Лабораторная работа 1

Язык разметки FXML и компоновка приложения

Цель работы: изучить создание javafx-приложений с использованием языка разметки fxml.

Теоретическая часть

В дополнение к процедурной конструкции пользовательского интерфейса приложения JavaFX могут использовать декларативную разметку XML.

Иерархическая структура XML — это способ описать иерархию компонентов в пользовательском интерфейсе.

Формат XML, специфичный для JavaFX, называется FXML. В нем вы возможно определить все компоненты приложения и их свойства, а также связать их с контроллером, который отвечает за управление взаимодействиями для данного представления. FXML – это язык разметки пользовательского интерфейса на основе XML, созданный корпорацией Oracle для определения пользовательского интерфейса приложения JavaFX.

Использование отдельного файла с описанием внешнего вида позволяет отделить уровень представления от остальной логики приложения.

Для создания интерфейса из файла fxml применяется метод `FXMLLoader.load()`. Чтобы получить описание интерфейса из файла, используется метод `getClass().getResource("имя_файла.fxml")`.

При использовании JDK, который не включает в себя JavaFX, его необходимо скачать отдельно и подключить к проекту. При запуске приложения в IDEA в проекте необходимо для среды исполнения добавить путь к библиотеке:

```
--module-path "Путь\до\javafx\lib"
--add-modules javafx.controls,javafx.fxml
```

Метод `FXMLLoader.load()` возвращает объект класса `Parent`, который можно передать в конструктор объекта `Scene`, и таким образом, приложение получит интерфейс из fxml.

Существуют элементы и атрибуты FXML, которые по умолчанию недоступны. Для их добавления в корневом элементе добавляется пространство имен FXML: *xmlns:fx="http://javafx.com/fxml"*

Основные составляющие окна приложения:

Stage — по сути это само окно, которое используется как начальное полотно и содержит в себе остальные компоненты. У приложения может быть несколько stage, но один такой компонент должен быть в любом случае. По сути Stage является основным контейнером и точкой входа.

Scene — отображает содержание stage. Каждый stage может содержать несколько компонентов — scene, которые можно между собой переключать. Внутри это реализуется графом объектов, который называется — Scene Graph (где каждый элемент — узел, называемый Node).

Node — это элементы управления, например, кнопки метки, или другие макеты (layout), внутри которых может быть несколько вложенных компонентов. У каждой сцены (scene) должен быть один вложенный узел (node), но это может быть макет (layout) с несколькими компонентами. Вложенность может быть многоуровневой, когда макеты содержат другие макеты и обычные компоненты. У каждого такого узла есть свой идентификатор, стиль, эффекты, состояние, обработчики событий.

Все элементы управления, которые используются в приложении, помещаются в специальные контейнеры – панели компоновки (layout panes). Все панели компоновки наследуются от класса javafx.scene.layout.Pane, который, в свою очередь, наследуется от Region.

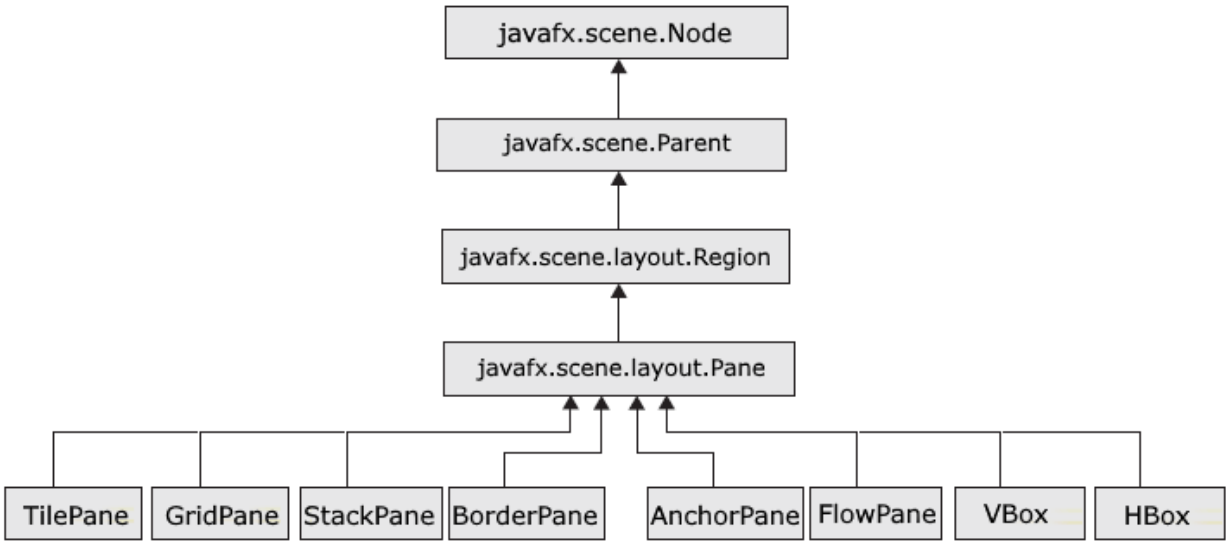

Рисунок 1 – Иерархия компонентов JavaFX

Панели компоновки позволяют упорядочить вложенные элементы определенным образом и могут определять их размеры. В JavaFX есть несколько встроенных панелей компоновки:

AnchorPane: позволяет прижимать вложенные элементы управления к одной из сторон контейнера или размещать их по центру

BorderPane: позволяет упорядочить вложенные элементы управления относительно одной из сторон контейнера

FlowPane: размещает все вложенные элементы либо по горизонтали (если элементы не помещаются, то они переносятся на новую строку), либо по вертикали (если элементы не помещаются, то они переносятся в новый столбик)

GridPane: образует сетку из строк и столбцов и размещает вложенные элементы ячейках полученной таблицы

HBox: размещает все элементы в виде одной строки, то есть горизонтальный стек

VBox: размещает все элементы в виде одного столбца, то есть вертикальный стек

StackPane: располагает одни элементы поверх других

TilePane: размещает все вложенные элементы в виде «плиток»

Контроллеры

Контроллер это обычный класс java, который может взамодействовать с FXML. Класс контроллера назначается корневому контейнеру в файле разметки используя тег `fx:controller="…"`.

У различных компонентов управления есть наборы свойств, определяющих их реакцию на действия пользователя.

Например, у класса Button есть свойство onAction, которое хранит обработчик события нажатия в виде объекта `EventHandler<ActionEvent>`. В FXML можно установить это свойство с помощью одноименного атрибута. В качестве значения ему передается метод из контроллера. Название метода в контроллере и значение атрибута в FXML должно совпадать.

Таким образом класс контроллера и интерфейс FXML могут взаимодействовать.

Также JavaFX пытается автоматически сопоставить компоненты с fx:id с полями определенным в контроллере с тем же именем. Предположим, в fxml есть кнопка с `fx:id="mainButton"`.

JavaFX пытается внедрить объект кнопки в контроллер в поле с именем mainButton:

```
@FXML
private Button mainButton;
```

или

```
public Button mainButton;
```

Поля должны иметь модификатор `public` или быть аннотированными `@FXML`.

Компонент выбора значений Slider

Slider (Слайдер) является компонентом интерфейса, позволяющим выбрать числовое значение в диапазоне значений. Slider включает один *бар* (*track*) и один *рычаг*, который можно перетаскивать (*Draggable thumb*). Он также может иметь отметки (*tick mark*) и марки (Рисунок 2).

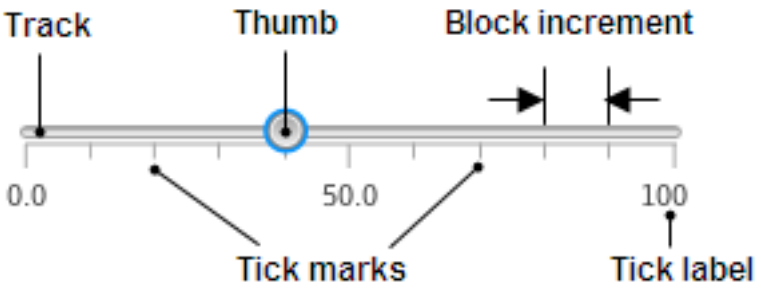

Рисунок 2 – Размещение элементов на компоненте Slider

Для создания слайдера можно использовать один из двух конструкторов класса:

`Slider()`: создает слайдер с настройками по умолчанию

`Slider(double min, double max, double value)`: создает слайдер, для которого установлены минимальное, максимальное и текущее значения.

Для программного изменения свойств компонента можно воспользоваться следующими методами:

`void setOrientation(Orientation how)`: устанавливает ориентацию слайдера - горизонтальную или вертикальную

`void setValue(double value)`: устанавливает текущее значение слайдера

`double getValue()`: возвращает текущее значение слайдера

`void setShowTickMarks(boolean value)`: значение `true` (по умолчанию) делает деления на шкале слайдера видимыми, соответственно значение `false` делает деления невидимыми

`void setMinorTickCount(int val)`: устанавливает количество делений между двумя числовыми отметками на шкале

`void setMajorTickUnit(int val)`: устанавливает расстояниие между двумя числовыми отметками на шкале

`void setShowTickLabels(boolean on)`: устанавливает видимость числовых меток на шкале

`void setBlockIncrement(double val)`: устанавливает, насколько значений будет перемещаться бегунок слайдера с помощью клавиатурных клавиш Вперед и Назад при горизонтальной ориентации и клавиш Вверх и Вниз при вертикальной ориентации

`void setSnapToTicks(boolean on)`: при передачи значения `true` позволяет переходить ровно по делениям (то есть вместо значения типа 4.5677687 будет 5)

```
public void onClick() {
   label.setText(String.valueOf(slider.getValue()));
}
```

В данном примере в обработчике кнопки используя метод *getValue*() считывается текущее значение слайдера и выводится на форму.

Сам метод возвращает значение типа double, поэтому его необходимо преобразовать к типу String. Это можно сделать двумя способами:

`String.valueOf(Object)`

или

`Object.toString()`, где вместо класса *Object* нужно указать соответствующий класс.

Можно сразу динамически изменять текст на метке при изменении значения слайдера. Для этого нужно назначить обработчик на событие «*On Mouse Dragged*» (рисунок 3).

```
public void onSliderChange() {
    label.setText(String.valueOf(slider.getValue()));
```

```
}
```

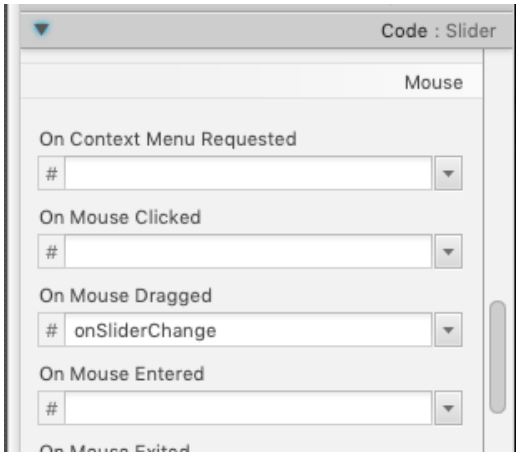

Рисунок 3 – Назначенный обработчик события

При установке “*Snap To Ticks*” значение ползунка будет останавливаться точно на метках. Но во время его движения он все равно будет перемещаться плавно. Чтобы при этом числа выводились в заданном виде, можно выполнить форматирование полученного значения:

```
double value = slider.getValue();
label.setText((Double.toString(value)).format("%.1f", value));
```

Использование стилей для оформления компонентов

Одним из важных преимуществ библиотеки JavaFX является использование стилей css. Используя концепцию стилей можно делать приложения с так называемыми «скинами», когда внешний вид полностью отделен от бизнес-логики, и даже разрабатывается отдельно, например, профессиональными дизайнерами.

Добавим стили в нашу программу. В файл sample.fxml добавим следующий текст:

```
stylesheets="@css/sample.css"
```

Данный путь подразумевает хранение файла sample.css в папке css, находящейся в папке resources.

Полученная строка будет выглядеть примерно так:

```
<GridPane alignment="center" hgap="10" vgap="10"
      xmlns:fx="http://javafx.com/fxml/1"
      xmlns="http://javafx.com/javafx/8.0.172-ea"
      fx:controller="sample.Controller"
      stylesheets="/sample/sample.css">
```

Имя файла */sample/sample.css* будет выделено красным, так как он отсутствует в проекте. Встроенными методами IDEA создадим файл *sample.css*, выбрав действие Create File sample.css

В полученный файл добавим следующие строки:

```
.root {
-fx-background-color: cadetblue; //Цвет заднего фона окна
}
.label {
-fx-font-size: 16; //Высота текста
}
.button {
-fx-font-size: 12;
-fx-text-fill: darkviolet;
}
.button:focused {
    -fx-background-color: chartreuse;
    -fx-text-fill: green;
}
```

Дополнительную информацию по стилям можно получить в любом справочнике по CSS, а также на официальной странице документации https://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html

Задания на лабораторную работу

Выполнить действия в соответствии с вариантом.

Оформить выравнивание и расположение элементов интерфейса используя различные панели компоновки. Значение свойств, влияющих на расположение элементов задать при построении интерфейса: Vgap, Hgap и padding, Row Valignment & Column Halignment.

Логику вычислений вынести в отдельный класс.

В приложении реализовать:

1. Разметка с использованием вложенных контейнеров
2. Ввод значений в текстовые поля
3. Дублировать ввод значение с использованием компонента Slider с фиксированным набором значений.

4. Использовать стили CSS для оформления приложения.
5. Добавить кнопку для сброса всех значений и установки Slider на начальное значение.

Вариант	**Задание**
1	Даны два числа. Найти среднее арифметическое кубов этих чисел и среднее геометрическое модулей этих чисел.
2	Заданы стороны прямоугольника. Вычислить его периметр, площадь и длину диагонали.
3	Даны вещественные положительные числа a, b, c. Если существует треугольник со сторонами a, b, c, то определить его вид (прямоугольный, остроугольный, тупоугольный)
4	Даны вещественные положительные числа a, b, c. Выяснить, существует ли треугольник со сторонами a, b, c.
5	Даны два числа. Вычислить их сумму, разность, произведение и среднее арифметическое.
6	Даны три числа. Найти среднее арифметическое квадратов этих чисел и сумму модулей этих чисел.
7	Заданы координаты трех вершин треугольника. Найти его периметр и площадь
8	Вычислить периметр и площадь прямоугольного треугольника по заданным длинам двух катетов a и b
9	Вычислить площадь и периметр прямоугольника, если задана длина одной стороны (a) и коэффициент n (%), позволяющий вычислить длину второй стороны (b=n*a)
10	Если известны две стороны и угол между ними, вычислить периметр и площадь треугольника.

Содержание отчета по лабораторной работе

- Титульный лист
- Описание задания
- Скриншоты разработанных экранных форм
- Разметка экранных форм на языке fxml
- Программный код основного класса и класса контроллера

Вопросы для самоконтроля

1. Что такое язык FXML?
2. Порядок загрузки и создания компонентов формы при использовании FXML?
3. Какова иерархия наследования компонентов формы?
4. Обязательно ли использование панелей компоновки (контейнеров, pane) при создании форм с помощью FXML?

Список литературы

1. Вязовик, Н. А. Программирование на Java : учебное пособие / Н. А. Вязовик. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 601 c. — ISBN 978-5-4497-0852-6. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/102048.html

2. Мухаметзянов, Р. Р. Основы программирования на Java : учебное пособие / Р. Р. Мухаметзянов. — Набережные Челны : Набережночелнинский государственный педагогический университет, 2017. — 114 c. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/66812.html

3. Блох, Дж. Java. Эффективное программирование / Дж. Блох ; перевод В. Стрельцов ; под редакцией Р. Усманов. — 2-е изд. — Саратов : Профобразование, 2019. — 310 c. — ISBN 978-5-4488-0127-3. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/89870.html