Лекция №11. Java. Разработка графического интерфейса

Содержание

1. AWT — Abstract Window Toolkit
2. Swing

Реализации графического пользовательского интерфейса

- В Java существует две реализации графического пользовательского интерфейса:
  - java.awt
  - javax.swing

1. AWT — Abstract Window Toolkit

- **java.awt** - набор классов-оберток компонентов GUI операционной системы, на которой выполняется Java-приложение.
- Компоненты AWT реализованы платформозависимым способом на каждой реализации java машины. При этом для пользователя существует иерархия оберток, обладающих разной реализацией графического представления на разных архитектурах.

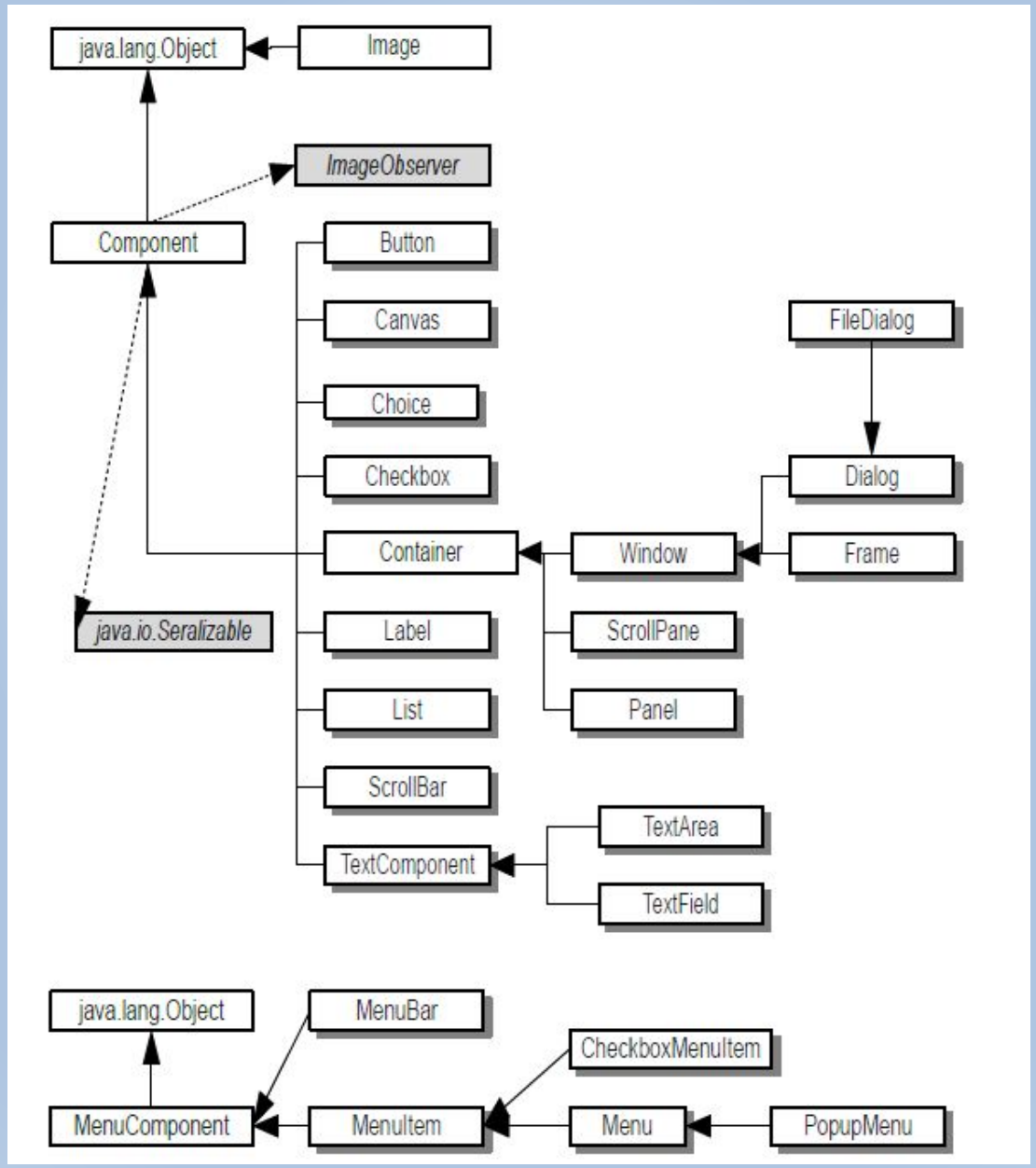
Компоненты AWT

- ***Компонент*** (java.awt.Component) — базовый класс, определяющий отображение на экране и поведение каждого элемента интерфейса при взаимодействии с пользователем. Задает размер, цвет, отступы, область отображения и порождает основные события. Класс лежит в основе иерархии.
- Для компонентов можно установить множество разных атрибутов, таких как цвет, шрифт и множество других.
- Например, для установки цвета фона и переднего плана используются следующие методы:

```
setBackground(Color); // установка цвета фона компонент
setForeground(Color); // установка цвета на переднем плане
                      // (например, надпись на кнопке)
```

- Для определения цвета используется класс java.awt.Color, в котором цвет можно задать либо при помощи таблицы RGB, либо при помощи переменных класса.
- Для установки шрифта используется метод setFont(Font). Все надписи, принадлежащие данному компоненту будут написаны этим шрифтом. Для определения шрифта используется класс java.awt.Font, в котором шрифт определяется через 3 параметра:
  - имя (Helvetica),
  - стиль (BOLD)
  - высота (12).

Структура пакета `java.awt`



- Возможность включать изображения в программу реализуется через потомков абстрактного класса Image.

- **Кнопка (Button)**

Простейший компонент пользовательского интерфейса, применяемый практически во всех приложениях.

Для создания нового компонента необходимо создать объект этого класса, в приведенном примере в качестве аргумента передается строка-заголовок. При создании новой кнопки можно использовать конструктор без аргументов.

```
Button b = new  Button ("New Button");
```

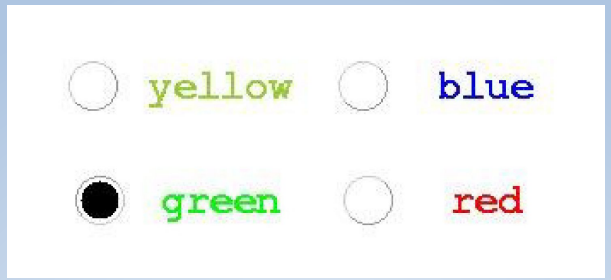
- Переключатели (Checkbox)

Данный компонент позволяет визуально устанавливать значения on или off для переменной. При использовании нескольких переключателей, несколько переменных могут принимать значение on.

При создании нового объекта необходимо задать как строку название, так и его значение. Необходимо помнить о том, что при использовании в качестве параметра объекта типа CheckboxGroup по крайней мере хотя бы один CheckBox должен быть в состоянии on.

```
Checkbox ch = new Checkbox ("One", true);
```

- **Группы переключателей (CheckboxGroup)**



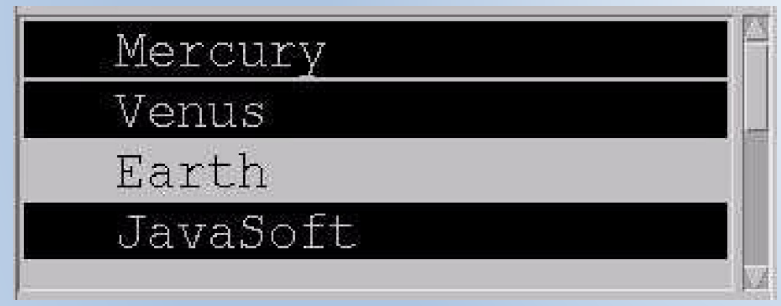

```
CheckboxGroup cbg = new CheckboxGroup();
    add(new Checkbox("yellow", cbg, false));
    add(new Checkbox("green", cbg, true));
    add(new Checkbox("blue", cbg, false));
    add(new Checkbox("red", cbg, false));
```

Для использования этого компонента необходимо наличие n-числа переключателей.

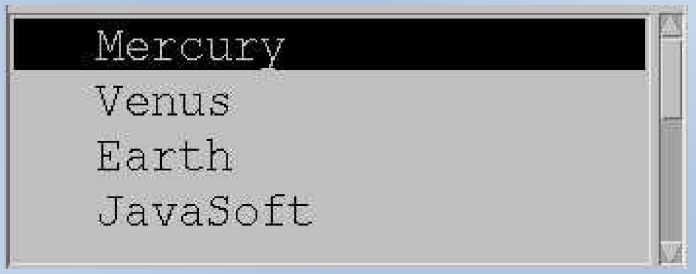
Алгоритм создания группы переключателей следующий:

- инициализация нового объекта типа CheckboxGroup
- добавление к нему переключателей при помощи метода add().

- **Списки (List)**
- Списком называется набор элементов, один или несколько из которых могут быть выбраны из создаваемого окна с прокруткой. Возможно использование пустого конструктора.
- Для создания списка необходимо:
  - инициализация объекта типа List (в конструкторе задаем число видимых строк и атрибут, определяющий возможен ли выбор нескольких строк одновременно);
  - добавление строк методом add().



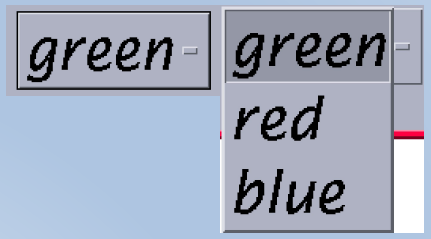

```
List lst = new List (4, false);
           lst.add("Mercury");
           lst.add("Venus");
           lst.add("Earth");
           lst.add("JavaSoft");
           lst.add("Mars");
```




```
           lst.add("Jupiter");
           lst.add("Saturn");
           lst.add("Uranus");
           lst.add("Neptune");
           lst.add("Pluto");
           lst.setMultipleMode (true);
```

Метод `setMultipleMode(boolean)` определяет возможность выбора множества строк

- **Выпадающие списки (Choice)**



Создается аналогично обычному списку, только при его использовании можно всегда выбирать только одну строку.

В данном классе используется только один пустой конструктор, который создает объект этого класса, для полноценной работы которого необходимо добавление n-числа строк, что реализуется при помощи метода `add()`.

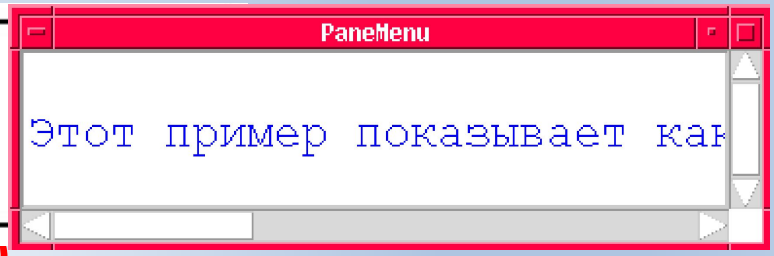
- **Надписи (Label)** This is Label
- Позволяет размещать статические текстовые надписи. При создании объекта можно использовать пустой конструктор, но тогда саму надпись необходимо задавать при помощи метода

```
Label l = new Label ("This is new Label", Label.CENTER);
    l.setText  ("This is Label");
    l.setAlignment  (Label.LEFT)
```

- **Контейнеры**
- *Контейнер (java.awt.Container*) - компонент, способный содержать в себе другие компоненты, и управлять их размещением и, возможно, размерами при помощи *менеджеров компоновки*.

- **Панель (Panel)**
- Работа с данным компонентом не отличается от работы с другими компонентами класса java.awt, за единственным исключением, что для панели можно установить менеджер компоновки при помощи метода setLayout(). Добавление компонентов осуществляется методом add().
- В отличии от других компонентов панель не прорисовывается, то есть не имеет внешнего вида.

```
Panel p1 = new Panel ();
    p1.setLayout (new FlowLayout ());
    p1.add (new Button ("Button1"));
    p1.add (new TextField (10));
    p1.add (new Label ("This is FlowLayout"));
```



- **Скроллирующие панели (ScrollPane)**

- Контейнер ScrollPane отображает ограниченный участок дочернего элемента и снабжен горизонтальными и вертикальными полосами прокрутки для его просмотра в окне просмотра.
- Необходимо создать его и добавить дочерний элемент. Отличие от обычного контейнера:
  - поддерживает лишь один дочерний элемент (как правило, используется панель со множеством компонент);
  - не предусмотрено использование менеджера компоновки
  - необходимо задавать размеры с помощью метода setSize()

- **Окно приложения (Frame)**
- Frame — контейнер, формирующий окна приложения с заголовком окна. Недоступен в апплетах.
- Фрейм позволяет добавлять к себе меню и обрабатывать события интерфейса пользователя, связанные с активированием, сворачиванием и закрытием окна.

Менеджеры компоновки

- Менеджер компоновки является незаменимым инструментом при использовании более одного компонента. Пакет java.awt включает следующие менеджеры:
  - FlowLayout - менеджер, используемый по умолчанию, размещает компоненты последовательно в строку. Его использование оправдано в том случае, когда известны точные размеры компонент.
  - BorderLayout — создает так называемое полярное расположение: разбивает панель на 5 зон (South, North, Center, West, East). Он учитывает разницу в размерах отдельных компонентов и пытается максимально использовать пространство контейнеров.
  - GridLayout - создает решетку, состоящую из прямоугольников одинакового размера, в каждом из которых располагается один компонент.
  - CardLayout - предназначен для последовательной визуализации различных панелей на одной основный.
  - GridBagLayout - данный менеджер является наиболее сложным. Он позволят реализовывать сложный интерфейс, в котором контейнер содержит много компонентов различных размеров, которые должны находиться в одном и том же заданном положении относительно других.

- **Меню (Menu)**
- Меню - компонент пользовательского интерфейса, позволяющий создавать в приложениях главное меню. Меню в java.awt неразрывно связано с содержащим ее фреймом.
- Процесс подсоединения меню к приложению состоит из нескольких частей:
  - Создание строки меню


```
MenuBar mb = new MenuBar();
```

  - Создание нового меню на строку мен


```
Menu m = new Menu("File");
```

  - Добавление опции в меню и присваивание ей име


```
m.add(new MenuItem ("Load")
```

  - Добавление меню к фрейму

```
Frame f = new Frame("окно");
f.setMenuBar(mb);
```

Модель обработки событий

- Компоненты AWT генерируют события в соответствии с воздействиями пользователя на графический интерфейс. Другие компоненты регистрируются для прослушивания этих событий и реагируют соответствующим образом.
- Существует 4 составные части модели обработки событий.
  1. Компонент-источник события
  2. Компоненты-слушатели или приемники события
  3. Интерфейс слушателя
  4. Класс события.

- Источник события
- *Источником события* является компонент, генерирующий событие и регистрирующий заинтересованные в прослушивании данного события компоненты.
- Источник события оповещает слушателя путем вызова специального метода интерфейса слушателя (actionPerformed) и передачи ему объекта события (ActionEvent). Все слушатели событий определенного события должны реализовывать соответствующий интерфейс.

- **Слушатель события**
- *Слушатель события* — компонент, регистрирующийся для прослушивания события у источника и реагирующий на него. Слушатель может события обрабатывать внутри собственного класса.
- **Интерфейс слушателя**
- *Интерфейс слушателя* — вспомогательный интерфейс, который обязаны реализовывать все слушатели события для возможности добавления себя в список у источника события.
- Интерфейс определяет методы, которые будут вызваны в момент доставки события слушателю. Все интерфейсы слушателей являются расширением класса java.util.EventListener. По установленному соглашению, методам слушателей может быть передан один единственный аргумент, который соответствует данному слушателю.

- **Событие**
- *Событие пользовательского интерфейса* — потомок класса java.awt.AWTEvent, предназначенный для передачи информации от источника события к слушателю.
- События пакета AWT вместе со слушателями находятся в пакете java.awt.event.
- В событии должна содержаться вся информация, необходимая программе для формирования реакции на данное событие.

Класс AWTEventMulticaster

- AWTEventMulticaster — класс, реализующий эффективную и потоко-безопасную диспетчеризацию событий для событий AWT.

Основные события AWT

Ссобытие, Интерфейс слушателя	Элемент	Методы слушателя	Значение
ActionEvent, ActionListener	Button	actionPerformed	нажатие кнопки
	List		двойной щелчок
	MenuItem		выбор пункта меню
	TextField		конец редактирования текста элемента, ENTER
AdjustmentEvent AdjustmentListener	Scrollbar	adjustmentValueChanged	реализация прокрутки
ComponentEvent ComponentListen	Component	componentHidden componentMoved	перемещение, изменение размеров, стал скрытым или

er		componentResized componentShown	видимым
ContainerEvent ContainerListener	Container	componentAdded componentRemoved	добавление или удаление элемента в контейнер
FocusEvent FocusListener	Component	focusGained focusLost	получение или потеря фокуса
ItemEvent ItemListener	Checkbox	itemStateChanged	установка (сброс) флажка
	CheckboxMenuItem		установка (сброс) флажка у пункта меню
	Choice		выбор элемента списка
	List		выбор элемента списка
KeyEvent, KeyListener	Component	keyPressed keyReleased keyTyped	нажатие или отпускание клавиши
MouseEvent, MouseListener MouseMotionListener	Component	mouseClocked mouseEntered mouseExited mousePressed mouseReleased mouseDragged mouseMoved	нажатие или отпускание кнопки мыши, курсор мыши вошел или покинул область элемента, перемещение мыши, перемещение мыши при нажатой кнопки мыши.
TextEvent, TextListener	TextComponent	textValueChahged	изменения в тексте элемента
WindowEvent, WindowListener	Window	windowActivated windowClosed windowClosing windowDeactivated windowDeiconified windowOpened	открытие, закрытие, минимизация, восстановление, запрос на обновление окна

JFC - Java Foundation Classes

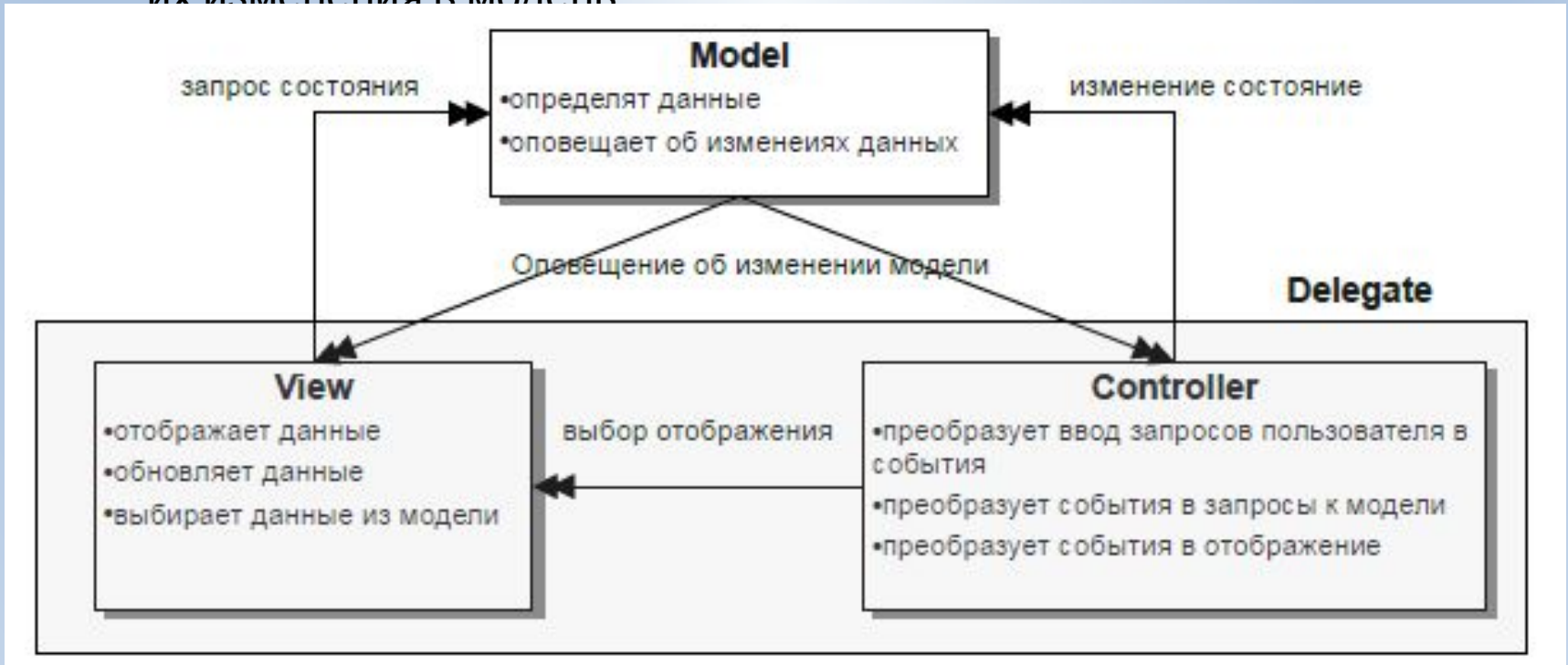
- JFC — набор базовых библиотек, предназначенный для построения эффективных графических приложений.
- Состоит из:
  - AWT - Содержит компоненты AWT 1.1
  - Java 2D - Обеспечивает расширенные возможности создания и обработки изображений.
  - Accessibility - Реализует вспомогательные технологии, такие как экранное считывание, обработка речи, и т.д.
  - Drag and Drop - Дает возможность взаимодействия приложений на языке Java с другими приложениями с использованием механизма Drag and Drop.
  - Swing - расширяет набор компонентов AWT 1.1 путем добавления легковесных компонентов, реализующих архитектуру MVC, в отличие от компонентов, основанных на использовании экспертов операционной системы.

2. Swing

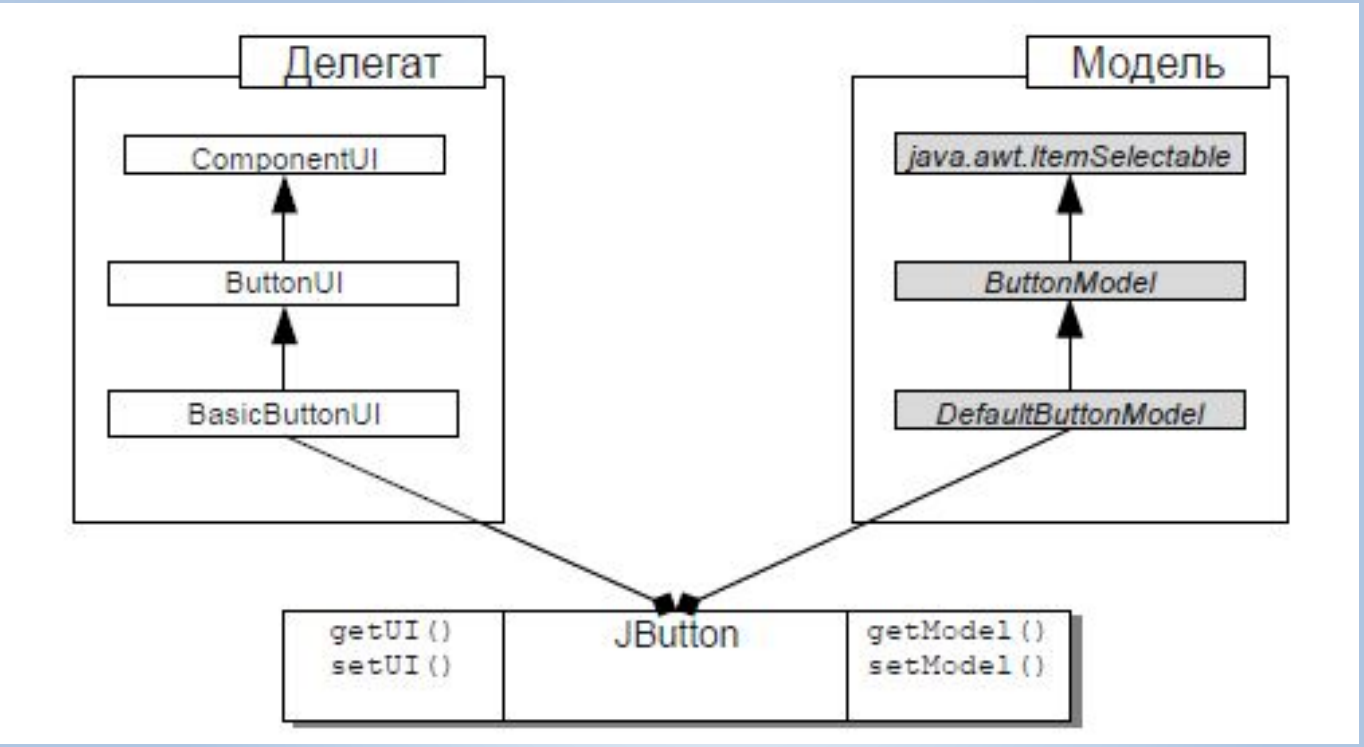
- Набор компонентов Swing - это разработанный на Java платформо-независимый набор компонентов графического пользовательского интерфейса.
- Особенности Swing-компонентов:
  - являются "легковесными", т.е. не используют средств операционной системы.
  - реализуют архитектуру PL&F (Pluggable Look and Feel), позволяя легко изменять вид и поведение работающего приложения.
  - реализуют компонентную технологию JavaBeans, и могут использоваться с любыми визуальными средствами разработки, поддерживающими работу с Beans-компонентами.

Архитектура MVC и модель Swing

- MVC (Model-View-Controller) - технология создания элементов пользовательского интерфейса, основанная на взаимодействии компонентов:
  - Model (модель) - логическое представление данных
  - View (представление) - визуальное представление данных
  - Controller (контроллер) - обрабатывает входные данные и передает их изменения в модель



- В Swing контроллер и представление объединяются в общий компонент под названием *делегат (delegate).*



- Компоненты Swing в любой момент времени связаны с определенной моделью и с определенным делегатом, специфическими для данного компонента. Модели и делегаты реализуют соответствующие интерфейсы.

Пакеты Swing

Пакет	Назначение
javax.swing	Содержит платформонезависимую реализацию компонентов, адаптеров, моделей компонент и интерфейсов для делегатов и моделей
javax.swing.border	Объявляет интерфейс Border и реализующие его классы, для отображения бордюров – визуального обрамления компонентов Swing
javax.swing.colorchooser	Реализует компонент JColorChooser
javax.swing.table	Содержит интерфейсы и классы для поддержки работы с таблицами
javax.swing.text	Содержит классы поддержки работы с текстовыми компонентами

Составные части окна

- В модели Swing внутренняя часть окна представляет собой корневую панель **JRootPane**, которая состоит из:
  - прозрачной панели (*glass pane*)
  - слоистой панели (*layered pane*).
- Прозрачная панель невидима, и используется для отображения подсказок и выплывающих меню.
- Слоистая панель состоит из двух компонентов - линейки меню и панели содержимого (*content pane)*.
- Компоновка осуществляется на панели содержимого:

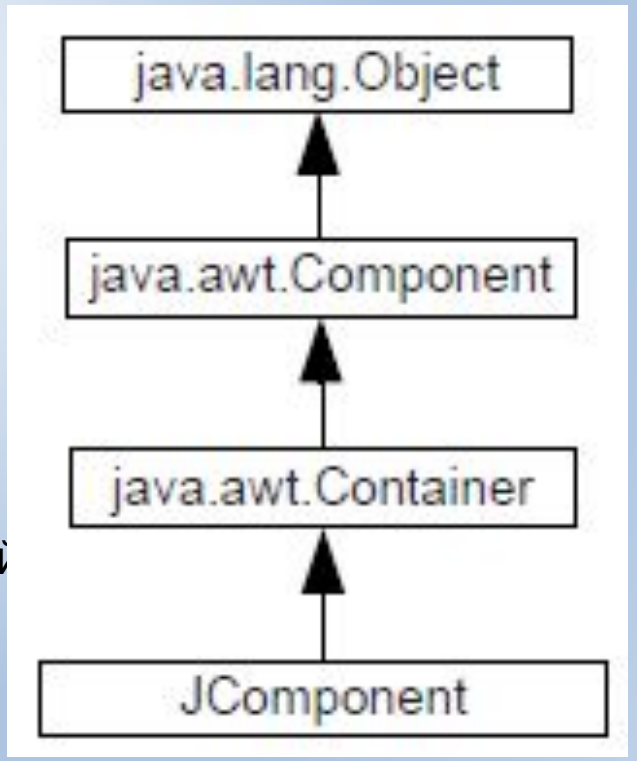
```
aFrame.getContentPane().setLayout (new FlowLayout());
aFrame.getContentPane().add(aComponent);
```

- При помещении компонента на слоистую панель можно указывать номер слоя, на который следует поместить компонент:

```
layeredPane.add (component, new Integer(5));
```

Класс JComponent

- Является базовым классом почти для всех Swing-компонентов пользовательского интерфейса (J-классов). Swing-компоненты пользовательского интерфейса наследуют от класса JComponent следующие свойства:
  - реализация PL&F
  - расширяемость
  - обработка событий клавиатуры
  - прорисовка границ
  - масштабируемость
  - отображение подсказок
  - автопрокрутка
  - возможность отладки графики
  - поддержка вспомогательных технологий
  - многоязыковая поддержка



Класс JPanel

- Является легковесным объектом, представляющим панель со встроенной поддержкой двойной буферизации.

Интерфейс Icon и класс ImageIcon

- Интерфейс Icon описывает изображения фиксированного размера. Класс ImageIcon реализует интерфейс Icon для объекта типа java.awt.Image.

```
public interface Icon {
  void paintIcon(Component c, Graphics g, int x, int y);
  int getIconWidth();
  int getIconHeight();
}
```

Класс JLabel

- Реализует однострочную текстовую метку с дополнительными возможностями:
  - наличие изображения;
  - возможность изменения взаимного расположения текста и изображения.

Класс JButton

- Является реализацией кнопки с возможностью включения изображения. Фон кнопки должен совпадать с фоном контейнера, в котором она находится.

Класс JCheckBox

- Соответствует объекту CheckBox, не входящему в группу. Можно задавать собственные изображения для выбранного и невыбранного состояния.

Класс JRadioButton

- Соответствует объекту CheckBox, входящему в группу, так, что в данный момент времени может быть выбран только один из них. Группа объектов задается с помощью ButtonGroup

Класс JTextComponent

- Класс обеспечивает основные функции простого текстового редактора. Основными подклассами JTextComponent являются JTextField, JTextArea, JTextPane.

Класс JComboBox

- Представляет собой выпадающий список с возможностью выбора и
- редактирования.

Класс JList

- Реализует список элементов.

Бордюры (Border)

- Рисование границ вокруг компонентов (бордюров) обеспечивается с помощью интерфейса Border. Border требует реализации следующих методо

```
// Определяет область, необходимую для рисования
public Insets getBorderInsets(Component c);
// Определяет прозрачность и непрозрачность границы
public boolean isBorderOpaque();
// Определяет каким образом рисовать границу
public void paintBorder (Component c, Graphics g,
                         int x, int y,
                         int width, int height)
```

- В пакет Swing входит 9 классов для рисования бордюров

BevelBorder	Объемный бордюр, поднятый или опущенный
CompoundBorder	Составной бордюр
DefaultBorder	Бордюр по умолчанию — реализация интерфейса
EmptyBorder	Пустой бордюр
EtchedBorder	Бордюр-канавка
LineBorder	Цветной бордюр произвольной толщины
MatteBorder	Бордюр с заполнением цветом или изображением
SoftBevelBorder	Объемный бордюр с закругленными углами
TitledBorder	Бордюр с заголовками в различных позициях

Классы меню

- Классы, обеспечивающие работу с меню (JCheckBoxMenuItem, JMenuItem, JRadioButtonMenuItem, JMenu, JMenuBar, JSeparator), являются подклассами компонента JComponent. Это позволяет в отличие от AWT добавить меню в любое место программы, к люб.ому контейнеру

Класс JPopupMenu

- Позволяет связать выпадающее меню с любым компонентом

События пакетов Swing

AncestorEvent	предок изменен, перемещен, удален
ChangeEvent	изменение состояния
DocumentEvent	изменение состояния документа
DragEvent	событие drag-and-drop
ListDataEvent	изменение данных списка
ListSelectionEvent	изменение состояния выбора списка
MenuEvent	событие меню
TableColumnModelEvent	изменение модели столбца таблицы
TableModelEvent	изменение модели таблицы
TreeExpansionEvent	свертывание и развертывание элементов дерева
TreeModelEvent	изменение модели дерева
TreeSelectionEvent	изменение выделенного элемента дерева