УНИВЕРСИТЕТ ИТМО

Программирование. 2 семестр

# Графический интерфейс

ITMO more than a UNIVERSITY

# UI - Пользовательский интерфейс



- ☑ Пользовательский интерфейс (Human Machine Interface)
  - средство взаимодействия пользователя и компьютера
- ☑ Текстовый интерфейс
  - Command Line Interface
- ☑ GUI (Graphical User Interface)

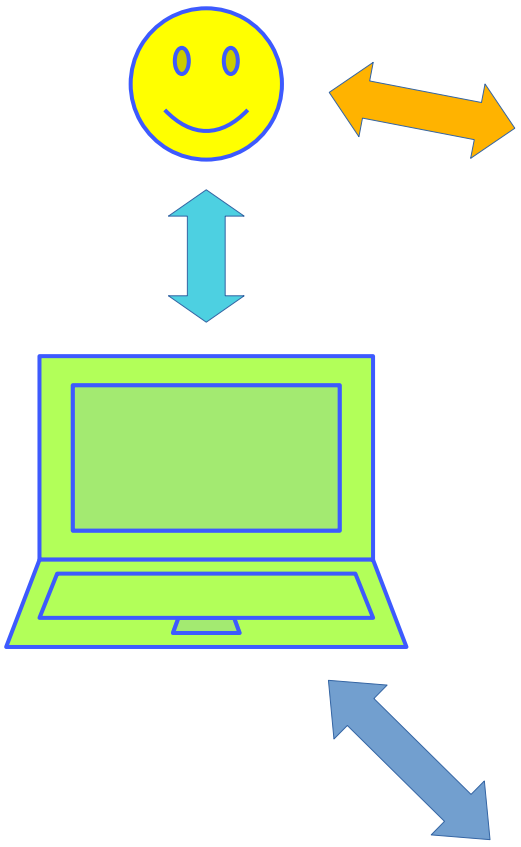
# GUI - Graphical User Interface



- ☑ WIMP
  - Window
  - Icon
  - Menu
  - Pointer

# Архитектура





GUI Applications

Widget Toolkits

Display Server

Window Manager

OS kernel / Graphics API

Hardware Drivers

Java

# AWT

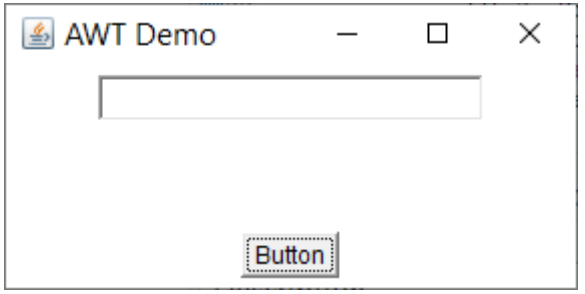


- ☑ AWT — Abstract Window Toolkit
  - библиотека, зависимая от графической подсистемы ОС
  - одинаково "хороший" вид на всех платформах

# AWT



- ☑ AWT — Abstract Window Toolkit
  - библиотека, зависимая от графической подсистемы ОС
  - одинаково "хороший" вид на всех платформах
  - оставили только общий функционал, остальное выпилили



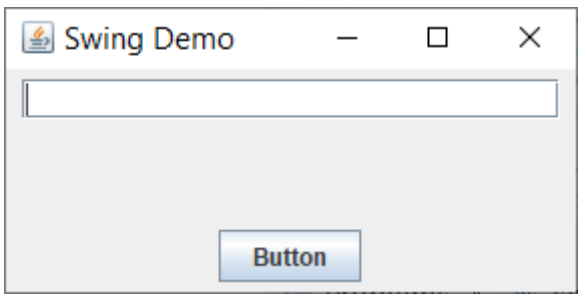

Java

# Swing



## ☑ Swing

- надстройка над AWT в виде легковесных Java-компонентов
- отрисовка кодом на Java
- изменяемый вид компонентов

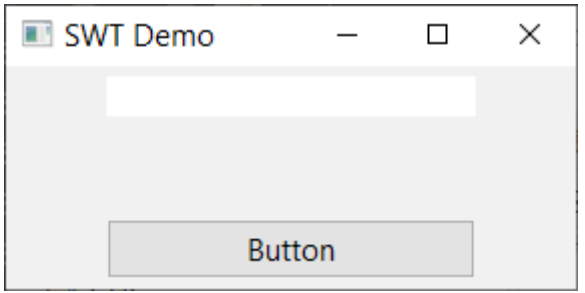


# SWT



## ☑ SWT

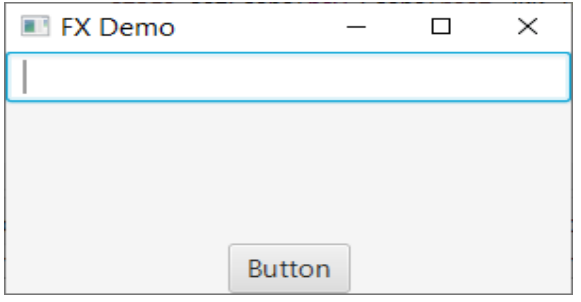
- Часть Eclipse, компоненты-оболочки для компонентов ОС
- Недостающий функционал написан на Java




Java

## ☑ JavaFX

- улучшенная поддержка анимации
- визуальные эффекты
- XML для задания интерфейса
- CSS для задания стилей



# Создание графических приложений



1) Создание основного окна
2) Создание остальных элементов интерфейса
3) Размещение элементов интерфейса
4) Обеспечение реакции на события
5) Все заработало!

Java

# Класс java.awt.Component



- ☑ Компонент (widget, control) — отображаемый и взаимодействующий с пользователем элемент GUI
  - java.awt.Component - абстрактный класс — элемент GUI
  - цвет, размер, местоположение
  - порождает основные события

Java

# Цвет



- ☑ Класс Color
  - Константы
    - Color.BLACK
    - Color.RED
    - ...
  - Конструкторы
    - Color(r, g, b [,a]) int (0-255), float (0.0-1.0)
    - Color(int [,boolean]) int (0x[AA]RRGGBB)
  - Методы
    - getRed(), getGreen(), getBlue(), getAlpha()
    - brighter(), darker()

| | | | |
|---|---|---|---|
| 255,0,0 | 0.5, 0, 0 | 0xFFA0A0 | 0,0,0 |
| 255,255,0 | 0.5, 0.5, 0 | 0xFFFFA0 | 0.2,0.2,0.2 |
| 0,255,0 | 0, 0.5, 0 | 0xA0FFA0 | 0x666666 |
| 0,255,255 | 0, 0.5, 0.5 | 0xA0FFFF | 153,153,153 |
| 0,0,255 | 0, 0, 0.5 | 0xA0A0FF | 0.8,0.8,0.8 |
| 255,0,255 | 0.5, 0, 0.5 | 0xFFA0FF | 0xFFFFFF |

Java

# Методы класса Component



- ☑ Цвет текста и цвет фона

```
Color getForeground()
void setForeground(Color)
Color getBackground()
void setBackground(Color)
```

# Положение и размер



- ☑ Класс Point (int x, int y)
  - getX(), getY(),
  - setLocation(x,y)
- ☑ Класс Dimension (int height, int width)
  - getHeight(), getWidth(),
  - setSize(h, w)
- ☑ Класс Rectangle (int x, int y, int height, int width)
  - getX(), getY(), getHeight(), getWIdth(), getLocation(), getSize()
  - setLocation(x,y), setSize(h,w), setBounds(x,y,h,w)

Java

# Методы класса Component



- ☑ Положение и размеры

```
void setBounds(Rectangle)
Rectangle getBounds()
void setLocation(Point)
Point getLocation()
void setSize(Dimension)
Dimension getSize()
```

# Шрифт



- ☑ Класс Font
  - физические (Arial, Times)
  - логические (Dialog, `DialogInput`, Serif, SansSerif, `Monospaced`)
  - Константы:
    - Font.DIALOG, `Font.MONOSPACED`, Font.SERIF, Font.SANS_SERIF
    - Font.PLAIN, **Font.BOLD**, *Font.ITALIC*
  - Конструктор Font(String name, int style, int size)
  - Методы
    - String getFontName(), int getStyle(), int getSize()

# Методы класса Component



- ☑ Шрифт

```
Font getFont()
void setFont(Font)
```

# Методы класса Component



- ☑ Видимость

```
boolean isVisible()
void setVisible(boolean)
```

  - Компоненты изначально видимы, кроме основных окон

- ☑ Активность

```
boolean isEnabled()
void setEnabled(boolean)
```

  - Компоненты изначально активны (воспринимают действия пользователя и порождают события)

# Методы класса Component



- ☑ Дополнительное рисование

```
void paint(Graphics)
void update(Graphics)
void repaint()
```

  - Graphics — графический контекст компонента

- ☑ Системный вызов paint
  - первое отображение
  - изменение размера
  - необходимость перерисовки
- ☑ JVM вызывает paint(Graphics)

- ☑ Программный вызов paint
  - изменение состояния компонента
- ☑ в программе вызывается repaint()
- ☑ регистрируется событие отрисовки
- ☑ JVM вызывает update(Graphics)

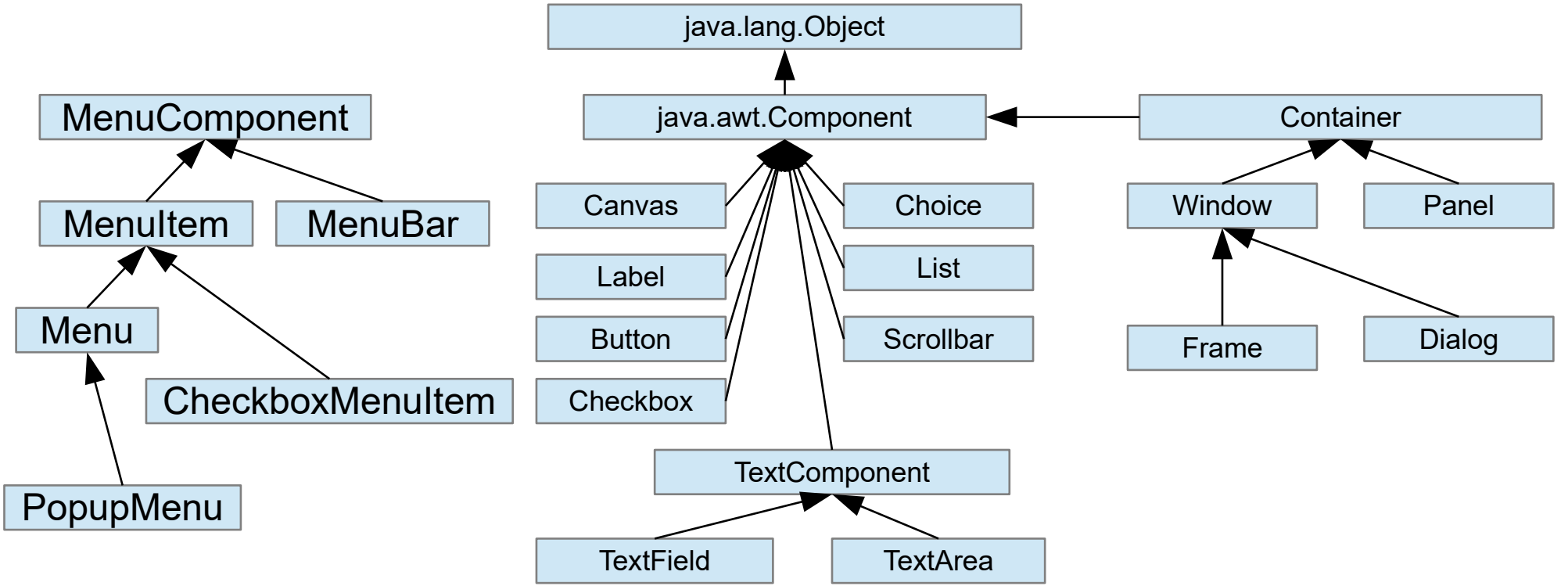
# Класс java.awt.Container



- ☑ Контейнер — компонент, который содержит другие компоненты
  - extends java.awt.Component
- ☑ Иерархия компонентов - дерево
- ☑ Компонент может находиться только в одном контейнере
- ☑ Методы:
  - add(Component)
  - setLayout(LayoutManager)
  - validate()

# Основные компоненты и контейнеры AWT





# Размещение компонентов в контейнере



- ☑ Абсолютное позиционирование
  - Отсутствует реакция на изменение размера контейнера
  - Проблемы с изменением шрифта или локали
- ☑ Менеджер компоновки
  - Управляет расположением и размером компонентов

# Менеджер компоновки



- ☑ Интерфейс LayoutManager
  - Container.setLayout(LayoutManager)
  - Container.add(Component)
- ☑ Интерфейс LayoutManager2
  - Container.setLayout(LayoutManager2, Object constraints)
  - Container.add(Component, Object constraint)

# Менеджер компоновки



- ☑ Расстановка элементов
  - Container.validate()
  - Container.invalidate()
  - Container.doLayout()
  - LayoutManager.layoutContainer(Container)
- ☑ Управление размером компонентов
  - Component.getPreferredSize()
  - Component.getMinimumSize()
  - Component.getMaximumSize()

Java

# FlowLayout

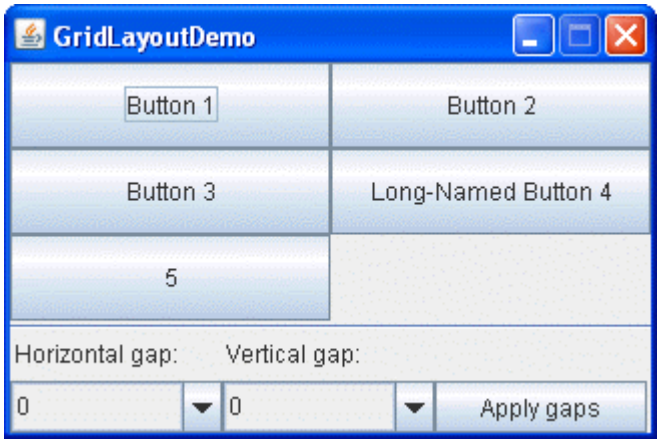


- ☑ Заполнение контейнера слева направо (или справа налево) построчно
- ☑ Компоненты сохраняют свой размер preferredSize
- ☑ Управление размещением:
  - `setHgap(int), setVgap(int) // 5`
  - `setAlignment(LEFT, RIGHT, CENTER)  // CENTER`

# GridLayout



- ☑ Контейнер делится на одинаковые ячейки по строкам и столбцам
- ☑ Все компоненты будут одного размера
- ☑ GridLayout(int rows, int cols)
- ☑ Управление размещением:
  - `setHgap(int), setVgap(int) // 0`
  - `setRows(int), setColumns(int) // 1, 0`

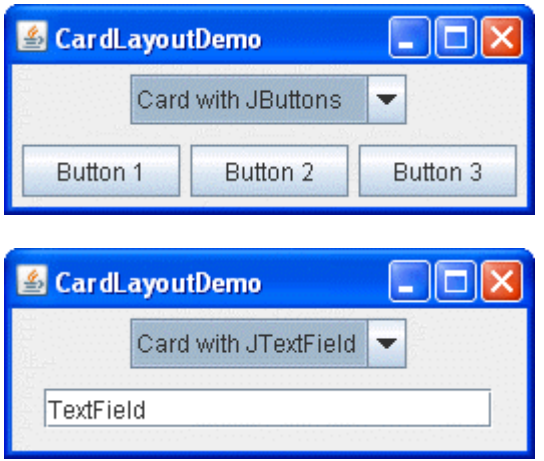



Java

# CardLayout



- ☑ Аналог колоды карт (виден только верхний компонент)
- ☑ Позволяет выбрать одну из панелей



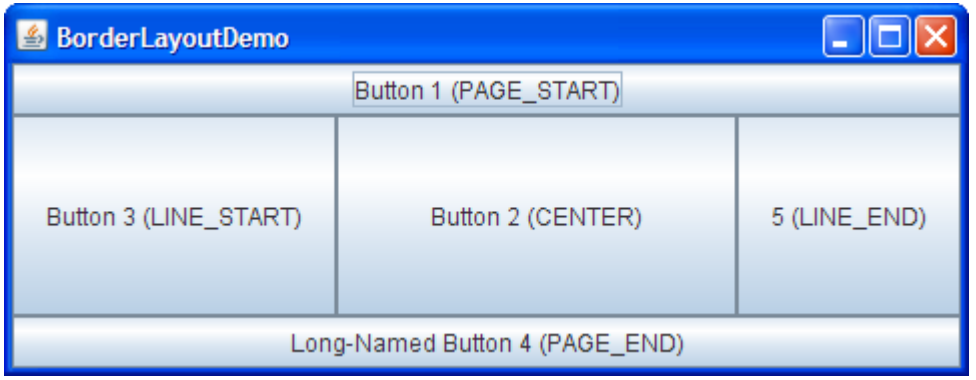

Java

# BorderLayout



☑ Компоненты располагаются в 5 областях:

- CENTER, NORTH, WEST, SOUTH, EAST;

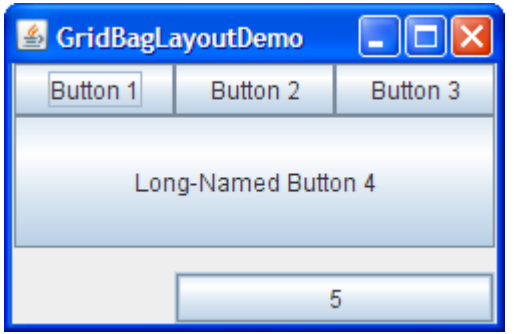



Java

# GridBagLayout + GridBagConstraints



- ☑ Контейнер делится на ячейки по строкам и столбцам
- ☑ Задаются ограничения
  - объединение ячеек
  - заполнение ячеек
  - привязка к краю ячеек
  - распределение пространства




Java

# Что еще нужно?



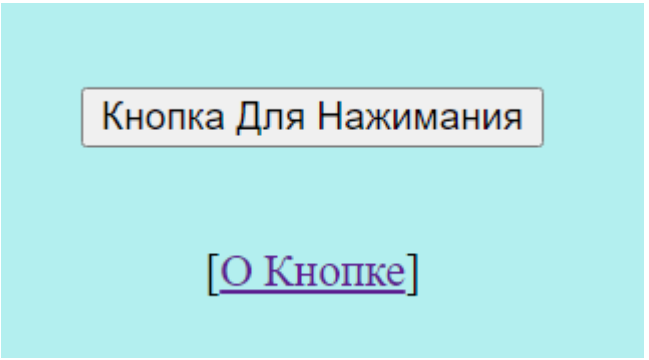
- ☑ Элементы созданы
- ☑ Размещены по контейнерам
- ☑ Все?

Java

# Что еще нужно?



- ☑ Элементы созданы
- ☑ Размещены по контейнерам
- ☑ Все?
- ☑ http://elmk.narod.ru/knopka.html



# Что еще нужно?



- ☑ Элеме
- ☑ Разме
- ☑ Все?
- ☑ http://

Кнопка Для Нажимания

Метафизическая конструкция, предназначенная для осознания индивидом тщетности человеческих усилий, иллюзорности собственного существования и эфемерности всего сущего. Благодаря конструктивным особенностям, при нажатии на Кнопку ровным счетом ничего не происходит, что дает нажимающему обильную пищу для размышлений на вышеперечисленные темы. Кроме того, Кнопка дает прекрасную возможность обрести заслугу, производя действие, не имеющее последствий и, соответственно, не порождающее новых причин.

---

[ Вернуться к Кнопке Для Нажимания ]

Java

# Обработка событий



- ☑ Событийно-ориентированное программирование
- ☑ Не задана последовательность выполнения кода
- ☑ Код выполняется асинхронно при наступлении определенных событий
- ☑ Шаблон Observer

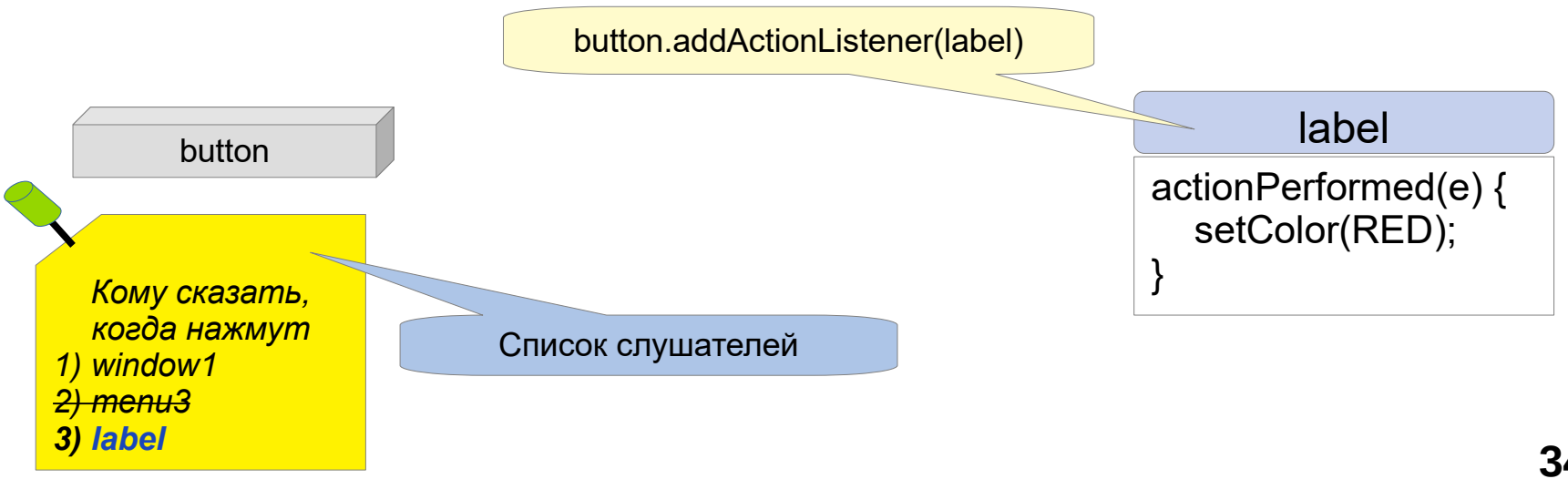
Java

# Обработка событий



- ☑ button.addActionListener(label)
- ☑ Перевод:
  - – Кнопка, когда тебя нажмут — скажи метке
  - – Ладно, записала...

То есть
вызови метод
actionPerformed()

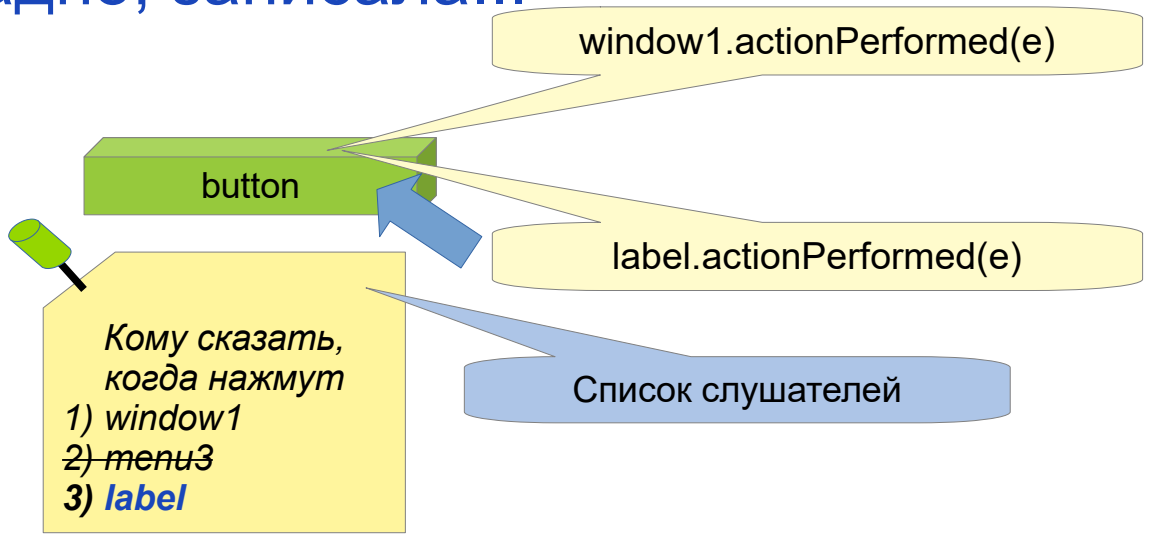


# Обработка событий



- ☑ button.addActionListener(label)
- ☑ Перевод:
  - – Кнопка, когда тебя нажмут — скажи метке
  - – Ладно, записала...

То есть
вызови метод
actionPerformed()




label

```
actionPerformed(e) {
    setColor(RED);
}
```

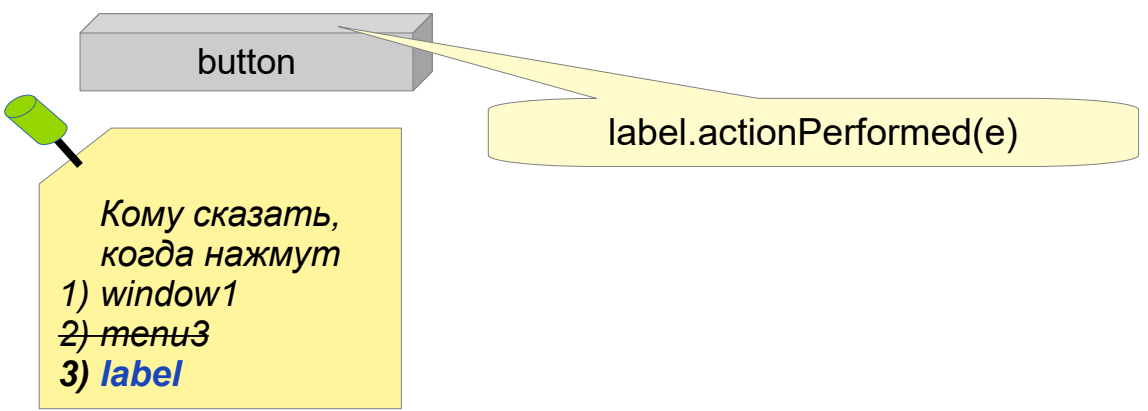
Java

# Обработка событий



- ☑ button.addActionListener(label)
- ☑ Перевод:
  - – Кнопка, когда тебя нажмут — скажи метке
  - – Ладно, записала...

То есть
вызови метод
actionPerformed()




label

```
actionPerformed(e) {
    setColor(RED);
}
```

# Обработка событий



- ☑ Источник события — любой компонент
- ☑ Событие — потомок класса AWTEvent
- ☑ Обработчик реализует интерфейс ...Listener и его методы
- ☑ Методу передается объект события для обработки

```
class A implements ActionListener {
    Button b = new Button("OK");
    Label l = new Label("Button pressed");
    l.setVisible(false);
    b.addActionListener(this);    - подписка на событие
......
public void actionPerformed(ActionEvent e) {
    l.setVisible("true"); - реакция на событие
}
```

# Обработка событий



- Анонимным классом

```
b.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    l.setVisible("true");
  }
});
```

- Лямбда-выражением

```
b.addActionListener((e) -> l.setVisible("true"));
```

# Mouse(Motion)Listener, MouseEvent



- ☑ MouseListener
  - mousePressed(MouseEvent)
  - mouseReleased(MouseEvent)
  - mouseClicked(MouseEvent)
  - mouseEntered(MouseEvent)
  - mouseExited(MouseEvent)
- ☑ MouseMotionListener
  - mouseDragged(MouseEvent)
  - mouseMoved(MouseEvent)

- ☑ MouseEvent
  - getPoint()
  - getLocationOnScreen()
  - getButton()
  - getClickCount()

# Классы-адаптеры



```
class X implements MouseListener {
   public void mousePressed(MouseEvent e) {
        // обработка нажатия кнопки мыши
    }
// обработка других событий не требуется
     public void mouseClicked(MouseEvent e) { }
     public void mouseReleased(MouseEvent e) { }
     ....
}
```

```
class Y extends MouseAdapter {
    public void mousePressed(MouseEvent e) { ... }
}
```

# KeyListener, KeyEvent



☑ KeyListener

- keyPressed(KeyEvent)
- keyReleased(KeyEvent)
- keyTyped(KeyEvent)

☑ KeyEvent

- getKeyChar()
  // для keyTyped()
- getKeyCode()
  // для keyPressed, keyReleased
- getModifiers()
  // Shift, Alt, Ctrl, Meta ...
- getKeyLocation()
  // Standard, Left, Right, Numpad

Java

# WindowListener, WindowEvent



☑ WindowListener

- windowOpened(WindowEvent)
- windowClosing(WindowEvent)
- windowClosed(WindowEvent)
- windowActivated(WindowEvent)
- windowDeactivated(WindowEvent)
- windowIconified(WindowEvent)
- windowDeiconified(WindowEvent)

☑ WindowEvent

- getNewState()
- getOldState()
- getOppositeWindow()

Java

# ActionListener, ActionEvent



- ☑ ActionListener
  - actionPerformed(ActionEvent)
- ☑ ActionEvent
  - нажата кнопка
  - двойной клик в списке
  - выбор пункта меню
  - клавиша Enter в текстовом поле

# AdjustmentListener, AdjustmentEvent



- ☑ AdjustmentListener
  - adjustmentValueChanged(AdjustmentEvent)
- ☑ AdjustmentEvent
  - int getValue()
  - boolean getValueIsAdjusting()

Java

# ItemListener, ItemEvent



- ☑ ItemListener
  - itemStateChanged(ItemEvent)
- ☑ ItemEvent
  - Object getItem()
  - int getStateChange() // selected-deselected
  - установка-сброс флажка
  - установка-сброс пункта меню
  - выбор элемента списка

# TextListener, TextEvent



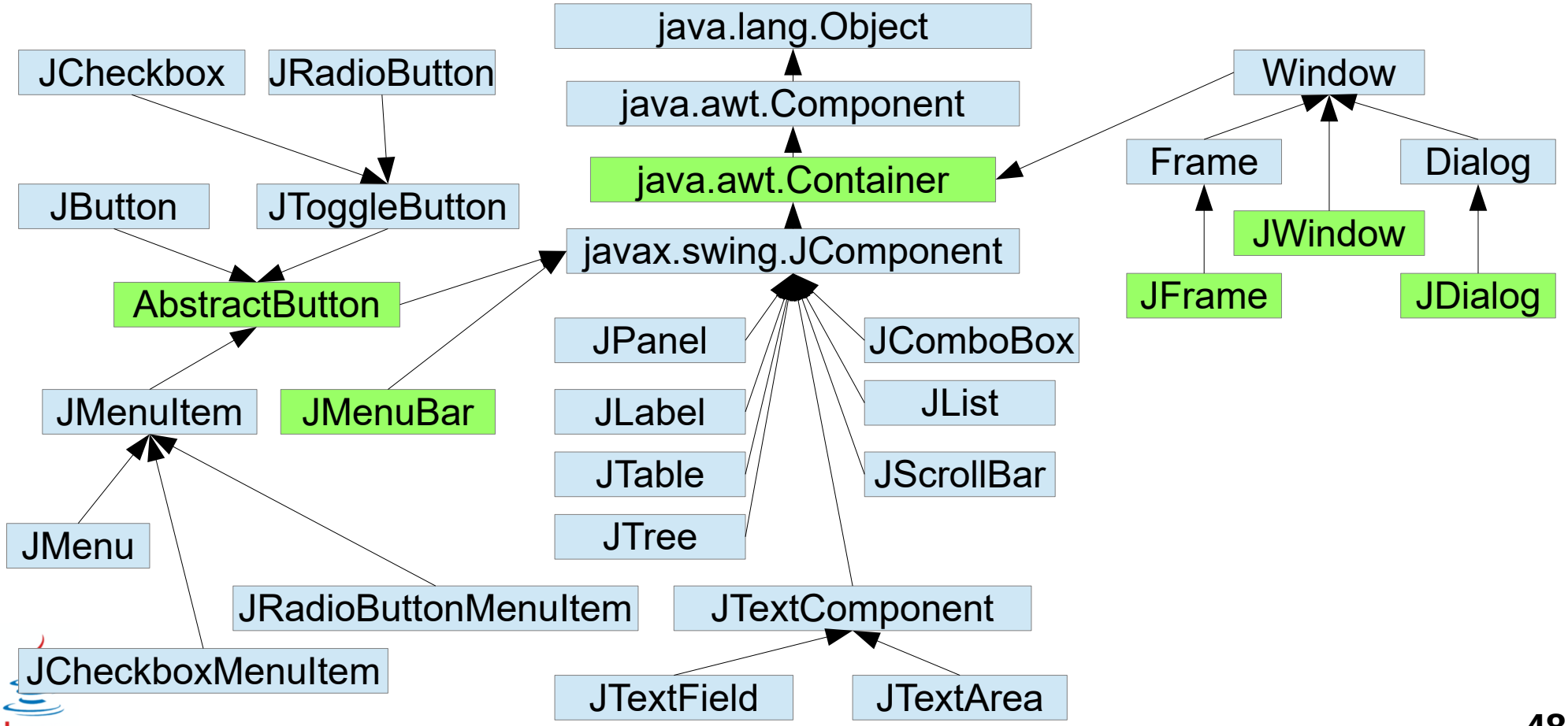
- ☑ TextListener
  - textValueChanged(TextEvent)
- ☑ TextEvent
  - изменился текст в текстовом компоненте

Java


УНИВЕРСИТЕТ ИТМО
Программирование. 2 семестр
Java Swing
ITMO re than a UNIVERSITY

# Основные компоненты Swing

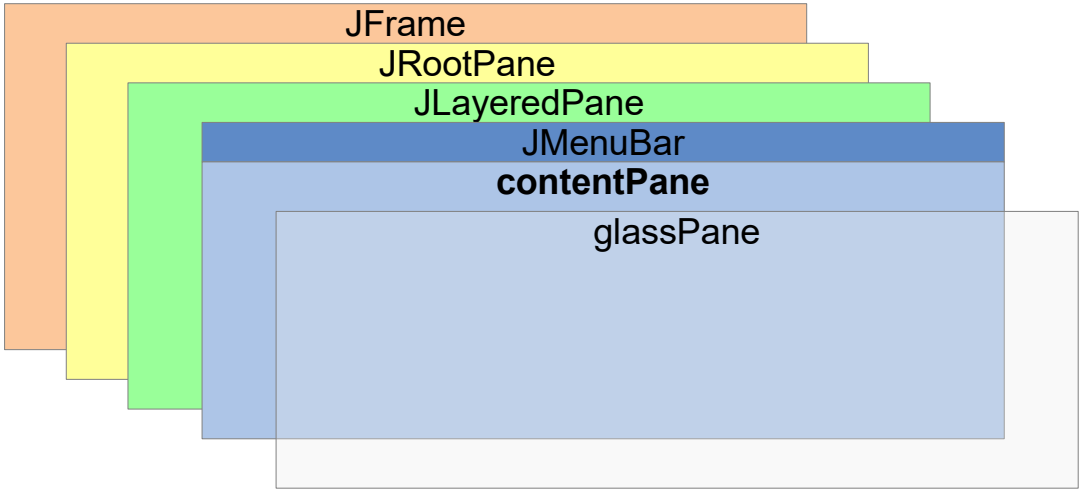




# JFrame



- ☑ Не является легковесным компонентом — это окно ОС
- ☑ Содержит набор панелей для размещения компонентов
- ☑ При создании — невидимый
- ☑ JFrame.add() = JFrame.getContentPane.add()




Java

# JFrame



```
JFrame f = new JFrame();
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.add(new JLabel("Hello!"), BorderLayout.CENTER);
f.setJMenuBar(new JMenuBar());
f.pack(); // установка размеров фрейма
f.setVisible(true);
```

# Потоки в приложениях Swing



- ☑ Основные потоки (initial)
  - Для выполнения основного кода приложения (main)
  - Запуск метода для создания элементов интерфейса в потоке обработки событий
- ☑ Поток обработки событий (event-dispatching)
  - для кода обработки событий
  - Все действия с элементами интерфейса здесь!
  - Действия должны выполняться быстро
- ☑ Фоновые потоки (worker)
  - для задач, требующих длительного времени
  - запускаются с помощью класса SwingWorker

# Организация приложения Swing



- ☑ InvokeLater(Runnable) – асинхронный старт
  - для приложений
- ☑ InvokeAndWait(Runnable) – синхронный старт
  - для апплетов

```
public class Main {
  public static void main(String... args) {
    SwingUtilities.invokeLater(() -> gui());
  }
  private void gui() {
    JFrame f = new JFrame();
    ...
    f.setVisible(true);
  }
}
```

# SwingWorker<T, V>



- ☑ Класс для долгих фоновых задач
- ☑ T — тип результата, V — тип промежуточных значений
- ☑ методы
  - abstract T doInBackground() - выполнить в фоновом потоке
  - done() - вызовется после завершения выполнения
  - T get() - возвращает результат
  - publish(V) – передать промежуточный результат
  - process(List<V>) - обработать
  - addPropertyChangeListener()
- ☑ Свойства
  - state (PENDING, STARTED, DONE)
  - progress (0 - 100)

# JComponent



- ☑ extends java.awt.Container — может содержать картинку
- ☑ всплывающие подсказки — setToolTipText()
- ☑ построение основано на шаблоне MVC
- ☑ встроенная двойная буферизация при отрисовке
- ☑ реализация метода paint

```
void paint(Graphics g) {
   paintComponent(g);
   paintBorder(g);
   paintChildren(g);
}
```

- Для отрисовки нужно переопределить paintComponent(g)
- Необходимо вызывать super.paintComponent(g);

# MVC и Swing

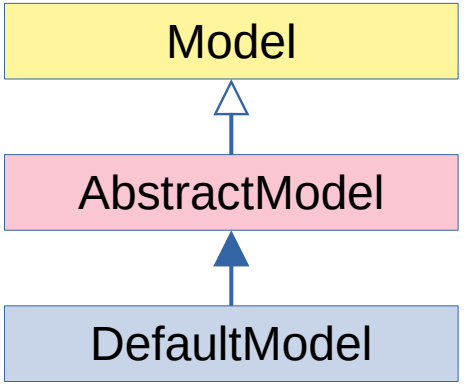


- ☑ MVC — Model, View, Controller
  - Модель отвечает за поведение
  - Представление — отвечает за отображение
  - Контроллер — связывает модель и представление и управляет ими
- ☑ Реализация Swing — Model + UI Delegate
  - UI Delegate = View + Controller
  - Модель может быть визуальной или моделью данных
  - Одну модель данных можно назначить разным компонентам
  - В случае большого числа событий можно использовать ChangeEvent — изменение в модели.

# Модели



- ☑ Модели — интерфейсы: ButtonModel, ListModel, ...
- ☑ Реализации моделей по умолчанию — DefaultListModel, DefaultTableModel
- ☑ Для сложных моделей дополнительно имеются классы абстрактных моделей. Например, AbstractTableModel, AbstractTreeModel




Java

# Делегаты



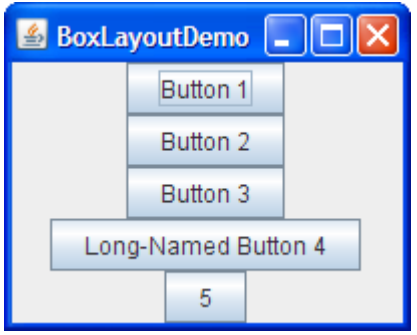
- ☑ Делегаты — потомки класса javax.swing.plaf.ComponentUI, например, ButtonUI, ListUI
- ☑ Напрямую в коде не используются
- ☑ Для управления делегатами предназначен класс javax.swing.UIManager

```
UIManager.setLookAndFeel(
    UIManager.getSystemLookAndFeelClassName());
SwingUtilities.updateComponentTreeUI(frame);
frame.pack();
```

# Менеджеры компоновки Swing



- ☑ BoxLayout
  - Компоненты располагаются в один ряд вертикально или горизонтально
- ☑ Класс Box — контейнер с BoxLayout
  - Box.createHorizontalBox()
  - Box.createVerticalBox()
- ☑ createRigidArea(Dimension)
- ☑ createHorizontalGlue()
- ☑ createVerticalGlue()
- ☑ Filler(minSize, prefSize, maxSize) - заполнитель



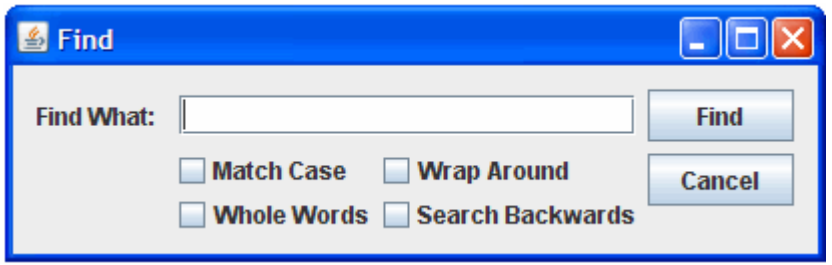

Java

# Менеджеры компоновки Swing



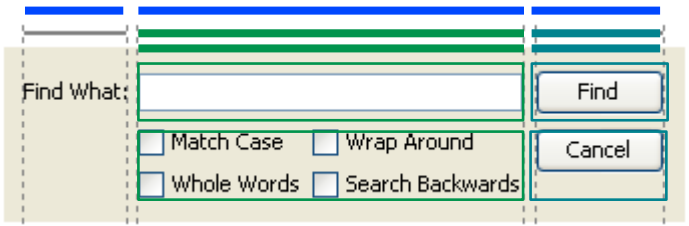
## ☑ GroupLayout

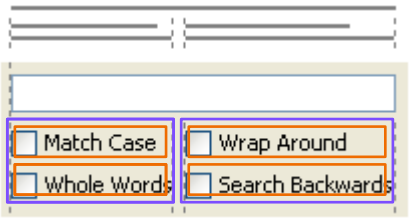
- Все компоненты описываются дважды — горизонтальное расположение и вертикальное расположение
- Все компоненты являются участниками групп — последовательных и параллельных



# GroupLayout



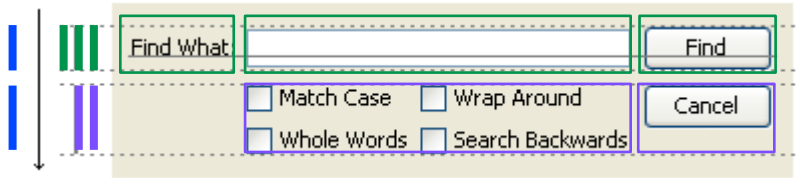



```
layout.setHorizontalGroup(layout.createSequentialGroup()
    .addComponent(label)
    .addGroup(layout.createParallelGroup(Alignment.LEADING)
          .addComponent(textField)
          .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(Alignment.LEADING)
                    .addComponent(caseCheckBox)
                    .addComponent(wholeCheckBox))
                .addGroup(layout.createParallelGroup(Alignment.LEADING)
                    .addComponent(wrapCheckBox)
                    .addComponent(backCheckBox))))
     .addGroup(layout.createParallelGroup(Alignment.LEADING)
        .addComponent(findButton)
        .addComponent(cancelButton))
);
```

# GroupLayout







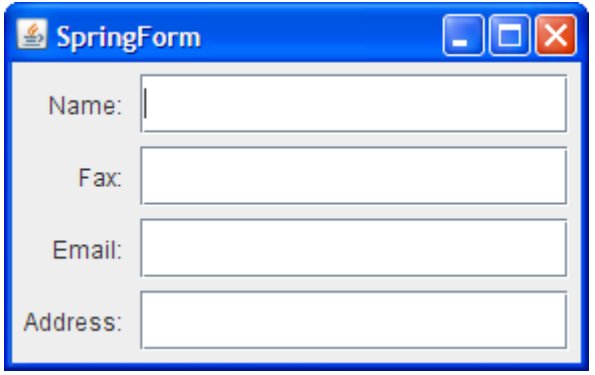
```
layout.setVerticalGroup(layout.createSequentialGroup()
    .addGroup(layout.createParallelGroup(Alignment.BASELINE)
        .addComponent(label)
        .addComponent(textField)
        .addComponent(findButton))
    .addGroup(layout.createParallelGroup(Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGroup(layout.createParallelGroup(Alignment.BASELINE)
                .addComponent(caseCheckBox)
                .addComponent(wrapCheckBox))
            .addGroup(layout.createParallelGroup(Alignment.BASELINE)
                .addComponent(wholeCheckBox)
                .addComponent(backCheckBox)))
        .addComponent(cancelButton))
);
```

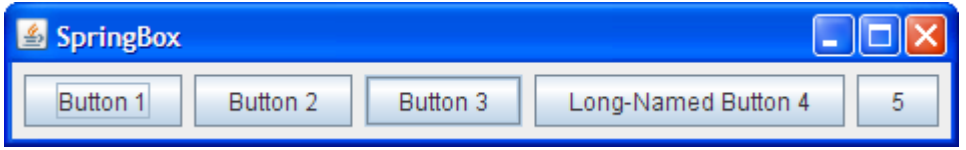
# Менеджеры компоновки Swing



## ☑ SpringLayout

- Все компоненты соединены пружинами (Spring), которые имеют минимальную, максимальную и предпочтительную длину
- Обычно используется автоматическими расстановщиками




Java

# JLabel



- ☑ Метка изначально прозрачная
- ☑ метод setOpaque(true) — сделать непрозрачной

# Простые текстовые элементы



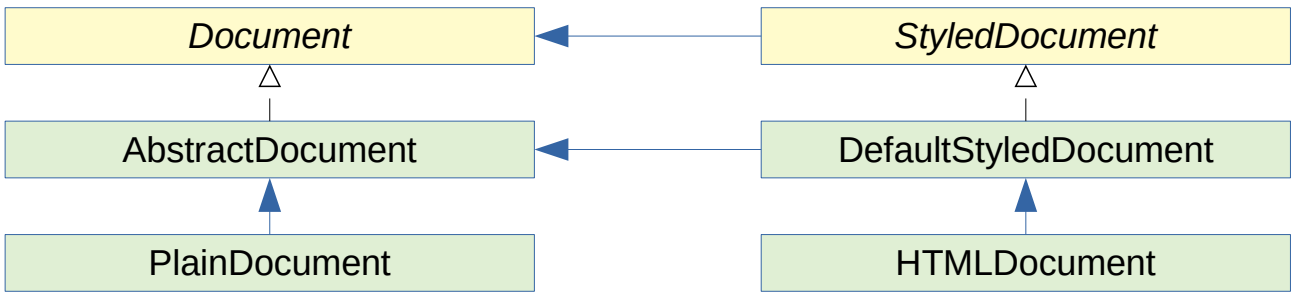
- ☑ JTextField — однострочное поле
  - JFormattedTextField — возможность проверки ввода
  - JPasswordField — не отображает введенные символы
  - Основное событие — ActionEvent
- ☑ JTextArea — многострочное поле
  - События — ActionEvent, UndoableEditEvent

Java

# Текстовые модели



- ☑ Модель для всех текстовых компонентов — Document



# Расширенные редакторы



- ☑ EditorPane — панель текста определенного формата с заданным редактором (поддерживается редактирование обычного текста, RTF, HTML)
  - TextPane — панель текста с поддержкой стилей
  - HyperlinkListener

# JButton, JCheckBox, JRadioButton



- ☑ Конструктор принимает строку
- ☑ Для JCheckBox и JRadioButton — еще состояние (boolean)
- ☑ JRadioButton используется в группе ButtonGroup
- ☑ События
  - ActionEvent для JButton, JRadioButton
  - ItemEvent для JCheckBox (позволяет отследить select-deselect)
- ☑ Модель — DefaultButtonModel — элемент с двумя состояниями
- ☑ Почти так же обрабатываются JMenuItem, JMenu, JCheckBoxMenuItem, JRadioButtonMenuItem

# JList



- ☑ Конструктор принимает массив или вектор объектов
- ☑ Основное событие — ListSelectionEvent
- ☑ Модели
  - ListModel ← AbstractListModel ← DefaultListModel
    - getElementAt()
    - getSize()
  - DefaultListModel — модель данных (вектор),
  - ListSelectionModel ← DefaultListSelectionModel
  - DefaultListSelectionModel — модель вариантов выбора (одиночный, интервальный, множественный)

# JComboBox



- ☑ Может быть редактируемым и нередактируемым
- ☑ Конструктор принимает массив или вектор объектов
- ☑ Основное событие — ActionEvent, иногда ItemEvent
- ☑ Модель DefaultComboBoxModel реализует 3 интерфейса — ListModel, ComboBoxModel и MutableComboBoxModel.
- ☑ По сравнению с ListModel — ComboBoxModel вводит понятие выбранный элемент (отображаемый)
- ☑ MutableComboBoxModel позволяет добавлять и удалять элементы

# JSpinner



- ☑ Составной компонент — 2 кнопки и редактор значений
- ☑ Конструктор принимает модель SpinnerModel
- ☑ Основное событие — ChangeEvent
- ☑ 3 готовых модели — SpinnerListModel, SpinnerDateModel, SpinnerNumberModel + AbstractSpinnerModel
- ☑ 3 готовых редактора — JSpinner.ListEditor, JSpinner.DateEditor, JSpinner.NumberEditor

# JSlider



- ☑ Составной компонент — 2 кнопки и редактор значений
- ☑ Конструктор принимает min и max значения
- ☑ Основное событие — ChangeEvent
- ☑ Модель — DefaultBoundedRangeModel — еще используется для JProgressBar

# JTable

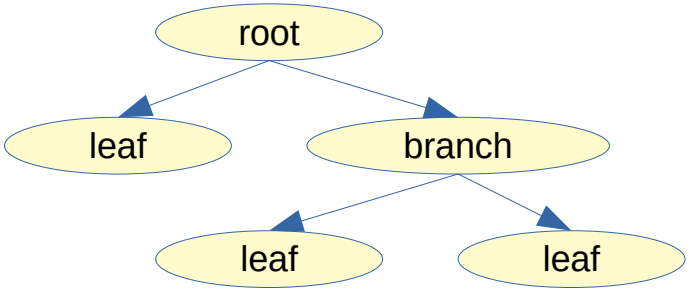


- ☑ TableModel ← AbstractTableModel ← DefaultTableModel
- ☑ DefaultTableModel — простая таблица
  - DefaultTableModel(Object[][] data, Object[] colNames)
  - DefaultTableModel(Vector data, Vector colNames)
- ☑ AbstractTableModel
  - Реализовать методы:
    - int getRowCount(), int getColumnCount(), Object getValueAt(int, int)
    - Class getColumnClass(), isCellEditable(r,c), setValueAt(r,c)
- ☑ TableModelEvent + TableModelListener
- ☑ Сортировка — TableRowSorter

# JTree



- ☑ TreeModel ← DefaultTreeModel
- ☑ TreeNode ← MutableTreeNode ← DefaultMutableTreeNode
- ☑ TreePath — путь к узлу
- ☑ TreeModelEvent, TreeSelectionEvent, TreeExpansionEvent




Java

# Контейнеры



- ☑ JPanel - универсальный контейнер — FlowLayout
- ☑ Box - BoxLayout
- ☑ JScrollPane — контейнер со скроллерами
- ☑ JSplitPane — контейнер из 2 частей
- ☑ JTabbedPane — контейнер с табуляторами
  - SingleSelectionModel


УНИВЕРСИТЕТ ИТМО
Программирование. 2 семестр
Java 2D
ITMOre than a UNIVERSITY

# paintComponent



- ☑ java.awt.Component

```
paint(Graphics g) { // код для рисования }
```

- ☑ javax.swing.JComponent

```
paint(Graphics g) {
  paintComponent(g);
  paintBorder(g);
  paintChildren(g);
}
paintComponent(Graphics g) {
  ui.update(g, this)
}
```

- ☑ javax.swing.plaf.ComponentUI

```
update(Graphics g, Component c) {
  // заполняет фон цветом фона
  this.paint(g, c);  // отрисовка компонента
}
```

# Свой код для рисования



- paintComponent(Graphics2D g) {

```
    super.paintComponent(g);
    Graphics2D g2d = (Graphics2D) g;
}
```

для вызова метода отрисовки

- repaint()

# Graphics2D



- ☑ Координаты (user-space)
- ☑ device-space


Java

# Graphics2D



- ☑ drawString(String s, int x, int y)
- ☑ drawImage(Image img, int x, int y, …)
- ☑ drawLine, drawRect, drawArc, drawOval, …
- ☑ draw(Shape)

# java.awt.geom



- ☑ Point2D, Point2D.Float, Point2D.Double
- ☑ *interface* Shape
- ☑ Line2D
- ☑ RectangularShape
- ☑ Rectangle2D, RoundRectangle2D, Ellipse2D, Arc2D
- ☑ QuadCurve2D, CubicCurve2D

# GeneralPath



- ☑ class GeneralPath implements Shape
  - moveTo()
  - lineTo()
  - quadTo()
  - curveTo()
  - closePath()

Java

# Стиль линии и заполнения

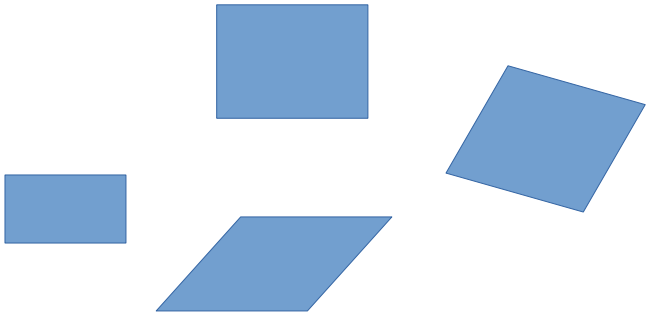


- ☑ interface Stroke
  - BasicStroke
- ☑ interface Paint
  - Color
  - GradientPaint
  - TexturePaint

Java

# Преобразования





- ☑ Graphics2D
  - rotate // вращение
  - scale // масштабирование
  - shear // сдвиг
  - translate // перенос координат
  - transform(AffineTransform)
- ☑ interface AffineTransform
  - getRotateInstance
  - getScaleInstance
  - getShearInstance
  - getTranslateInstance

Java

# Анимация



- ☑ Метод отрисовки объекта

  paintComponent(Graphics g) { drawObject(g); }

- ☑ Метод изменения объекта (размера, координаты, цвет)

  change() { x++; y--; color.darker(); width += 2; }

- ☑ Аниматор
  - javax.swing.Timer
    - + actionPerformed() { change(); repaint(); }
  - java.util.Timer
    - + TimerTask.run() { change(); repaint(); }
  - Thread
    - + Runnable.run() { change(); repaint(); sleep(); }

Java

# Приложение Swing



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

public class SwingApp {
    public SwingApp() {
        JFrame frame = new JFrame("Hello");
        JLabel label = new JLabel("");
        JButton button = new JButton("OK");
        frame.getContentPane().setLayout(new FlowLayout());
        frame.getContentPane().add(label);
        frame.getContentPane().add(button);
        button.addActionListener((ae) -> {label.setText("Привет!");});
        frame.setSize(240, 120);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> { new SwingApp(); });
    }
}
```

УНИВЕРСИТЕТ ИТМО

Программирование. 2 семестр

# JavaFX

ITMO more than a UNIVERSITY

# JavaFX



- ☑ JavaFX — новая библиотека для разработки RIA (Rich Internet Applications)
- ☑ Поддержка XML для создания интерфейса
- ☑ Поддержка стилей CSS
- ☑ Поддержка 2D- и 3D-графики
- ☑ Легковесные компоненты
- ☑ Интеграция с библиотекой Swing

Java

# javafx.application



- ☑ javafx.application.Application — класс-предок всех приложений JavaFX
  - void init() — инициализация приложения (стартовый поток)
  - abstract void start(Stage s) — основной поток приложения
  - void stop() — освобождение ресурсов
  - public static void launch(String args) — запуск приложения

# javafx.stage



- ☑ javafx.stage.Stage — основная платформа
- ☑ Контейнер верхнего уровня (аналог JFrame)
- ☑ Предоставляется системой при запуске приложения
- ☑ Обеспечивает связь с графической подсистемой ОС
  - setTitle(String)
  - setScene(Scene)
  - show()

# javafx.scene



- ☑ javafc.scene.Scene — контейнер для элементов сцены
- ☑ Должен быть хотя бы один объект класса Scene
- ☑ Элементы сцены — узлы (Node)
- ☑ Узлы образуют граф (scene graph)
- ☑ Граф включает не только контейнеры и компоненты, но также графические примитивы (текст и графические примитивы)
- ☑ Узел с дочерними узлами — Parent (extends Node)
- ☑ Корневой узел (root node) — узел без родительского узла

- ☑ Scene sc = new Scene(*root node*,300,150);

# Hello World!



```
import javafx.application.*;
import javafx.stage.*;
import javafx.scene.*;
import javafx.scene.control.*;
import javafx.scene.layout.*;

public class Hello extends Application {
    public void start(Stage stage) {
        FlowPane fp = new FlowPane();
        fp.getChildren().add(new Label("Hello World!"));
        stage.setScene(new Scene(fp,100,200));
        stage.show();
    }
    public static void main(String... args) {
        launch(args);
    }
}
```

# Node

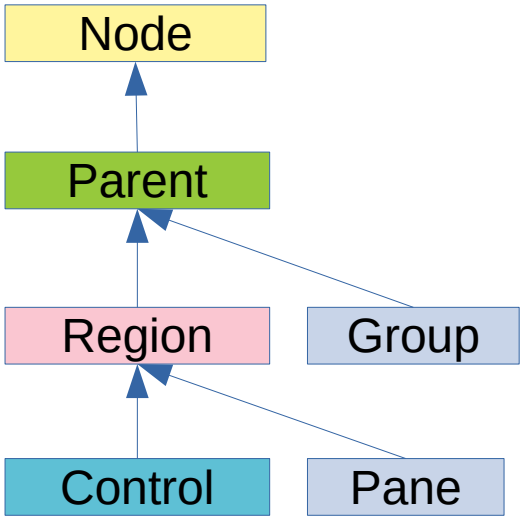


- ☑ Свойства (properties)
  - String id
  - Parent (только один)
  - Scene
  - Стиль (styleClass, style)
  - Видимость, активность, прозрачность
  - Размеры (min, max, preferred)
  - Границы (boundsInLocal, boundsInParent, layoutBounds)
  - Трансформации (сдвиг, вращение, масштаб, наклон)
  - Эффекты
  - События (mouse, key, drag, touch, rotate, scroll, swipe, zoom)

# Основные типы узлов JavaFX



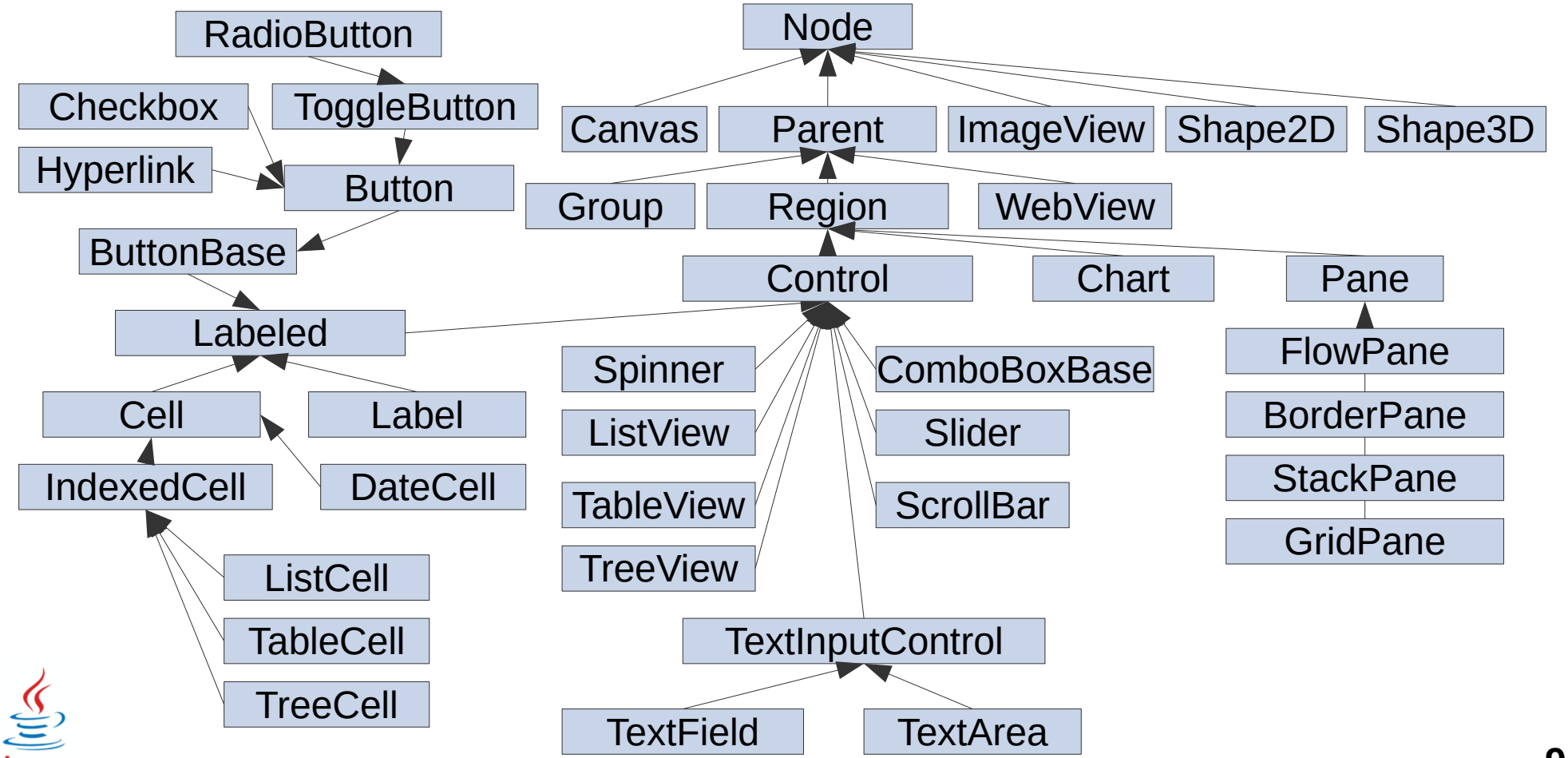
- ☑ Node - узел
- ☑ Parent - содержит другие узлы
- ☑ Region - имеет стиль
- ☑ Group - общие эффекты
- ☑ Control - управление пользователем
- ☑ Pane - панель с компоновкой




Java

# Основные компоненты JavaFX





# Контейнеры с компоновкой



- ☑ BorderPane — top, bottom, left, right, center
- ☑ HBox, VBox — в один ряд по горизонтали/вертикали
- ☑ StackPane — один над другим
- ☑ GridPane — сетка (таблица)
- ☑ FlowPane — последовательно с переносом
- ☑ TilePane — равномерные ячейки
- ☑ AnchorPane — привязка к границам родителя

# Обработка событий



- ☑ Событие: javafx.event.Event
  - ActionEvent extends Event
- ☑ Обработчик: javafx.event.EventHandler<T extends Event>
  - void handle(T event)
- ☑ Регистрация:
  - setOnAction(EventHandler<T>)

```
Label label = new Label();
Button button = new Button("Нажми меня");
button.setOnAction((ae) -> { label.setText("Спасибо");} )
```

# Обработка событий подробно



- ☑ class Event - событие:
  - source - источник в цепочке обработки, может меняться
  - target - цель события (interface EventTarget)
  - type - тип события (class EventType)
- ☑ Обработчики - addEventFilter, addEventHandler
- ☑ Фаза перехвата (capturing) - сверху вниз - EventFilters
- ☑ Фаза всплытия (bubbling) - снизу вверх - EventHandlers
- ☑ метод consume() - остановка события

# FXML



```
FlowPane fp = new FlowPane();
fp.getChildren().add(new Label("Hello World!"));
stage.setScene(new Scene(fp,100,200));

FXMLLoader loader = new FXMLLoader();
loader.setLocation("file.xml");
FlowPane fp = loader.<FlowPane>load();
stage.setScene(new Scene(fp,100,200));
```

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.layout.FlowPane?>
<?import javafx.scene.control.Label?>
<FlowPane>
   <children>
      <Label text="Hello World!"/>
   </children>
</FlowPane>
```

# CSS



```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.layout.FlowPane?>
<?import javafx.scene.control.Label?>
<FlowPane>
   <children>
      <Label text="Hello World!">
         <style>
           -fx-padding: 10px;
           -fx-background: rgb(255,127,255);
         </style>
   </children>
</FlowPane>
```

```
label.setStyle("-fx-background-color: #ff77ff");
label.setStyle("-fx-padding: 10px");
```

# Анимация FX



- ☑ javafx.animation
  - Timeline - KeyFrame... frames
    - setCycleCount()
    - play()
  - KeyFrame - Duration time, KeyValue... values
  - KeyValue - WritableValue<T> target, T endValue

# Приложение JavaFX



```
import javafx.application.*;
import javafx.event.*;
import javafx.geometry.Pos;
import javafx.scene.control.*;
import javafx.scene.*;
import javafx.stage.*;
import javafx.scene.layout.*;
import javafx.scene.effect.*;

public class FXApp extends Application {
    public void start(Stage stage) {
        stage.setTitle("Hello");
        FlowPane root = new FlowPane();
        Label label = new Label();
        Button button = new Button("OK");
        root.getChildren().add(label);
        root.getChildren().add(button);
        button.setOnAction((ae) -> label.setText("Привет!"));
        stage.setScene(new Scene(root,240,120));
        stage.show();
    }
    public static void main(String... args) {
        launch(args);
    }
}
```