

SWING

Графические библиотеки

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

Окна верхнего уровня

- Окно приложения
 - Класс `JFrame`
- Диалоговое окно
 - Класс `JDialog`
- Окно апплета
 - Класс `JApplet`
- Вложенное окно
 - Класс `JInternalFrame`

Demo

- testFrame.java
- testFrame2.java

Заккрытие окна

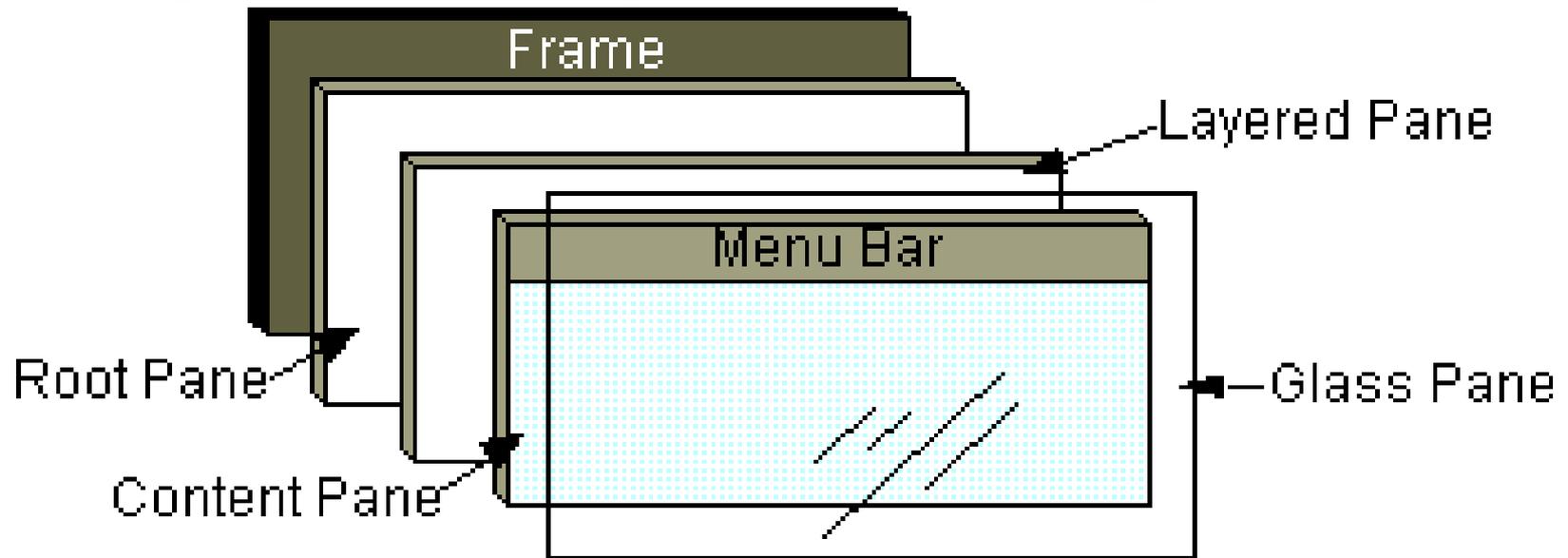
- Метод
 - `setDefaultCloseOperation(operation)` – установить действие при закрытии окна
 - `DO_NOTHING_ON_CLOSE`
 - `HIDE_ON_CLOSE`
 - `DISPOSE_ON_CLOSE`
 - `EXIT_ON_CLOSE (JFrame)`

Стандартные диалоги

- 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

Панель содержимого

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



Demo

- testFrameButton

Контейнеры

- **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.

КОМПОНОВЩИКИ

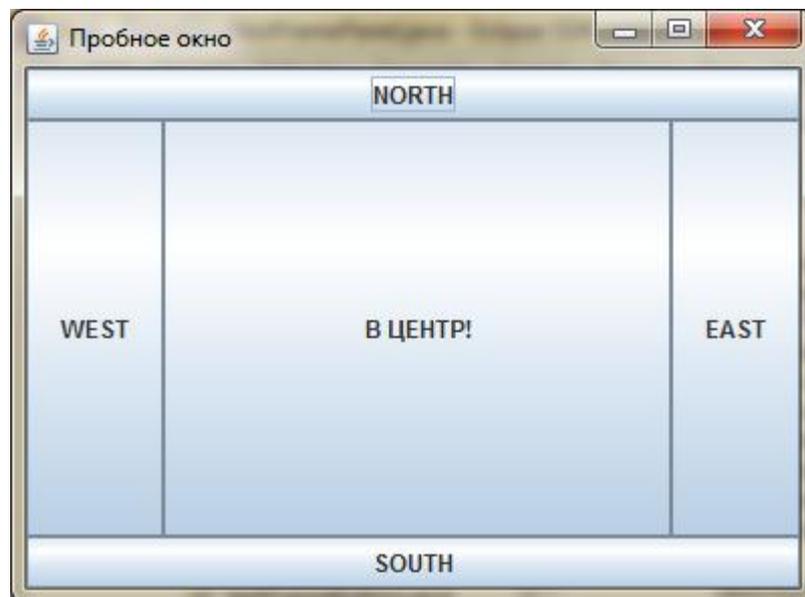
- Размещают компоненты внутри контейнера
- Интерфейс `java.awt.LayoutManager`
- `panel.setLayout(new FlowLayout());`

FlowLayout

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

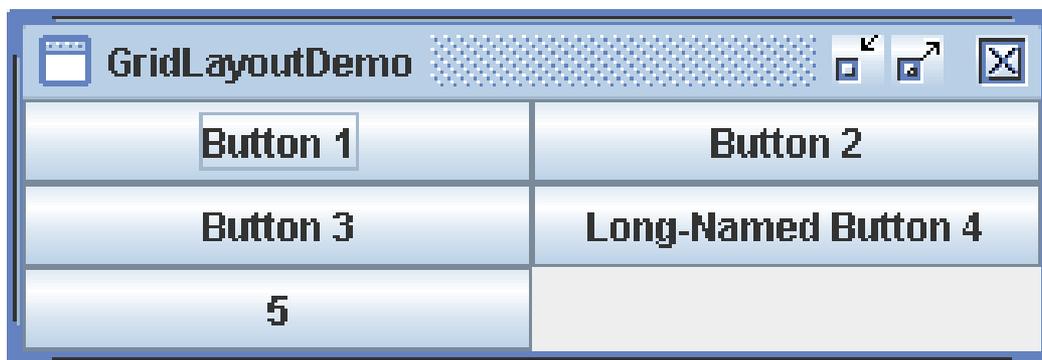
BorderLayout

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



GridLayout

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



BoxLayout

- 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());`

Demo

- `testFramePanel.java`

Обрамление

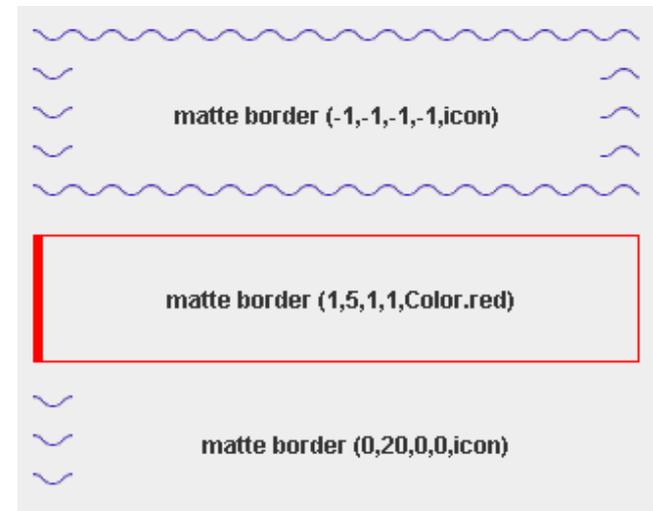
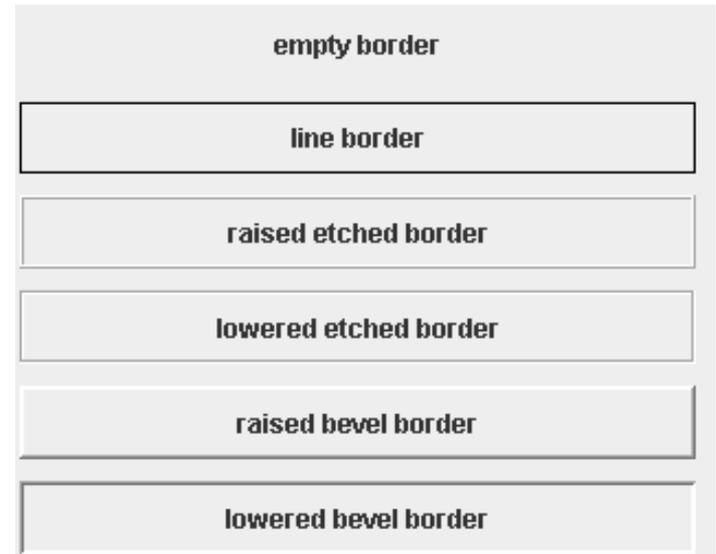
- Каждая компонента может иметь обрамление в виде рамки
- Пакет `javax.swing.border`
- Класс `Border`
- Метод
 - `Component.setBorder(Border)`

Размер оформления

- Размер оформления 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

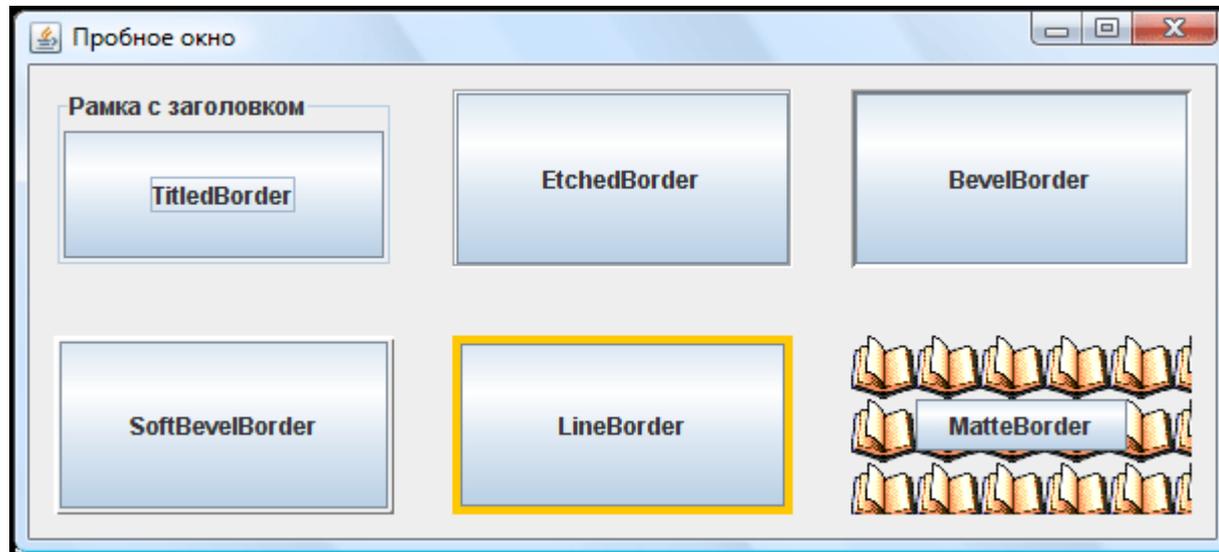
Типы обрaмлений (простые)

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



Типы обрамлений (составные)

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



Визуальные компоненты

- **JLabel** - Метка с текстом
- Конструктор
 - **JLabel(text?, icon?)**
- Свойства
 - **text** – надпись на метке
 - **icon** – картинка
- Demo: `testFrameComponents.java`

JButton - кнопка

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

JToggleButton, JCheckBox, JRadioButton

- JToggleButton - кнопка, которая может находиться в двух состояниях: нажатом и отпущенном
- JCheckBox, JRadioButton – наследники
- ButtonGroup – взаимоисключающий контейнер
- Demo: [SimpleWindowToggles](#)
- Ask to change

Визуальные компоненты

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

Панель прокрутки JScrollPane

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

Визуальные компоненты

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

Обработка событий

Слушатели

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

Создание слушателя

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

MouseListener

- 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

Demo

- `MouseListener`

Слушатели

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

DEMO

ActionListener

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

Действия

- 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

Свойства действий

- 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
- DEMO

Swing и потоки

- 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**

Видимые компоненты

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

Исполнение действий в EventThread

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