Java GUI

BOOKS

* Хорстманн Java. Библиотека профессионала Т1 (576)
* Шилдт Java8. Полное руководство (201-210)(286-438)(470-563)(564)
* Файн Java24 (233)

SITES

* Java course ( http://java-course.ru/begin/simpleclasses/)
* Metanit (https://metanit.com/java/tutorial/8.1.php)

VIDEO

* Файн ([https://www.youtube.com/playlist?list=PLkKunJj\_bZefB1\_hhS68092rbF4HFtKjW) (5](https://www.youtube.com/playlist?list=PLkKunJj_bZefB1_hhS68092rbF4HFtKjW)%20(5) (33M)-6)(8 – 0m)
* Уроки java (<https://www.youtube.com/user/husivm/videos>) (44- 56)(58-95)(121 – 0m)(328 – 0m)
* Мирончик (<https://www.youtube.com/playlist?list=PL27FCF69EBF48228B>) (1.2 – 2.25m)

**Основы**

* Сначала все строилось на библиотеке AWT (Abstract Window Toolkit), она взаимодействовала со стандартными средствами ОС, далее от этого отказались и внедрили Swing
* Окно верхнего уровня называется фрейм (отвечает класс JFrame)
* Каждая программа должна регистрироваться в потоке диспетчеризации событий, поэтому правильно программу запускать в потоке EventQueue.invokeLater(new Runnable() { public void run() { … } , либо EventQueue.invokeLater(() –> {..})
* Иерархия наследования JFrame -> Frame -> Window -> Container -> Component -> Object (аналогично JPanel -> JComponent -> Container)
* Фрейм содержит 4 области (2 прозрачных, область меню и область содержимого)
* Все операции рисования графики делаются с помощью класса Graphics
* Как строится модель событий: есть источник, при наступлении события он извещает всех зарегистрированных приемников
* События делятся на низкоуровневые и семантические. Семантические – те что совершает пользователь (щелчок мыши и т.д.), а низкоуровневые – просто обеспечивают возможность этих действий

**Swing**

* Создание простого фрейма   
  JFrame frame = new FirstFrame();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

* Обычно еще устанавливают панель содержимого setContentPane(new JPanel())
* Важные методы фрейма setLocation() и setBounds(), dispose(), setlconlmage(), setTitle(), setResizable()
* Этот класс содержит много методов для оконной системы ОС Toolkit kit = Toolkit.getDefaultToolkit();
* Вместо JComponent можно рисовать в JPanel, но она непрозрачна, поэтому стоит сначала залить цветом фона super.paintComponent(g);
* Все события происходят от класса java.util.EventObject от него уже наследуется ActionEvent (события от кнопки) и WindowEvent (события от окна фрейма)
* Наиболее часто употребляемые события: ActionEvent (щелчок на кнопке, выбор пункта меню, выбор элемента из списка, нажатие клавиши <Enter> при вводе текста в поле), AdjustmentEvent (перемещение бегунка на полосе прокрутки), ItemEvent (выбор одной из кнопок-переключателей, установка одного из флажков или выбор элемента из списка). Из низкоуровневых: KeyEvent (нажатие или отпускание клавиши), MouseEvent (нажатие и отпускание кнопки мыши, перемещение курсора мыши, перетаскивание курсора), MouseWheelEvent (вращение колесика мыши), FocusEvent (получение или потеря фокуса ввода), WindowEvent (изменение состояния окна)
* Для приемника событий реализуем интерфейс ActionListener и реализуем метод actionPerformed, а для того чтобы зарегистрировать приемник делаем у объекта .addActionListener
* Метод actionPerformed(ActionEvent e) получает на вход информацию об источнике события, получить компонент, который вызвал событие можно как (JButton) e.getSource()
* По сути actionPerformed это колбек
* У WindowListener есть 7 методов, которые надо реализовать, но лучше использовать класс адаптер и переопределить только нужный нам (WindowListener –> WindowAdapter)
* Мы можем создать приемника событий и использовать для разных компонент (т.к. ActionListener как правило один на компонент), для этого нам нужно создать действие (Action) точнее его абстрактный класс AbstractAction и передать его в конструкторе действия
* У Action есть важный метод putValue, где мы можем задавать свойства
* Для того чтобы связать действия с нажатием клавиши, необходимо создать объект класса KeyStroke, но там еще нужно учитывать привязку ввода (JComponent.WHEN\_FOCUSED, WHEN\_ANCESTOR\_OF\_FOCUSED\_COMPONENT, WHEN\_lN\_FOCUSED\_WlNDOW) для этого получаем InputMap из контейнера (панели), далее связываем imputMap с каким-нибудь идентификатором и помещаем этот идентификатор и Action в ActionMap для контейнера (панели)
* Большинство компонент построено по шаблону MVC, получить модель компонента можно как ButtonModel model = button.getModel()
* Компоновка интерфейса: компоненты размещаются в контейнерах, а диспетчер компоновок определяет порядок расположения и размеры контейнера. Установить компоновку у контейнеры - setLayout
* По умолчанию используется FlowLayout (компоненты идут друг за другом по горизонтали и располагаются по центру), так же есть BorderLayout(выбираю север, юг), GridLayout(4,2) (сетка, все компоненты одного размера), BoxLayout, CardLayout, GridBagLayout (нужно описывать GridBagConstraints)
* Можно указать абсолютные значения в контейнере, ставим setLayout(null), а затем у каждого компонента ставим setBound(x, y, w, h)
* Компоненты:  
  - текст: классы JTextField, JTextArea, JPasswordField  
  - метки: класс JLabel, расположить их можно указав JLabel.LEFT и т.д.  
  - флажки: JCheckBox  
  - кнопки переключатели: создаем класс ButtonGroup и туда накидываем JRadioButton  
  - модальное окно: JOptionPane (так же у него есть метод showConfirmDialog который показывает различные уведомления)  
  - диалоговое окно: JDialog