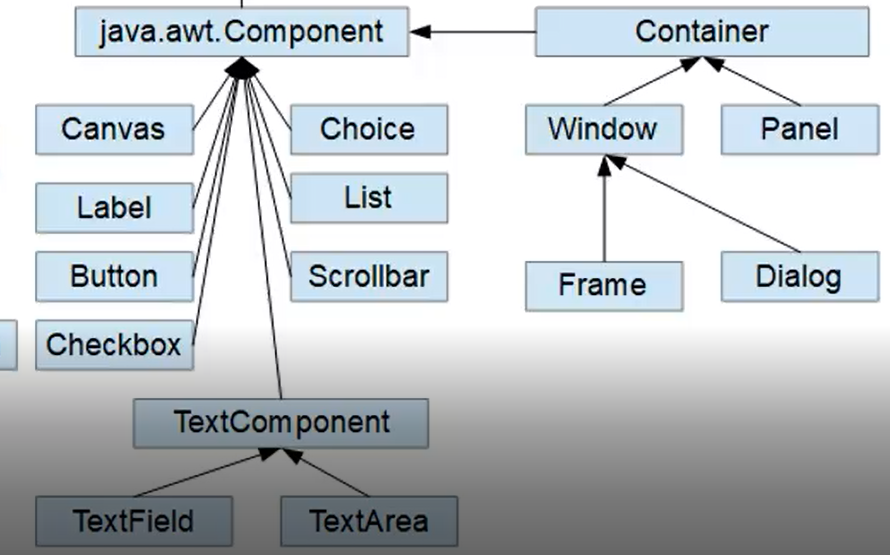
**Компоненты пользовательского интерфейса. Иерархия компонентов.**



**Базовые классы Component, Container, JComponent.**

* Component – абстрактный класс, элемент GUI, который взаимодействует с пользователем. Определять цвет размер и местоположение компонента. Порождает основные события.
  + Цвет – Color(RGB), константа
  + Размер и местоположение – Point(x, y) Demention(high, width) Rectangle(x, y, high, width)
  + Шрифт – Font
  + Видимость – isVisible(), setVisible() Frame Window не видимы после создания
  + Активность – isEnabled, setEnabled()
* Container – потомок класса Component, который содержит в себе другие компоненты.
  + Иерархия компонентов – дерево
  + Компонент находится только в 1 контейнере
* JComponent – расширяет Container может содержать картинку, всплывающие подсказки, построение основано на MVC

Основные компоненты

* Canvas - является простейшим наследником Component. Он не добавляет никакой новой функциональности, но именно его нужно использовать в качестве суперкласса для создания пользовательского компонента с некоторым нестандартным внешним видом.
* Label - Как понятно из названия, этот компонент отображает надпись.
* Button - Этот компонент позволяет добавить в интерфейс стандартные кнопки.
* Checkbox - Элемент, который может быть выделен или нет
* Choice служит для выбора пользователем одного из нескольких возможных вариантов (выпадающий список)
* TextComponent является наследником Component и базовым классом для компонент, работающих с текстом
  + TextField позволяет вводить и редактировать одну строку текста
  + TextArea позволяет вводить и просматривать многострочный текст.
* Scrollbar позволяет работать с полосами прокрутки, которые используются для перемещения внутренней области от начальной до конечной позиции. Полоса может быть расположена горизонтально или вертикально. Стрелки на каждом из ее концов служат для перемещения "на один шаг" в соответствующем направлении.

Потомки контейнера

* Panel является суперклассом для новых контейнеров с особой работой с вложенными компонентами.
* Window служит базовым классом для всех окон, порождаемых из Java.
  + Frame предназначен для создания полнофункциональных окон приложений – с полосой заголовка, рамкой, кнопками закрытия, минимизации и максимизации окна.
  + Dialog позволяют открывать дополнительные окна для взаимодействия с пользователем.

**Менеджеры компоновки.**

Управляют местоположением компонентов и их размером

* Layout
* Layout2 -можно задать дополнительную характеристику.

Мы можем каждый компонент добавлять в менеджер компановки с помощью метода add. И менять с помощью set

Основные менеджеры компановки

* FlowLayout – заполняет контейнер слева направо построчно. Если компонент не влезает в строку – начинается следующая, они сохраняют свой предпочитаемый размер.
* GridLayout – таблица с ячейками одинакового размера. Если не хватит элементов – пустые ячейки можно задать число строк столбцов, границы. Элементы растягиваются под размеры ячейки.
* CardLayout – понель с имитацией вкладок. Мы добавляем вкладки и в каждый момент времени отображается только одна из них. Выбрать - show
* BorderLayout – Используется для окон верхнего уровня. Делит область на 5 частей. Добавление add(компонент , место)
* GridBagLayout – Контейнер делится на ячейки по строкам и столбцам. Вы сами все реализуете.

**Модель обработки событий. Класс-слушатель и класс-событие.**

Основано на событийно -ориентированном программировании.

В начале мы пишем что то типа button.AddActionLisener(Label) т.е. Как только с кнопкой что то произойдет об этом сразу узнает метка. Для этого метка должна реализовавать метод ActionListener. Она реализует метод actionperfomed, где мы прописываем логику того что ей делать.

Источник события – любой компонент

Событие - AWTEwent

**Технология JavaFX. Особенности архитектуры, отличия от AWT / Swing.**

* AWT – кроссплатфрмена, разработчики оставили только классы и интерфейсы, которые поддерживались во всех ос, поэтому она очень урезана.
  + библиотека, которая зависит от графической подсистемы ОС
  + Должна выглядеть на всех платформах одинокого хорошо, но это не так.
* Swing – все компоненты написаны на java
  + Надстройка над AWT в виде легковесных компонентов.
  + Изменяемый вид компанентов
* JavaFX
  + Новая графическая библиотека
  + Поддерживает анимацию
  + Есть визуальные эффекты
  + XML – интерфейс
  + CSS - стиль

**Интернационализация. Локализация. Хранение локализованных ресурсов.**

**Локализация** — адаптация программы для определенного языка или местности.

* Локализация обычно включает, перевод текста на другой язык,
* использование определенных форматов чисел, отображения даты и времени,
* замена звуковой и визуальной информации.

**Интернационализация** – это не процесс, это стиль, как правильно писать программу, чтобы потом ее было удобно локализовывать. Интернационализация обычно заключается в том, что текстовые данные отделяются от кода и хранятся отдельно, а числа и даты отображаются не напрямую, а прогоняются через форматтеры, которые преобразуют данные в нужный формат.

Есть у нас некое приложение, есть дополнительный текстовый файл, в нем есть некоторый раздел, и дальше мы пишем, что при использовании английского языка нужно будет вывести строку «hello». Это уже интернационализованное приложение, но оно пока не локализовано. Чтобы оно стало локализованным, надо текстовый файл дополнить. Код уже трогать не надо. Остается только раздать текстовые файлы переводчикам, которые добавят поддержку языков. Это приложение уже интернационализованное и локализованное одновременно.

Локаль — это объект, задающий местность и язык. Элементы локали — это язык, страна и вариант, изредка встречаются дополнииельные элементы. Язык обозначается двумя строчными буквами: ru, en. Страна - обычно двумя заглавными буквами, иногда тремя цифрами. Вариант - показывает либо кодировку, либо применяемый календарь

Хранение данных

* PropertyResourceBundle - используется для хранения исключительно текствой информации (при этом его проще использовать)
  + При использовании PropertyResourceBundle достаточно создать текстовые файлы с опредленным именем. Имя должно состоять из названия ресурса, локали и расширения .properties. Несколько файлов с одним именем ресурса, но разными локалями образуют набор ресурсов (ResourceBundle). С помощью метода getBundle набор ресурсов загружается и после этого его можно использовать. Текстовые данные берутся из набора ресурсов по ключу, возможно с указанием локали. Основное преимущество файлов свойств - простой формат файлов с данными. Недостаток - возможность использовать только текстовые данные.
* ListResourceBundle - можно применять для хранения любых объектов.
  + Нужно создать наследников класса ListResourceBundle с именами этих классов, сформированными по тому же принципу как и у файлов свойств, Затем в каждом классе реализовать метод getContents(), возвращающий двумерный массив объектов, содержащий ключи и соответствующие им значения. Такой способ немного сложнее, потому что формат класса более сложный, после того, как класс создан, его необходимо скомпилировать. Но при этом можно хранить любые объекты, не только текстовые.

**Форматирование локализованных числовых данных, текста, даты и времени. Классы NumberFormat, DateFormat, MessageFormat, ChoiceFormat.**

NumberFormat - абстрактный класс с фабричными методами, которые возвращают одного из потомков NumberFormat, умеющего форматировать числа в соответствии с заданными правилами (для числа, для валюты и для процентов). Обычно этим потомком является объект класса DecimalFormat. С помощью метода format можно получить число в нужном формате. Наиболее общий формат для всех локалей выдают фабричные методы NumberFormat.

Абстрактный класс DateFormat имеет фабричные методы для получения форматтера для даты, времени, и совмещенного. Можно задать тип формата: полный, длинный, средний, короткий. Если локаль не задана, используется системная. Данные методы возвращают объект класса SimpleDateFormat, являющийся потомком DateFormat. SimpleDateFormat позволяет указать произвольный шаблон формата. И класс DateFormatSymbols используется для замены стандартных элементов формата даты или времени.

MessageFormat, который позволяет использовать параметры в строке. Соответственно, можно задать для разных языков строки с параметрами в разных местах. На слайде показан пример для фраз на русском и английском языках, где в строку подставляется время, дата и другая строка.

Класс ChoiceFormat позволяет выбрать и подставитьт один вариант из нескольких в зависимости от используемого значения. Его можно использовать, например, для выбора нужной формы слова при числительных (хотя для русского языка это реализовать сложнее, чем для английского). Для создания формата нужно задать массив чисел, задающих диапазоны, и соответствующие этим диапазонам ключи.