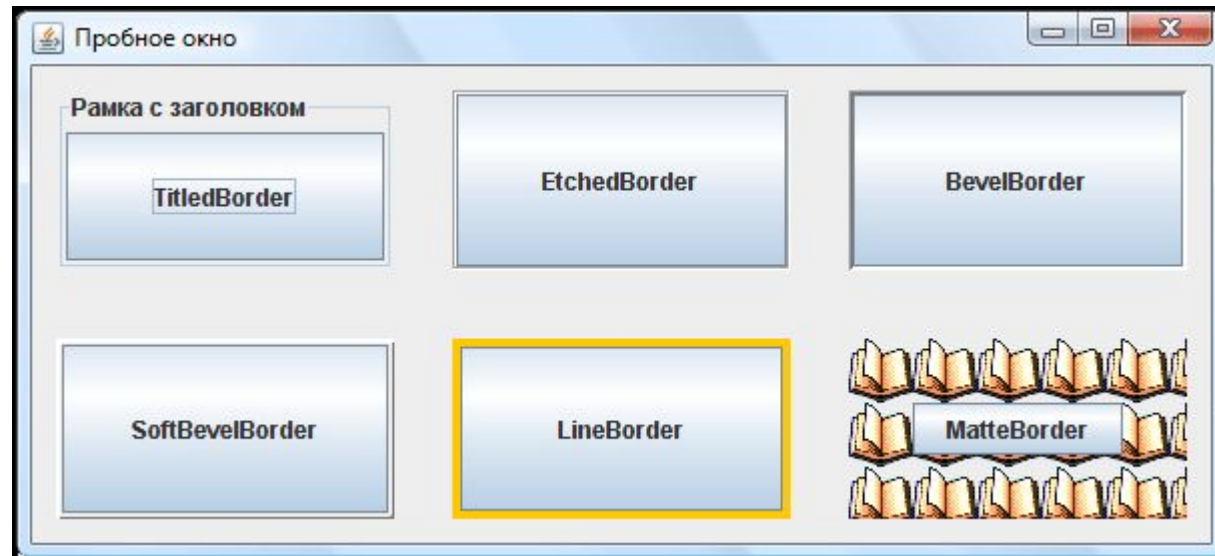
- Каждая компонента может иметь обрамление в виде рамки
- Пакет `javax.swing.border`
- Класс `Border`

- Размер обрамления вычитается из размера компоненты
- Класс `Insets`
- Конструктор `Insets(left, right, bottom, top)`
- Поля
 - `left` – отступ слева
 - `right` – отступ справа
 - `bottom` – отступ снизу
 - `top` – отступ сверху

- Классы
 - **EmptyBorder** – пустое место
 - **LineBorder** – линия
 - **EtchedBorder** – объемность
 - **BevelBorder** – выпуклость / вдавленность
 - **MatteBorder** - Обрамление “набирается” из рисунка



- **TitledBorder** – обрамление с заголовком. Создается на основе другого обрамления
- **CompoundBorder** – объединяет два обрамления
 - **CompoundBorder(insideBorder, outsideBorder)**



- Простейший контейнер
- Конструктор
 - JPanel(LayoutManager)
- Свойства
 - `layoutManager` -- КОМПОНОВЩИК

- **JLabel** - Метка с текстом
- Конструктор
 - **JLabel(text?, icon?)**
- Свойства
 - **text** – надпись на метке
 - **icon** – картинка

- JButton(String text?, Icon icon?)
- setRolloverIcon(Icon icon)
- setPressedIcon(Icon icon)
- setMargin(Insets margin)

- **JToggleButton** – кнопка, которая может находиться в двух состояниях:
нажатом и отпущенном
- **JCheckBox, JRadioButton** – наследники
- **ButtonGroup** – взаимоисключающий контейнер

- JTextField
 - setText(String text)
 - getText(int offset, int length)
- JPasswordField
 - set(get)EchoChar(char echo)
- JTextArea
 - append(String text)
 - insert(String text, int position)

- Панель с полосами прокрутки
- Конструктор
 - **JScrollPane(Component?, vsbPolicy?, hsbPolicy?)**
 - <dir>_SCROLLBAR_AS_NEEDED
 - <dir>_SCROLLBAR_NEVER
 - <dir>_SCROLLBAR_ALWAYS
- `getContentPane().add(new JScrollPane(textArea));`

- Класс `ImageIcon`
- Конструктор
 - `ImageIcon(url)` – загрузить по URL
 - `ImageIcon(file)` – загрузить из файла
- Методы
 - `getIconHeight()` – высота иконки
 - `getIconWidth()` – ширина иконки
 - `getImage()` – платформозависимый рисунок
- Применение
 - `frame.setIconImage(icon.getImage())`
 - `new JLabel(icon);`

- **JToolBar**
- **JComboBox**
- **JSlider**
- **JTabbedPane**
- **JList**
- **JProgressBar**

- Низкоуровневые события
 - Создаются системой на основе действий пользователя
 - Инициатор события – текущая компонента
- Высокоуровневые события
 - Создаются компонентами на основе других событий
 - Инициатор события – компонента создавшая событие

- Ввод пользователя
 - **InputEvent** – базовый класс
 - **KeyEvent** – событие клавиатуры
 - **MouseEvent** – событие мыши
 - **MouseEvent** – событие колеса прокрутки
- Изменение состояния компоненты
 - **ComponentEvent** – изменение видимости / размера / местоположения компонента
 - **FocusEvent** – изменение фокуса
 - **ContainerEvent** – добавление / удаление КОМПОНЕНТ
 - **WindowEvent** – операции с окнами

- Генерация событий
 - Клавиатурные – для компоненты владеющей фокусом
 - Мыши – для компоненты, над которой находится мышь
 - Прочие – для компоненты с которой произошли
- Событие ввода может быть поглощено
 - Метод `consume()`

- Примеры
 - **ActionEvent** – нажатие на кнопку
 - **MenuItemEvent** – операции с меню
 - **PopupMenuEvent** – операции с всплывающим меню
 - ...

- Оповещаются о возникновении события
- Интерфейсы `XXXListener`
- Управление слушателями
 - Метод `addXXXListener(XXXListener listener)` – добавить слушателя
 - Метод `removeXXXListener(XXXListener listener)` – убрать слушателя

- Реализация слушателя
 1. Реализовать интерфейс
 2. Добавить слушателя к компоненту
 3. Реагировать на события
- Вспомогательные классы
 - **XXXAdapter** – для реализации слушателей с несколькими методами

- Слушатель событий от мыши должен реализовать интерфейс `MouseListener`. В этом интерфейсе перечислены следующие методы:
- `mouseClicked(MouseEvent event)` — выполнен щелчок мышкой на наблюдаемом объекте
- `mouseEntered(MouseEvent event)` — курсор мыши вошел в область наблюдаемого объекта
- `mouseExited(MouseEvent event)` — курсор мыши вышел из области наблюдаемого объекта
- `mousePressed(MouseEvent event)` — кнопка мыши нажата в момент, когда курсор находится над наблюдаемым объектом
- `mouseReleased(MouseEvent event)` — кнопка мыши отпущена в момент, когда курсор находится над наблюдаемым объектом

- FocusListener
- MouseWheelListener
- KeyListener
- ChangeListener
- WindowListener
- ComponentListener – смена положения, размера...
- ActionListener – универсальный слушатель
 - actionPerformed(ActionEvent event)

- Событие `ActionEvent`
 - СВОЙСТВА
 - `getActionCommand()` – название команды
 - `getModifiers()` – состояние клавиш-модификаторов
 - `getWhen()` – когда произошло
- Слушатель `ActionListener`
 - Метод `actionPerformed(ActionEvent e)`

- Действие – абстракция действия которое можно произвести
- Интерфейс `Action`
- Методы
 - `actionPerformed(ActionEvent)` – совершить действие
 - `setEnabled(boolean)` – запретить / разрешить
 - `isEnabled()` – проверить разрешение
 - `putValue(key, value)` – записать значение свойства
 - `getValue(key)` – прочитать значение свойства

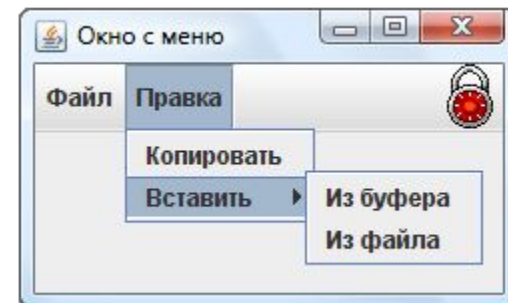
- Константы интерфейса **Action**
 - **NAME** – название действия
 - **SHORT_DESCRIPTION** – описание для всплывающих подсказок
 - **LONG_DESCRIPTION** – описание для контекстной помощи
 - **ACTION_COMMAND_KEY** – имя команды
 - **SMALL_ICON** – иконка

- Основное меню
 - Класс `JMenuBar`
- Раскрывающееся меню
 - Класс `JMenu`
- Элементы меню
 - Класс `JMenuItem` – простой
 - Класс `JCheckBoxMenuItem` – помечаемый
 - Класс `JRadioButtonMenuItem` – один из
 - Класс `JSeparator` – разделитель

```

SimpleWindow() {
super("Окно с меню");
setDefaultCloseOperation(EXIT_ON_CLOSE);
JMenuBar menuBar = new JMenuBar();
JMenu fileMenu = new JMenu("Файл");
fileMenu.add(new JMenuItem("Новый"));
fileMenu.add(new JMenuItem("Открыть", new ImageIcon("1.gif")));
fileMenu.add(new JMenuItem("Сохранить"));
fileMenu.addSeparator();
fileMenu.add(new JMenuItem("Выйти"));
JMenu editMenu = new JMenu("Правка");
editMenu.add(new JMenuItem("Копировать"));
JMenu pasteMenu = new JMenu("Вставить");
pasteMenu.add(new JMenuItem("Из буфера"));
pasteMenu.add(new JMenuItem("Из файла"));
editMenu.add(pasteMenu)
;menuBar.add(fileMenu);
menuBar.add(editMenu);
menuBar.add(Box.createHorizontalGlue());
menuBar.add(new JLabel(new ImageIcon("2.gif")));
setJMenuBar(menuBar);
setSize(250,150);}

```



- Обработка сообщений и перерисовка интерфейса пользователя происходят в потоке событий (**EventThread**)
- Если занять **EventThread**, GUI “зависнет”
- С видимыми компонентами можно оперировать только в **EventThread**
- GUI рекомендуется создавать в **EventThread**

- Компонента считается видимой, если
 - Она добавлена к видимому контейнеру
- Окна считаются видимой
 - После вызова метода `pack()`
 - После вызова `setVisible(true)`

- Класс `SwingUtilities`
- Методы
 - `invokeLater(Runnable)` – выполнить метод `run()` в `EventThread`
 - `invokeAndWait(Runnable)` – выполнить метод `run()` в `EventThread` и дождаться окончания

Используемая литература:

- Аллен П., Бамбара Дж. J2EE. Разработка бизнес-приложений.

Thank you!

