Графічний інтерфейс Swing мови Java

Основные компоненты Swing

Для использования языка Java в различных предметных областях была разработана группа пакетов с общим именем **JFC – Java Foundation Classes** (базовые классы Java).

В JFC включены следующие компоненты:

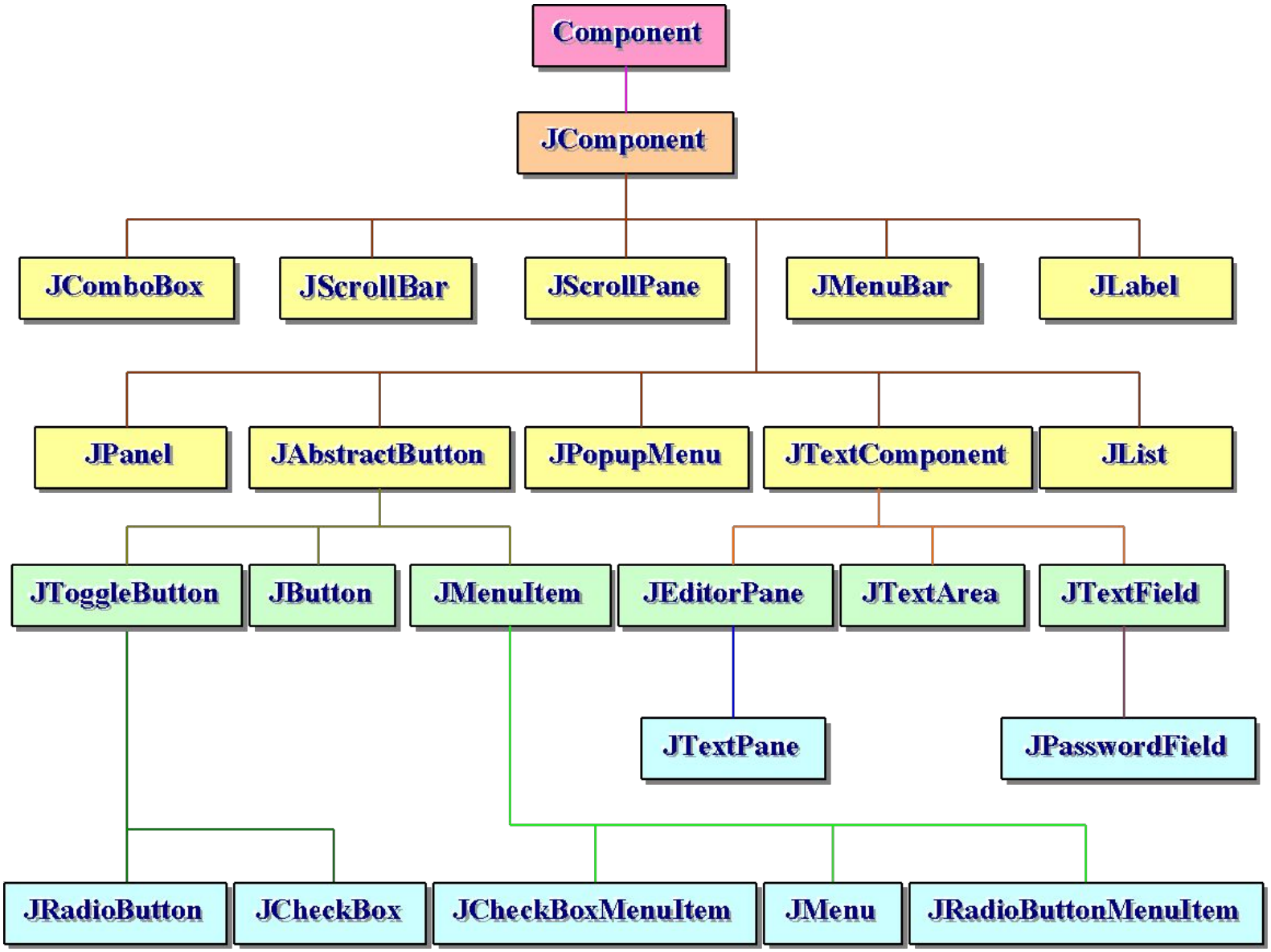
- облегченные компоненты графического интерфейса пользователя (**Swing**);
- модель делегирования событий;
- дополнительные функции печати;
- средства обмена данными с буфером обмена Clipboard;
- улучшенные средства интеграции с системой цветопередачи;
- поддержка операций, выполняемых без участия мыши;
- поддержка операции "перетащить и опустить" (drag-and-drop);
- поддержка специальных функций, предназначенных для людей с ограниченными возможностями;
- улучшенные операции двухмерной графики.

Каждый компонент AWT в среде Java должен быть связан с соответствующим ему объектом (peer-объектом) графического интерфейса пользователя в той операционной системы, в которой выполняется программа, написанная на языке Java - программа, выполняющаяся в среде Windows, будет иметь графический интерфейс Windows и та же программа, выполняющаяся в среде Unix, будет иметь графический интерфейс этой версии Unix.

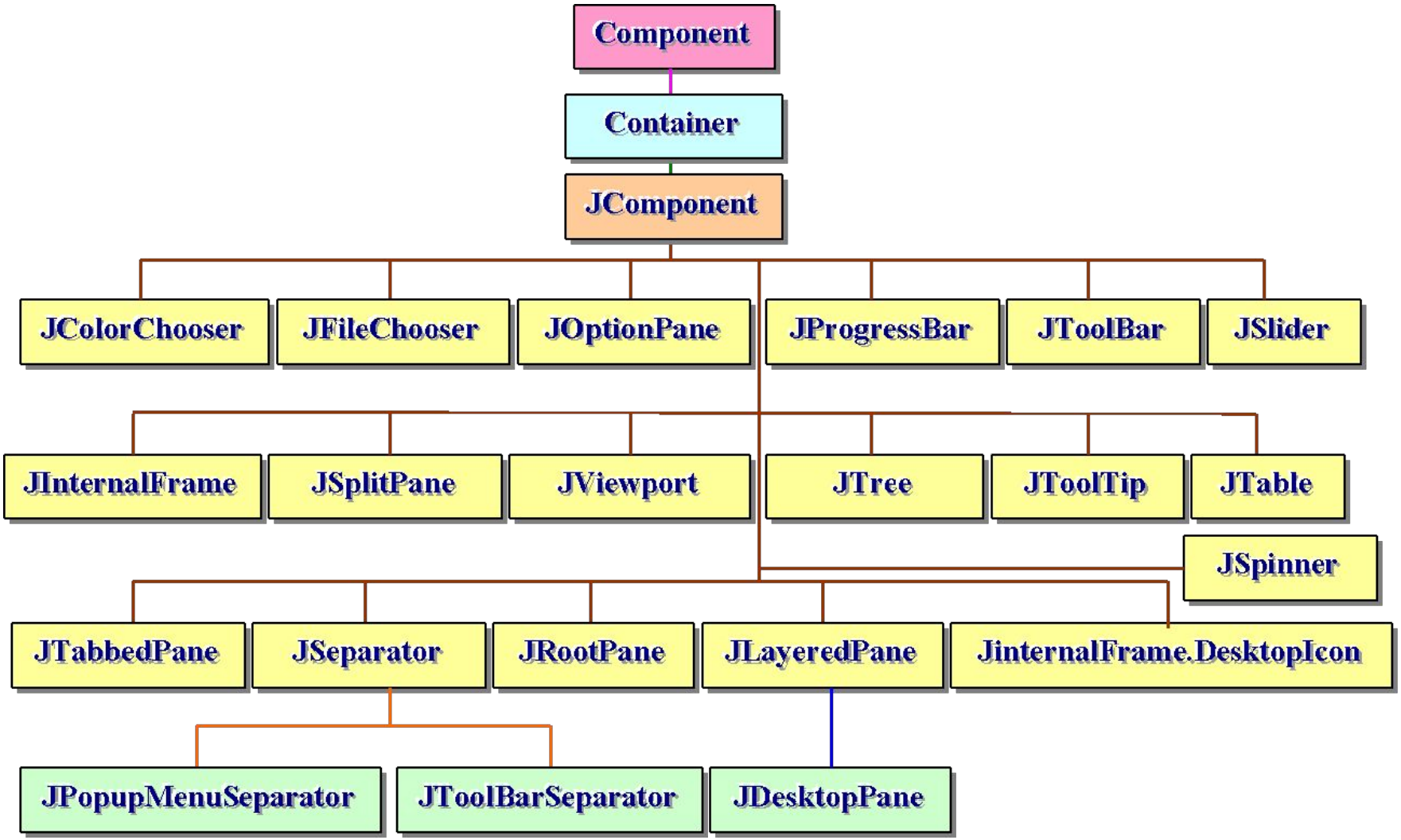
Графические объекты, создаваемые с использованием Swing, не нуждаются в создании вспомогательных объектов. Вместо создания парного компонента в среде операционной системы компьютера они обращаются к библиотеке классов модуля-приложения, соответствующего затребованному стилю представления компонента. Поэтому **пользовательский интерфейс Swing не зависит от операционной системы, под управлением которой работает Java, и его можно менять по желанию пользователя.**

Компоненты Swing входят в Java в составе группы пакетов с префиксом **javax.swing**, но многие из них являются потомками компонентов пакета **java.awt**.

Класи графічного інтерфейсу Swing, що виконують ті ж функції, що і компоненти AWT



Класи графічного інтерфейсу Swing, відсутні в AWT



Компоненты Swing можно разделить на следующие типы:

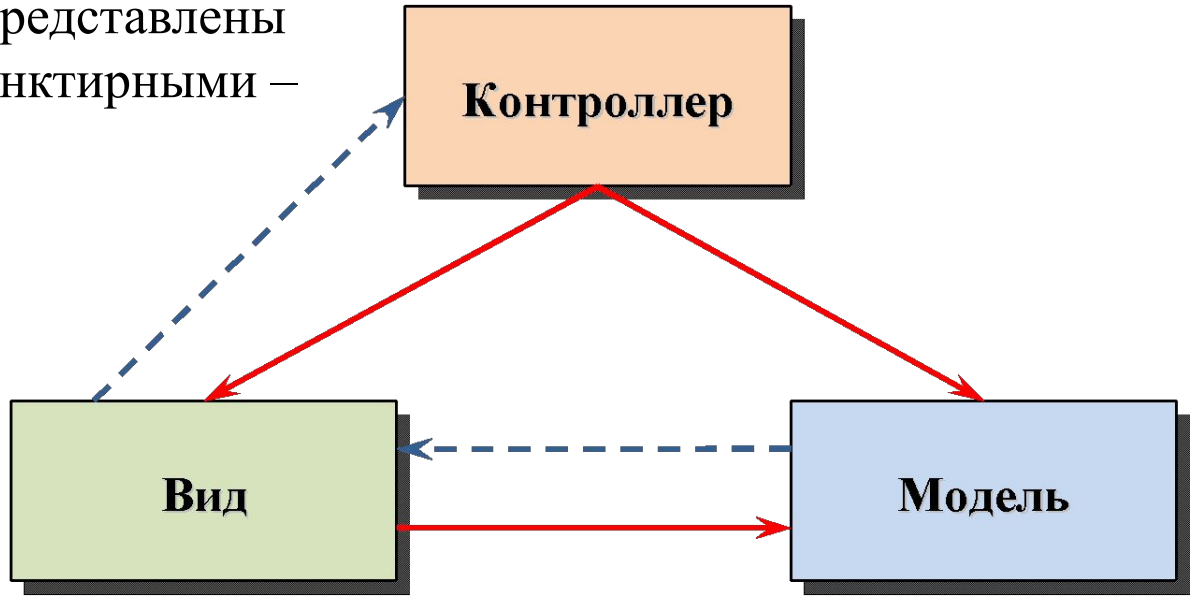
- контейнеры верхнего уровня (**JWindow**, **JFrame**, **JDialog, JApplet**);
- специализированные контейнеры (**JInternalFrame**, **JLayeredPane**, **JRootPane, JOptionPane**);
- общецелевые контейнеры (**JPanel**, **JScrollPane**, **JSplitPane**, **JTabbedPane, JToolBar**);
- компоненты управления (**JButton**, **JCheckBox**, **JRadioButton**, **JToggleButton**, **JComboBox**, **JList**, **JMenuBar**, **JMenu**, **JMenuItem**, **JCheckboxMenuItem**, **JRadioButtonMenuItem**, **JSeparator**, **JSlider**);
- нередактируемые информационные компоненты (**JLabel**, **JProgressBar, JToolTip**);
- редактируемые информационные компоненты (**JColorChooser**, **JFileChooser**, **JTable**, **JTree**, **JTextField**, **JPasswordField**, **JTextArea**, **JEditorPane**, **JTextPane**).

В отличие от компонентов AWT, **компоненты системы Swing способны работать только по модели делегирования событий**.

Схема MVC в компонентах Swing

Схема MVC (Model-View-Controller – Модель-Вид-Контроллер) является одним из **шаблонов проектирования**, разработанным для графических программ. В этой схемы компоненты (модули) разбиваются на три по возможности независимых компонента: **модель данных**, **интерфейс пользователя** и **управляющую компоненту**.

сплошными стрелками представлены связи по управлению, пунктирными – связи по данным



Компонент «Модель» предоставляет данные для **компонента «Вид»** и управляет **компонентом «Контроллер»,** изменяя свое состояние в ответ на его запросы.

Компонент «Вид» предоставляет графический интерфейс для пользователя, получая данные от **компонента «Модель».**

Компонент «Контроллер» обрабатывает данные, введенные пользователем с помощью **компонента «Вид»,** и управляет функционированием **компонентов «Модель» и «Вид».**

В Java **компонент «Модель»** представляется классами, определяющими представление данных, а также сведения о текущем состоянии этих данных. Кроме того, в этих классах заданы методы ввода и изменения данных.
В **компоненте «Вид»** задаются классы, определяющие вид представления данных на определенном устройстве (например, дисплее или печати). Для одной модели может быть задано несколько видов представления для разных устройств. В **компоненте «Вид»** задаются методы для получения данных от **компонента «Модель».**

В **компоненте «Контроллер»** задаются классы, организующие ввод данных и изменения состояния объекта. Эти классы реагируют на различные события, генерируемые пользователем или системой и обращаются к методам **компонент «Вид» и «Модель»** для получения или изменения данных. Один и тот же **компонент «Модель»** может использоваться разными **компонентами «Контроллер».**

В библиотеке **Swing компоненты «Модель»** описаны с помощью интерфейсов, в которых задаются методы, необходимые для функционирования модели. Для того, чтобы облегчить программирование, для каждого интерфейса существует хотя бы один класс, реализующий данный интерфейс.

В большинстве случаев при программировании графического приложения используются уже имеющиеся в библиотеке **Swing** классы моделей. Для задания собственной модели необходимо создать собственный класс, либо реализующий соответствующий интерфейс для модели, либо являющийся наследником одного из существующих классов моделей.

Контейнери верхнього рівня і спеціалізовані контейнери

Классы **JWindow**, **JFrame**, **JDialog** и **JApplet** являются подклассами соответственно классов **Window**, **Frame**, **Dialog** и **Applet**, а не **JComponent**. По этой причине **интерфейс пользователя, реализуемый объектами данных классов, зависит от операционной системы, под управлением которой выполняется программа на языке Java**.

Для создания окон графических приложений используется класс **JFrame**. Приложения с графическим интерфейсом использует, по крайней мере, один фрейм. Апплеты также могут использовать фреймы.

Для создания окон, которые зависят от другого окна применяются диалоговые окна класса **JDialog**.

Апплеты, использующие компоненты **Swing**, должны быть подклассами класса **JApplet**. В апплетах **Swing** можно использовать все графические компоненты **Swing**, включая меню.

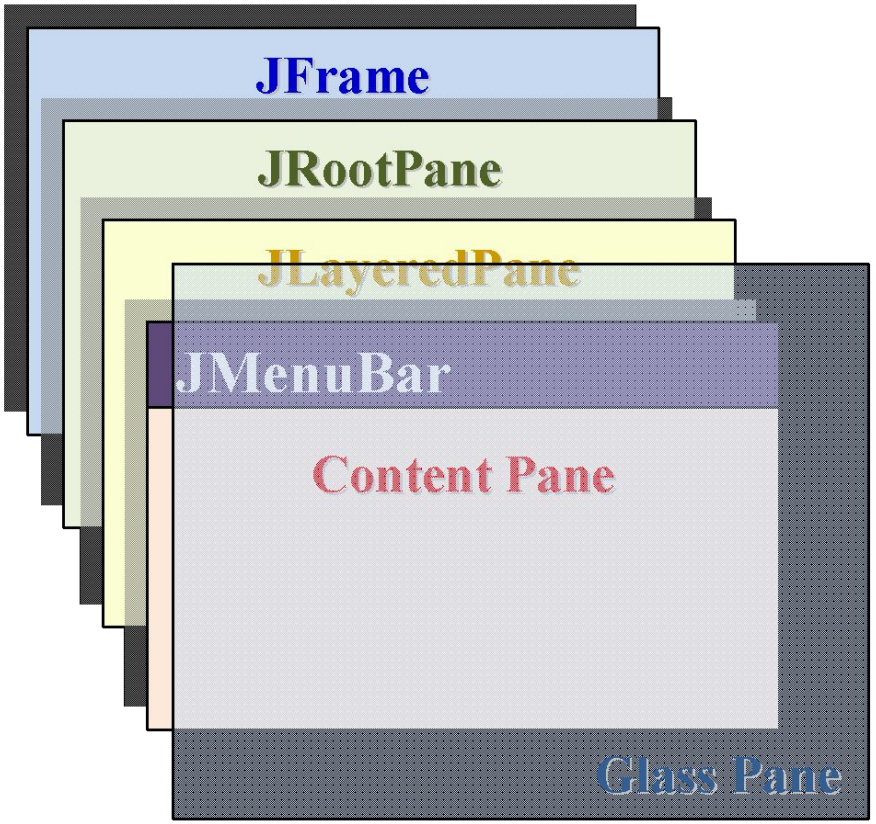
Любая программа, которая использует компоненты **Swing**, содержит, по крайней мере, один контейнер верхнего уровня. Этот контейнер является корнем иерархии контейнеров, содержащих все компоненты **Swing**.

Коренева панель

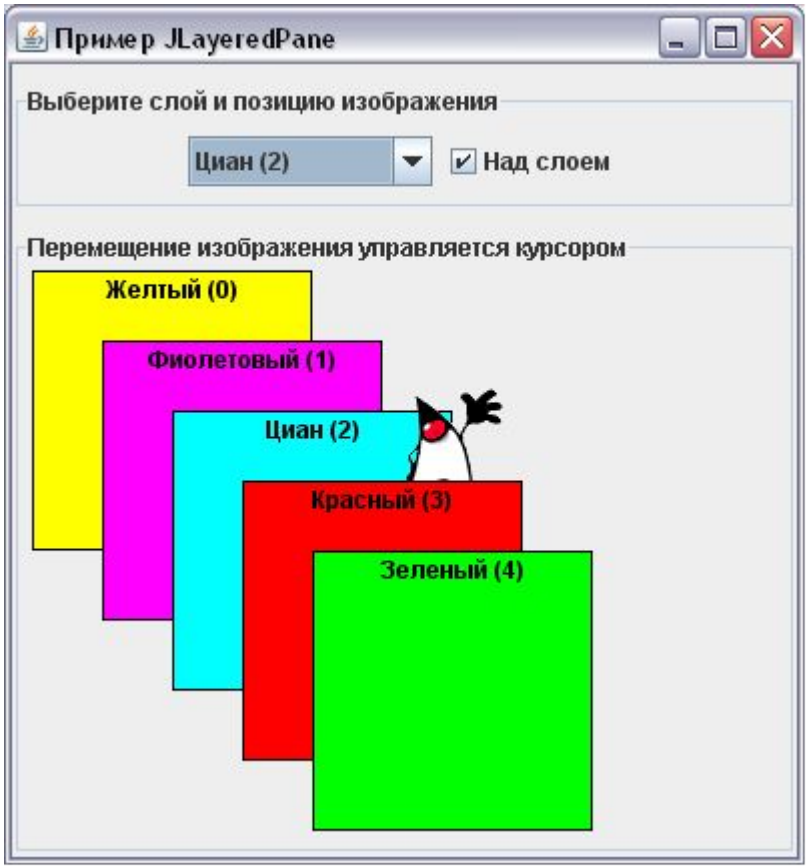
Каждый контейнер верхнего уровня базируется на промежуточном, скрытом, контейнере, называемом **корневой панелью** (root pane), которая определена в классе **JRootPane**. Сама корневая панель обычно не используется, а используются ее компоненты, которые корневая панель предоставляет контейнеру верхнего уровня.

Корневая панель содержит следующие компоненты:

- слоистая панель (layered pane);
- панель содержимого (content pane);
- строка меню (menu bar) – необязательный компонент;
- стеклянная панель (glass bar).



Слоистая панель служит для позиционирования панели содержимого и строки меню. Кроме того, она обеспечивает позиционирование панелей по оси *z* - компоненты, формируемые с помощью слоистой панели, могут полностью или частично перекрывать друг друга.

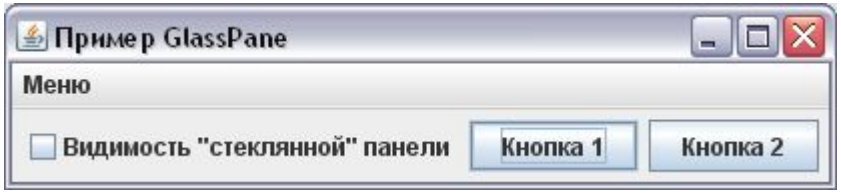


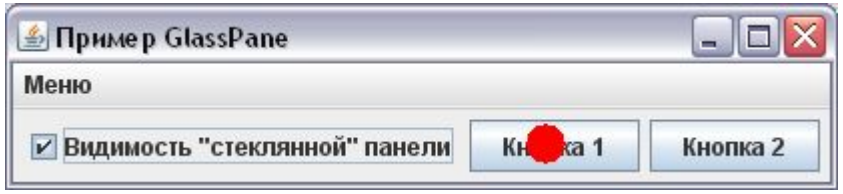
В пакете Swing определены два класса слоистых панелей: **JLayeredPane** и **JDesktopPane**.

В слоистой панели, задаваемой с помощью класса **JLayeredPane**, слой, в котором располагается та или иная панель, задается с помощью индекса.

Панель содержимого содержит все компоненты **Swing** (кнопки, надписи, текстовые поля и т.д.).
Для всех контейнеров верхнего уровня (включая **JApplet**) менеджером компоновки по умолчанию является **BorderLayout**.
К контейнеру верхнего уровня может быть добавлена **строка меню** (menu bar). Строка меню также располагается внутри главного контейнера, но за пределами панели содержимого.

Стеклянная панель обычно используется, когда необходимо перехватывать события над областью, которая содержит один компонент или несколько компонентов. Например, для области, состоящей из нескольких компонент, можно отключить события мыши, организовав перехват событий стеклянной панелью, или вывести изображения поверх компонент.





Класс JFrame

Для фрейма можно использовать методы получения и установки корневой панели, слоистой панели, панели содержимого, строки меню и стеклянной панели, причем должно быть получено, по крайней мере, значение панели содержимого.
Отличием **JFrame** от **Frame** является наличие у **JFrame** свойства, определяющего операцию закрытия окна по умолчанию. Для **Frame** по умолчанию ничего не происходит, если делается попытка закрыть окно, а **JFrame** закрывается.

Диалоговые окна

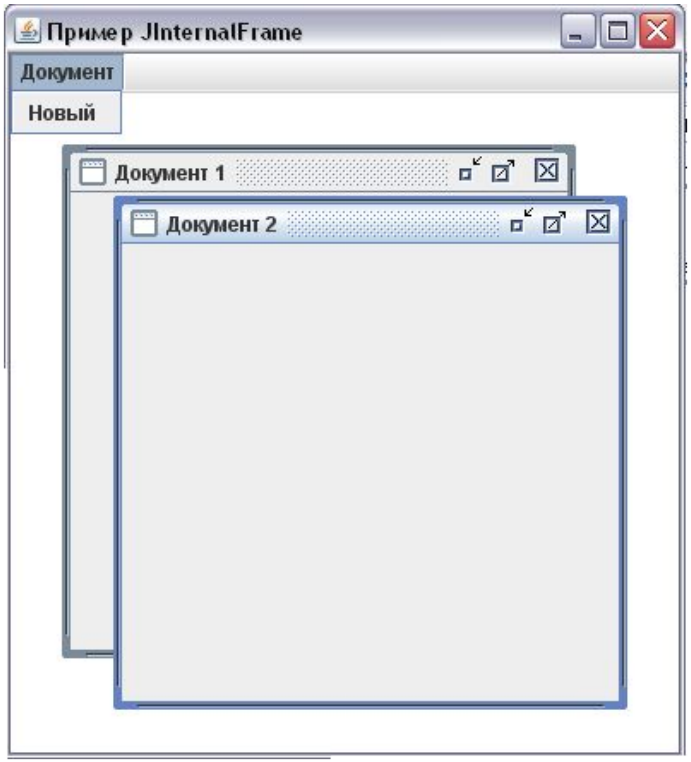
Диалоговые окна можно создавать либо с помощью класса **JDialog**, либо используя класс **JOptionPane**.

Окно класса **JDialog** имеет те же черты, что **JFrame** и использование объектов класса **JDialog** аналогично прямому использованию объекта **JFrame**.
Класс **JOptionPane** облегчает вывод стандартных диалоговых или информационных окон.

Классы JInternalFrame и JDesktopPane

Класс **JInternalFrame** позволяет создать внутренний фрейм, который имеет те же свойства, что и обычный фрейм.

Обычно создаваемые объекты класса **JInternalFrame** помещаются в виртуальный рабочий стол, реализованный в **Swing** с помощью класса **JDesktopPane**. Этот класс расширяет класс **JLayeredPane** для того, чтобы управлять перекрывающимися внутренними фреймами.

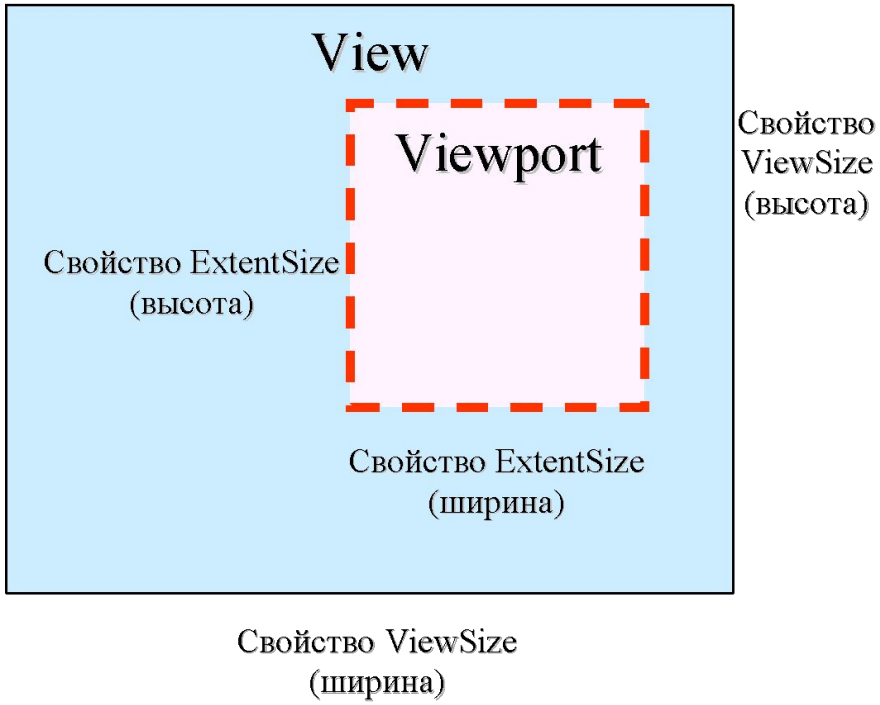


Контейнери загального призначення

Классы JScrollPane и Viewport

Как и компонент **ScrollPane** в AWT, компонент **JScrollPane** обеспечивает горизонтальную и вертикальную прокрутку содержимого. Он выводит содержимое с использованием одного из менеджеров компоновки Swing – **ScrollPaneLayout**.

Кроме того, важную роль при выводе объектов с помощью **JScrollPane** и других классов просмотра играет класс порта просмотра – **JViewport**.

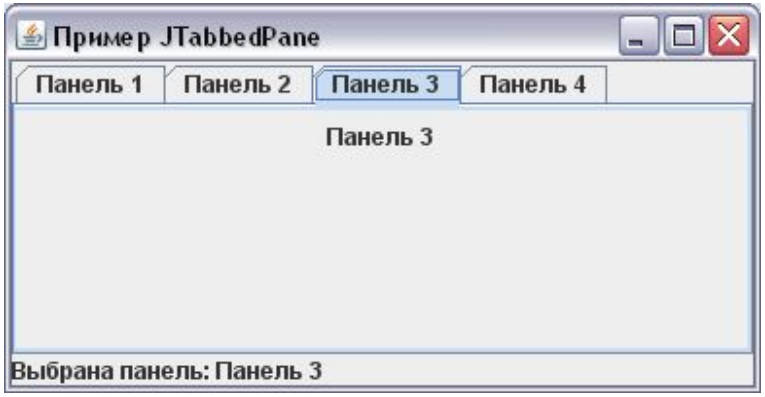


Класс JPanel

Класс **JPanel** является вариантом класса **Panel** в AWT. Работа с объектами класса **JPanel** не отличается от работы с объектами класса **Panel**, за исключением того, что панель добавляются не во фрейм, а в панель содержимого фрейма.

Класс JTabbedPane

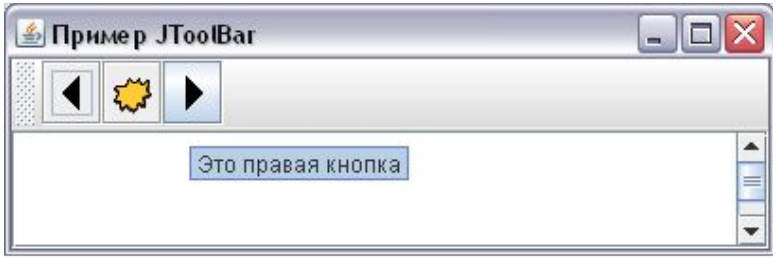
Панель с вкладками (tabbed pane) введена в Swing вместо менеджера компоновки **CardLayout**, который в Swing не реализован.



Каждая вкладка имеет заголовок. Когда пользователь выбирает вкладку, ее содержимое становится видимым. Только одна из вкладок может быть выбрана одновременно. Панель с вкладками обычно используется для установки параметров конфигурации или функционирования апплета или графического приложения.
Панель с вкладками реализована классом **JTabbedPane**

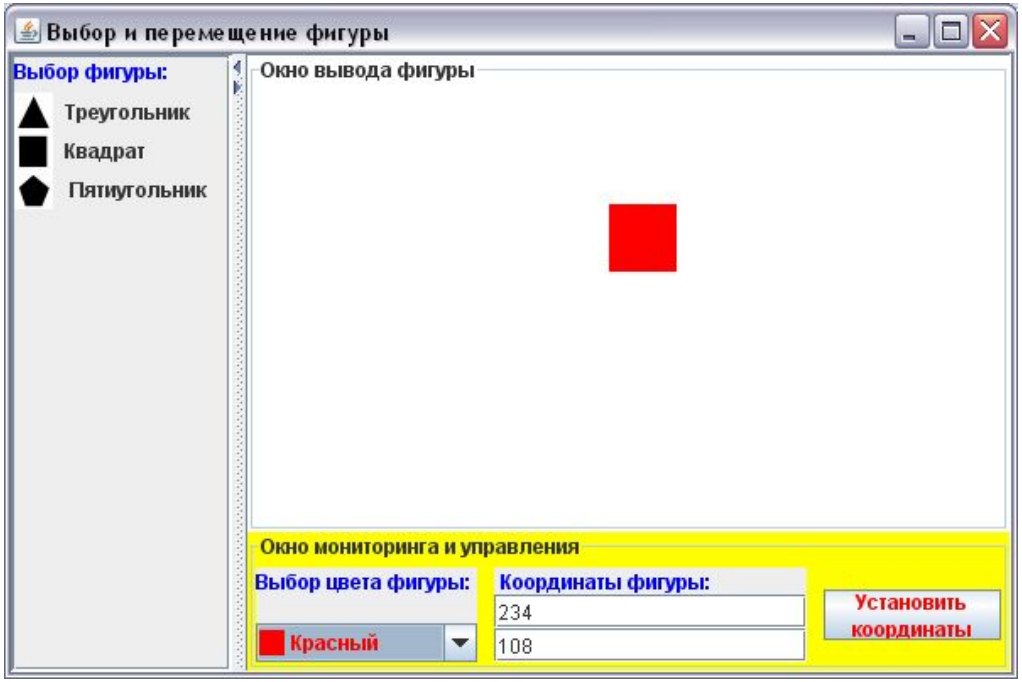
Класс JToolBar

Панель инструментов (класс) создается в **Swing** методами класса **JToolBar.** В панель инструментов обычно помещаются кнопки, текстовые строки и другие элементы управления.



Класс JSplitPane

Расщепленные панели (класс **JSplitPane**) используются для разделения выводимой панели на две компоненты.



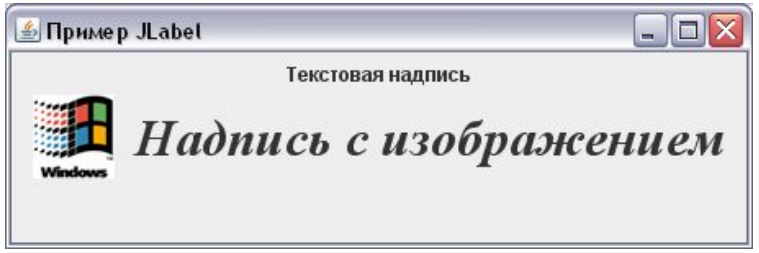
Значки, написи і кнопки

Интерфейс Icon и класс ImageIcon

Значки (icons) реализуются в **Swing** с помощью интерфейса **Icon** (на самом деле они не являются компонентами и их можно использовать практически со всеми компонентами **Swing**).
Реализацией интерфейса **Icon**, создающей значок из изображения, является класс **ImageIcon**.
В отличие от класса **Image**, в классе **ImageIcon** изображение загружается предварительно с использованием методов класса **MediaTracker**.
Для создания собственных изображений можно непосредственно реализовывать интерфейс **Icon**.

Класс JLabel

Надписи **Swing** являются объектами класса **JLabel**. Этот объект может отображать тексты и/или значки.



Класс AbstractButton

Кнопки Swing являются подклассами класса **AbstractButton**. Этот класс содержит много методов, которые позволяют управлять поведением кнопок, флажков и переключателей. Например, можно определять различные значки для отображения компонента, когда он отжат (disabled), нажат (pressed), или выбран (selected). Можно выбрать также значок "наезда" (rollover).

Класс JButton

Класс **JButton** обеспечивает функциональные возможности кнопки. Этот класс позволяет связать с кнопкой изображение, текст или и то и другое.

Прапорці та перемикачі

Класс **JCheckBox**, который обеспечивает функциональные возможности флажка, является конкретной реализацией класса **AbstractButton**.

Переключатели поддерживаются классом **JRadioButton**, который также является конкретной реализацией класса **AbstractButton**.

Переключатели должны быть объединены в группу, где в каждый момент может быть выбран только один элемент. Например, если пользователь щелкает по переключателю, который находится в группе, любой предварительно нажатый переключатель в этой группе автоматически сбрасывается. Чтобы создать группу кнопок, создается объект класса **ButtonGroup**.

Родительским классом и для **JCheckBox** и для **JRadioButton** является класс **JToggleButton**, который не имеет эквивалента в AWT. Он ведет себя как объект класса **Button**, который после нажатия на него остается в нажатом состоянии.