Создание графического интерфейса на Java

Графические библиотеки Java

В Java есть следующие пакеты для создания графических интерфейсов:

- Abstract Windows Toolkit (AWT) - поставляется с JDK, каждый AWT-компонент имеет свой визуальный компонент (peer) для конкретной ОС, переносимость обеспечивается пакетом java.awt.peer; ограничен набор графических компонентов; внешний вид зависит от ОС.
- Standard Widget Toolkit (SWT) – поставляется отдельно для конкретных ОС, включена в среду Eclipce, взаимодействуют с ОС с помощью peer-интерфейсов, в отличии от AWT, расширен ассортимент компонентов.
- Swing – поставляется с JDK, расширяет классы AWT, не зависит от peer-компонентов ОС.
- Java 3D – трехмерная графика.

Тяжело- и легковесные компоненты

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

Архитектура Модель-Представление-Контроллер (MVC)

Шаблон проектирования MVC предполагает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: *модель*, *представление* и *контроллер* – таким образом, что модификация каждого компонента может осуществляться независимо.

Модель *(model)* хранит данные компонента и позволяет легко, не обращаясь к самому компоненту, изменять или получать эти данные.

Вид (view) выводит данные на экран для представления их пользователю.

Контроллер (controller) определяет, как должны реагировать вид и данные модели в ответ на действия пользователя.

Преимущества MVC

- К одной *модели* можно присоединить несколько *видов*, при этом не затрагивая реализацию *модели*. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы.
- Не затрагивая реализацию *видов*, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных), для этого достаточно использовать другой *контроллер*.
- Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (модели), вообще не будут осведомлены о том, какое представление будет использоваться.

Взаимодействия между моделью, видом и контроллером

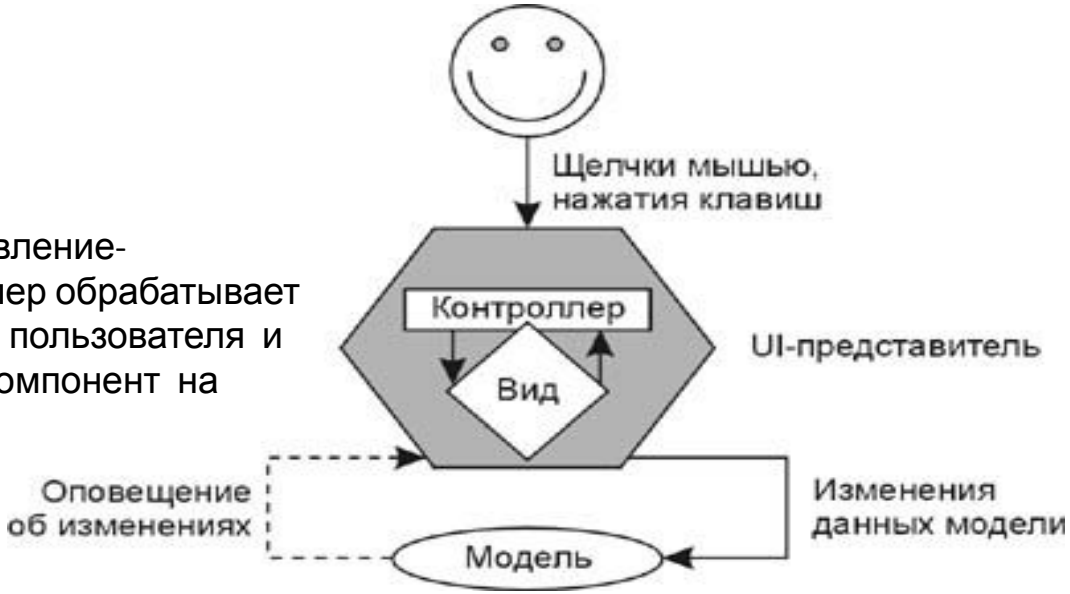
Классическая модель

Тесная связь между контроллером и моделью и контроллером и видом. Представление (вид) сопоставлено с единственным Контроллером и каждый Контроллер с единственным Представлением. Представление и Контроллер имеют прямую ссылку на

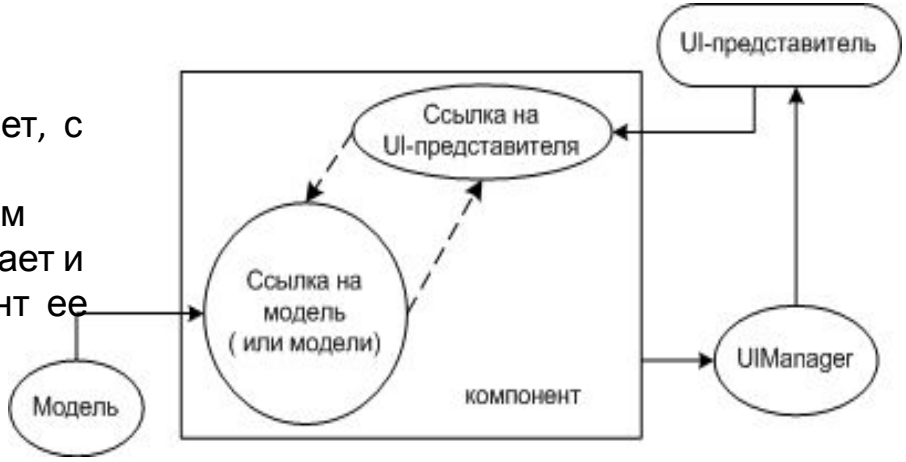
Пример MVC

```
public class Model {
private int[] intArray = {1,2,3,4,5};
public String getStringArray() {
return "intArray=" + Arrays.toString(intArray);    }
public void setIntArray(int index, int value) {
 this.intArray[index] = value;    } }

public class Controller {
Model model = new Model();
View view = new View();
 Controller(){       updateView();    }
void setArrayValue(int index, int value) {
model.setIntArray(index, value);       updateView();    }
void updateView() {       view.showArray(model.getStringArray());    } }

public class View {    public void showArray(String arrayString){
System.out.println("View");   System.out.println(arrayString);    System.out.println();    } }

public class User {    public static void main(String[] args) {
Controller controller = new Controller();         controller.setArrayValue(1, 4);    } }
```

Модель Swing упрощает реализацию



Представление-контроллер обрабатывает события пользователя и рисуют компонент на экране



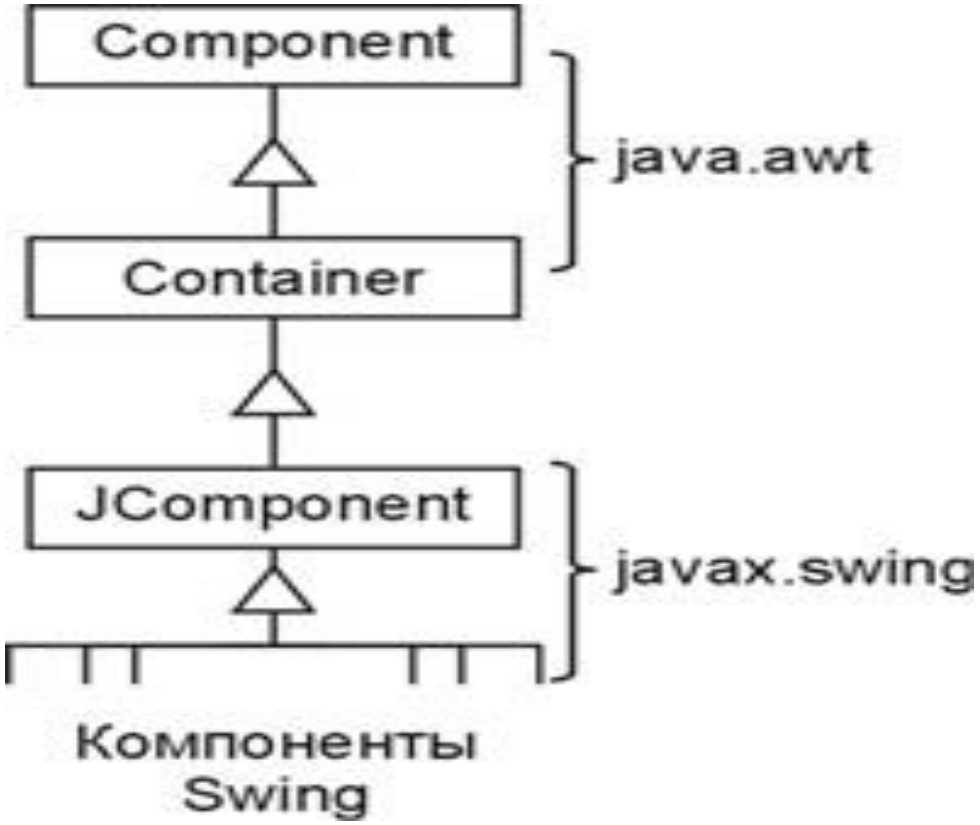
Модель не знает, с каким UI-представителем она сотрудничает и какой компонент ее использует

связывается с моделью только через класс компонента

устанавливает внешний вид и поведение для всех компонентов библиотеки

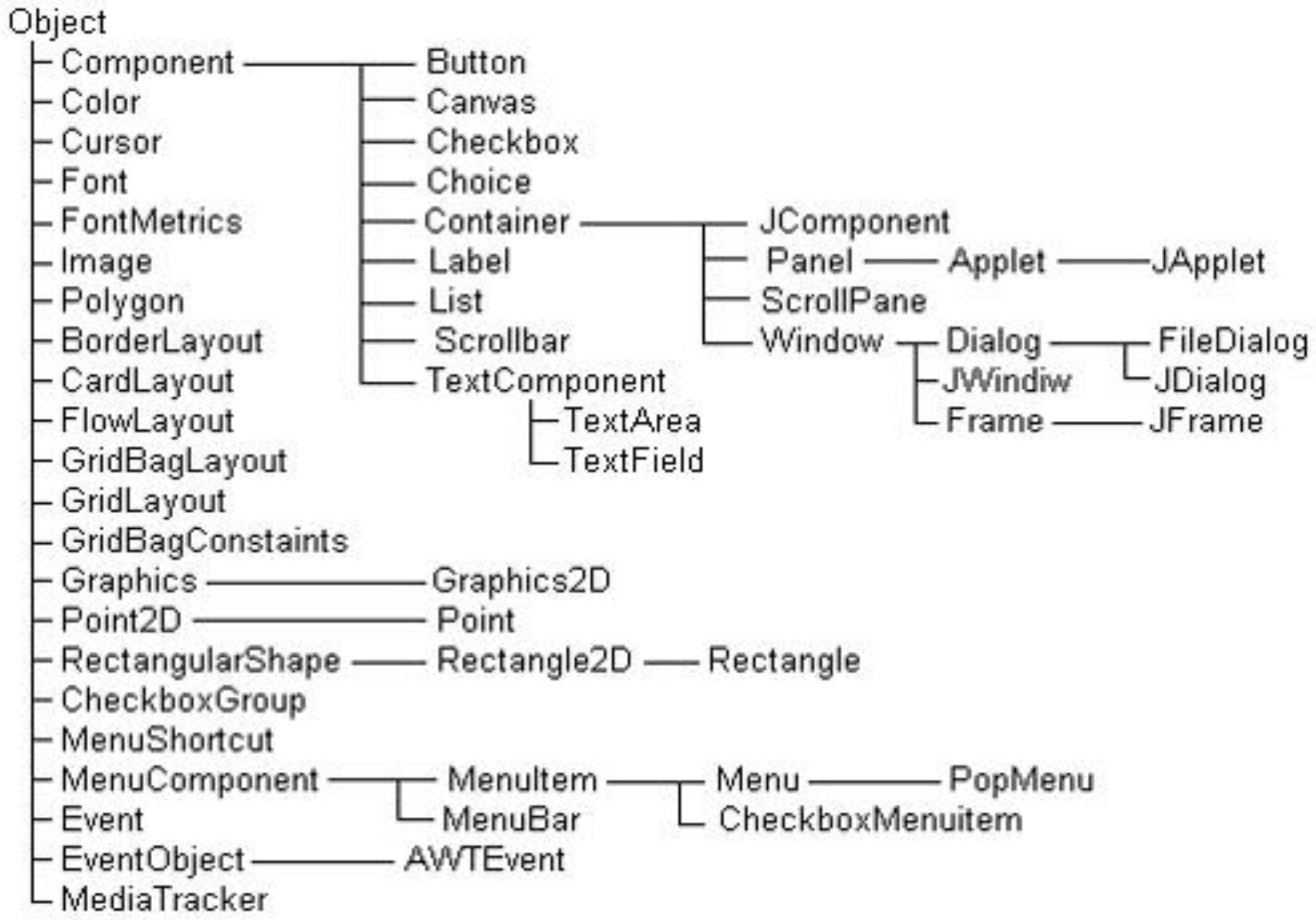
Кнопки, списки, таблицы, текстовые поля …

Диаграмма наследования компонентов библиотеки Swing



Компоненты Swing — это легковесные компоненты AWT

Иерархия основных классов AWT



Компоненты интерфейса

Button - кнопка;

JCheckBox - кнопка-флажок;

JComboBox - выпадающий список;

JLabel - метка, надпись;

JList - список;

JPasswordField - текстовое поле для скрытого ввода;

JProgressBar - компонент для отображения числа в некотором диапазоне;

JRadioButton - преключатели, радио-кнопки, обычно используется с компонентом ButtonGroup;

JSlider - компонент позволяющий выбрать значение из заданного диапазона;

JTable - таблица;

JTextField - однострочное текстовое поле;

JTextArea - многострочное текстовое поле;

JTree - дерево.

Контейнеры интерфейса

Части пользовательского интерфейса, содержащие другие компоненты

Контейнеры верхнего уровня:

Frame, JFrame - окно приложения;

JDialog - диалог приложения;

JColorChooser - диалог выбора цвета;

JFileChooser - диалог выбора файлов и директорий;

FileDialog - диалог выбора файлов и директорий (awt компонент).

Простые контейнеры:

JPanel - простая панель для группировки элементов, включая вложенные панели;

JToolBar - панель инструментов (обычно это кнопки);

JScroolPane - панель прокрутки, позволяющая прокручивать содержимое дочернего элемента;

JDesktopPane - контейнер для создания виртуального рабочего стола или приложений на основе MDI (multiple-document interface);

JEditorPane, JTextPane - контейнеры для отображения сложного документа как HTML или RTF;

JTabbedPane - контейнер для управления закладками;

Создание окна

```
import java.awt.*;
class TooSimpleFrame extends Frame{
  public static void main(String[] args){
    Frame fr = new TooSimpleFrame();
  fr.setSize(400, 150);  // размер окна
  fr.setVisible(true);  // визуализация окна
  }  // кнопка закрытия не работает
}
```

Окно с нестандартной иконкой

```
import javax.swing.*;
public class FrameClosing extends JFrame {
public FrameClosing() {
super("Заголовок Окна");
// операция при закрытии окна
setDefaultCloseOperation(EXIT_ON_CLOSE); // при закрытии окна – выход
// значок для окна
setIconImage(getToolkit().getImage("icon.gif"));  //C:/icons/icon.png
// вывод на экран
setSize(300, 100); // размеры окна ширина и высота
setVisible(true); // визуализация окна
}
public static void main(String[] args) {
new FrameClosing();
}
}
```

Swing15

Стандартные диалоговые окна

Диалоговые окна могут быть модальными (фокус на окне, пока не нажата кнопка) или немодальными

Тип диалогового окна	Описание
NFORMATION_MESSAGE 	Диалоговое окно выводит информацию общего назначения со значком соответствующего вида
WARNIN GE 	Диалоговое окно выводит на экран предупреждающую информацию со значком соответствующего вида
QUESTION SAGE 	Диалоговое окно вопроса для ввода информации
ERROR_M EERROR_MESSAGE 	Диалоговое окно выводит на экран информацию об ошибке со значком соответствующего вида
PLAIN_MESSAGE	Указывает, что диалоговое окно не

Окна ввода и сообщений

```
import java.awt.*;
    import javax.swing.*;
    public class Solution {
        public static void main(String[] args) {
                JOptionPane.showMessageDialog(null,
    "Hello, World");
String s  = JOptionPane.showInputDialog("Введите
    ваше имя");
        }
    }
```

Swing20

Стандартные компоновщики Java

1. Компоновщик BorderLayout (полярное размещение).
2. Компоновщик FlowLayout (последовательное размещение).
3. КомпоновщикGridLayout (табличное размещение).
4. Компоновщик SpringLayout (относительное размещение).
5. Компоновщик BoxLayout (блочное размещение).

Полярное расположение (компоновщик BorderLayout)

- Значение BorderLayout.NORTH или строка "North" — компонент располагается вдоль верхней (северной) границы окна и растягивается на всю его ширину. Обычно так размещается панель инструментов.
- Значение BorderLayout.SOUTH или строка "South" — компонент располагается вдоль нижней (южной) границы и растягивается на всю ширину окна. Такое положение идеально для строки состояния.
- Значение BorderLayout.WEST или строка "West" — компонент располагается вдоль левой (западной) границы окна и растягивается на всю его высоту, однако при этом учитываются размеры северных и южных компонентов (они имеют приоритет).
- Значение BorderLayout.EAST или строка "East" — компонент располагается вдоль правой (восточной) границы окна. В остальном его расположение аналогично западному компоненту.
- Значение BorderLayout.CENTER или строка "Center" — компонент помещается в центр окна, занимая максимально возможное пространство.

Пример использования компоновщика BorderLayout

```
import javax.swing.*;
import java.awt.*;
public class BorderLayoutSample extends JFrame {
public BorderLayoutSample() {
super("BorderLayoutSample");
setSize(400, 300);
setDefaultCloseOperation( EXIT_ON_CLOSE );
// получаем панель содержимого класса JFrame
Container c = getContentPane();
// По умолчанию в Swing используется менеджер BorderLayout
// добавляем компоненты в панель, используя строковые константы
c.add(new JButton("Север"), "North");
c.add(new JButton("Юг"), "South");
// или константы из класса BorderLayout
// JLabelэлемент для отображения текста
c.add(new JLabel("Запад"), BorderLayout.WEST);
c.add(new JLabel("Восток"), BorderLayout.EAST);
// если параметр не указывать вовсе, компонент автоматически добавится в центр
c.add(new JButton("Центр"));
// выводим окно на экран
setVisible(true);
}
public static void main(String[] args) {
new BorderLayoutSample();
}
}
```

Последовательное размещение (компоновщик FlowLayout)

Компоновщик размещает компоненты слева направо, сверху вниз (по умолчанию в панелях Jpanel).

```
import javax.swing.*;
import java.awt.*;
public class FlowLayoutSample extends JFrame {
public FlowLayoutSample() {
super("FlowLayout1");
setSize(400, 200);
setDefaultCloseOperation( EXIT_ON_CLOSE );
// получаем панель содержимого
Container c = getContentPane();
// устанавливаем последовательное расположение с выравниванием компонентов по центру
c.setLayout( new FlowLayout( FlowLayout.CENTER ));
// добавляем компоненты
c.add( new JButton("Один"));
c.add( new JButton("Два"));
c.add( new JButton("Три"));
// выводим окно на экран
setVisible(true);
}
public static void main(String[] args) {
new FlowLayoutSample();
}
}
```

```
import java.awt.*;
    import javax.swing.*;
    public class Solution {
       public static void main(String[] args) {
            JOptionPane.showMessageDialog(null, "Hello, World");
       }
    }
```

Табличное расположение (компоновщик GridLayout)

- все компоненты имеют одинаковый размер. Доступное пространство разбивается на одинаковое количество ячеек, в каждую из которых помещается компонент;
- все компоненты всегда выводятся на экран, как бы ни было велико или мало доступное пространство.

```
import java.awt.*;
import javax.swing.*;
import java.util.*;
class GridTest extends JFrame{
GridTest(String s){
super(s);
Container c = getContentPane();
// 4 строки  4 столбца расстояния между  строками и столбцами в пикселях
c.setLayout(new GridLayout(4, 4, 5, 5));
StringTokenizer st = new StringTokenizer("7 8 9 / 4 5 6 * 1 2 3 - 0 . = +");
while(st.hasMoreTokens())
c.add(new Button(st.nextToken()));
setSize(200, 200);
setVisible(true);
}
public static void main(String[] args){
new GridTest(" Менеджер GridLayout");

}
}
```

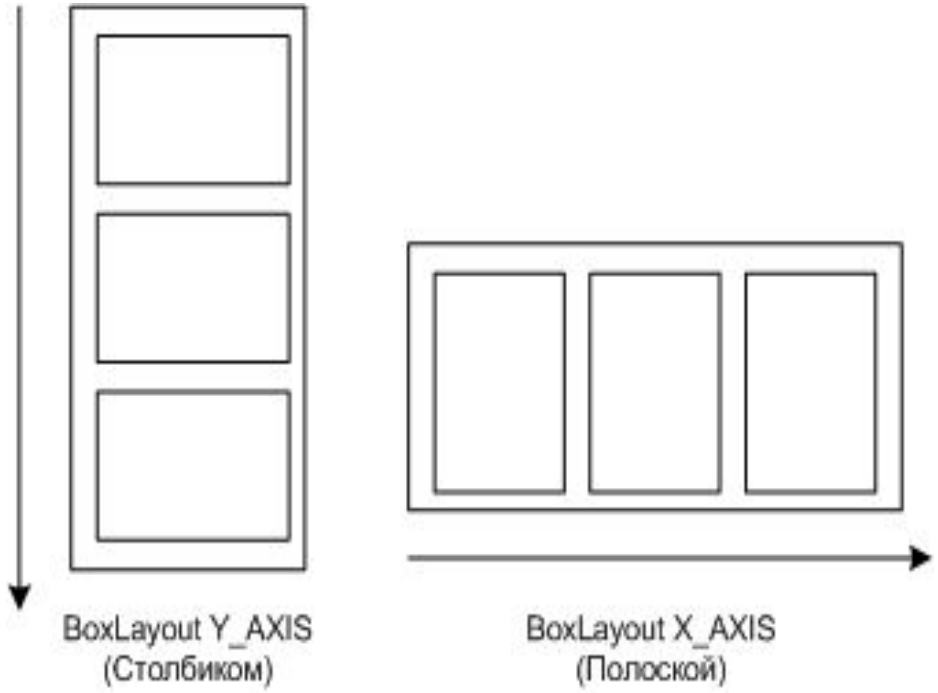
Swing7

Табличное расположение придаст кнопкам одинаковый размер, а последовательное расположение не даст им «расплыться» и заодно выровняет их по правому краю

```
I
mport java.awt.*;
import javax.swing.*;
public class CommandButtons extends JFrame {
public CommandButtons() {
super("CommandButtons");
setSize(350, 250);
setLocation(150, 100);
setDefaultCloseOperation( EXIT_ON_CLOSE );
// создаем панель с табличным расположением для выравнивания размеров кнопок
JPanel grid = new JPanel(new GridLayout(1, 2, 5, 0) ); // 1 строка, 2 столбца, промежутки 5 пикс. по гориз, 0 по верт.
// добавляем компоненты
grid.add( new JButton("OK"));
grid.add( new JButton("Отмена"));
// помещаем полученное в панель с последовательным  расположением, выровненным по правому краю
JPanel flow = new JPanel(new FlowLayout( FlowLayout.RIGHT ));
flow.add(grid);
// получаем панель содержимого
Container c = getContentPane();
// помещаем строку кнопок вниз окна
c.add(flow, BorderLayout.SOUTH );
// выводим окно на экран
setVisible(true);
}
public static void main(String[] args) {
new CommandButtons();
}
}
```

Swing4

Блочное расположение (компоновщик BoxLayout)



Менеджер блочного расположения выкладывает компоненты в контейнер блоками:
столбиком (по оси Y) или полоской (по оси X), при этом каждый отдельный компонент можно выравнивать по центру, по левому или по правому краям, а также по верху или по низу.

Пример блочного размещения

```
import java.awt.*;
import javax.swing.*;
public class Box1 extends JFrame {
public Box1() {
super("Box1 - Y");
setSize(400, 200);
setDefaultCloseOperation( EXIT_ON_CLOSE );
// получаем панель содержимого
Container c = getContentPane();
// устанавливаем блочное расположение по   оси Y (столбиком)
BoxLayout boxy = new BoxLayout(c, BoxLayout.Y_AXIS);
c.setLayout(boxy);
// добавляем компоненты
c.add( new JButton("Один")); c.add( new JButton("Два")); c.add( new JButton("Три"));
// выводим окно на экран
setVisible(true);
}
static class Box2 extends JFrame {
public Box2() {
super("Box2 - X");
// устанавливаем размер и позицию окна
setSize(400, 200);
setLocation(100, 100);
setDefaultCloseOperation( EXIT_ON_CLOSE);
// получаем панель содержимого
Container c = getContentPane();
// устанавливаем блочное расположение по оси X (полоской)
BoxLayout boxx = new BoxLayout(c, BoxLayout.X_AXIS);
c.setLayout(boxx);
// добавляем компоненты
c.add( new JButton("Один")); c.add( new JButton("Два")); c.add( new JButton("Три"));
// выводим окно на экран
setVisible(true);
}
}
public static void main(String[] args) {
new Box1();
new Box2();
}
} Swing5 В этом примере создаются два окна. В одном из них реализовано блочное расположение по оси Y, в другом — блочное
        расположение по оси X.
```