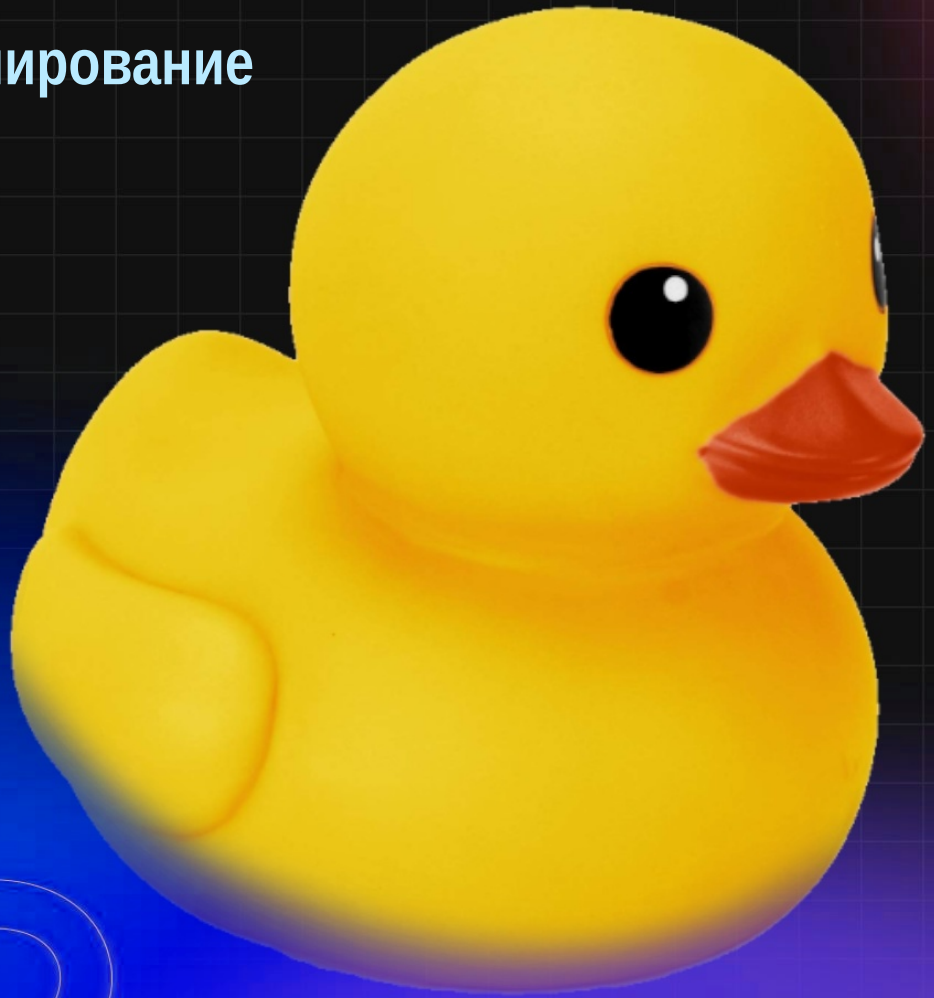
Программирование
2 семестр
2025

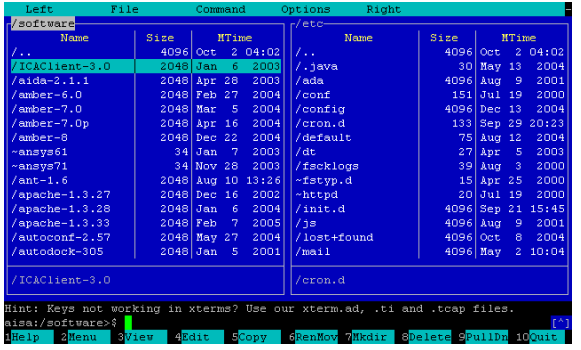


ИТМО

Графический интерфейс

UI - Пользовательский интерфейс

- Пользовательский интерфейс (Human Machine Interface)
  - средство взаимодействия пользователя и компьютера
- Текстовый интерфейс
  - CLI - Command Line Interface
  - Интерактивные интерфейсы
  - Полноэкранный текстовый интерфейс
- GUI (Graphical User Interface)



GUI - Graphical User Interface

ИТМО

- WIMP
  - Window
  - Icon
  - Menu
  - Pointer
- Touch
- Multitouch
- VR/AR, 3D, ...

Java

UX/UI - User Experience / User Interface

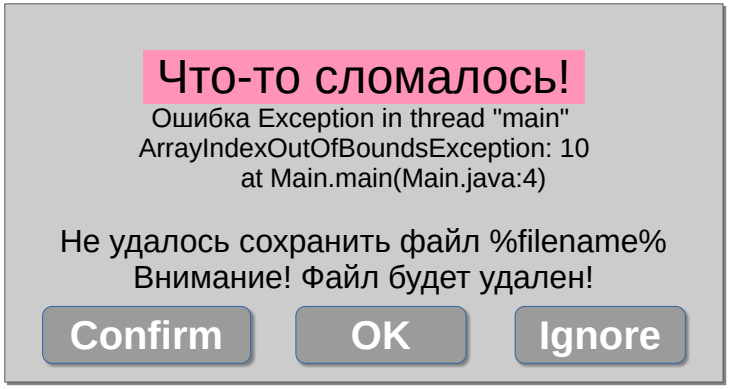
ИТМО

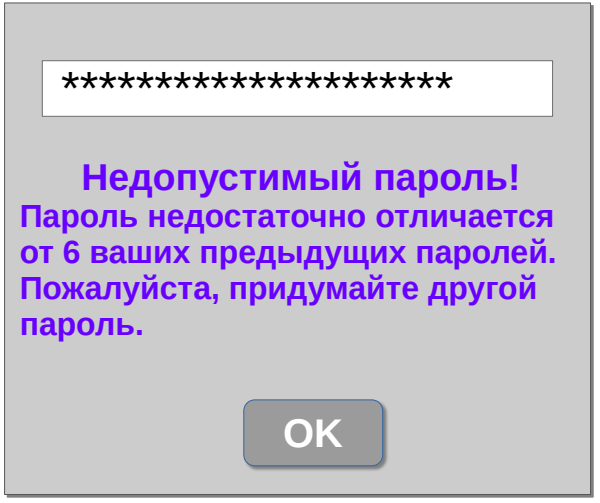
- Хороший интерфейс:
  - удобство
  - понятность
  - обратимость
  - управляемость
  - согласованность
  - исключение ошибок
  - эстетика
  - минимализм

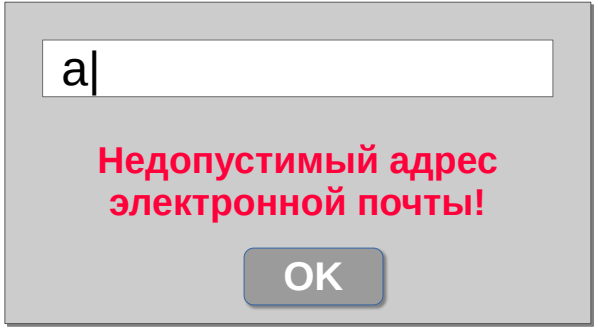
Java

Usability

- Хороший интерфейс:
  - удобство
  - понятность
  - обратимость
  - управляемость
  - согласованность
  - исключение ошибок
  - эстетика
  - минимализм



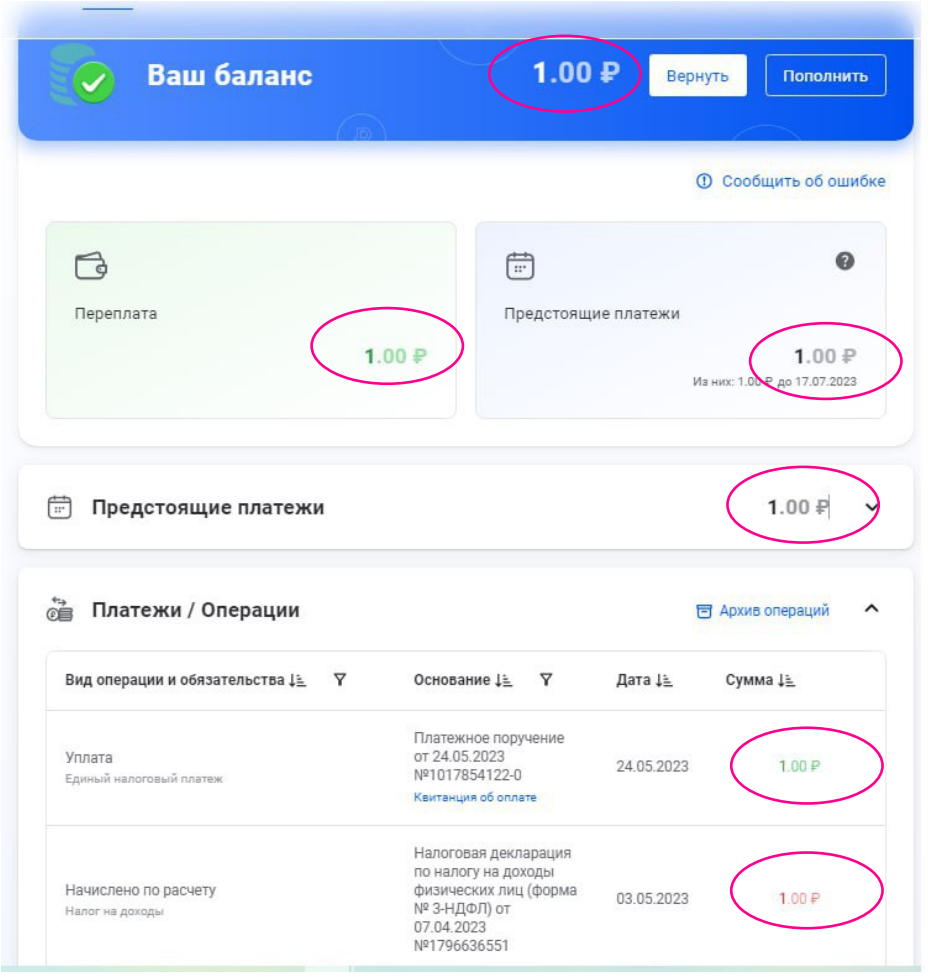




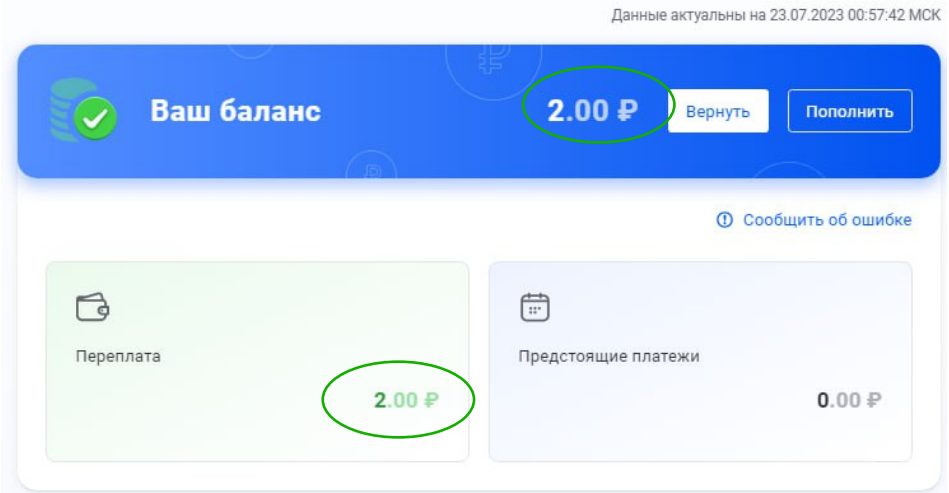
UX/UI -

- https://userinyerface.com/

ЛК ФНС - баланс



ЛК ФНС - баланс



ЛК ФНС - вызов

Уведомление № 6300
о вызове в налоговый орган налогоплательщика (плательщика сбора, плательщика страховых взносов, налогового агента)

27.06.2023
(дата)

Межрайонная инспекция Федеральной налоговой службы №19 по Санкт-Петербургу
(наименование налогового органа)

в соответствии с подпунктом 4 пункта 1 статьи 31 Налогового кодекса Российской Федерации (далее - Кодекс) вызывает на основании настоящего уведомления в налоговый орган, находящийся по адресу:

,198216,Санкт-Петербург г.,,,Трамвайный пр-кт,23,1, , кабинет № 205
(адрес и номер кабинета)

17.07.2023 с 14:00 до 16:00
(дата и время или приемные дни и часы)

в связи с налоговой проверкой

Налоговая декларация по налогу на доходы физических лиц (форма № 3-НДФЛ) (первичная декларация)
(наименование налоговой декларации (расчета, заявления))

07.04.2023
(дата представления / день истечения срока представления)

за год,2022
(налоговый (расчетный, отчетный) период)

для дачи пояснений по вопросу:

иное:

по налоговой декларации по форме 3-НДФЛ за 2022 год для вручения Акта налоговой проверки.

ЛК ФНС - вызов

ИТМО

В случае не явки, Акт налоговой проверки, в соответствии с п.5 ст.100 Налогового Кодекса Российской Федерации будет направлен в Ваш адрес почтовым отправлением.

(подробное описание оснований для вызова налогоплательщика
(плательщика сбора, плательщика страховых взносов, налогового агента)

Java

ЛК ФНС - акт

3. Итоговая часть (выводы и предложения проверяющих).

3.1. По результатам налоговой проверки проверяющими установлена неуплата следующих налогов, сборов, страховых взносов (недоимка), предлагается привлечь к ответственности за совершение налогового правонарушения:

ЛК ФНС - акт

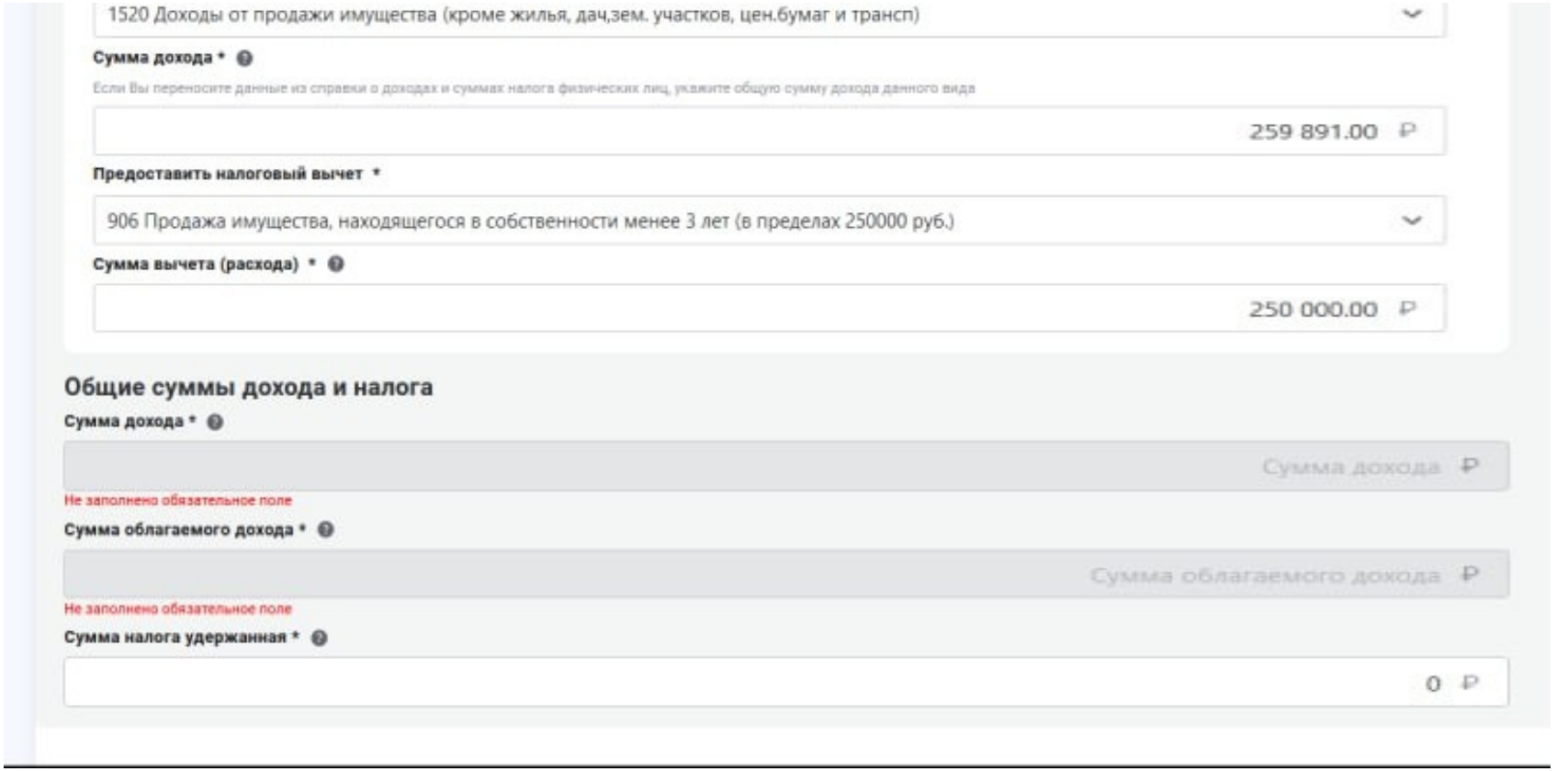
ИТМО

3. Итоговая часть (выводы и предложения проверяющих).

3.1. По результатам налоговой проверки проверяющими установлена неуплата следующих налогов, сборов, страховых взносов (недоимка), предлагается привлечь к ответственности за совершение налогового правонарушения:

№	Пункт (при наличии) и статья Налогового кодекса Российской Федерации	Состав налогового правонарушения	Налог (сбор, страховые взносы)	Налоговый (расчетный, отчетный) период	Срок уплаты налога, сбора, страховых взносов, установленный законодательством о налогах и сборах, установленный закнодательством о налогах и сборах	Недоимка, рублей	Код бюджетной классификации	Код по ОКТМО	ИНН	КПП	Код налогового органа
1	2	3	4	5	6	7	8	9	10	11	12
					Итого:	0					

A-a-a-a! Javascript не заполнил поля...



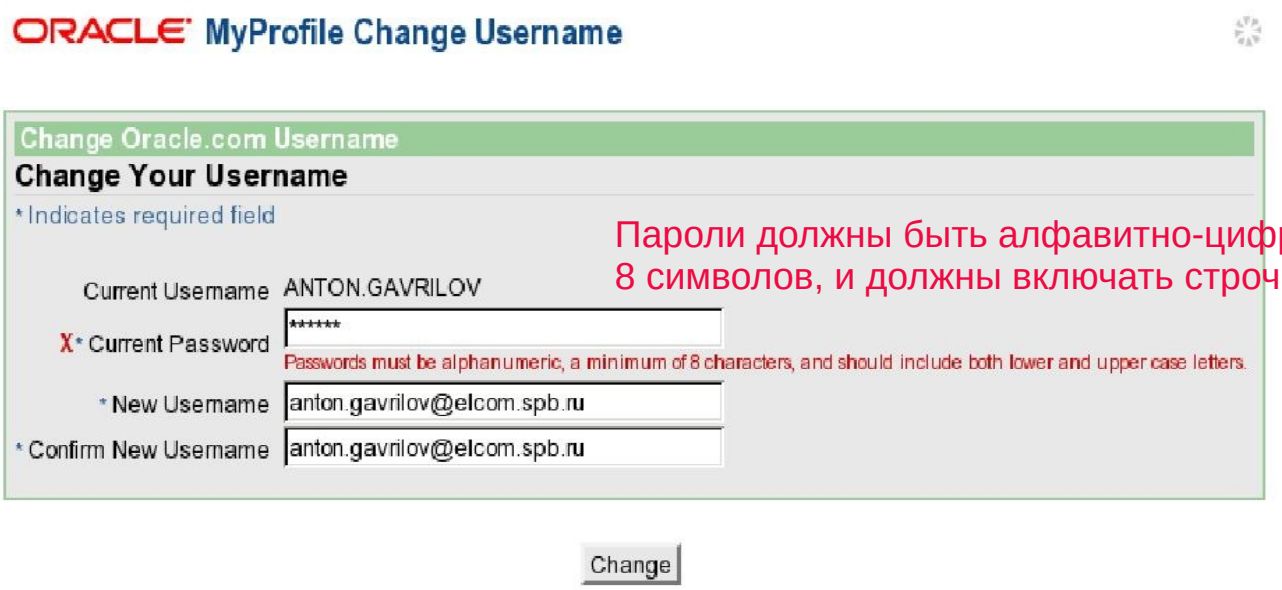
Техподдержка Oracle, 2010 год

ИТМО

- Внезапно перестал работать один из ваших сервисов.
  Я захожу через SSO, все работает, кроме одного сайта.
  - Это проблема с вашим аккаунтом!
- С моим? Это же SSO, если бы проблема была в моем аккаунте, тогда бы не работало все...
  - С вашим! Вот видите, у вас неправильный логин.
    Он не совпадает с адресом email! Год назад введены новые требования к логинам.
- Но этот логин у меня уже 3 года, и все это время проблем не было.
  - Логин неправильный! Поменяйте логин через форму смены логина.

Java

Техподдержка Oracle, 2010 год



Пароли должны быть алфавитно-цифровыми, минимум из 8 символов, и должны включать строчные и заглавные буквы.

- Я не могу поменять логин, система почему-то ругается на пароль, пишет, что он должен быть не менее 8 символов!
  - Да. это тоже новое требование, поменяйте пароль через форму смены пароля.

С таким паролем нельзя менять пароль!

ИТМО

ORACLE MyProfile Change Password

Change Oracle.com Password

Change Your Password

* Indicates required field

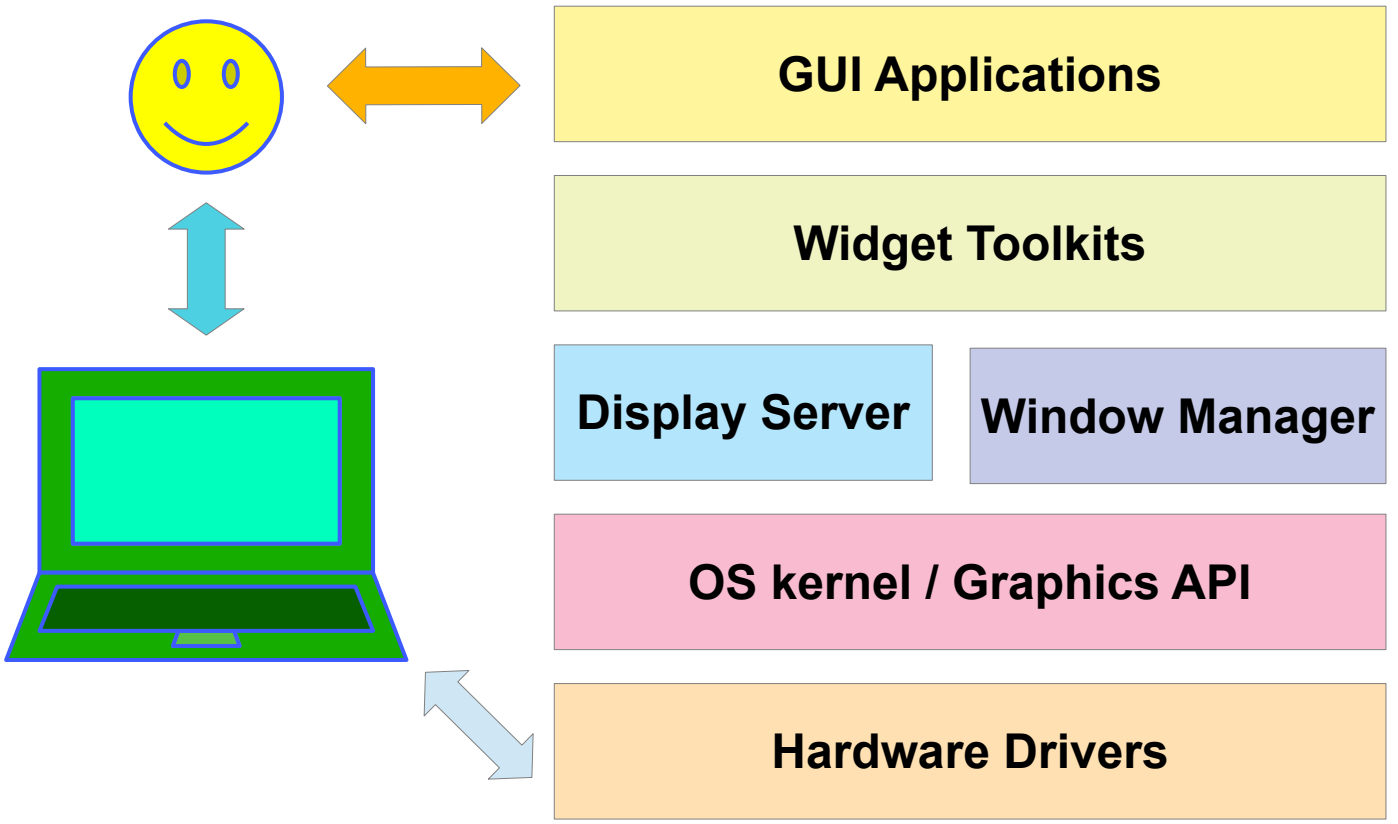
Пароли должны быть алфавитно-цифровыми, минимум из 8 символов, и должны включать строчные и заглавные буквы.

Username	ANTON.GAVRILOV
X * Current Password	******
	Passwords must be alphanumeric, a minimum of 8 characters, and should include both lower and upper case letters.
* New Password	********
* Confirm New Password	********

Change

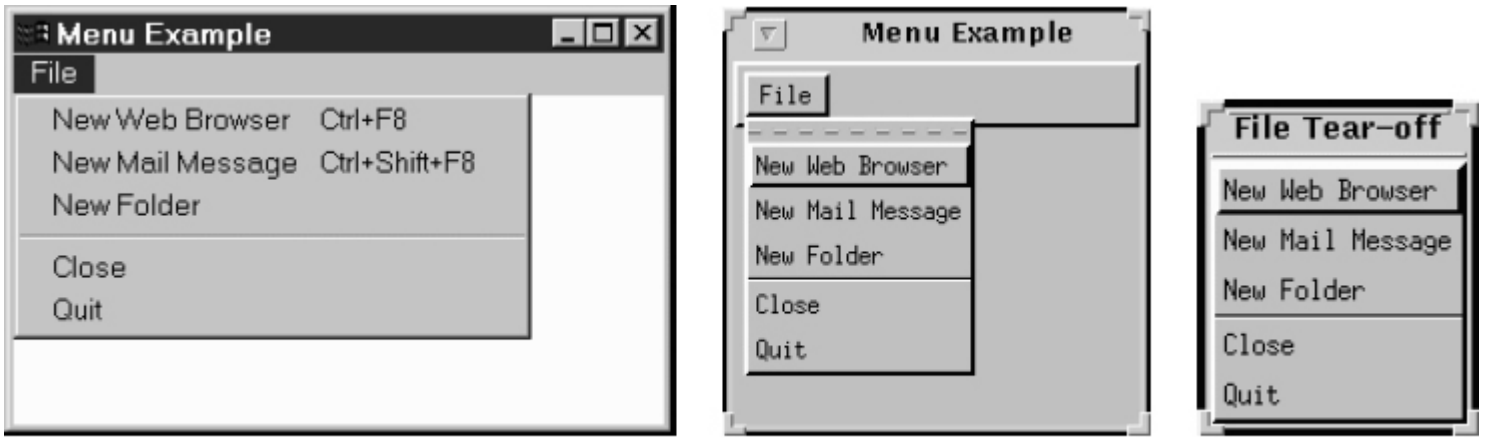
Java

Архитектура



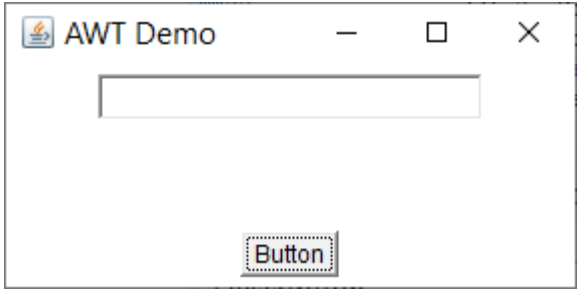
AWT

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



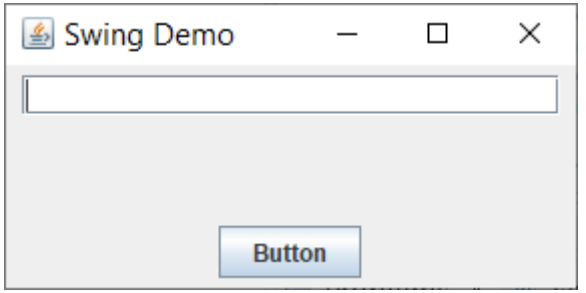
AWT

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



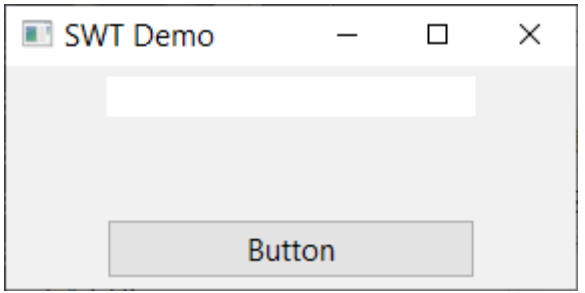
Swing

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



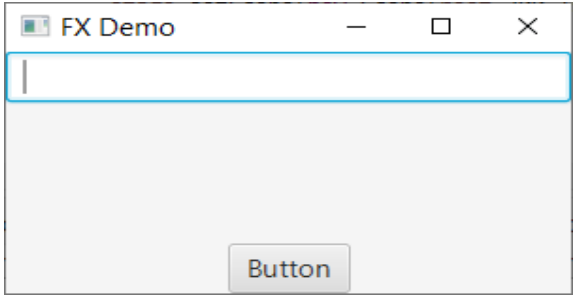
SWT

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



JavaFX

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



Создание графических приложений

ИТМО

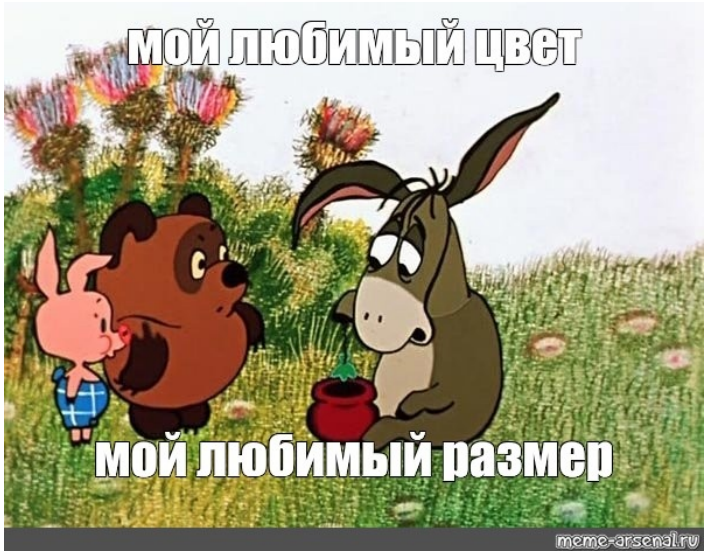
1) Создание основного окна
2) Создание остальных элементов интерфейса
3) Размещение элементов интерфейса
4) Обеспечение реакции на события
5) Все заработало!

Java

Класс java.awt.Component

ИТМО

- 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

ИТМО

- 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
    - get**Red**(), get**Green**(), get**Blue**(), 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

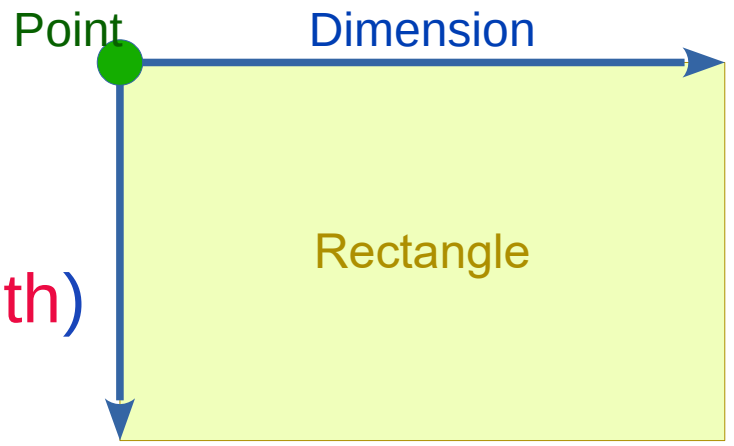
- Цвет текста и цвет фона

```
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)



Методы класса Component

ИТМО

- Положение и размеры

```
Rectangle getBounds()
void setBounds(Rectangle)

Point getLocation()
void setLocation(Point)

Dimension getSize()
void setSize(Dimension)
```

Java

Шрифт

ИТМО

- Класс Font
  - физические (Arial, Times, Courier)
  - 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()

Java

Методы класса Component

ИТМО

- Шрифт

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

Java

Методы класса Component

ИТМО

- 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)

Java

Методы класса Component

ИТМО

- 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

ИТМО

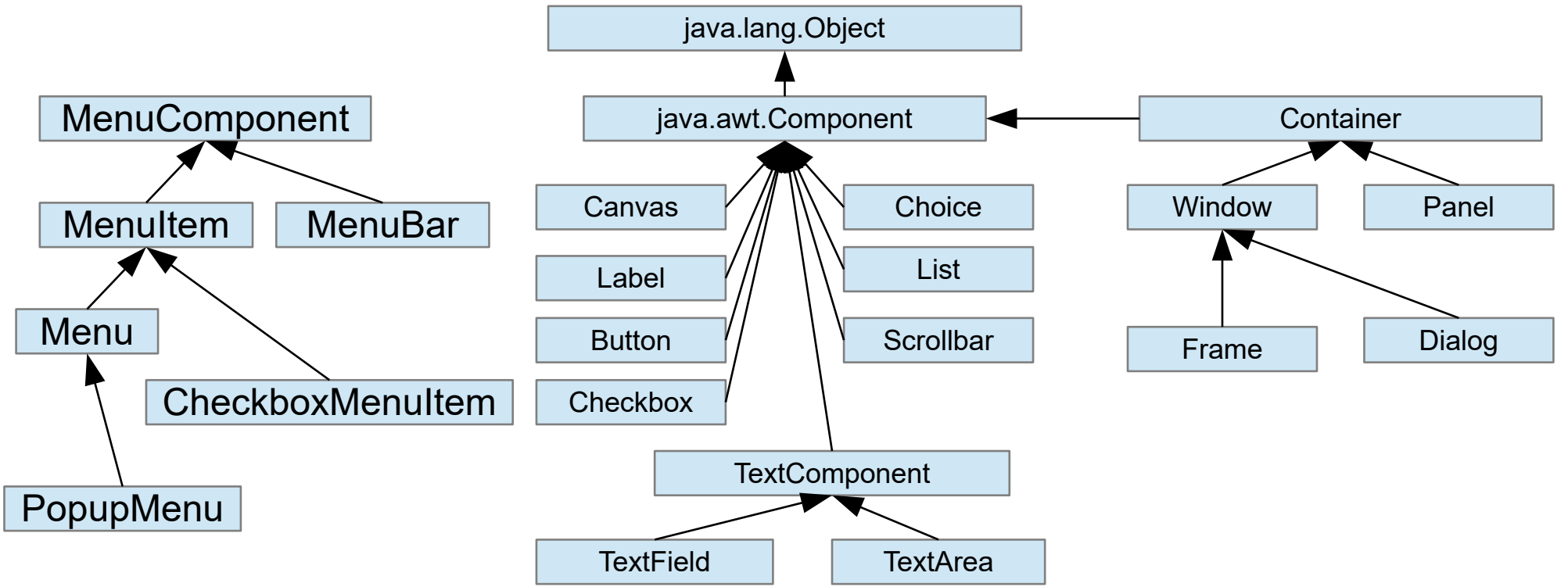
- Контейнер — компонент, который содержит другие компоненты

```
class Container extends 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

ИТМО

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

Java

Менеджер компоновки

ИТМО

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

Java

Менеджер компоновки

ИТМО

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

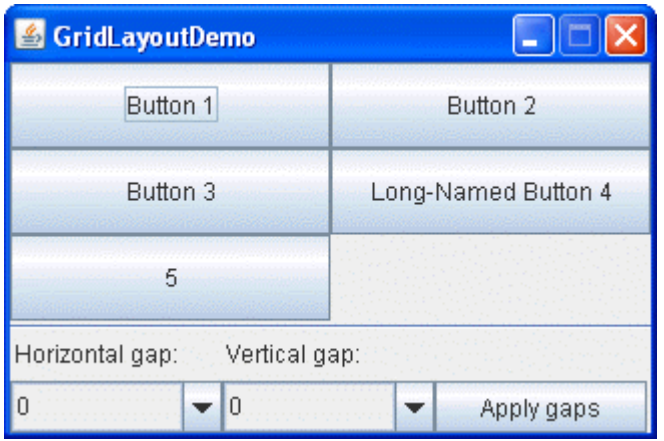
Java

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

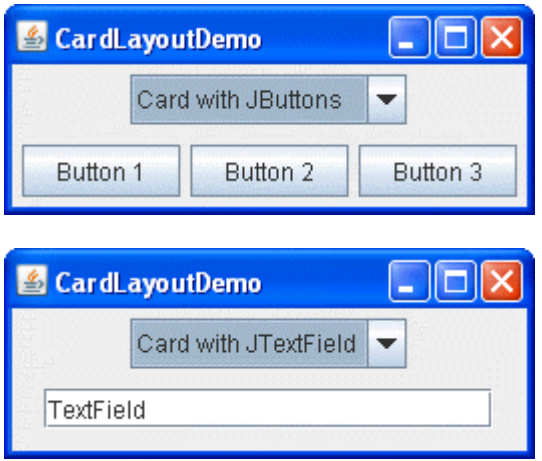
- Контейнер делится на одинаковые ячейки по строкам и столбцам
- Все компоненты будут одного размера
- GridLayout(int rows, int cols)
- Управление размещением:
  - `setHgap(int), setVgap(int) // 0`
  - `setRows(int), setColumns(int) // 1, 0`



CardLayout

ИТМО

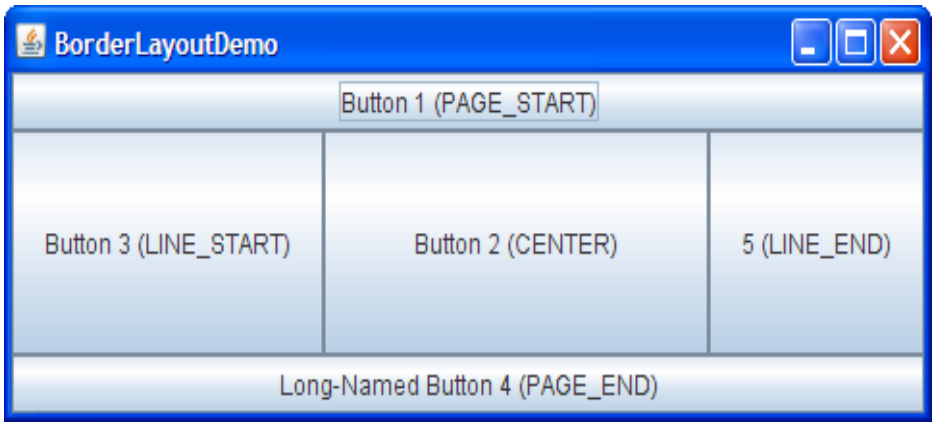
- Аналог колоды карт (виден только верхний компонент)
- Позволяет выбрать одну из панелей
- Базовый аналог вкладок
- Переключение между картами нужно реализовывать отдельно

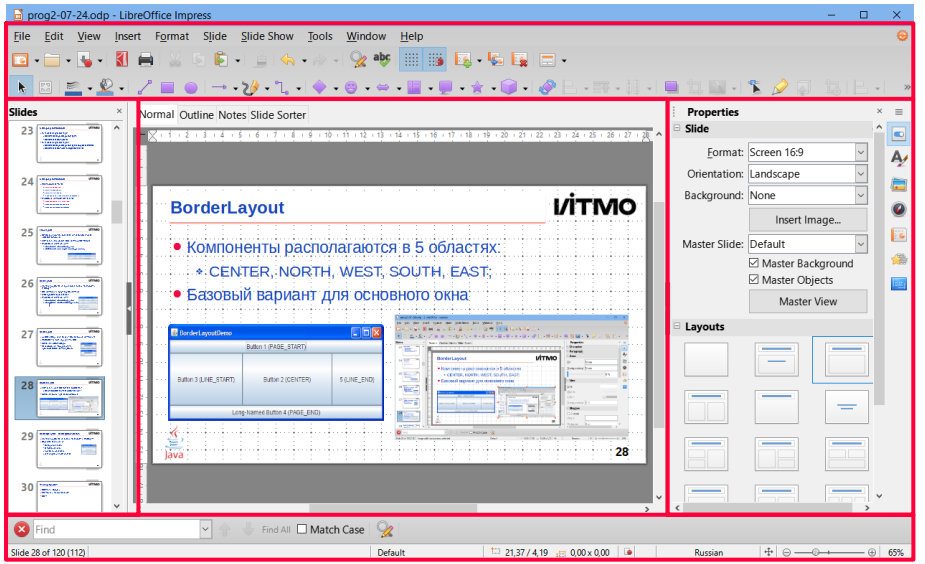


Java

BorderLayout

- Компоненты располагаются в 5 областях:
  - CENTER, NORTH, WEST, SOUTH, EAST;
- Базовый вариант для основного окна

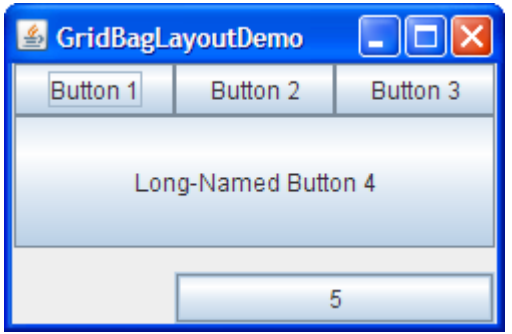




GridBagLayout + GridBagConstraints

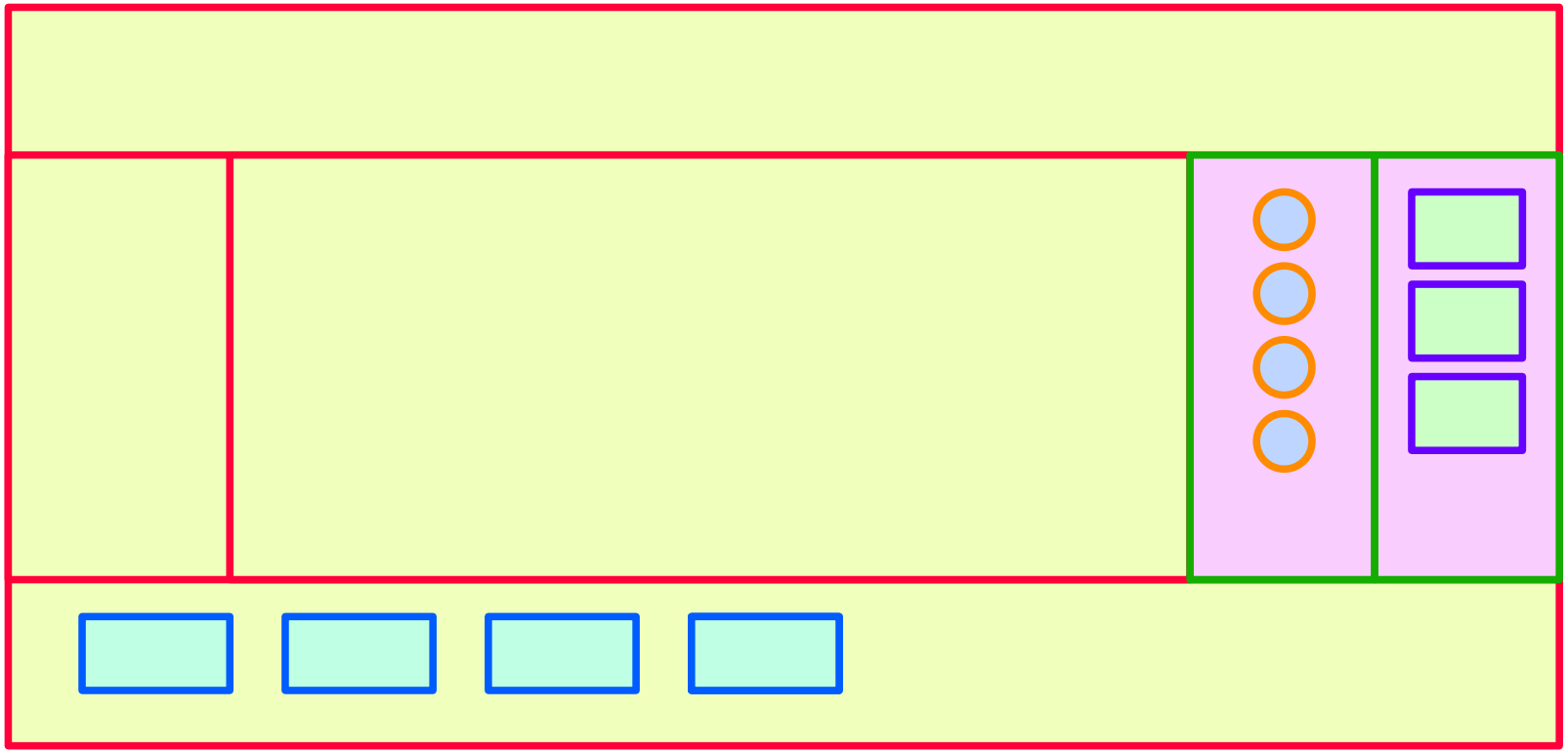
ИТМО

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



Java

Составные варианты

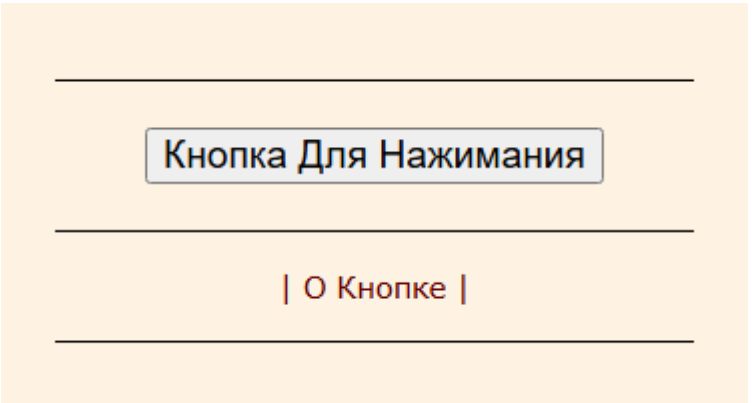


Что еще нужно?

- Компоненты созданы
- Компоненты размещены
- Все?

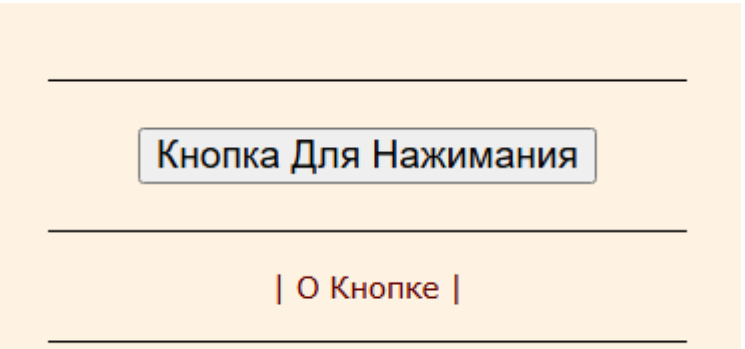
Что еще нужно?

- Компоненты созданы
- Компоненты размещены
- Все?



Что еще нужно?

- Компоненты созданы
- Компоненты размещены
- Все?



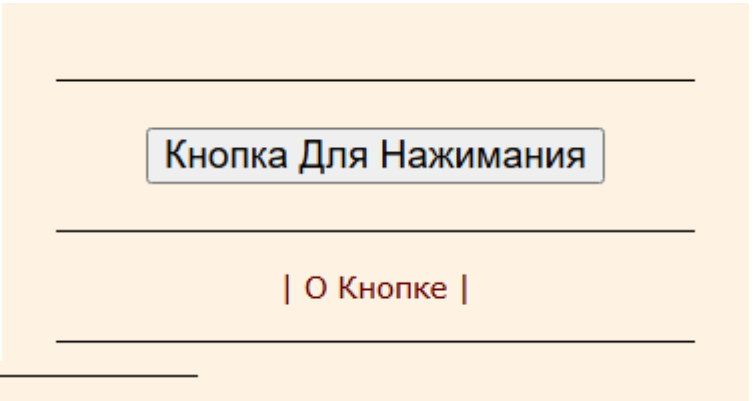
Кнопка Для Нажимания

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.

Борис Юшманов, Мастер Нажимания на Кнопку

Что еще нужно?

- Компоненты созданы
- Компоненты размещены
- Все?



Кнопка

- Кнопка Для Нажимания не предназначена для развлечения, но она не предназначена и для созерцания - она не предназначена ни для чего, она такова каков Нажимающий.
- Кнопка Для Нажимания не является буддийской, христианской, мусульманской или зароострийской - она вне традиции, вне пространства, вне времени.
- Позади Кнопки находится Пустота. Сама Кнопка суть Пустота. Тот, кто поймет, что и впреди Кнопки Пустота - достиг просветления.

Обработка событий

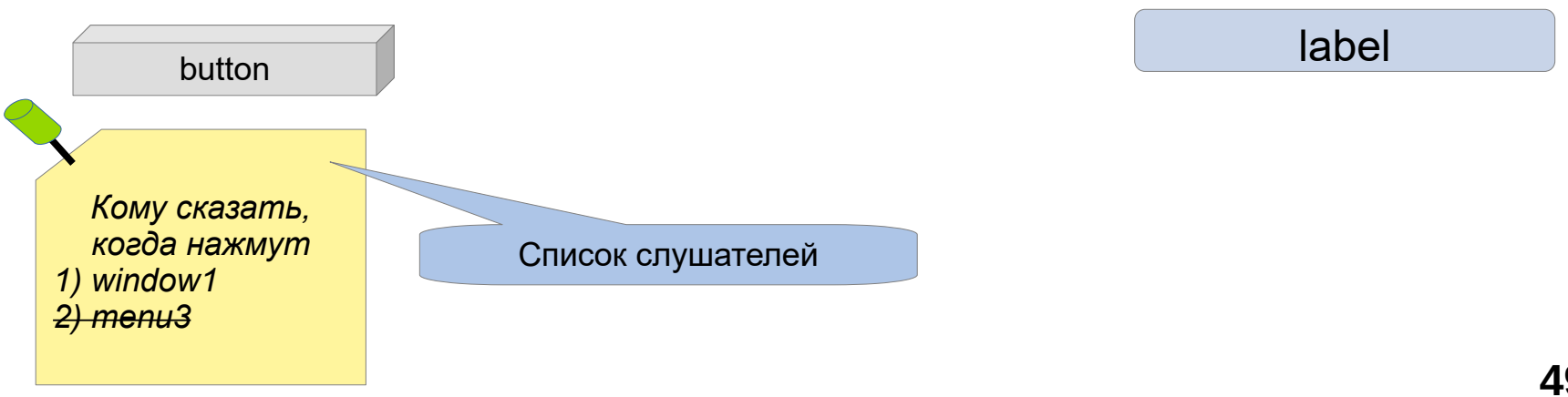
ИТМО

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

Java

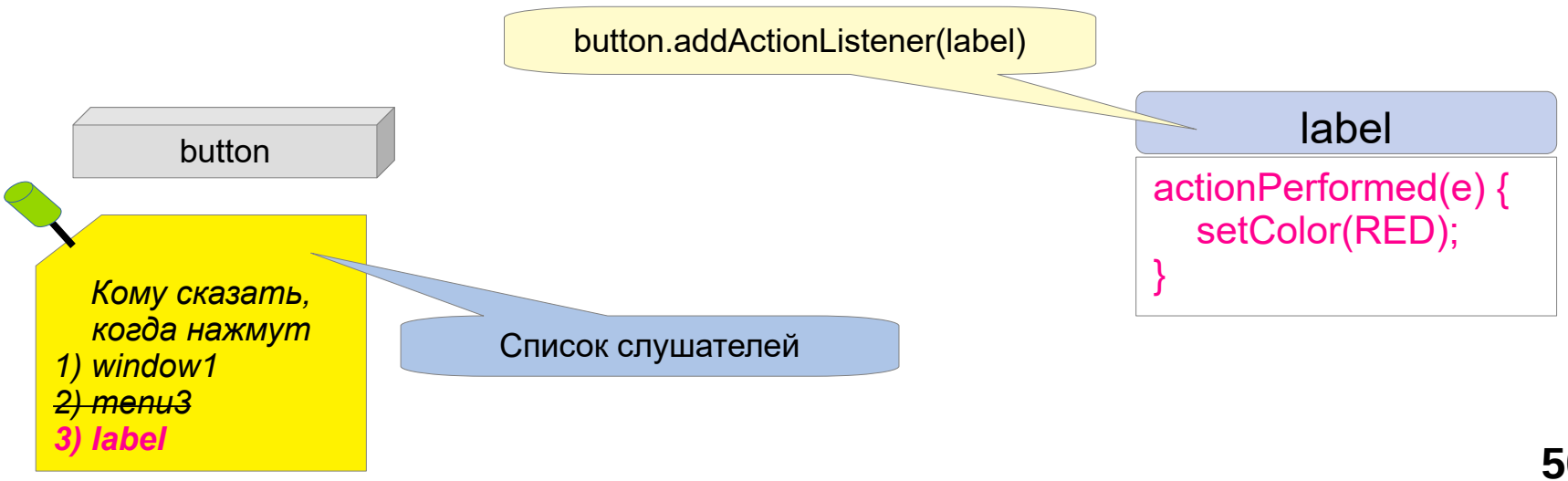
Обработка событий

- button - кнопка
- label - метка
- Хочется при нажатии на кнопку сделать метку красной



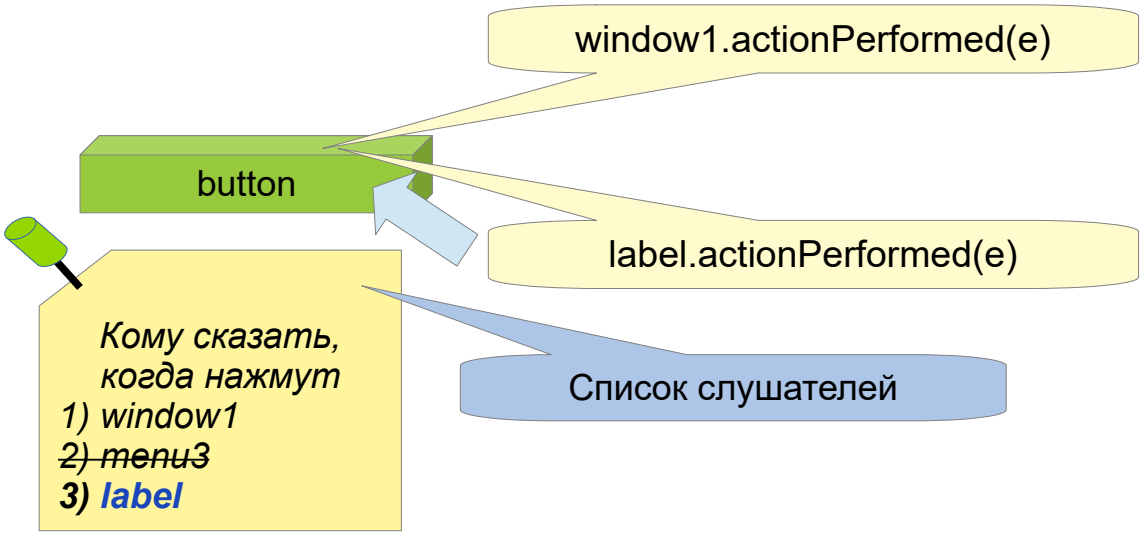
Обработка событий

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



Обработка событий

- Нажимаем кнопку!
- У всех слушателей (ActionListener-ов) из списка вызываем метод actionPerformed и передаем в него событие ActionEvent e

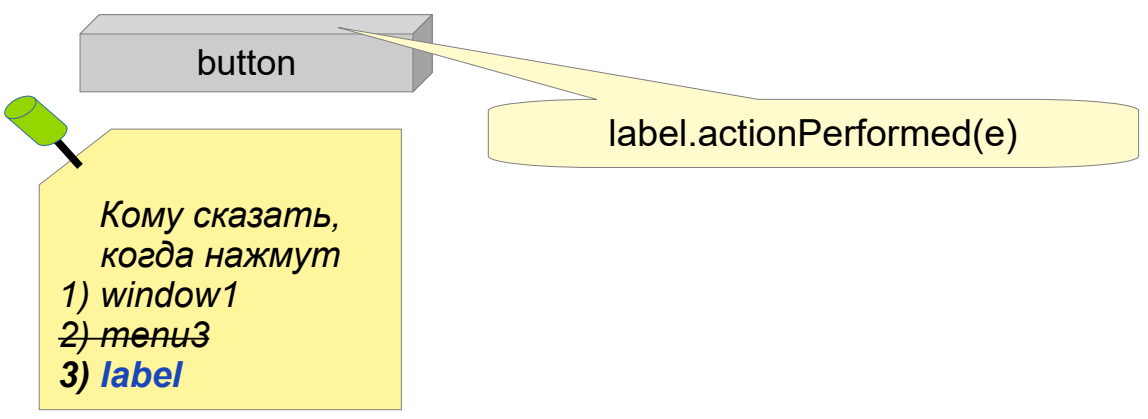


label

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

Обработка событий

- Среди слушателей есть метка
- Вызываем и у нее actionPerformed
- Метка перекрашивается в красный
- Из аргумента-события можно получить больше информации



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

ИТМО

- Анонимным классом

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

- Лямбда-выражением

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

Java

Mouse(Motion)Listener, MouseEvent

ITMO

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

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

Классы-адаптеры

ИТМО

```
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) { ... }
}
```

Java

MouseWheelListener, MouseWheelEvent

ИТМО

- MouseWheelListener
  - mouseWheelMoved(MouseEvent)

- MouseWheelEvent
  - getWheelRotation()

Java

KeyListener, KeyEvent

ИТМО

- 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()

ActionListener, ActionEvent

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

AdjustmentListener, AdjustmentEvent

ИТМО

- AdjustmentListener
  - adjustmentValueChanged(AdjustmentEvent)
- AdjustmentEvent
  - int getValue()
  - boolean getValueIsAdjusting()
  - перемещение слайдера

Java

ItemListener, ItemEvent

ИТМО

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

Java

TextListener, TextEvent

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

Свойства и JavaBeans

ИТМО

- Java Beans (это еще не Enterprise Java Beans)
  - спецификация JavaBeans
  - пакет java.beans.*
  - публичный конструктор без аргументов
  - приватные свойства + геттеры и сеттеры
- Компоненты Swing - Java Beans

Java

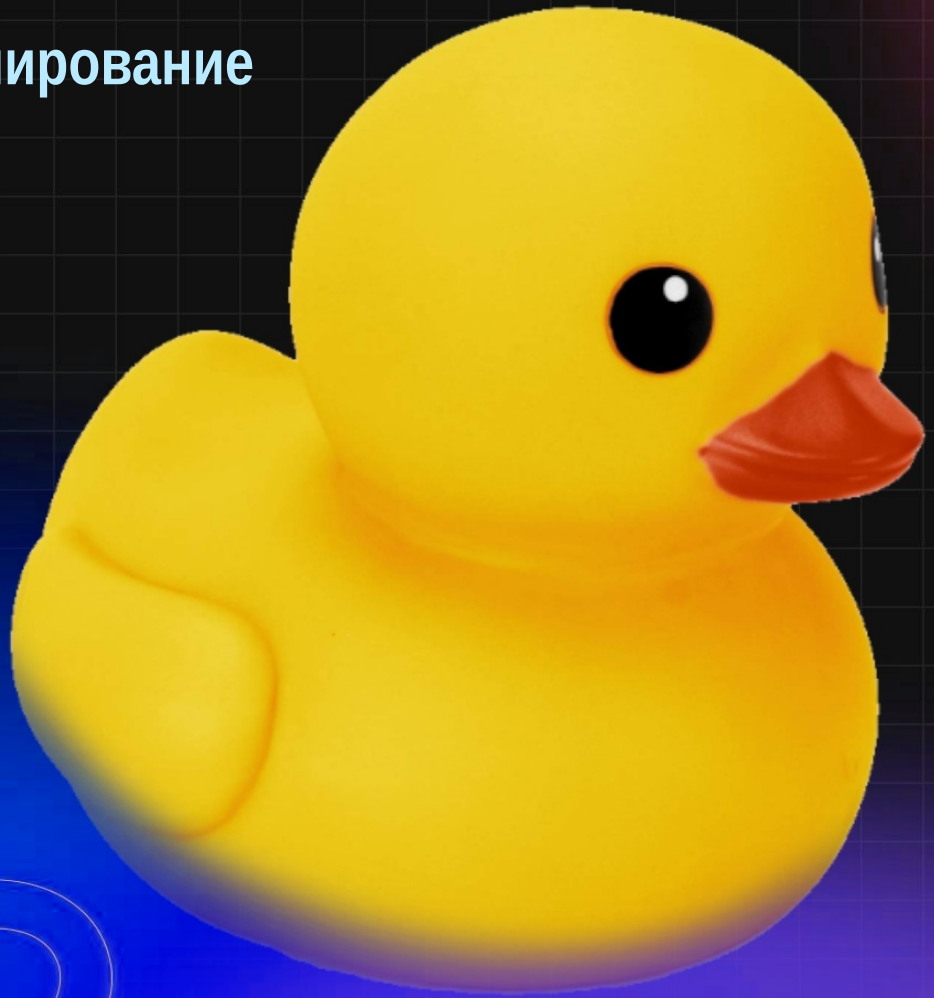
PropertyChangeListener

ИТМО

- Свойства и JavaBeans
  - спецификация JavaBeans (не Enterprise Java Beans)
    - конструктор по умолчанию
    - приватные свойства с геттером и сеттером
- Связанные свойства
  - java.beans.PropertyChangeListener
  - java.beans.PropertyChangeEvent
  - java.beans.PropertyChangeSupport
- Свойства с ограничениями - VetoableChangeListener

Java

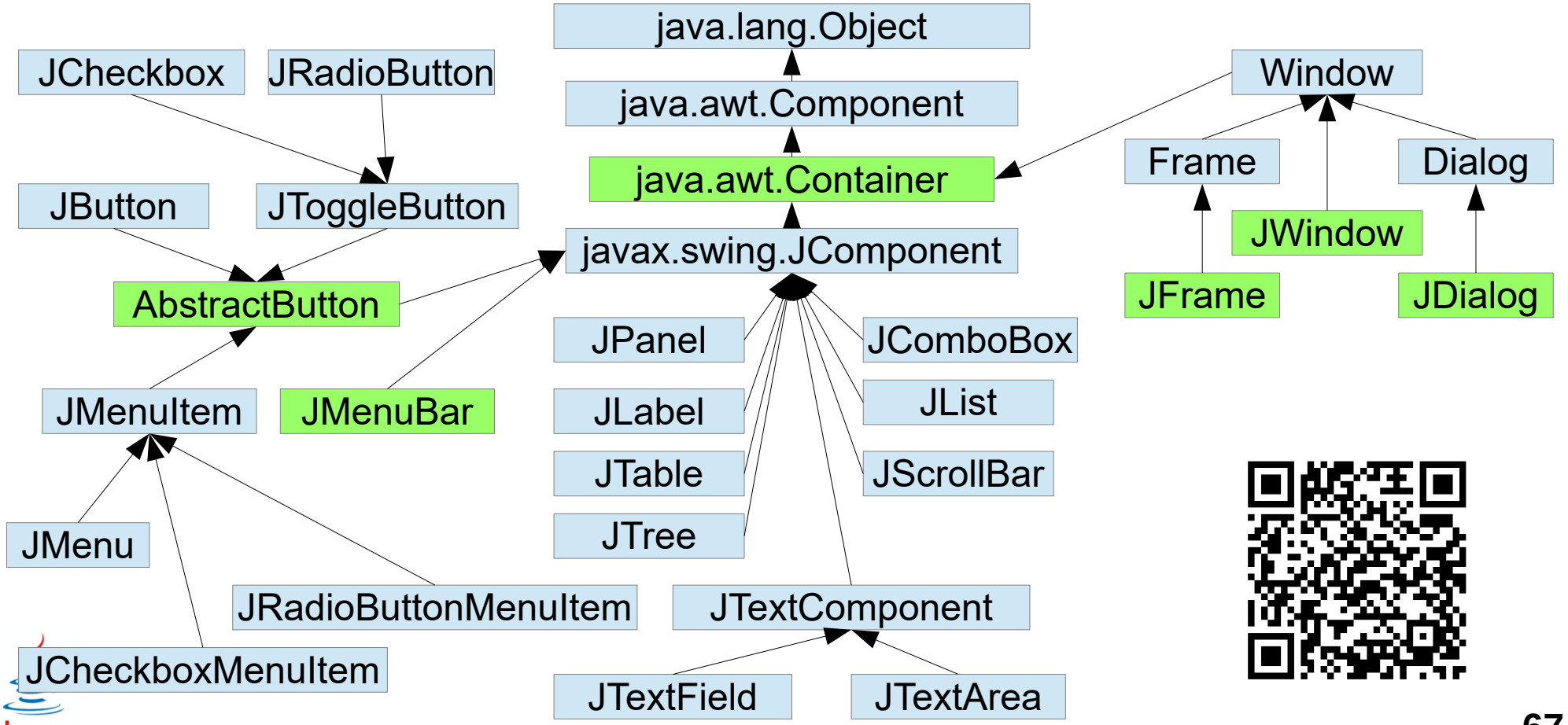
Программирование
2 семестр
2025



ИТМО

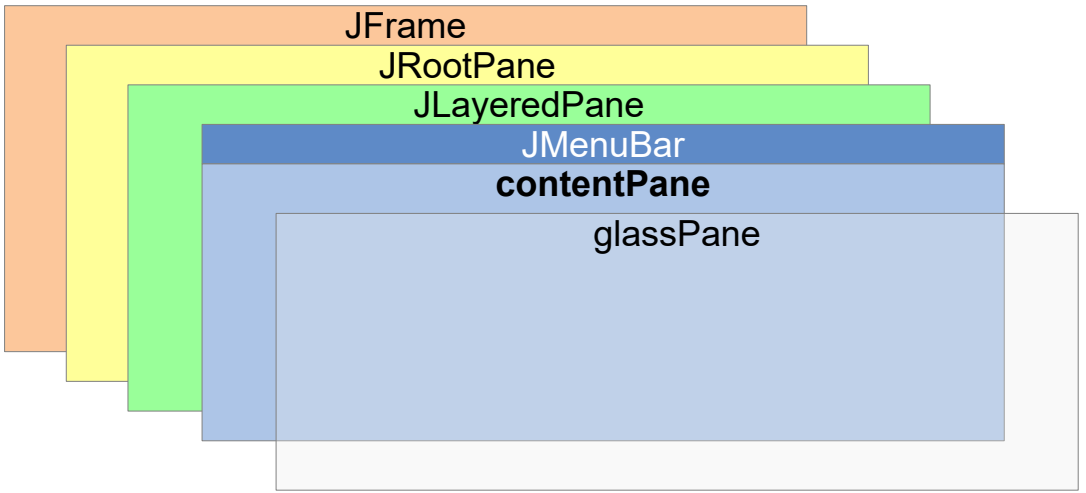
Java Swing

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()



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

ИТМО

- Основные потоки (initial)
  - для кода основного приложения (Main thread)
  - запуск создания элементов интерфейса в EDT
- Поток обработки событий (EDT - Event Dispatch Thread)
  - для кода обработки событий
  - все действия с элементами интерфейса здесь!
  - действия должны выполняться быстро
- Фоновые потоки (Worker threads)
  - для долгих задач
  - запускаются с помощью SwingWorker

Организация приложения Swing

ИТМО

- .invokeLater(Runnable) – асинхронный запуск
  - помещает задачу в очередь EDT

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

Java

SwingWorker<T, V>

ИТМО

- 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

ИТМО

- 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

ИТМО

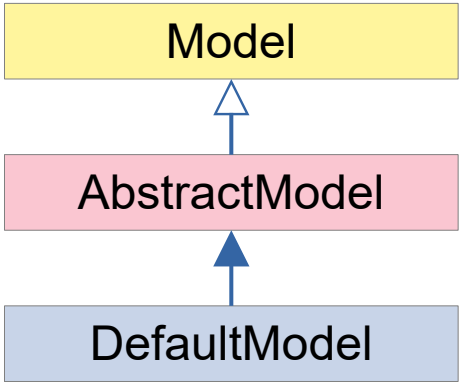
- MVC — Model, View, Controller
  - **Модель** отвечает за поведение
  - **Представление** — отвечает за отображение
  - **Контроллер** — связывает модель и представление и управляет ими
- Реализация Swing — Model + UI Delegate
  - UI Delegate = View + Controller
  - Модель может быть визуальной или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

Java

Модели

ИТМО

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



Java

Делегаты

ИТМО

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

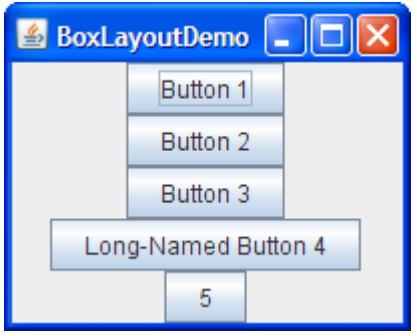
```
UIManager.setLookAndFeel(
   UIManager.getSystemLookAndFeelClassName());
SwingUtilities.updateComponentTreeUI(frame);
frame.pack();
```

Java

Менеджеры компоновки Swing

ИТМО

- 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

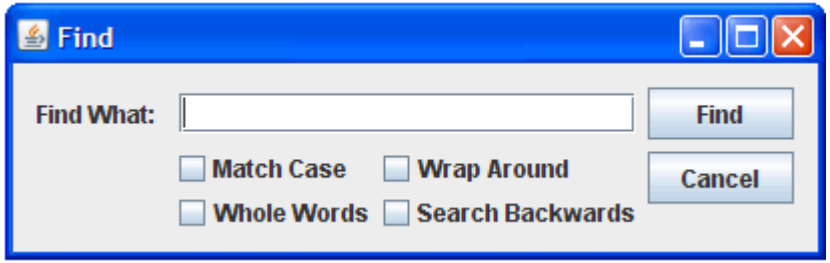


Java

Менеджеры компоновки Swing

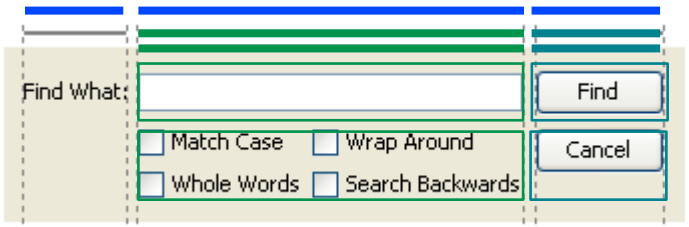
ИТМО

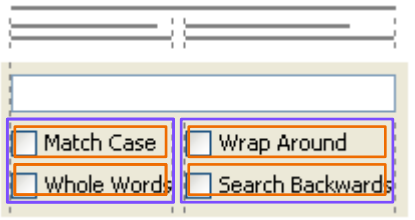
- GroupLayout
  - Все компоненты описываются дважды — горизонтальное расположение и вертикальное расположение
  - Все компоненты являются участниками групп — последовательных и параллельных



Java

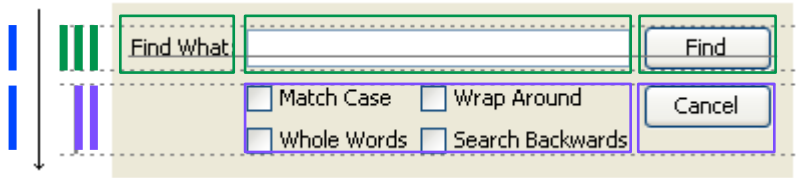
GroupLayout






```
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

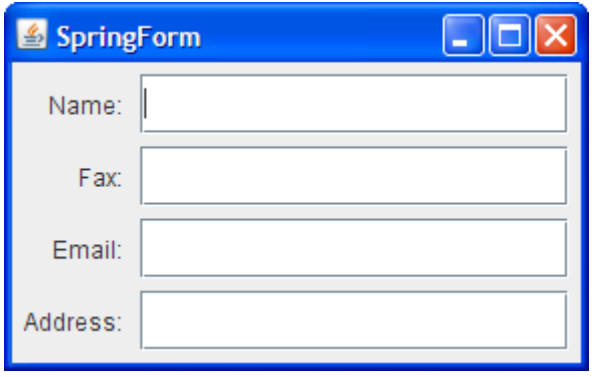


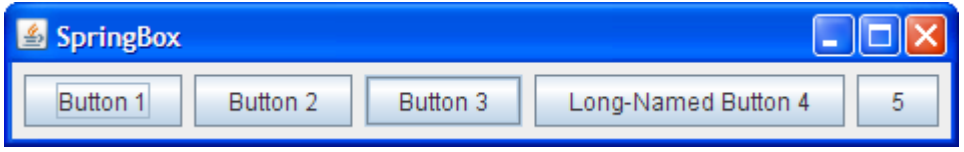

```
layout.setVerticalGroup(layout.createSequentialGroup()
    .addGroup(layout.createParallelGroup(Alignment.BASELINE)
        .addComponent(label)
        .addComponent(textField)
        .addComponent(findButton))
    .addGroup(layout.createParallelGroup(Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()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

Менеджеры компоновки Swing

ИТМО

- 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

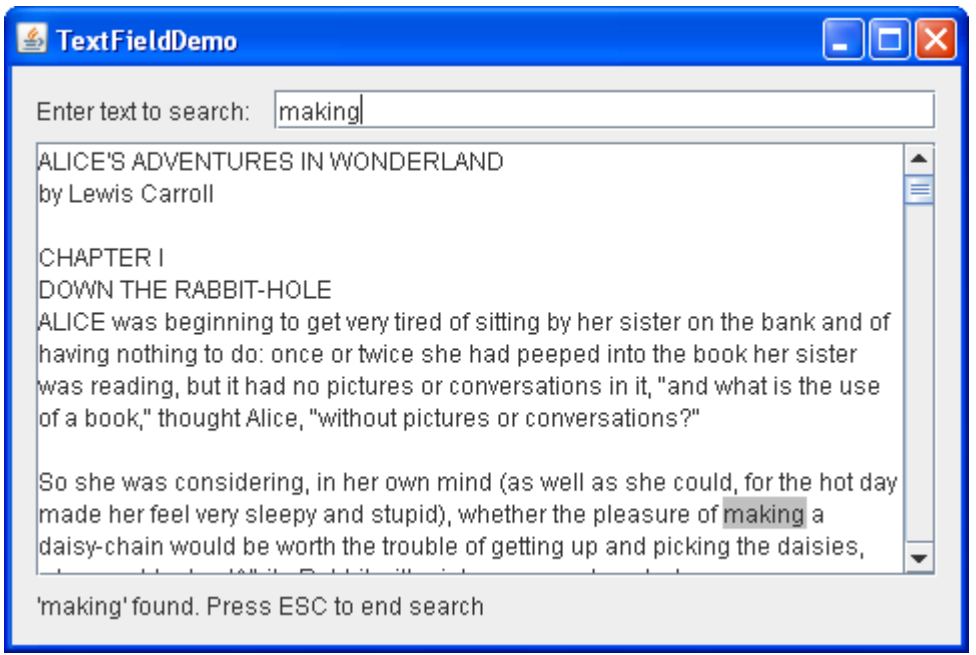
ИТМО

- JTextField — однострочное поле
  - JFormattedTextField — возможность проверки ввода
  - JPasswordField — не отображает введенные символы
  - Основное событие — ActionEvent

Java

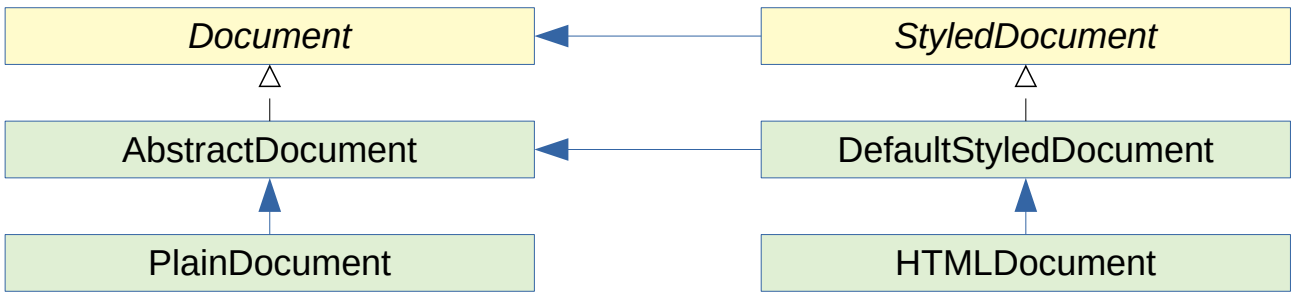
Простые текстовые элементы

- JTextArea — многострочное поле
  - События — ActionEvent, UndoableEditEvent



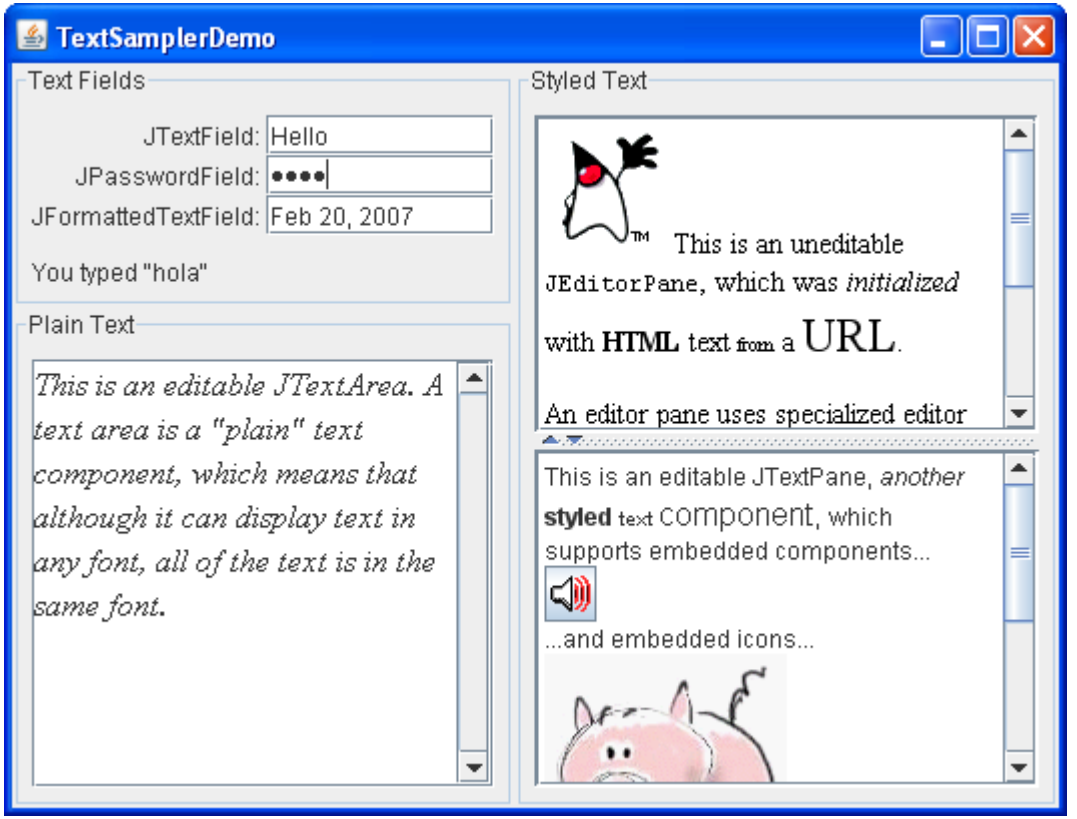
Текстовые модели

- Модель для всех текстовых компонентов — Document



Расширенные редакторы

- EditorPane — панель с редактором (текст, RTF, HTML)
  - TextPane — стили
  - HyperlinkListener



JButton, JCheckBox, JRadioButton

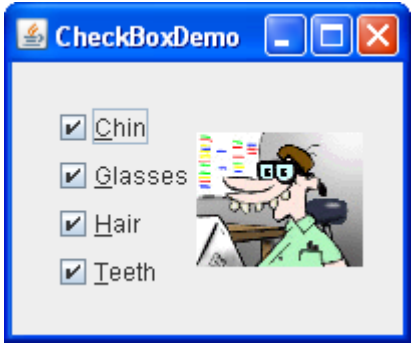
ИТМО

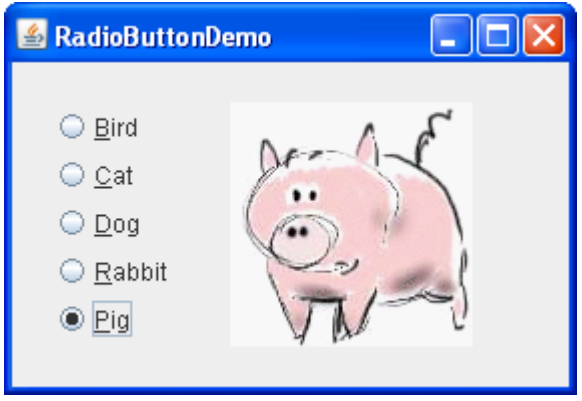
- Конструктор принимает строку
- Для JCheckBox и JRadioButton — еще состояние (boolean)
- JRadioButton используется в группе ButtonGroup
- События
  - ActionEvent для JButton, JRadioButton
  - ItemEvent для JCheckBox (позволяет отследить select-deselect)
- Модель — DefaultButtonModel — элемент с двумя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

Java

Кнопки

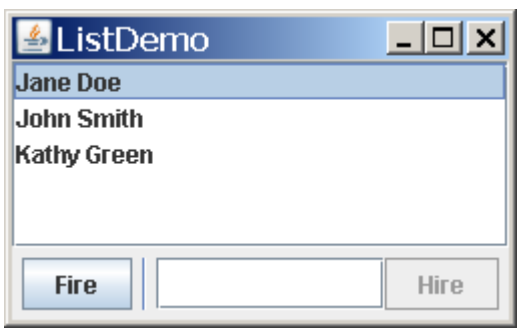

ButtonDemo
Disable middle button
Middle button
Enable middle button


CheckBoxDemo
Chin
Glasses
Hair
Teeth


RadioButtonDemo
Bird
Cat
Dog
Rabbit
Pig

JList

- Конструктор принимает массив или вектор объектов
- Основное событие — ListSelectionEvent



JList

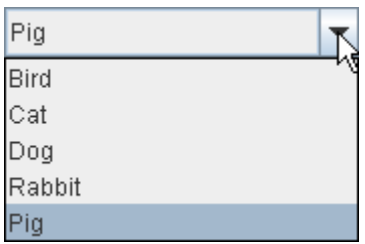
ИТМО

- Модели
  - ListModel ← AbstractListModel ← DefaultListModel
    - getElementAt()
    - getSize()
  - DefaultListModel — модель данных (вектор),
  - ListSelectionModel ← DefaultListSelectionModel
  - DefaultListSelectionModel — модель вариантов выбора (одиночный, интервальный, множественный)

Java

JComboBox

- Может быть редактируемым и нередактируемым
- Конструктор принимает массив или вектор объектов
- Основное событие — ActionEvent, иногда ItemEvent



JComboBox

ИТМО

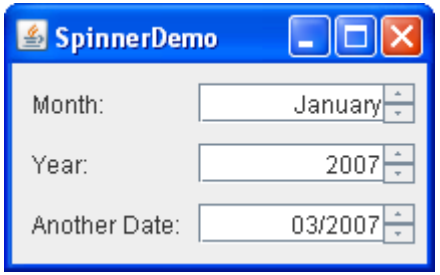
- Модель DefaultComboBoxModel реализует 3 интерфейса — ListModel, ComboBoxModel и MutableComboBoxModel.
- По сравнению с ListModel — ComboBoxModel вводит понятие выбранный элемент (отображаемый)
- MutableComboBoxModel позволяет добавлять и удалять элементы

Java

JSpinner

ИТМО

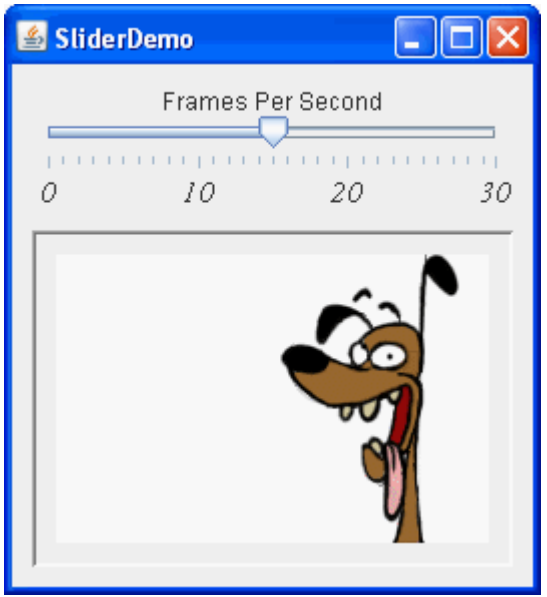
- Составной компонент — 2 кнопки и редактор значений
- Конструктор принимает модель SpinnerModel
- Основное событие — ChangeEvent
- 3 готовых модели — SpinnerListModel, SpinnerDateModel, SpinnerNumberModel + AbstractSpinnerModel
- 3 готовых редактора — JSpinner.ListEditor, JSpinner.DateEditor, JSpinner.NumberEditor



Java

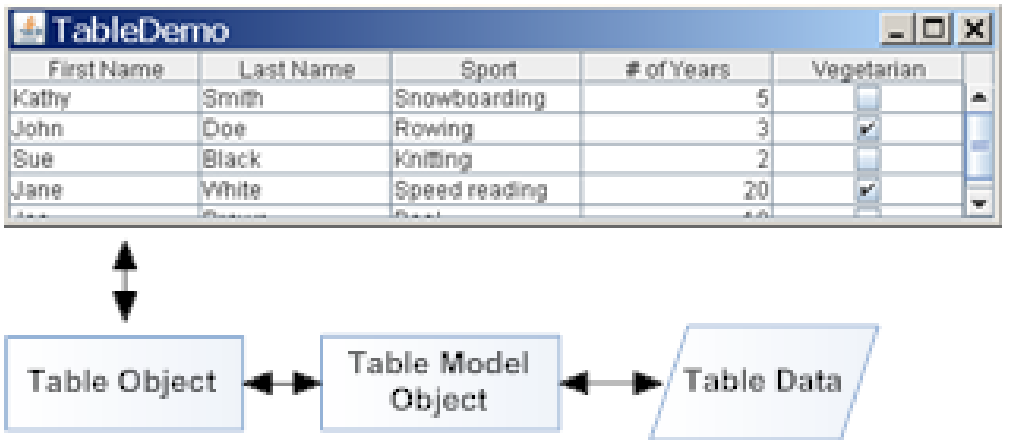
JSlider

- Конструктор принимает min и max значения
- Основное событие — ChangeEvent
- Модель — DefaultBoundedRangeModel — еще используется для JProgressBar



JTable

- Таблица с моделью данных
- Обычно расположена на JScrollPane
- Данные - в модели (interface TableModel)
- JTable - отображение данных
- TableModel
- TableColumnModel
- LIstSelectionModel



JTable

ИТМО

- 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

Java

JTable - сортировка и фильтрация

ИТМО

- Сортировка — TableRowSorter

```
var sorter = new TableRowSorter<TableModel>(tableModel);
jTable.setRowSorter(sorter);
```

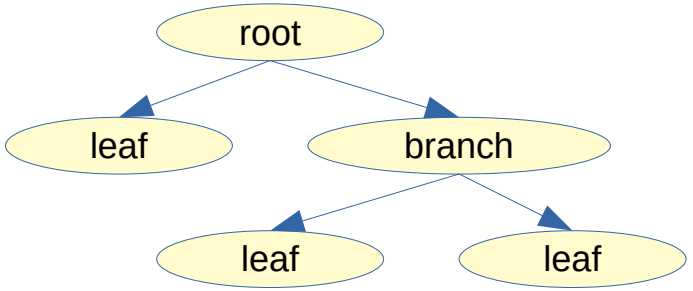
- Фильтрация - RowFilter

```
var rowFilter = RowFilter.regexFilter(text);
sorter.setRowFilter(rowFilter);
```

Java

JTree

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



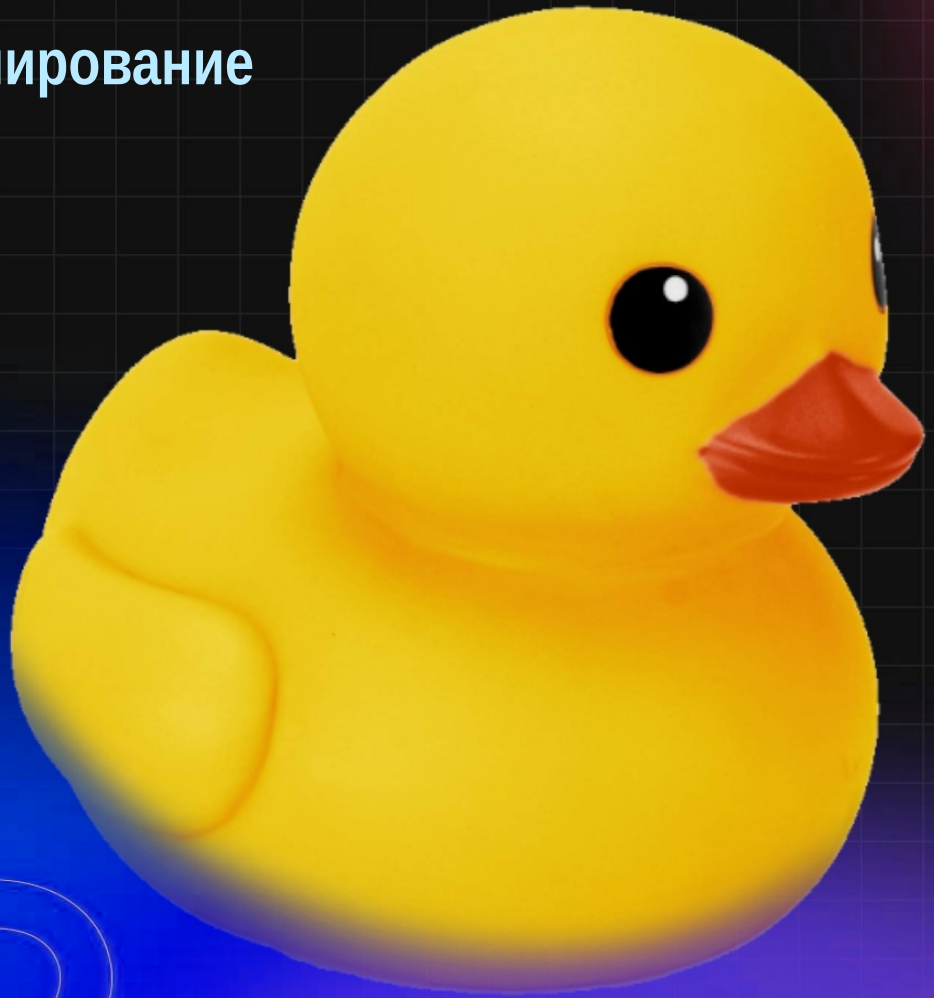
Контейнеры

ИТМО

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

Java

Программирование
2 семестр
2025



ИТМО

Java 2D

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()

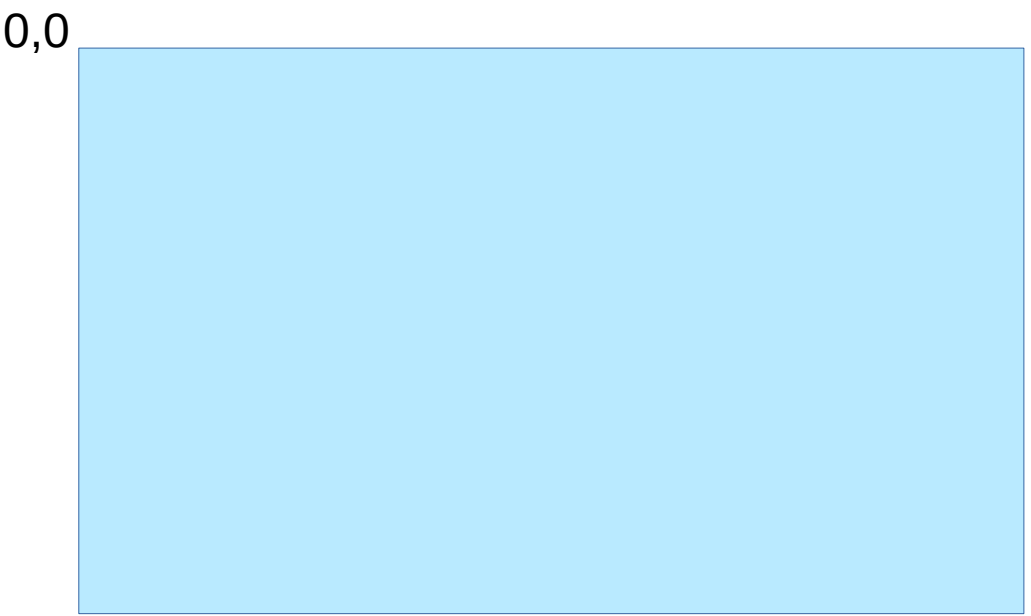
Graphics2D

ИТМО

- Координаты в программе (user-space)

  Rendering

- Координаты устройства (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

ИТМО

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

Java

GeneralPath

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

Стиль линии и заполнения

ИТМО

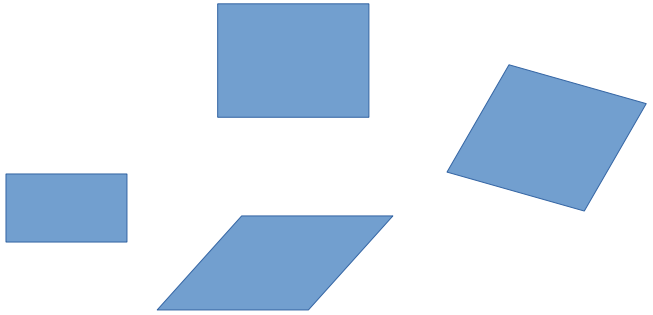
- interface Stroke
  - BasicStroke
- interface Paint
  - Color
  - GradientPaint
  - TexturePaint

Java

Преобразования

ИТМО

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



Java

Анимация

ИТМО

- 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

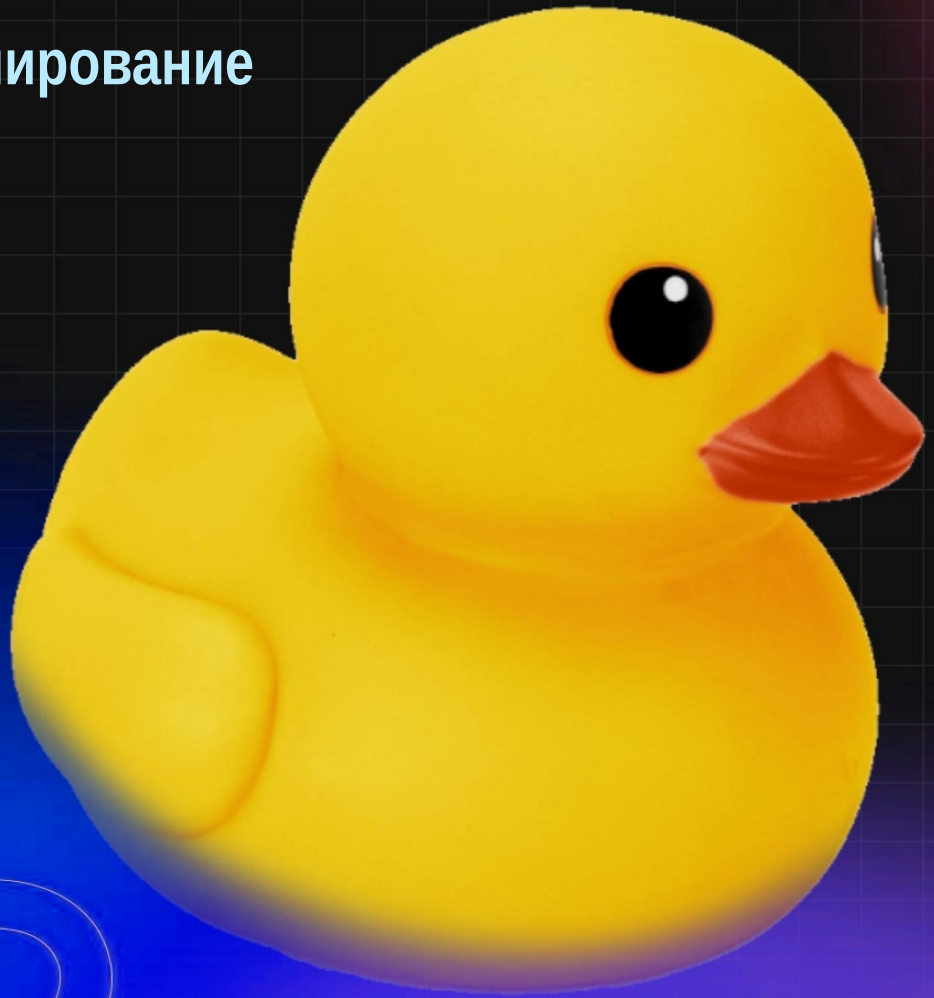
ИТМО

```
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(); });
    }
}
```

Программирование
2 семестр
2025

ИТМО



JavaFX

JavaFX

ИТМО

- 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

- JavaFX 1.1 - 2008
- JavaFX 2.2 - в составе JDK 7 uodate 6
- Начиная с Java 8, JavaFX 8 - часть JDK
- Начиная с Java 11, JavaFX - снова отдельный проект
- https://openjfx.io/
- https://gluonhq.com/

javafx.application

ИТМО

- 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

Java

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

ИТМО

- 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);

Java

Hello World!

ITMO

```
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);
    }
}
```

Java

Node

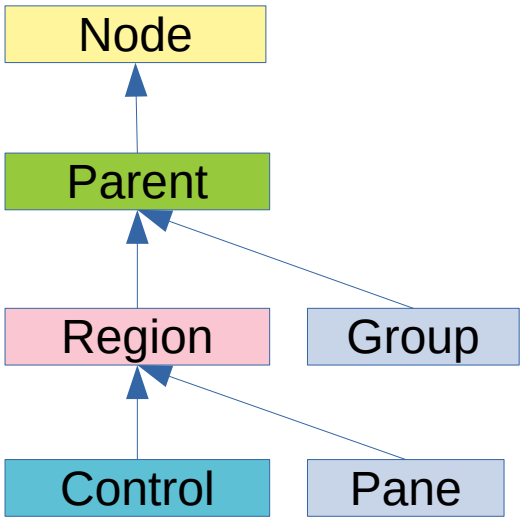
ИТМО

- 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)

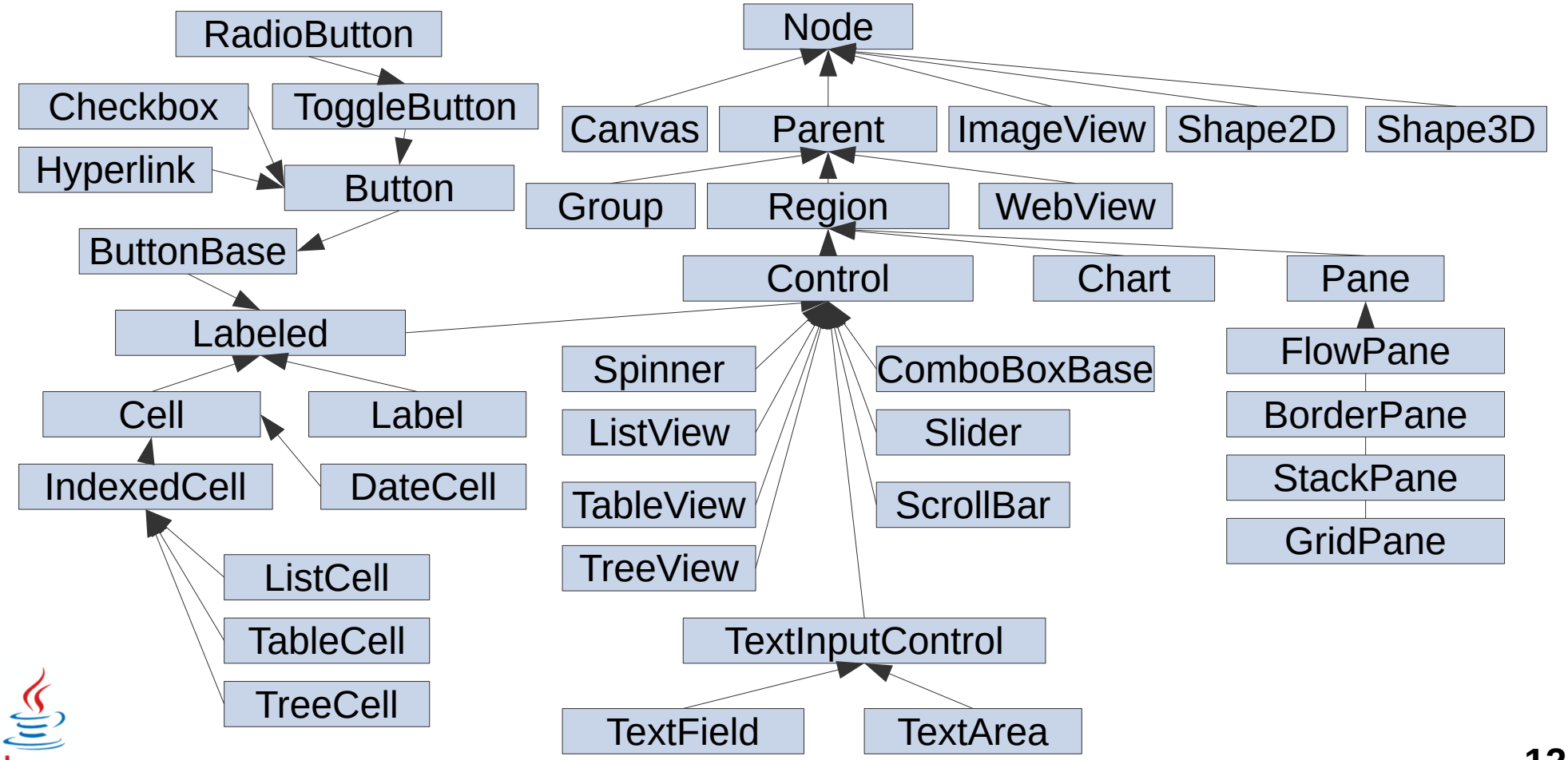
Java

Основные типы узлов JavaFX

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



Основные компоненты JavaFX



javafx.scene.control.*

- Labeled
  - Label
  - ButtonBase
  - Cell
- TextInputControl
  - TextField
    - PasswordField
  - TextArea

Buttons

- ButtonBase
  - Button
  - CheckBox
  - Hyperlink
  - ToggleButton
    - RadioButton // ToggleGroup
  - MenuButton

Cell

- Cell
  - DateCell
  - IndexedCell
    - ListCell
    - TableCell
    - TreeCell

Controls

- ChoiceBox
- ComboBoxBase
- MenuBar
- Pagination
- Spinner
- Slider
- Separator
- ...

Контейнеры с компоновкой

ИТМО

- 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 (аналог GridLayout)
- AnchorPane — привязка к границам родителя

Java

Обработка событий

ИТМО

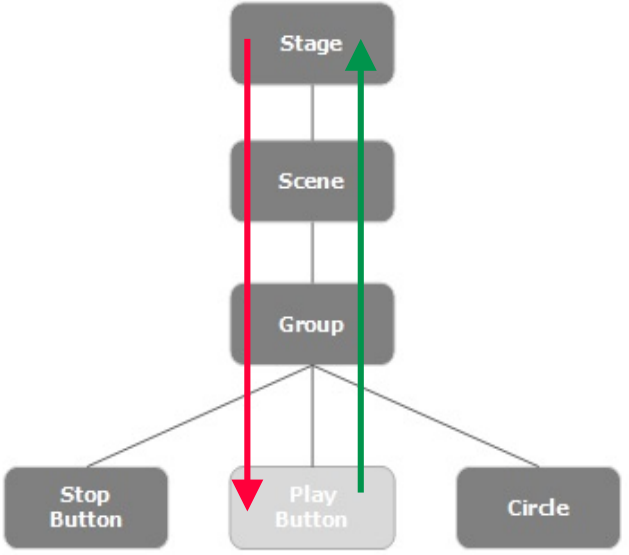
- Событие: javafx.event.Event
  - ActionEvent extends Event
- Обработчик: javafx.event.EventHandler<T extends Event>
  - void handle(T event)
- Регистрация:
  - setOnAction(EventHandler<T>)

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

Обработка событий подробно

ИТМО

- class Event - событие:
  - source - источник, может меняться
  - target - цель (interface EventTarget)
  - type - тип (class EventType)
- Обработчики - addEventHandler/Filter
- Фаза перехвата (capturing) - EventFilter
- Фаза всплытия (bubbling) - EventHandler
- метод consume() - остановка события



Java

События JavaFX

- ActionEvent
  - setOnAction
- InputEvent
  - MouseEvent
    - setOnMousePressed (Released/Clicked/Moved/Dragged/...)
  - KeyEvent
    - setOnKeyPressed (Released/Typed)
  - DragEvent, TouchEvent
  - GestureEvent
    - SwipeEvent, ScrollEvent, RotateEvent, ZoomEvent
- WindowEvent

Свойства JavaFX

ИТМО

- javafx.beans.properties

```
IntegerProperty num1 = new SimpleIntegerProperty(1);
IntegerProperty num2 = new SimpleIntegerProperty(2);
NumberBinding sum = Bindings.add(num1,num2);
System.out.println(sum.getValue());
num1.setValue(2);
System.err.println(sum.getValue());
```

Java

Collections

- javafx.collections.*
- FXCollections
  - ObservableList
  - ObservableSet
  - ObservableMap

Views

- ListView
- ComboBoxView
- TableView
  - setItems(ObservableList<Object>)
- TreeView

TableView

- TableView<T> - таблица
  - tableView.setItems(ObservableList<T> data)
- TableColumn - столбец таблицы
  - column.setCellValueFactory(PropertyValueFactory factory)
  - tableView.getColumns().addAll(...)
- TableCell - ячейка таблицы

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>
```

Scene Builder

- IDE для JavaFX
- https://gluonhq.com/products/scene-builder/

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

ИТМО

- javafx.animation
  - Timeline - KeyFrame... frames - ось времени
    - setCycleCount()
    - play()
  - KeyFrame - Duration time, KeyValue... values - состояние
  - KeyValue - WritableValue<T> target, T endValue - значение
  - Transition - процесс перехода

Java

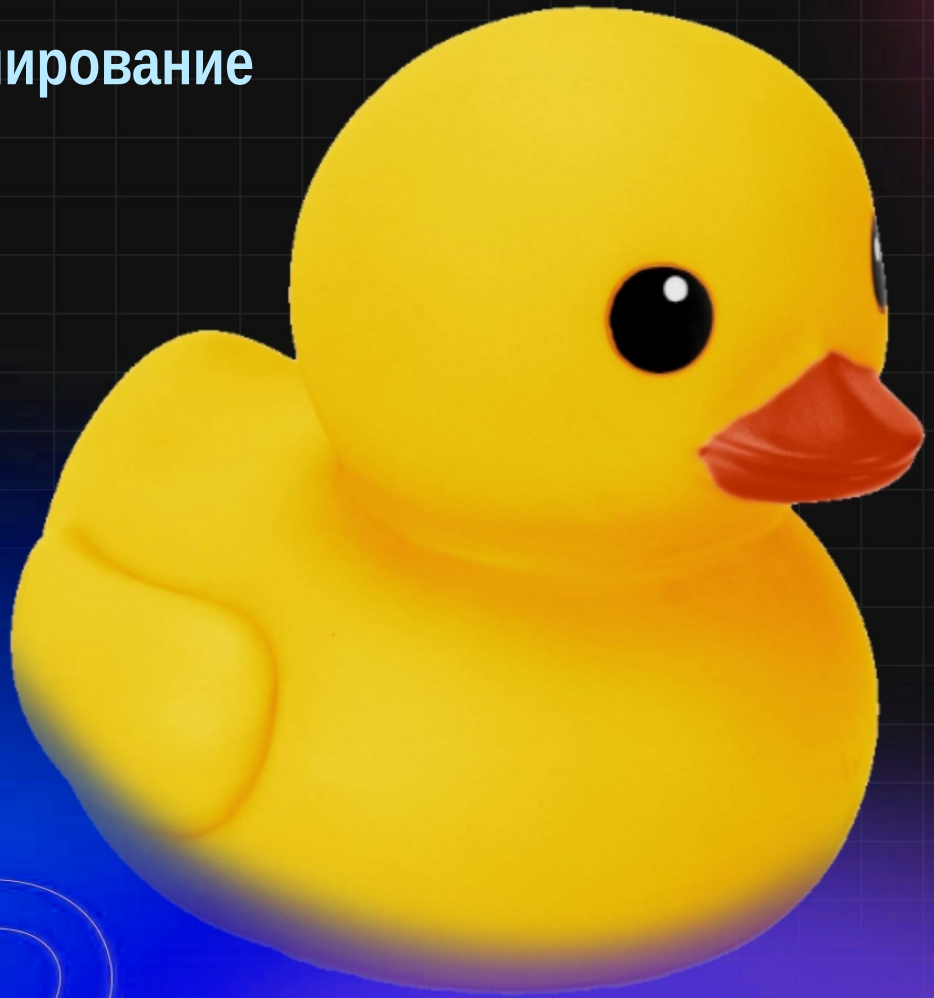
Приложение JavaFX

```
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);
    }
}
```

Программирование
2 семестр
2025

ИТМО



L10n и i18n

Локализация

ИТМО

- Локализация — адаптация программных продуктов для определенной культуры (совокупность языка, местности, других аспектов)
  - Перевод текста
  - Использование соответствующих форматов данных
  - Замена звуковой и визуальной информации
  - **l**ocalizatio**n** = **l**10**n**

Java

Локализация (кустарно-пиратская)

```
0005610  001  \0  \0  \0   h   e   l   l
0005620   o   ,   w   o   r   l   d  \0

0005610  001  \0  \0  \0   п   р   и   в
0005620   е   т   ,       м   и   р  \0
```

Интернационализация

ИТМО

- Интернационализация — проектирование программы так, чтобы ее можно было локализовать без конструктивных изменений
  - текстовые сообщения не хардкодятся, а динамически подгружаются
  - культурно-зависимые данные отображаются в надлежащем формате
  - **i**nternationalizatio**n** = **i**18**n**
- Простая локализация

Интернационализация

ИТМО

- application.exe
- messages.txt

```
[hello]
en=hello
```

Java

i18n + l10n = g11n

- application.exe

- messages.txt

```
[hello]
en=hello
ru=привет
it=ciao
es=hola
fr=salut
de=hallo
fi=hei
zh=你好
vn=xin chào
jp=こんにちは
kr=안녕하세요
hi=नमस्ते
ar=مرحبًا
```

Проблемы

ИТМО

- Шрифты, направление письма, сложные символы
- Разные правила (pluralization): 1 кот, 3 кота, 11 котов
- Форматы: листов бумаги, телефонных номеров, адресов
- Единицы измерения, размеры батареек, напряжение в розетках
- Платежные системы, национальные праздники, имена и титулы, цветовой символизм
- Культурные и религиозные ограничения, цензура
- И т. д.

Java

Средства поддержки локализации

ИТМО

- CLDR - Common Locale Data Repository
  - названия языков, стран, валют, элементов даты и времени
  - шаблоны форматов чисел, даты и времени
  - правила сортировки, записи числа прописью
  - правила транслитерации
- формат XML
- https://cldr.unicode.org/

Java

Локаль

ИТМО

- Локаль — совокупность характеристик, определяющих географический, политический или культурный регион
- Класс java.util.Locale
- Примеры: it, en_UK, ru_RU.CP1251
- Элементы локали:
  - язык — 2 строчные буквы (иногда 3) (ru)
  - страна (регион) — 2 заглавные буквы (RU) или 3 цифры
  - вариант (например, кодировка для русской локали)
  - письменность — 4 буквы, первая заглавная (Cyrl)
  - расширение

Java

Создание локали

ИТМО

- Конструкторы класса Locale
  - new Locale(String lang)
  - new Locale(String lang, String country)
  - new Locale(String lang, String country, String variant)
- Класс Locale.Builder
  - new Locale.Builder().setLanguage("ru").setRegion("RU").build()
- Метод forLanguageTag()
  - Locale.forLanguageTag("ru-RU");
- Константы класса Locale
  - Locale.FRENCH
  - для русского константы нет

Методы для работы с локалью

ИТМО

- Получение списка локалей и локали по умолчанию
  - `static Locale[] getAvailableLocales()`
  - `static Locale getDefault()`
- Преобразование в строку
  - `String toString() // ru_RU`
  - `String getDisplayName() // Russian (Russia)`

Java

Наборы ресурсов

ИТМО

- Класс java.util.ResourceBundle
- Формат "ключ-значение"
  - ключ - условная строка, значение - перевод
- 2 варианта:
  - Файл свойств *.properies (PropertyResourceBundle)
  - Класс со списком (ListResourceBundle)

Java

Свойства - PropertyResourceBundle

ИТМО

- GuiLabels_en.properties GuiLabels_ru.properties
  s1 = Yes s1 = Да
  s2 = No s2 = Нет

```
ResourceBundle r = ResourceBundle.getBundle("GuiLabels");
JButton b1 = new JButton(r.getString("s1"));
JButton b2 = new JButton(r.getString("s2"));
```

- \+ отдельные текстовые файлы
- \- только String
- Кодировка **ISO-8859-1** — необходима обработка с помощью native2ascii (до Java 9)

Списки - ListResourceBundle

ИТМО

```
public class GuiLabels_en extends ListResourceBundle {
    public Object[][] getContents() { return contents; }
    private Object[][] contents = { {"s1", "Yes"}, {"s2", "No"} };
}
public class GuiLabels_ru extends ListResourceBundle {
    public Object[][] getContents() { return contents; }
    private Object[][] contents = { {"s1", "Да"}, {"s2", "Нет"} };
}
ResourceBundle r = ResourceBundle.getBundle("GuiLabels");
JButton b1 = new JButton(r.getString("s1"));
JButton b2 = new JButton(r.getString("s2"));
```

- \+ любые типы объектов
- \- нужна компиляция файлов

Формирование списка ресурсов

ИТМО

- ResourceBundle.getBundle(String name, Locale locale)
  - locale = language_script_country_variant
  - Кандидаты - сначала .class, если его нет - .properties
    - name_language_script_country_variant
    - name_language_scirpt_country
    - name_language_script
    - name_language_country_varuant
    - name_language_country
    - name_language
  - Если таких ресурсов нет - пробуем дефолтную локаль по той же схеме
  - Последний кандидат - name
  - Если ничего не нашлось - MissingResourceException

Java

Получение нужного ресурса

ИТМО

- getString(key)
- getStringArray(key)
- getObject(key)

- Просматривается список кандидатов
- Возвращается первая найденная по ключу строка
- MissingResourceException

Java

Пример

```
var bundle = ResourceBundle.getBundle("msg", locale);
```

```
// msg.properties
delete = Delete
ok = OK
cancel = Cancel
```

```
// msg_ru_RU.properties
delete = Удалить

cancel = Отменить
```

```
// msg_it.properties

ok = Va bene
cancel = Cancellare
```

Пример

```
// Locale.getDefault() == ru_RU

var bundle = ResourceBundle.getBundle("msg", locale);
```

```
locale = de

msg_ru_RU.properties
msg.properties
```

```
// msg.properties
delete = Delete
ok = OK
cancel = Cancel
```

```
// msg_ru_RU.properties
delete = Удалить

cancel = Отменить
```

```
// msg_it.properties

ok = Va bene
cancel = Cancellare
```

Пример

```
// Locale.getDefault() == ru_RU

var bundle = ResourceBundle.getBundle("msg", locale);
```

```
locale = de

msg_ru_RU.properties
msg.properties
```

```
locale = ru_RU

msg_ru_RU.properties
msg.properties
```

```
// msg.properties
delete = Delete
ok = OK
cancel = Cancel
```

```
// msg_ru_RU.properties
delete = Удалить

cancel = Отменить
```

```
// msg_it.properties

ok = Va bene
cancel = Cancellare
```

Пример

```
// Locale.getDefault() == ru_RU

var bundle = ResourceBundle.getBundle("msg", locale);
```

```
locale = de

msg_ru_RU.properties
msg.properties
```

```
locale = ru


msg.properties
```

```
// msg.properties
delete = Delete
ok = OK
cancel = Cancel
```

```
// msg_ru_RU.properties
delete = Удалить

cancel = Отменить
```

```
// msg_it.properties

ok = Va bene
cancel = Cancellare
```

Пример

ИТМО

```
// Locale.getDefault() == ru_RU

var bundle = ResourceBundle.getBundle("msg", locale);
```

```
locale = de

msg_ru_RU.properties
msg.properties
```

```
locale = ru


msg.properties
```

```
locale = it_IT

msg_it.properties
msg.properties
```

```
// msg.properties
delete = Delete
ok = OK
cancel = Cancel
```

```
// msg_ru_RU.properties
delete = Удалить

cancel = Отменить
```

```
// msg_it.properties

ok = Va bene
cancel = Cancellare
```

Пример

ИТМО

```
// Locale.getDefault() == ru_RU

var bundle = ResourceBundle.getBundle("msg", locale);
var message = bundle.getString("delete");
```

```
locale = de
Удалить
msg_ru_RU.properties
msg.properties
```

```
locale = ru
Delete

msg.properties
```

```
locale = ït_IT
Delete
msg_it.properties
msg.properties
```

```
// msg.properties
delete = Delete
ok = OK
cancel = Cancel
```

```
// msg_ru_RU.properties
delete = Удалить

cancel = Отменить
```

```
// msg_it.properties

ok = Va bene
cancel = Cancellare
```

Java

Пример

```
// Locale.getDefault() == ru_RU

var bundle = ResourceBundle.getBundle("msg", locale);
var message = bundle.getString("cancel");
```

```
locale = de
Отменить
msg_ru_RU.properties
msg.properties
```

```
locale = ru
Cancel

msg.properties
```

```
locale = ït_IT
Cancellare
msg_it.properties
msg.properties
```

```
// msg.properties
delete = Delete
ok = OK
cancel = Cancel
```

```
// msg_ru_RU.properties
delete = Удалить

cancel = Отменить
```

```
// msg_it.properties

ok = Va bene
cancel = Cancellare
```

Форматирование числовых данных

ИТМО

- Класс java.text.NumberFormat — абстрактный
  - NumberFormat nf = NumberFormat.getNumberInstance()
  - NumberFormat cf = NumberFormat.getCurrencyInstance()
  - NumberFormat pf = NumberFormat.getPercentInstance()
  - nf.format(new Float(999.8));
- Класс java.text.DecimalFormat
  - df = (DecimalFormat) nf;
  - df.applyPattern("##,#0.00");
  - df.format(new Float(888.7));
- Класс DecimalFormatSymbols
  - ds.setDecimalSeparator('=');
  - df.setDecimalFormatSymbols(ds);
  - df.format(new Float(777.6));

Символы шаблона

ИТМО

0 — цифра, 0 отображается

— цифра, 0 не отображается

. — разделитель десятичной дроби

, — разделитель групп разрядов

E — разделитель мантиссы и порядка

- — знак минус

; — разделитель подшаблонов

% — умножить на 100 и отобразить как процент

‰ — умножить на 1000 и отобразить как промилле

¤ — символ валюты

Форматирование даты и времени

ИТМО

- Класс java.text.DateFormat — абстрактный
  - DateFormat df = DateFormat.getDateInstance(DateFormat.FULL)
  - DateFormat tf = DateFormat.getTimeInstance(DateFormat.LONG)
  - DateFormat dtf = DateFormat.getDateTimeInstance(DateFormat.SHORT)
  - df.format(new Date());
- Класс java.text.SimpleDateFormat
  - sdf = (SimpleDateFormat) df;
  - sdf.applyPattern("yyyy-MM-dd");
  - sdf.format(new Date());
- Класс DateFormatSymbols
  - ds.setShortWeekdays("пнд","втр","срд","чтв","птн","сбт","вск");
  - sdf.setDateFormatSymbols(ds);
  - sdf.format(new Date());

Символы шаблона

Более 4 символов — полный формат, 3 — сокращенный, 2 - число

G — эра
y — год
M — месяц после числа
L — месяц (название)
d — число
E — название дня недели
H — часы
m — минуты
s — секунды
S — миллисекунды
z — временная зона

Сообщения - MessageFormat

ИТМО

- 8 апреля 2024 в 18:17 произошло полное солнечное затмение.
- Total solar eclipse happened at 18:17PM on April 8, 2024.

Eclipse_en.properties

```
msg = {0} solar eclipse happened at
{1,time,short} on {1,date,short}.

full = Total
part = Partial
ring = Annular
```

Eclipse_ru.properties

```
msg = {1,date,short} в {1,time,short}
произошло {0} солнечное затмение.

full = полное
part = частное
ring = кольцеобразное
```

```
ResourceBundle r = ResourceBundle.getBundle("Eclipse");
MessageFormat mf = new MessageFormat(r.getString("msg");
Date date = Date.from(Instant.parse("2024-04-08T18:17"));
Object[] args = {r.getString("full", date.getTime()};
mf.format(args);
```

Java

Формат с выбором

- Класс ChoiceFormat extends NumberFormat
  - 0 friends like it
  - 1 friend likes it
  - 1000 friends like it

Like_en.properties

```
msg = {0} it
one = {0,number} friend likes
many = {0,number} friends like
```

```
ResourceBundle r = ResourceBundle.getBundle("Like");
MessageFormat mf = new MessageFormat(r.getString("msg");
double[] lims = { 0, 1, 2 };
String one = r.getString("one");
String many = r.getString("many");
String[] msgs = { many, one, many };
ChoiceFormat cf = new ChoiceFormat(lims, msgs);
mf.setFormatByArgumentIndex(0, cf);
Object[] args = { new Integer(15) };
mf.format(args);
```

```
15 friends like it
```

Сравнение строк

ИТМО

- Класс Collator - абстрактный
- Collator getInstance()
- int compare()
- RuleBasedCollator

```
List<String> lst = Arrays.asList({"Fluor", "Chlor","Brom","Jod"});
Collator c1 = Collator.getInstance(Locale.EN);
Collator c2 = Collator.getInstance(new Locale("cz","CZ");
lst.sort(c1);    // Brom, Chlor, Flour, Jod
lst.sort(c2);    // Brom, Fluor, Chlor, Jod
// A, Á, B, C, Č, D, Ď, E, É, Ě, F, G, H, Ch, I, Í, J, K, L, M, N,
// Ň, O, Ó, P, Q, R, Ř, S, Š, T, Ť, U, Ú, Ů, V, W, X, Y, Ý, Z, Ž
```

Java

Разбор текста

ИТМО

- Класс BreakIterator — поиск границ
- Методы
  - getCharacterInstance()
  - getWordInstance()
  - getSentenceInstance()
  - getLineInstance()
  - int first()
  - int last()
  - int next()
  - int previous()

Java