НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Объектно-ориентированное программирование на языке Java

Часть 4. GUI: JavaFX

Yevhen Berkunskyi, NUoS
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua
Tatiana Smykodub

Что такое JavaFX і?

- JavaFX - это платформа и (большой) набор объектов, которые можно использовать для разработки приложений на основе графического интерфейса
- API JavaFX является хорошим примером того, как объектно-ориентированные принципы могут применяться в разработке программного обеспечения

JavaFX vs Swing vs AWT

- В первой версии Java для поддержки графического интерфейса в имелась только библиотека **AWT** (Abstract Windows Toolkit)
- **AWT** не была плохой, но существовали некоторые ограничения и проблемы с реализацией на некоторых платформах
- В конечном итоге **AWT** (которая все еще существует, но больше не используется) была заменена новой библиотекой **Swing**, которая была более универсальной, более надежной и более гибкой.

JavaFX vs Swing vs AWT

- **Swing** была разработана, прежде всего, для использования в настольных приложениях (хотя ее также можно использовать и для разработки веб-приложений).
- В настоящее время **Swing** полностью заменена новой графической библиотекой **JavaFX**
- Вы все еще можете использовать **Swing** (в обозримом будущем), но Oracle не собирается ее развивать дальше - это, по сути, умирающая технология
- Java заменил **Swing** на **JavaFX**
- Когда **JavaFX** заменится чем-то другим? Никто не знает, вероятно, через много лет

Введение в JavaFX

С помощью JavaFX вы можете создавать графический интерфейс пользователя двумя способами:

- Написав код на Java «вручную».
- С помощью инструмента визуальной компоновки, называемого Scene Builder, где вы будете перетаскивать компоненты в окне и сохранять дизайн в формате FXML в файле с расширением .fxml. Затем программа Java загрузит этот .fxml-файл и отобразит ваше творение.

Для этого вам нужно:

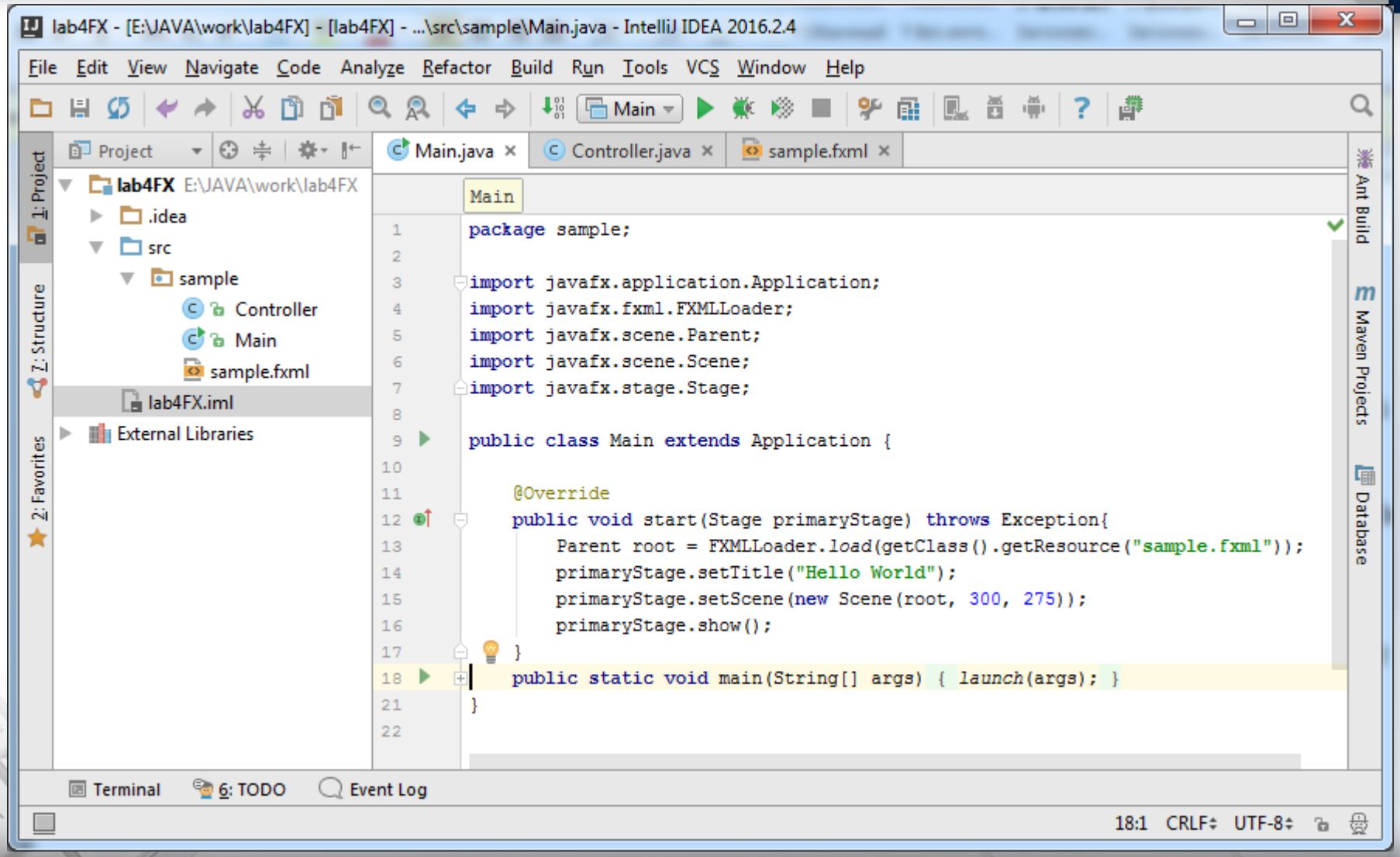
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Введение в JavaFX

В IntelliJ IDEA создайте новый проект (меню File | New Project), выберите приложение JavaFX в качестве типа проекта и дайте ему имя. В новом проекте будут созданы два класса Java(Main и Controller) и файл с именем sample.fxml

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Введение в JavaFX



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Структура JavaFX программ

- Все программы JavaFX начинаются не как «обычный» класс, как мы делали раньше, а как расширение абстрактного класса Application в JavaFX, javafx.application.Application

```
public class MyProgram {
    // Body of class
}
```

Теперь:

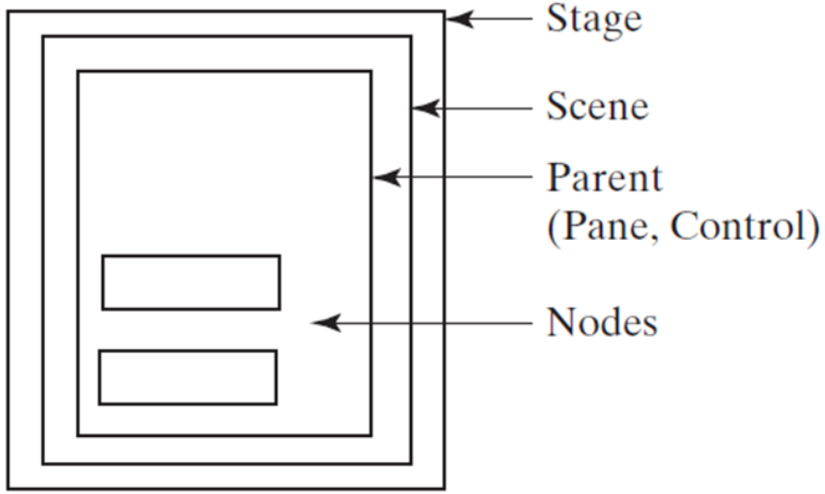
```
import javafx.application.Application;
…
public class MyProgram extends Application {
    // Body of class
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Введение в JavaFX

- Main наследует класс Application и содержит два метода. Это базовая структура, которая нужна для для запуска JavaFX приложения. Для нас важен метод start (Stage primaryStage). Он автоматически вызывается когда приложение запускается из метода main.
- Примечание: в последних версиях JavaFX метод main уже не требуется.
- Как видите, метод start (...) принимает экземпляр класса Stage в качестве параметра. На рисунке снизу представлена структура любого JavaFX приложения:

Panes, UI Controls, и Shapes



- Мы можем поместить кнопку непосредственно на сцену, которая центрировала кнопку и заставила ее занимать все окно.
- Вряд ли это то, что мы действительно хотели сделать
- Один из подходов - указать размер и расположение каждого элемента пользовательского интерфейса (например, кнопки)
- Лучшим решением является поместить элементы пользовательского интерфейса (известные как узлы) в контейнеры, называемые панелями, а затем добавить панели в сцену

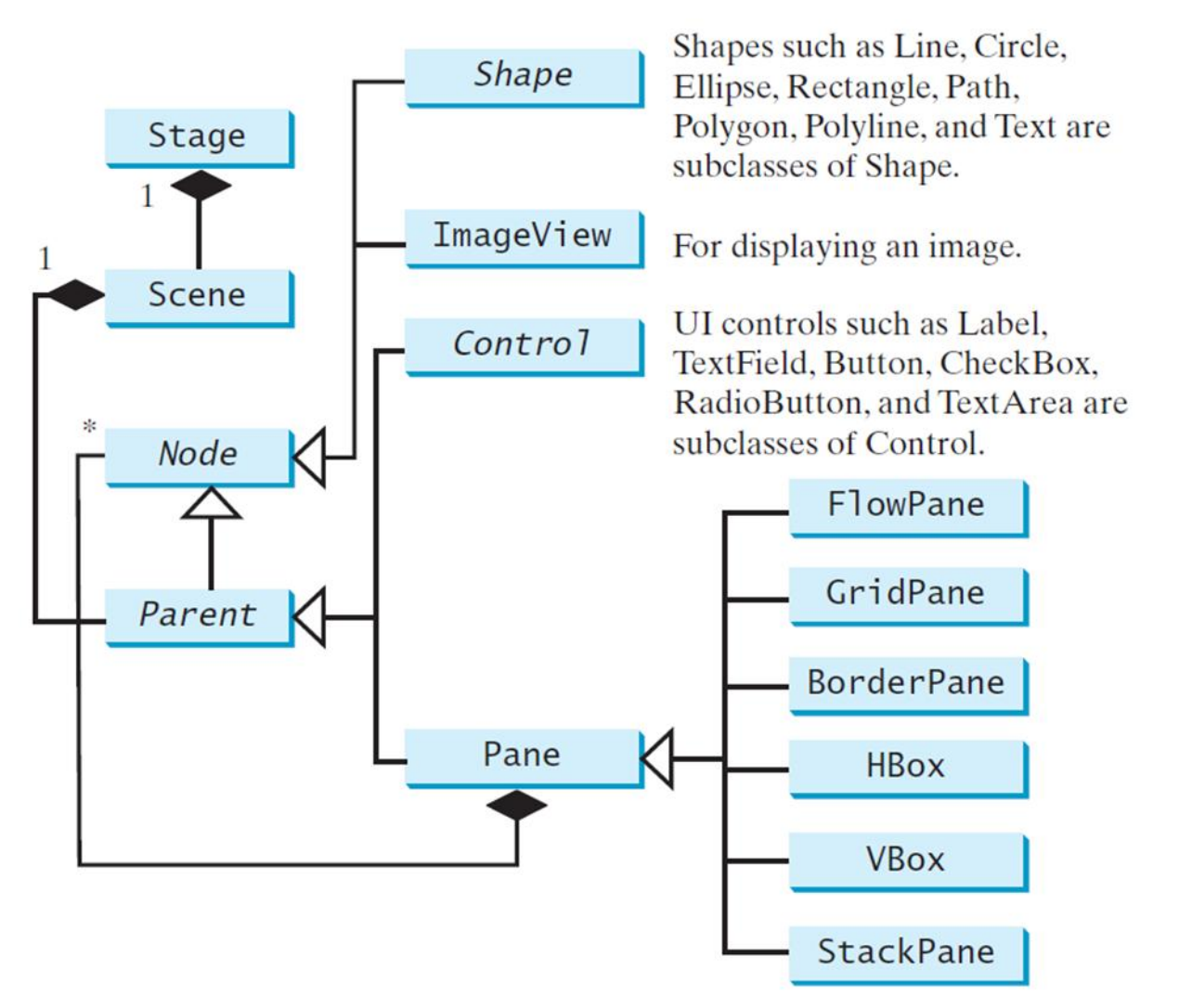
Panes, UI Controls, и Shapes

- Это как театральная пьеса: Stage является основным контейнером, который, как правило, представляет окно с рамками и стандартными кнопками закрыть, минимизировать и максимизировать. Внутрь Stage добавляется Scene, которая, конечно, может быть заменена другой Scene. Внутрь Scene уже прилагаются стандартные компоненты типа AnchorPane, TextBox и другие.
- Для получения более подробной информации, обратитесь к этому материалу.

sample.fxml

- Метод start загружает файл sample.fxml, устанавливает заголовок сцены и задает размер сцены 300 на 275 пикселей.
- FXML - это язык разметки на основе XML. Идея состоит в том, чтобы отделить создание GUI на декларативном языке (FXML) от логики приложения (Java). В FXML GUI представлен тегами в угловых скобках.

Panes, UI Controls, и Shapes



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Panes, UI Controls, и Shapes

- Помимо очевидных типичных «активных» элементов пользовательского интерфейса (с которыми мы можем взаимодействовать, например кнопки и т. д.), есть еще и статические формы - линии, круги и т. д.
- Прежде чем работать с фигурами, мы должны погово-рить о координатах внутри панели.
- Верхний левый угол сцены всегда (0, 0), положительная ось X направлена вправо, а положительная ось Y направлена вниз. Визуально мы находимся в декартовом квадранте IV, но Y остается положительным.
- Все координаты измеряются в пикселях

Example

```
package sample;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;
import javafx.scene.text.Font;
import javafx.stage.Stage;
public class Main extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception{
        primaryStage.setTitle("Hello World");
        BorderPane pane=new BorderPane();
        Label l1= new Label("Hello");
        l1.setFont(new Font(100)); pane.setLeft(l1);
        Label l2= new Label("fffff");
        l2.setFont(new Font(200)); pane.setRight(l2);
        primaryStage.setScene(new Scene(pane));
        primaryStage.show();
    }
}
```

Panes

- Как мы уже говорили (и делали), мы добавляем наши узлы в область, а затем добавляем панель на сцену, а затем сцену на подмостки.
- Как мы организуем (т. е. выкладываем) узлы на панели?
- Java имеет несколько разных видов панелей.

Компоновки в JavaFX

Класс панели	Описание
HBox, VBox	Выравнивает дочерние элементы по горизонтали или по вертикали
GridPane	Располагает дочерние элементы в виде табличной сетки аналогично компоненту **GridBagLayout** в Swing
TilePane	Располагает дочерние элементы в виде сетки и одинакового размера аналогично компоненту **GridLayout** в Swing
BorderPane	Предоставляет для компоновки северную, восточную, южную, западную и центральную области аналогично компоненту **BorderLayout** в Swing
FlowPane	Располагает дочерние элементы рядами, создавая новые ряды, если недостаточно свободного места, аналогично компоненту **FlowLayout** в Swing
AnchorPane	Дочерние элементы могут быть расположены на абсолютных позициях или относительно границ панели. Такой режим выбирается по умолчанию в инструменте компоновки SceneBuilder
StackPane	Располагает дочерние элементы друг над другом. Такая компоновка может оказаться полезной для оформления компонентов, например, для размещения кнопки на цветном прямоугольнике

Общие свойства и методы для Node

- Узлы имеют много общих свойств
- Свойства стиля JavaFX очень похожи на CSS (каскадные таблицы стилей), используемые для указания стилей на страницах HTML (Web).

Свойства JavaFX

- В спецификации JavaBeans определяются также *привязанные свойства*, где объекты инициируют события об изменениях в свойствах при вызове методов установки.
- В JavaFX эта часть спецификации не используется. Вместо этого свойства, в JavaFX, помимо методов получения и установки, снабжается третьим методом, возвращающим объект класса, реализующего интерфейс Property.
- Например, у свойства text в JavaFX имеется метод **Property<String> textProperty**(). К объекту свойства можно присоединить приемник событий. Этим JavaFX отличается от прежней технологии JavaBeans.

Свойства JavaFX

- В JavaFX объект свойства, а не компонент JavaBeans, посылает уведомления об изменениях.

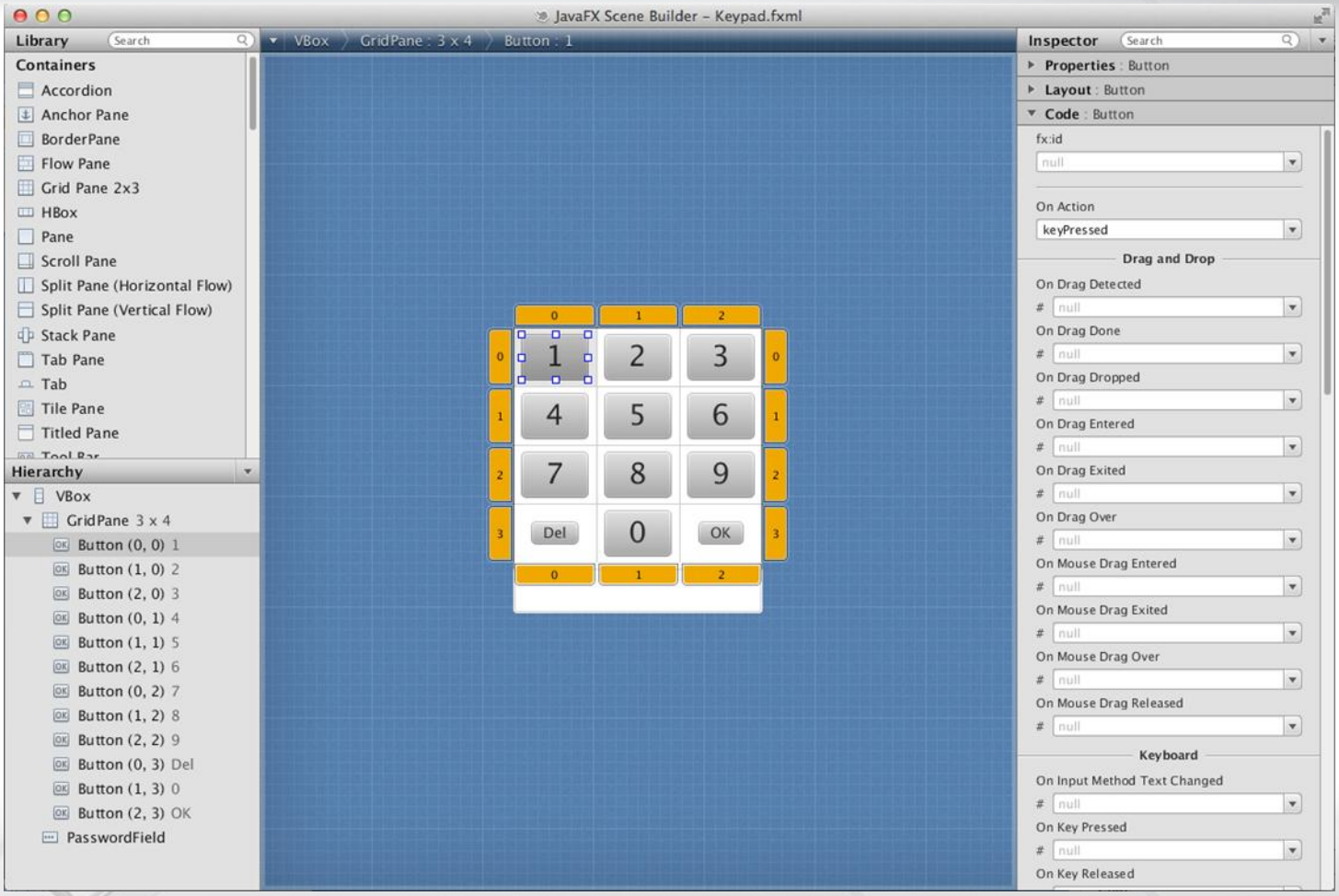
  Для реализации привязанных свойств в JavaBeans требовался шаблонный код, выполнявший ввод, удаление и запуск приемников событий. А в JavaFX все сделано намного проще, поскольку все бремя хлопот берут на себя библиотечные классы.

Scene Builder

- GUI создаются намного быстрее, чем в Swing и AWT
- Более сложные и эстетически приятные пользовательские интерфейсы
- Легкая интеграция звуков, изображений и видеороликов и веб-контента
- Код упрощается в JavaFX, отделяя пользовательский интерфейс от логики приложения
- JavaFX может быть интегрирован в приложения Swing, что обеспечивает более плавный переход

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Scene Builder



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Дизайн приложения в Scene Builder

- Создадим окно входа в систему, где пользователь будет вводить ID, пароль и нажимать кнопку «Sing In».
- На сцене расположим макет GridPane. Первая строка будет содержать Label и TextField для ID пользователя, вторая строка будет иметь аналогичную пару для пароля, а третья строка сетки будет содержать одну кнопку Button, которая будет охватывать два столбца.

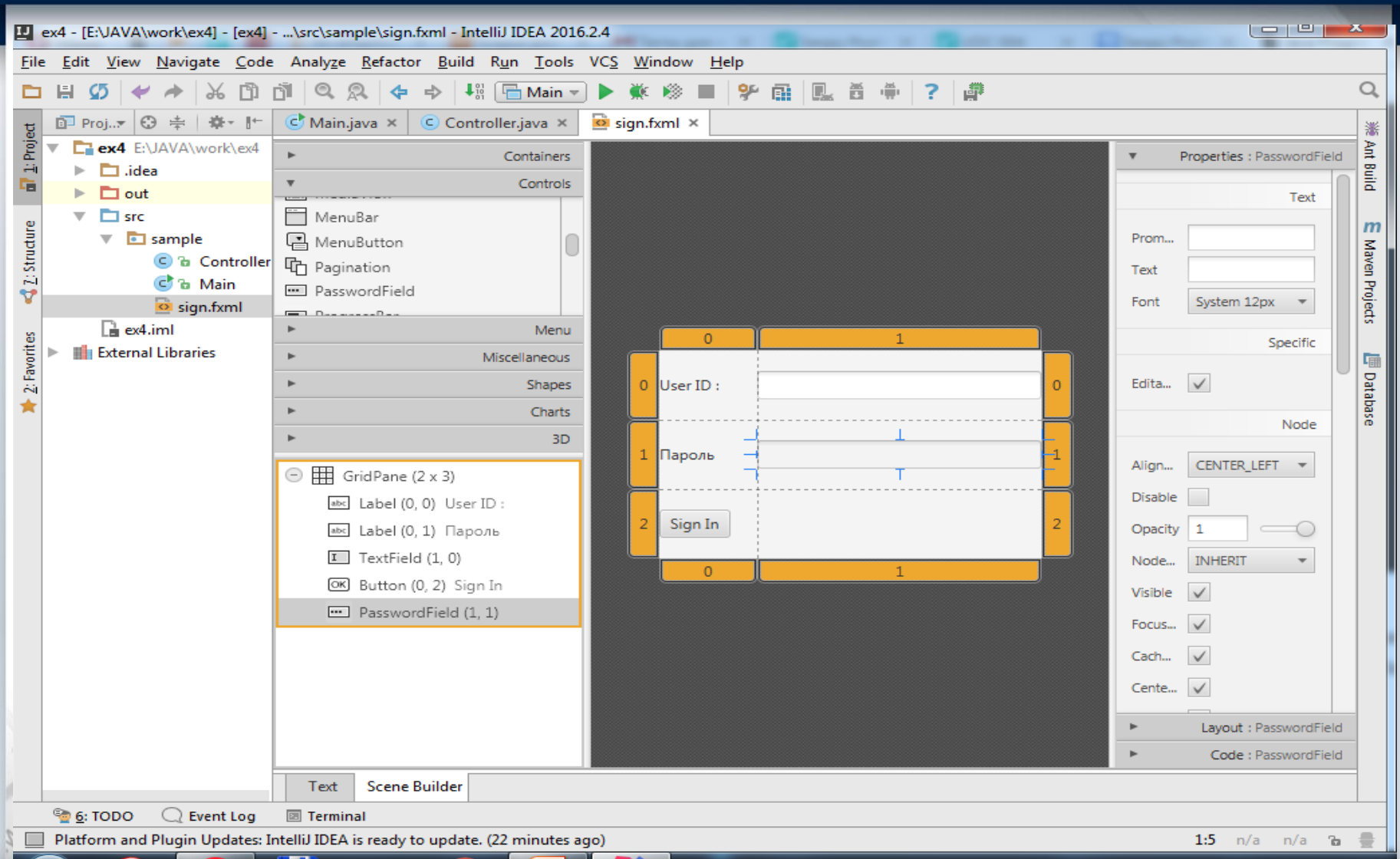
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Дизайн приложения в Scene Builder

- Откройте сгенерированный sign.fxml в Scene Builder. Поскольку этот файл уже имеет пустой тег <GrigPane> теперь перетащите два элемента управления Label и Button из раздела «Controls» в соответствующие ячейки в первом столбце сетки. Измените текст на этих компонентах как User ID :, Пароль и Sign In. Затем нужно перетащить два объекта TextField в ячейки второго столбца. Правда для пароля, это не очень хорошая идея, здесь больше подойдет объект Password (он отметит ввод пользователя) для пароля.

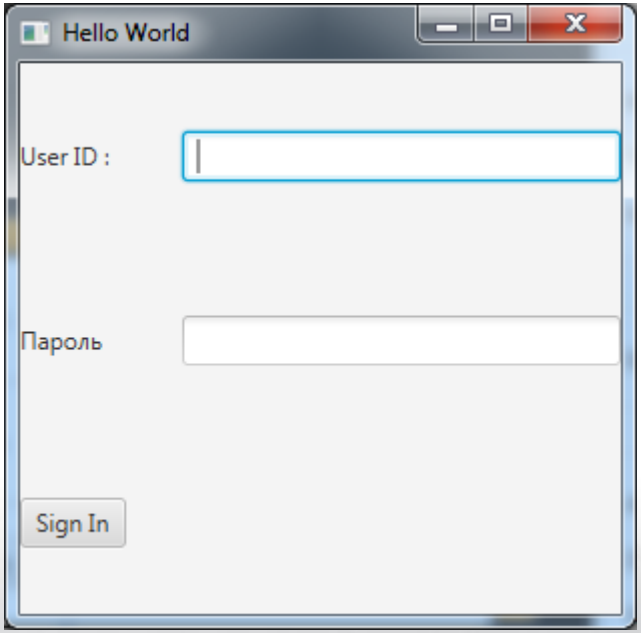
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Дизайн приложения в Scene Builder



Дизайн приложения в Scene Builder

Обратите внимание, что иерархия узлов показана в нижней левой панели Scene Builder. Если вы хотите изменить свойства компонента на сцене со сложным GUI, элемент можно выбрать на панели «Иерархия».



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Дизайн приложения в Scene Builder

- В Scene Builder панель «Properties» расположена справа и имеет три раздела: «Properties», «Layout» и «Code.
- На левой панели Scene Builder выберите GridPane, а выравнивание правой панели - TOP_LEFT.
- Введите 10 в поле padding, раздела Layout: сверху, справа, снизу и слева для GridPane. Нам нужно некоторое расстояние между границами сцены и сеткой.
- Выберите Button слева, а затем измените column span (ширину столбца) на 2, а pref width - на большое число, скажем 300. Это сделает кнопку широкой.
- Выберите первый столбец строки сетки, нажав на маленькое 0 поверх сетки. Установите для обеих сторон как pref и maximum width : 70.
- Выберите второй столбец строки сетки. Установите для обеих столбцов pref и maximum width: 100.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Дизайн приложения в Scene Builder

После того, как вы сделаете все эти изменения и сохраните их, файл sign.fxml будет выглядеть так:

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>

<GridPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="183.0" xmlns="http://javafx.com/javafx/8.0.60" xmlns:fx="http://javafx.com/fxml/1">
  <columnConstraints>
    <ColumnConstraints hgrow="SOMETIMES" maxWidth="70.0" minWidth="10.0" prefWidth="70.0"
/>
    <ColumnConstraints hgrow="SOMETIMES" maxWidth="100.0" minWidth="10.0"
prefWidth="100.0" />
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Дизайн приложения в Scene Builder

```
 </columnConstraints>
 <rowConstraints>
  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
 </rowConstraints>
 <children>
    <Label text="User ID :" />
    <Label text="Пароль " GridPane.rowIndex="1" />
    <TextField prefHeight="25.0" prefWidth="174.0" GridPane.columnIndex="1" />
    <Button mnemonicParsing="false" prefWidth="300.0" text="Sign In"
GridPane.columnSpan="2" GridPane.rowIndex="2" />
    <PasswordField GridPane.columnIndex="1" GridPane.rowIndex="1" />
  </children>
  <padding>
    <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
  </padding>
</GridPane>
```

События

Приложения JavaFX имеют дело с различными событиями. Например, нажатие кнопки, перемещение мыши и т. д. Когда событие происходит, объект события представлен одним из подклассов класса Event, расположенный в пакетах javafx.event и javafx.scene.input. Вам нужно решить, какие события важны для вашей программы, и, как говорят программисты, «слушать эти события». Есть разные способы назначения обработчиков событий элементам GUI. Если ваш GUI разработан с использованием FXML, вам нужно указать имя своего класса контроллера прямо в файле FXML. Если ваш GUI разработан с использованием классов Java, использование отдельного класса контроллера не обязательно, но это хорошая идея.

Назначение обработчиков событий в FXML

Scene Builder позволяет не только создавать и компоновать компоненты GUI на сцене, но также назначать Java-код, который будет передавать события. Мы будем придерживаться принципов разделения GUI и логики приложений. И сохраним GUI в файле .fxml, в то время как логика приложения будет запрограммирована на Java.

Обратите внимание на строку в файле sign.fxml, который включает тег GridPane в качестве корневого элемента:

```
<GridPane maxHeight="-Infinity" maxWidth="-Infinity"
      minHeight="-Infinity" minWidth="-Infinity"
      prefHeight="183.0"
      xmlns="http://javafx.com/javafx/8.0.60"
      xmlns:fx=http://javafx.com/fxml/1
      fx:controller = "sample.Controller">
```

Назначение обработчиков событий в FXML

Атрибуты hgap и vgap определяют заполнение между столбцами сетки и строками. Атрибуты xmlns определяют так называемые пространства имен, необходимые для правильного разбора файлов FXML. А атрибут fx: controller задает полное имя класса Controller, который в нашем проекте находится в пакете **sample**.

Когда файл sign.fxml загружен, FXMLLoader создает экземпляр класса Java на основе содержимого FXML, а затем создает экземпляр и вводит объект “sample.Controller” в объект GUI. Слово inject описывает процесс, когда среда выполнения автоматически создает экземпляр класса B и делает его доступным в классе A. Для нас это означает, что нет необходимости вручную создавать экземпляр sample.Controller с помощью нового оператора.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Назначение обработчиков событий в FXML

Класс Controller:

```
package sample;

import javafx.event.ActionEvent;

public class Controller {
    public void signInClicked(ActionEvent
        evt){
        System.out.println("Hello from
        signInClicked method in controller"); }
}
```

Назначение обработчиков событий в FXML

Контроллер класса использует класс ActionEvent из пакета javafx.event.

Мы хотим, чтобы метод signInClicked вызывался, если пользователь нажмет кнопку. Чтобы это произошло, мы укажем имя этого метода как обработчик свойства onAction в элементе <Button> в signin.fxml, как показано ниже.

```
<Button mnemonicParsing="false"
prefWidth="300.0" text="Sign In"
GridPane.columnSpan="2" GridPane.rowIndex="2"
   onAction="#signInClicked"/>
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Обмен данными между компонентами GUI и контроллером

Чтобы прочитать значения из компонентов GUI, нам нужно предоставить им уникальный идентификатор в файле FXML. Нам не нужно указывать id всем компонентам - только тем, которые мы хотим обработать в коде контроллера. Взгляните на раздел <childrens> файла sign.fxml:

```
<children>
   <Label text="User ID :" />
   <Label text="Пароль " GridPane.rowIndex="1" />
   <TextField prefHeight="25.0" prefWidth="174.0"
       GridPane.columnIndex="1" fx:id="id" />
   <Button mnemonicParsing="false" prefWidth="300.0"
text="Sign In" GridPane.columnSpan="2" GridPane.rowIndex="2"
      onAction="#signInClicked"/>
   <PasswordField GridPane.columnIndex="1"
       GridPane.rowIndex="1" fx:id="pwd"/>
</children>
```

Обмен данными между компонентами GUI и контроллером

- Только TextField и PasswordField имеют атрибут fx: id - так вы можете назначить id в FXML.
- Следующий шаг - добавить ссылки на эти компоненты GUI в переменные класса Controller.
- Следующий фрагмент кода показывает, как вводить ссылки на компоненты в поля id и pwd.
- Нужно объявить две переменные Java в классе Controller с именами, которые соответствуют именам из sign.fxml.
- Java имеет так называемые аннотации. Они начинаются с знака @ и могут быть помещены перед объявлением переменной, класса или метода в зависимости от того, как была определена аннотация.

Обмен данными между компонентами GUI и контроллером

- Некоторые аннотации используются компилятором Java, а некоторые - JVM. Аннотации JavaFX @FXML инструктируют среду выполнения внедрять ссылки на объекты GUI в переменные.
- Другими словами, среда выполнения JavaFX будет читать строку (1): «*Мне нужно получить объект, который был создан после загрузки файла sign.fxml, и найти компонент с fx: id =“id” внутри этого объекта. Потом, необходимо вставить ссылку на этот компонент в идентификатор переменной Java*". То же самое относится к переменной pwd.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Обмен данными между компонентами GUI и контроллером

Controller

```
1  package sample;
2
3  import javafx.event.ActionEvent;
4  import javafx.fxml.FXML;
5  import javafx.scene.control.PasswordField;
6  import javafx.scene.control.TextField;
7
8  public class Controller {
9      @FXML    private TextField id;          //(1)
10     @FXML    private PasswordField pwd;     //(1)
11     public void signInClicked(ActionEvent evt){
12         String userID = id.getText();       // (2)
13         String password = pwd.getText();    // (2)
14
15         if (!"Yakov".equals(userID)){       // (3)
16             id.setStyle("-fx-background-color:lightpink;"); //(4)
17         } else{
18             id.setStyle("-fx-background-color:white;");    // (5)
19         }
20
21         System.out.println("got id:" + userID + ", got password: " + password);
22
23         System.out.println("Hello from signInClicked method in controller");
24     }
25 }
26
```

Назначение обработчиков событий в Java

- Если ваша программа полностью написана на Java без использования FXML, тогда вы будете назначать обработчики событий, используя имена, начинающиеся с setOn: setOnAction, setOnKeyTyped, setOnEditStart и т. д.
- Вы можете писать код обработки событий с использованием анонимных внутренних классов, лямбда-выражений или ссылок на методы.
- Реализация обработчиков событий в виде лямбда-выражений проще, чем с анонимными внутренними классами.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Назначение обработчиков событий в Java

Способ 2:

Задание: пусть при наведении на кнопку в текстовое поле будет добавляться текст.

Для этого заходим в **Controller.java** и привязываемся к нашей кнопке через аннотацию **@FXML fx.id**:

Метод **initialize()** будет вызываться как только приложение запустится.

В **строке 14** мы добавляем **addEventHandler**, который будет определять событие и выполнять его.

MouseEvent. ***MOUSE_ENTERED*** – указывает, что это событие будет срабатывать при наведении на кнопку.

handle(MouseEvent mouseEvent**)** – в этом методе мы описываем действие, которое выполнится при наведении на кнопку

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Назначение обработчиков событий в Java

Main.java | Controller.java | sample.fxml

```
1  package sample;
2  import javafx.event.EventHandler;
3  import javafx.fxml.FXML;
4  import javafx.scene.control.Button;
5  import javafx.scene.control.TextArea;
6  import javafx.scene.input.MouseEvent;
7
8  public class Controller  {
9      @FXML
10     private Button button1;
11     @FXML
12     private TextArea text1;
13     public void initialize(){
14         button1.addEventHandler(MouseEvent.MOUSE_ENTERED, new EventHandler<MouseEvent>() {
15             @Override
16             public void handle(MouseEvent mouseEvent) {
17                 text1.appendText("Mouse!");
18             }
19         });
20     }
21
22 }
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Назначение обработчиков событий в Java

Способ 3:

Задание: пусть при нажатии на кнопку в текстовое поле будет добавляться текст.

Для этого класс **Controller** делаем реализующим интерфейс Initializable :

```
public class Controller   implements Initializable {}
```

После этого реализуем предлагаемые средой IDE методы:

```
@Override
public void initialize(URL location, ResourceBundle
resources) {}
```

В уже имеющемся методе **initialize** добавляем обработку нажатия на кнопку **button1**:

Назначение обработчиков событий в Java

При генерации событии, например, при нажатии на кнопку, создается объект **Event**, через который передается информация о событии. В данном случае при нажатии на кнопку будет генерироваться событие типа **javafx.event.ActionEvent** (который наследуется от Event).

Интерфейс **EventHandler** типизируется типом, который унаследован от класса **Event** и который по сути представляет событие. В нашем случае это класс **ActionEvent**.

Для прикрепления обработчика события **EventHandler** к событию элемента управления применяется метод **button1.setOnAction(),** в который передается реализация интерфейса **EventHandler**. В методе **handle** определяются действия, которые будут вызываться при нажатии на кнопку.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Назначение обработчиков событий в Java

Main.java | Controller.java | sample.fxml

```
2
3   import javafx.event.ActionEvent;
4   import javafx.event.EventHandler;
5   import javafx.fxml.FXML;
6   import javafx.fxml.Initializable;
7   import javafx.scene.control.Button;
8   import javafx.scene.control.TextArea;
9   import java.net.URL;
10  import java.util.ResourceBundle;
11
12  public class Controller   implements Initializable {
13      @FXML  private Button button1;
14      @FXML  private TextArea text1;
15
16      @Override
17      public void initialize(URL location, ResourceBundle resources) {
18          button1.setOnAction(new EventHandler<ActionEvent>() {
19              @Override
20              public void handle(ActionEvent event) {
21                  text1.appendText("You've clicked!");
22              }
23          });
24      }
25  }
26
```

Назначение обработчиков событий в Java

Способ 4:

Нередко вместо определения явной реализации интерфейса **EventHandler** применяются лямбда-выражения. Например, мы могли бы переписать метод **button1.setOnAction**() следующим образом:

```
package sample;
      import javafx.fxml.FXML;
      import javafx.scene.control.Button;
      import javafx.scene.control.TextArea;
public class Controller  {
   @FXML  private Button button1;
   @FXML  private TextArea text1;
   public void initialize() {
      button1.setOnAction(evt -> {
          text1.appendText("You've clicked!");
       });       }          }
```

Назначение обработчиков событий в Java

- Вышеприведенный фрагмент кода означает, что выражение лямбда получает объект события в качестве аргумента.
- Какой тип аргумента **evt**? Компилятор Java выяснит это автоматически. Поскольку метод **setOnAction** ожидает получить объект **ActionEvent** из JVM, компилятор угадывает, что тип **evt - ActionEvent**, поэтому вам даже не нужно записывать его в программный код.

Выражение Lambda позволяет создать анонимный класс, который реализует определенный тип интерфейса - функциональный интерфейс, который имеет один и только один абстрактный метод. Вы просто определяете типы параметров и тело метода, а компилятор Java сам описывает все остальное, основываясь на контексте, в котором вы используете Лямбда-выражение.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Questions?