 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №6**

з дисципліни «Крос-платформне програмування»

на тему: «Візуалізація даних. Побудова графіків»

Варіант №12

**Виконав:**

студент гр. БС-81

Погребенко В.О.

**Перевірив:**

ас. каф. БМК Рисін С.В.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2021

**Ціль**

Ознайомитися з можливостями побудови діаграм на платформі JavaFX; навчитися наочно представляти залежності між наборами даних у двовимірному вигляді; розібратися з організацією та роботою класу ScatterChart; побудувати точковий графік за набором даних двух змінних.

**Завдання:**

1. Завантажити дані з наданого файлу.

2. Створити серії на основі значення результату (0 – негативна серія, 1 – позитивна серія).

3. На основі вибору користувача (чи значеннями по замовчанню) створити осі з відповідним даним масштабом та екземпляр класу ScatterChart.

4. Додати відповідні серії даних до графіку з відображення їх назв.

5 Створити дві групи радіо-кнопок із значеннями, відповідними назвам змінних, по одній для кожної координатної осі.

6. Відобразити графік та елементи інтерфейсу відповідно до варіанту.

7. Надати знімок екрану з осями, відповідними варіанту

Таблиця вибору змінних за варіантом

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Змінна по X** | | | | | | | |
| **Змінна по Y** |  | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **1** | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **2** |  | 8 | 9 | 10 | 11 | 12 | 13 |
| **3** |  |  | 14 | 15 | 16 | 17 | 18 |
| **4** |  |  |  | 19 | 20 | 21 | 22 |
| **5** |  |  |  |  | 23 | 24 | 25 |
| **6** |  |  |  |  |  | 26 | 27 |
| **7** |  |  |  |  |  |  | 28 |

**Лістинг:**

**Весь код можна побачити на:**

https://github.com/PogrebenkoBS81/cross\_platform/tree/main/lab6

# **Package “pogrebenko.labsix.controller”:**

## **InitializationController.java:**

package pogrebenko.labsix.controller;  
  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.control.Alert;  
import javafx.scene.control.ButtonType;  
import javafx.scene.control.TextField;  
import javafx.stage.FileChooser;  
import javafx.stage.Stage;  
import pogrebenko.labsix.util.CommonUtil;  
import pogrebenko.labsix.util.JavaFxUtil;  
import pogrebenko.labsix.view.ViewRoutes;  
import pogrebenko.loggerwrapper.LoggerWrapper;  
  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileReader;  
import java.io.IOException;  
import java.net.URL;  
import java.util.ArrayList;  
import java.util.ResourceBundle;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
public class InitializationController {  
 private static final Logger *LOGGER* = LoggerWrapper.*getLogger*();  
  
 // Settings for log file browse window.  
 private final static String[] *extensionsLogFile* = {"\*.txt", "\*.log"};  
 private final static String *filterLabelLogFile* = "Choose or create \*.txt or \*.log file";  
  
 // Settings for data file browse window.  
 private final static String[] *extensionsDataFile* = {"\*.csv"};  
 private final static String *filterLabelDataFile* = "Choose \*.csv file";  
  
 @FXML // ResourceBundle that was given to the FXMLLoader  
 private ResourceBundle resources;  
  
 @FXML // URL location of the FXML file that was given to the FXMLLoader  
 private URL location;  
  
 @FXML // fx:id="tfLogFilePath"  
 private TextField tfLogFilePath; // Value injected by FXMLLoader  
  
 @FXML // fx:id="tfDataFilePath"  
 private TextField tfDataFilePath; // Value injected by FXMLLoader  
  
 @FXML  
 void onBrowseDataFile(ActionEvent event) {  
 *LOGGER*.info("Browsing data file...");  
 FileChooser fc = JavaFxUtil.*browseForFile*(*filterLabelDataFile*, *extensionsDataFile*);  
 File logFile = fc.showOpenDialog(new Stage());  
 // Check if any file has been selected.  
 if (logFile == null) {  
 *LOGGER*.info("No data file chosen...");  
  
 return;  
 }  
 // Get path to the selected file.  
 String absPath = logFile.getAbsolutePath();  
 tfDataFilePath.setText(logFile.getAbsolutePath());  
 *LOGGER*.info("Chosen data file is: " + absPath);  
 }  
  
 */\*\*  
 \* On log file browse event. Chooses log file to load in the app.  
 \*  
 \** ***@param*** *event browse event.  
 \*/* @FXML  
 void onBrowseLogFile(ActionEvent event) {  
 *LOGGER*.info("Browsing for log file...");  
 FileChooser fc = JavaFxUtil.*browseForFile*(*filterLabelLogFile*, *extensionsLogFile*);  
 File logFile = fc.showOpenDialog(new Stage());  
 // Check if any file has been selected.  
 if (logFile == null) {  
 *LOGGER*.info("No log file chosen...");  
  
 return;  
 }  
 // Get path to the selected file.  
 String absPath = logFile.getAbsolutePath();  
 tfLogFilePath.setText(logFile.getAbsolutePath());  
 *LOGGER*.info("Chosen log file is: " + absPath);  
 }  
  
 @FXML  
 void onContinuePressed(ActionEvent event) {  
 *LOGGER*.info("Continue button pressed, validating input...");  
 // Check if the selected files exist.  
 if (!CommonUtil.*isFileExists*(getLogFilePath())) {  
 *LOGGER*.severe("Given log file path is invalid: " + getLogFilePath());  
 new Alert(Alert.AlertType.*ERROR*, "Invalid log file path!", ButtonType.*YES*).showAndWait();  
  
 return;  
 }  
  
 if (!CommonUtil.*isFileExists*(getDataFilePath())) {  
 *LOGGER*.severe("Given data file path is invalid: " + getDataFilePath());  
 new Alert(Alert.AlertType.*ERROR*, "Invalid data file path!", ButtonType.*YES*).showAndWait();  
  
 return;  
 }  
  
 // Open main page, and load data to it.  
 try {  
 prepareMainController(event);  
 } catch (IOException e) {  
 *LOGGER*.log(Level.*INFO*, e.getMessage(), e);  
 new Alert(Alert.AlertType.*ERROR*, "Cannot start app! " + e.getMessage(), ButtonType.*YES*).showAndWait();  
 }  
 }  
  
 @FXML  
 void onExitPressed(ActionEvent event) {  
 // Alternatively i could use: Stage stage = (Stage)btnExit.getScene().getWindow(); stage.close();  
 *LOGGER*.info("Exit button on init page pressed...");  
  
 JavaFxUtil.*closeWindow*(event);  
 }  
  
 @FXML  
 // This method is called by the FXMLLoader when initialization is complete  
 void initialize() {  
 *LOGGER*.info("Initialization...");  
 }  
  
 private void prepareMainController(ActionEvent event) throws IOException {  
 *LOGGER*.info("Switching to the new window, loading scene...");  
 FXMLLoader loader = JavaFxUtil.*getSceneLoader*(ViewRoutes.*APP\_MAIN\_SCREEN*);  
 Parent root = loader.load();  
  
 *LOGGER*.info("Retrieving main controller and setting the init values...");  
 MainController mainController = loader.getController();  
  
 *LOGGER*.info("Setting main controller params...");  
 setMainControllerParams(mainController);  
  
 *LOGGER*.info("Preparing main controller...");  
 mainController.prepareController();  
  
 *LOGGER*.info("Opening new window...");  
 Stage mainStage = JavaFxUtil.*getNewStage*(root, Main.*MAIN\_SCREEN\_TITLE*);  
 mainStage.show();  
  
 *LOGGER*.info("Closing old window...");  
 JavaFxUtil.*closeWindow*(event);  
 }  
  
 private void setMainControllerParams(MainController mainController) throws IOException {  
 *LOGGER*.info("Setting the init values for the main controller...");  
 mainController.setLogPath(getLogFilePath());  
 mainController.setStringPlotData(loadCSV(getDataFilePath(), ","));  
 }  
  
 */\*\*  
 \* Loads data from the given file.  
 \*  
 \** ***@param*** *fileName file to load from.  
 \** ***@param*** *delimiter delimiters of the file.  
 \** ***@throws*** *IOException if some I/O error occurred.  
 \*/* private ArrayList<String[]> loadCSV(String fileName, String delimiter) throws IOException {  
 *LOGGER*.info(String.*format*("Loading data from: '%s', with delimiter: '%s'", fileName, delimiter));  
 ArrayList<String[]> values = new ArrayList<>();  
  
 File loadFile = new File(fileName);  
 FileReader reader = new FileReader(loadFile);  
  
 try (BufferedReader buffReader = new BufferedReader(reader)) {  
 for (String line = buffReader.readLine(); line != null; line = buffReader.readLine()) {  
 values.add(line.split(delimiter));  
 }  
 }  
  
 return values;  
 }  
  
 */\*\*  
 \* Return the path to the selected log file.  
 \*  
 \** ***@return*** *path to the selected log file.  
 \*/* public String getLogFilePath() {  
 *LOGGER*.finest("Retrieving log file path...");  
  
 return tfLogFilePath.getText();  
 }  
  
 */\*\*  
 \* Return the path to the selected log file.  
 \*  
 \** ***@return*** *path to the selected log file.  
 \*/* public String getDataFilePath() {  
 *LOGGER*.finest("Retrieving data file path...");  
  
 return tfDataFilePath.getText();  
 }  
}

## **MainController.java:**

package pogrebenko.labsix.controller;  
  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.scene.chart.NumberAxis;  
import javafx.scene.chart.ScatterChart;  
import javafx.scene.chart.XYChart;  
import javafx.scene.control.RadioButton;  
import javafx.scene.layout.AnchorPane;  
import pogrebenko.loggerwrapper.LoggerWrapper;  
  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.logging.Logger;  
  
public class MainController {  
 private static final Logger *LOGGER* = LoggerWrapper.*getLogger*();  
  
 private String logPath;  
 private ArrayList<Double[]> plotData;  
  
 private int selectedXAxis = 7;  
 private int selectedYAxis = 2;  
  
 private ScatterChart<Number, Number> scDataPlot; // Value injected by FXMLLoader  
  
 @FXML // fx:id="apPlotSite"  
 private AnchorPane apPlotSite; // Value injected by FXMLLoader  
  
 @FXML  
 void onXSelected(ActionEvent event) {  
 RadioButton selectedRadioButton = (RadioButton) event.getSource();  
 selectedXAxis = Integer.*parseInt*(selectedRadioButton.getId());  
  
 *LOGGER*.info("New X axis selected: " + selectedXAxis);  
 }  
  
 @FXML  
 void onYSelected(ActionEvent event) {  
 RadioButton selectedRadioButton = (RadioButton) event.getSource();  
 selectedYAxis = Integer.*parseInt*(selectedRadioButton.getId());  
  
 *LOGGER*.info("New Y axis selected: " + selectedXAxis);  
 }  
  
 @FXML  
 void onDataDraw(ActionEvent event) {  
 *LOGGER*.info(String.*format*("Drawing data by axis X: %d and Y: %d ", selectedXAxis, selectedYAxis));  
  
 drawScatterPlot();  
 }  
  
 @FXML  
 // This method is called by the FXMLLoader when initialization is complete  
 void initialize() {  
 *LOGGER*.info("Initializing....");  
  
 *LOGGER*.info("Creating scatter chart...");  
 scDataPlot = new ScatterChart<>(new NumberAxis(), new NumberAxis());  
 scDataPlot.setPrefSize(apPlotSite.getPrefWidth(), apPlotSite.getPrefHeight());  
  
 *LOGGER*.info("Adding scatter chart to plot");  
 apPlotSite.getChildren().add(scDataPlot);  
 }  
  
 private ChartSeries prepareDataSeries() {  
 *LOGGER*.info("Preparing data series...");  
  
 XYChart.Series<Number, Number> positiveSeries = new XYChart.Series<>();  
 positiveSeries.setName("Positive series");  
  
 XYChart.Series<Number, Number> negativeSeries = new XYChart.Series<>();  
 negativeSeries.setName("Negative series");  
  
 for (var stringPoints : getPlotData()) {  
 Double x = stringPoints[getSelectedXAxis()];  
 Double y = stringPoints[getSelectedYAxis()];  
  
 if (x == null || y == null) {  
 *LOGGER*.warning("Null value appeared during plot drawing!");  
  
 continue;  
 }  
  
 XYChart.Data<Number, Number> data = new XYChart.Data<>(x, y);  
  
 if (stringPoints[stringPoints.length - 1].equals(1.0)) {  
 positiveSeries.getData().add(data);  
 } else {  
 negativeSeries.getData().add(data);  
 }  
 }  
  
 return new ChartSeries(positiveSeries, negativeSeries);  
 }  
  
 private void drawScatterPlot() {  
 *LOGGER*.warning("Drawing new scatter plot...");  
  
 scDataPlot.getData().clear();  
 ChartSeries plotSeries = prepareDataSeries();  
  
 scDataPlot.getData().add(plotSeries.positiveSeries);  
 scDataPlot.getData().add(plotSeries.negativeSeries);  
 }  
  
 public void prepareController() throws IOException {  
 *LOGGER*.info("Preparing controller...");  
  
 LoggerWrapper.*getWrapper*().addFileHandler(getLogFilePath());  
 drawScatterPlot();  
 }  
  
  
 public void setLogPath(String logPath) {  
 *LOGGER*.finest("Setting log file path...");  
  
 this.logPath = logPath;  
 }  
  
 public String getLogFilePath() {  
 *LOGGER*.finest("Retrieving log file path...");  
  
 return logPath;  
 }  
  
 private Double parseDataDouble(String toParse) {  
 try {  
 return Double.*parseDouble*(toParse);  
 } catch (NumberFormatException | NullPointerException e) {  
 *LOGGER*.warning("Invalid double value to parse from string: " + toParse);  
 }  
  
 return null;  
 }  
  
 public void setStringPlotData(ArrayList<String[]> plotData) {  
 *LOGGER*.finest("Setting plot data...");  
 ArrayList<Double[]> doubleData = new ArrayList<>();  
  
 for (String[] dataRow : plotData) {  
 doubleData.add(  
 Arrays.*stream*(dataRow)  
 .map(this::parseDataDouble)  
 .toArray(Double[]::new)  
 );  
 }  
  
 setPlotData(doubleData);  
 }  
  
 private int getSelectedXAxis() {  
 return selectedXAxis - 1;  
 }  
  
 private int getSelectedYAxis() {  
 return selectedYAxis - 1;  
 }  
  
 public ArrayList<Double[]> getPlotData() {  
 *LOGGER*.finest("Retrieving plot data...");  
  
 return this.plotData;  
 }  
  
 public void setPlotData(ArrayList<Double[]> plotData) {  
 *LOGGER*.finest("Setting plot data...");  
  
 this.plotData = plotData;  
 }  
}  
  
class ChartSeries {  
 public final XYChart.Series<Number, Number> positiveSeries;  
 public final XYChart.Series<Number, Number> negativeSeries;  
  
 ChartSeries(XYChart.Series<Number, Number> positiveSeries, XYChart.Series<Number, Number> negativeSeries) {  
 this.positiveSeries = positiveSeries;  
 this.negativeSeries = negativeSeries;  
 }  
}

## **Main.java:**

package pogrebenko.labsix.controller;  
  
import javafx.application.Application;  
import javafx.stage.Stage;  
import pogrebenko.labsix.util.JavaFxUtil;  
import pogrebenko.labsix.view.ViewRoutes;  
import pogrebenko.loggerwrapper.LoggerWrapper;  
  
import java.util.logging.Level;  
  
// *TODO: Add JavaFx tasks on "hard" operations, add documentation.*public class Main extends Application {  
 private static final String *APP\_NAME* = "Data Plotter";  
 public static final String *MAIN\_SCREEN\_TITLE* = *APP\_NAME* + ": Main";  
 public static final String *INIT\_SCREEN\_TITLE* = *APP\_NAME* + ": Initialization";  
  
 // Initialize common logger.  
 static {  
 LoggerWrapper LOGGER\_WRAPPER = LoggerWrapper.*getWrapper*();  
 LOGGER\_WRAPPER.setLoggingLevel(Level.*INFO*);  
 LOGGER\_WRAPPER.setLogFormat();  
 }  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
  
 */\*\*  
 \* JavaFX entry method.  
 \*  
 \** ***@param*** *primaryStage - initial stage.  
 \*/* @Override  
 public void start(Stage primaryStage) throws Exception {  
 primaryStage.hide(); // Hide given window.  
 // And open initialization and log window.  
 // Alternatively i could just change scene of the primaryStage,  
 // But this approach with the new windows looks a little bit nicer.  
 JavaFxUtil.*openCleanWindow*(ViewRoutes.*APP\_INIT\_SCREEN*, *INIT\_SCREEN\_TITLE*);  
 }  
}

# **Package “pogrebenko.labsix.util”:**

## **CommonUtil.java:**

package pogrebenko.labsix.util;  
  
import java.io.File;  
  
*/\*\*  
 \* Util methods that may be used in various labs.  
 \*  
 \** ***@author*** *Pogrebenko Vasily, BS-81  
 \** ***@version*** *1.0  
 \** ***@since*** *1.0  
 \*/*public final class CommonUtil {  
 */\*\*  
 \* Private constructor that throws an exception to prevent instantiation.  
 \*/* private CommonUtil() {  
 throw new UnsupportedOperationException();  
 }  
  
 */\*\*  
 \* Checks if the given file exists.  
 \*  
 \** ***@param*** *fileName file name to check.  
 \** ***@return*** *true if file exists, false otherwise.  
 \*/* @SuppressWarnings("BooleanMethodIsAlwaysInverted")  
 public static boolean isFileExists(String fileName) {  
 if (fileName == null) {  
 return false;  
 }  
  
 File f = new File(fileName);  
  
 return f.exists() && !f.isDirectory();  
 }  
}

## **JavaFxUtil.java:**

package pogrebenko.labsix.util;  
  
import javafx.event.ActionEvent;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Node;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.stage.FileChooser;  
import javafx.stage.Stage;  
import pogrebenko.labsix.view.ViewRoutes;  
  
import java.io.IOException;  
  
*/\*\*  
 \* Frontend util methods that may be used in various labs.  
 \*  
 \** ***@author*** *Pogrebenko Vasily, BS-81  
 \** ***@version*** *1.0  
 \** ***@since*** *1.0  
 \*/*public class JavaFxUtil {  
 */\*\*  
 \* Private constructor that throws an exception to prevent instantiation.  
 \*/* private JavaFxUtil() {  
 throw new UnsupportedOperationException();  
 }  
  
 */\*\*  
 \* Creates "browse for file" dialog.  
 \*  
 \** ***@param*** *label label message of the dialog.  
 \** ***@param*** *extensions extensions to browse for.  
 \** ***@return*** *chosen file.  
 \*/* public static FileChooser browseForFile(String label, String... extensions) {  
 FileChooser fileChooser = new FileChooser();  
 fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter(label, extensions));  
  
 return fileChooser;  
 }  
  
 */\*\*  
 \* Returns FXML loader for given window.  
 \*  
 \** ***@param*** *route route of view to load.  
 \** ***@return*** *FXML loader for given window.  
 \*/* public static FXMLLoader getSceneLoader(ViewRoutes route) {  
 return new FXMLLoader(route.getURL());  
 }  
  
 public static Stage getNewStage(Parent root, String title) {  
 Stage stage = new Stage();  
 stage.setTitle(title);  
 stage.setScene(new Scene(root));  
 return stage;  
 }  
  
 */\*\*  
 \* Opens new window with given title, and sets it's scene to "root" scene.  
 \*  
 \** ***@param*** *root root scene to display in the new window.  
 \** ***@param*** *title title of the new window.  
 \*/* public static void openNewWindow(Parent root, String title) {  
 *getNewStage*(root, title).show();  
 }  
  
 */\*\*  
 \* Opens new window with view from given FXML route and given title.  
 \* No settings or fields is set here, so new window will be "clean".  
 \*  
 \** ***@param*** *route route of view to load.  
 \** ***@param*** *title title of the new window.  
 \*/* public static void openCleanWindow(ViewRoutes route, String title) throws IOException {  
 FXMLLoader loader = *getSceneLoader*(route);  
 *openNewWindow*(loader.load(), title);  
 }  
  
 */\*\*  
 \* Closes window via it's event.  
 \*  
 \** ***@param*** *event event of the window to close.  
 \*/* public static void closeWindow(ActionEvent event) {  
 ((Node) (event.getSource())).getScene().getWindow().hide();  
 }  
}

# **Package “pogrebenko.labsix.view”:**

## **app\_init.fxml**

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.text.Font?>  
  
<AnchorPane prefHeight="243.0" prefWidth="600.0" style="-fx-background-color: #ffffff;" xmlns="http://javafx.com/javafx/15.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="pogrebenko.labsix.controller.InitializationController">  
 <AnchorPane layoutX="15.0" layoutY="61.0" prefHeight="69.0" prefWidth="570.0" style="-fx-background-color: #ffffff;">  
 <TextField fx:id="tfLogFilePath" editable="false" layoutX="14.0" layoutY="22.0" prefHeight="25.0" prefWidth="435.0" text="D:\test.log" />  
 <Label layoutX="14.0" prefHeight="19.0" prefWidth="130.0" text="Specify your log file:">  
 <font>  
 <Font name="System Bold Italic" size="13.0" />  
 </font>  
 </Label>  
 <Button layoutX="466.0" layoutY="22.0" mnemonicParsing="false" onAction="#onBrowseLogFile" prefHeight="25.0" prefWidth="90.0" text="Browse" />  
 </AnchorPane>  
 <Button layoutX="313.0" layoutY="199.0" mnemonicParsing="false" onAction="#onContinuePressed" prefHeight="25.0" prefWidth="70.0" text="Continue" />  
 <Button layoutX="218.0" layoutY="199.0" mnemonicParsing="false" onAction="#onExitPressed" prefHeight="25.0" prefWidth="70.0" text="Exit" />  
 <Label layoutX="236.0" layoutY="26.0" prefHeight="25.0" prefWidth="169.0" style="-fx-background-color: #ffffff;" text="APP INITIALIZATION">  
 <font>  
 <Font name="System Bold" size="16.0" />  
 </font>  
 </Label>  
 <AnchorPane layoutX="15.0" layoutY="130.0" prefHeight="69.0" prefWidth="570.0" style="-fx-background-color: #ffffff;">  
 <children>  
 <TextField fx:id="tfDataFilePath" editable="false" layoutX="14.0" layoutY="22.0" prefHeight="25.0" prefWidth="435.0" />  
 <Label layoutX="14.0" prefHeight="19.0" prefWidth="169.0" text="Specify your data file:">  
 <font>  
 <Font name="System Bold Italic" size="13.0" />  
 </font>  
 </Label>  
 <Button layoutX="466.0" layoutY="22.0" mnemonicParsing="false" onAction="#onBrowseDataFile" prefHeight="25.0" prefWidth="90.0" text="Browse" />  
 </children>  
 </AnchorPane>  
</AnchorPane>

## **app\_main.fxml**

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.RadioButton?>  
<?import javafx.scene.control.ToggleGroup?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.ColumnConstraints?>  
<?import javafx.scene.layout.GridPane?>  
<?import javafx.scene.layout.RowConstraints?>  
<?import javafx.scene.text.Font?>  
  
<AnchorPane prefHeight="635.0" prefWidth="1197.0" xmlns="http://javafx.com/javafx/15.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="pogrebenko.labsix.controller.MainController">  
 <children>  
 <AnchorPane fx:id="apPlotSite" layoutX="14.0" layoutY="49.0" prefHeight="573.0" prefWidth="893.0" style="-fx-border-color: black;" />  
 <AnchorPane layoutX="919.0" layoutY="49.0" prefHeight="573.0" prefWidth="270.0" style="-fx-border-color: black;">  
 <children>  
 <GridPane alignment="CENTER" layoutX="2.0" layoutY="50.0" prefHeight="396.0" prefWidth="266.0">  
 <columnConstraints>  
 <ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" />  
 <ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" />  
 </columnConstraints>  
 <rowConstraints>  
 <RowConstraints minHeight="10.0" vgrow="SOMETIMES" />  
 <RowConstraints minHeight="10.0" vgrow="SOMETIMES" />  
 <RowConstraints minHeight="10.0" vgrow="SOMETIMES" />  
 <RowConstraints minHeight="10.0" vgrow="SOMETIMES" />  
 <RowConstraints minHeight="10.0" vgrow="SOMETIMES" />  
 <RowConstraints minHeight="10.0" vgrow="SOMETIMES" />  
 <RowConstraints minHeight="10.0" vgrow="SOMETIMES" />  
 <RowConstraints minHeight="10.0" vgrow="SOMETIMES" />  
 </rowConstraints>  
 <children>  
 <Label text="Y" GridPane.columnIndex="1">  
 <font>  
 <Font name="System Bold" size="17.0" />  
 </font>  
 </Label>  
 <Label prefHeight="22.0" prefWidth="3.0" text="X">  
 <font>  
 <Font name="System Bold" size="17.0" />  
 </font>  
 </Label>  
 <RadioButton id="1" mnemonicParsing="false" onAction="#onYSelected" text="1" GridPane.columnIndex="1" GridPane.rowIndex="1">  
 <toggleGroup>  
 <ToggleGroup fx:id="axisY" />  
 </toggleGroup>  
 </RadioButton>  
 <RadioButton id="2" mnemonicParsing="false" onAction="#onYSelected" selected="true" text="2" toggleGroup="$axisY" GridPane.columnIndex="1" GridPane.rowIndex="2" />  
 <RadioButton id="3" mnemonicParsing="false" onAction="#onYSelected" text="3" toggleGroup="$axisY" GridPane.columnIndex="1" GridPane.rowIndex="3" />  
 <RadioButton id="4" mnemonicParsing="false" onAction="#onYSelected" text="4" toggleGroup="$axisY" GridPane.columnIndex="1" GridPane.rowIndex="4" />  
 <RadioButton id="5" mnemonicParsing="false" onAction="#onYSelected" text="5" toggleGroup="$axisY" GridPane.columnIndex="1" GridPane.rowIndex="5" />  
 <RadioButton id="6" mnemonicParsing="false" onAction="#onYSelected" text="6" toggleGroup="$axisY" GridPane.columnIndex="1" GridPane.rowIndex="6" />  
 <RadioButton id="2" mnemonicParsing="false" onAction="#onXSelected" text="2" GridPane.rowIndex="1">  
 <toggleGroup>  
 <ToggleGroup fx:id="axisX" />  
 </toggleGroup>  
 </RadioButton>  
 <RadioButton id="3" mnemonicParsing="false" onAction="#onXSelected" text="3" toggleGroup="$axisX" GridPane.rowIndex="2" />  
 <RadioButton id="4" mnemonicParsing="false" onAction="#onXSelected" text="4" toggleGroup="$axisX" GridPane.rowIndex="3" />  
 <RadioButton id="5" mnemonicParsing="false" onAction="#onXSelected" text="5" toggleGroup="$axisX" GridPane.rowIndex="4" />  
 <RadioButton id="6" mnemonicParsing="false" onAction="#onXSelected" text="6" toggleGroup="$axisX" GridPane.rowIndex="5" />  
 <RadioButton id="7" mnemonicParsing="false" onAction="#onXSelected" selected="true" text="7" toggleGroup="$axisX" GridPane.rowIndex="6" />  
 <RadioButton id="8" mnemonicParsing="false" onAction="#onXSelected" text="8" toggleGroup="$axisX" GridPane.rowIndex="7" />  
 <RadioButton id="7" mnemonicParsing="false" onAction="#onYSelected" text="7" toggleGroup="$axisY" GridPane.columnIndex="1" GridPane.rowIndex="7" />  
 </children>  
 </GridPane>  
 <Button layoutX="83.0" layoutY="504.0" mnemonicParsing="false" onAction="#onDataDraw" prefHeight="26.0" prefWidth="104.0" text="Draw" />  
 <Label layoutX="14.0" layoutY="14.0" text="Data by axis:">  
 <font>  
 <Font name="System Bold" size="16.0" />  
 </font>  
 </Label>  
 </children>  
 </AnchorPane>  
 <Label layoutX="14.0" layoutY="14.0" prefHeight="26.0" prefWidth="173.0" text="Data Viewer">  
 <font>  
 <Font name="System Bold" size="18.0" />  
 </font>  
 </Label>  
 </children>  
</AnchorPane>

## **ViewRoutes.java**

package pogrebenko.labsix.view;  
  
import java.net.URL;  
  
*/\*\*  
 \* Routes of the different windows of the app.  
 \* Enum is used to ensure that only given routes will be used.  
 \*  
 \** ***@author*** *Pogrebenko Vasily, BS-81  
 \** ***@version*** *1.3.0  
 \** ***@since*** *1.2  
 \*/*public enum ViewRoutes {  
 // Initialize paths to the every view in the project.  
 *APP\_INIT\_SCREEN*("app\_init.fxml"),  
 *APP\_MAIN\_SCREEN*("app\_main.fxml");  
  
 private final URL url;  
  
 */\*\*  
 \* Constructs view url.  
 \*  
 \** ***@param*** *url url of the view.  
 \*/* ViewRoutes(final String url) {  
 this.url = getClass().getResource(url);  
 }  
  
 */\*\*  
 \* Returns the url of the view.  
 \*  
 \** ***@return*** *url of the view.  
 \*/* public URL getURL() {  
 return url;  
 }  
  
 */\*\*  
 \* Returns the path to the view.  
 \*  
 \** ***@return*** *string representation of the view url.  
 \*/* @Override  
 public String toString() {  
 return url.toString();  
 }  
}

# **Package “pogrebenko.loggerwrapper”**

package pogrebenko.loggerwrapper;  
  
import java.io.IOException;  
import java.util.logging.\*;  
  
*/\*\*  
 \* Wrapper for default logger.  
 \* Allows to simply log to a single file across different classes.  
 \*  
 \** ***@author*** *Pogrebenko Vasily, BS-81  
 \** ***@version*** *1.0  
 \** ***@since*** *1.0  
 \*/*public class LoggerWrapper {  
 // Init default logger.  
 private static final Logger *logger* = Logger.*getLogger*(LoggerWrapper.class.getName());  
 // Static wrapper variable for implementing singleton pattern.  
 private static LoggerWrapper *wrapper* = null;  
  
 */\*\*  
 \* Private constructor to ensure singleton.  
 \*/* private LoggerWrapper() {  
 }  
  
 */\*\*  
 \* Returns singleton instance of the logger wrapper.  
 \* No need in "synchronized" keyword for now,  
 \* but if app will be multithreaded, it will be necessary.  
 \*  
 \** ***@return*** *LoggerWrapper the singleton instance of the logger wrapper.  
 \*/* public static synchronized LoggerWrapper getWrapper() {  
 if (*wrapper* == null) {  
 *wrapper* = new LoggerWrapper();  
 }  
  
 return *wrapper*;  
 }  
  
 */\*\*  
 \* Returns singleton logger instance of the logger.  
 \*  
 \** ***@return*** *logger instance.  
 \*/* public static synchronized Logger getLogger() {  
 return *logger*;  
 }  
  
 */\*\*  
 \* Adds file handler to the logger.  
 \*  
 \** ***@param*** *LogName name of the log file.  
 \*/* public synchronized void addFileHandler(String LogName) throws IOException {  
 FileHandler fh = new FileHandler(LogName);  
 fh.setFormatter(new SimpleFormatter());  
 addHandler(fh);  
 }  
  
 */\*\*  
 \* Adds custom handler to the logger.  
 \*  
 \** ***@param*** *handler handler to add.  
 \*/* public synchronized void addHandler(Handler handler) {  
 *logger*.addHandler(handler);  
 }  
  
 */\*\*  
 \* Sets the logger logging level.  
 \*  
 \** ***@param*** *level logging level.  
 \*/* public synchronized void setLoggingLevel(Level level) {  
 *logger*.setLevel(level);  
 }  
  
 */\*\*  
 \* Sets the logger log format.  
 \*/* public synchronized void setLogFormat() {  
 // Set timestamp format for logger.  
 System.*setProperty*("java.util.logging.SimpleFormatter.format",  
 "%1$tY-%1$tm-%1$td %1$tH:%1$tM:%1$tS %4$s %2$s %5$s%6$s%n");  
 }  
}

**Результат роботи програми:**

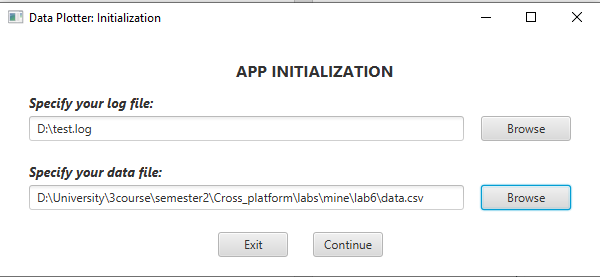


Рис. 1 – Ініціалізація додатку.

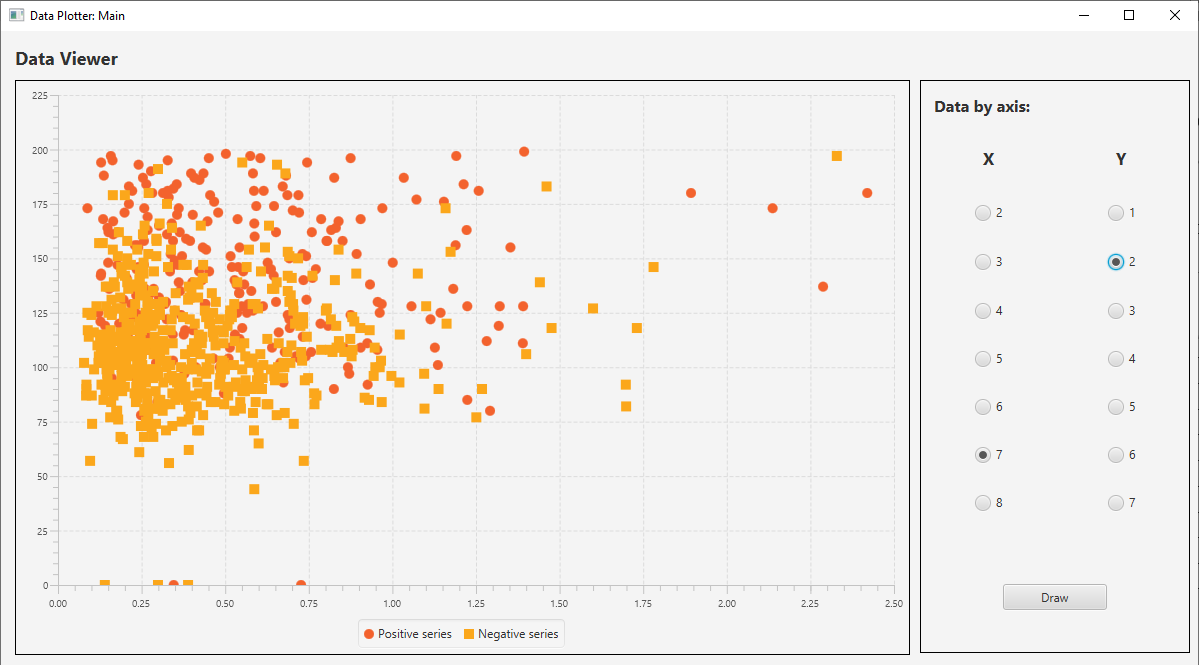


Рис. 2 – Початкове вікно.

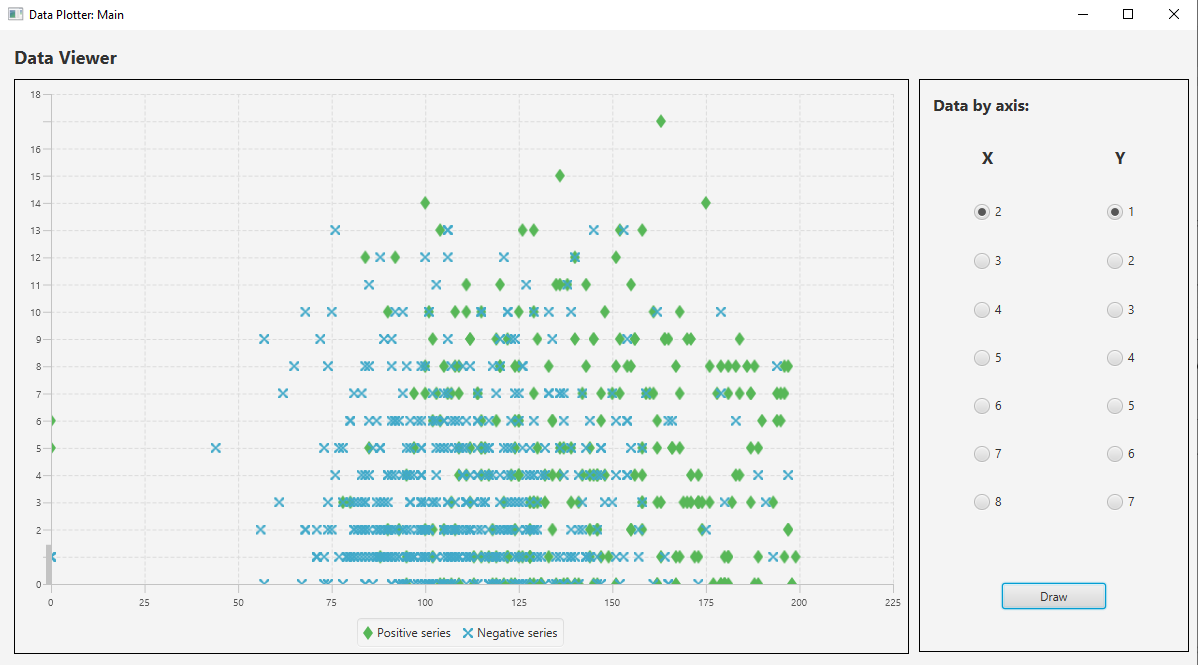


Рис. 3 – Вибір відображення данних по іншим вісям.

**Контрольні запитання:**

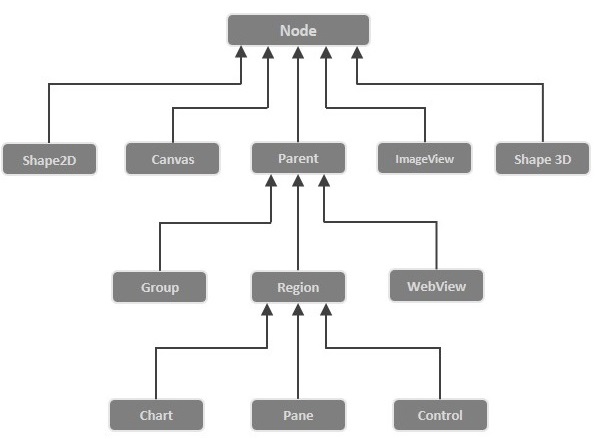
1. **В чому полягяють переваги платформи JavaFX над ії попередниками.**

JavaFX має підтримку налаштування за допомогою CSS та XML, за допомогою JavaFX можна також створювати тривимірні додатки, які, наприклад, в Swing або AWT дуже складні до реалізації.

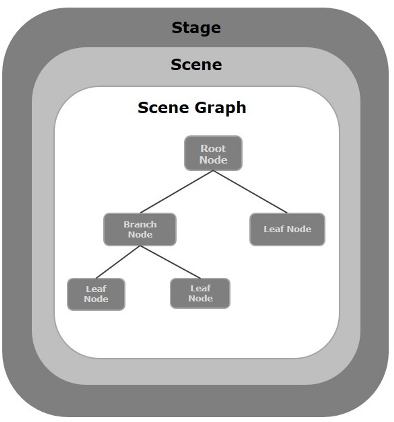
Крім того, в загалому, наявнйи більший функціонал, scene builder для зручного створення файлів розмітки, та інше.

1. **Опишіть основні частини JavaFX та іх призначення.**

Базовий об’єкт у Javafx – Node, від якого наслідуються усі інші обьекти. У Node визначені базові для всіх елементів методи, як, наприклад, isDisabled().



У загальному випадку, програмний додаток JavaFX буде мати три основні компоненти, а саме Stage, Scene та Nodes, як показано на наступній діаграмі:



Stage - містить усі об'єкти додатків JavaFX.

Scene представляє собою вміст додатків JavaFX.

Scene Graph - це древовидна структура даних (іерархічна), що представляє вміст сцени.

1. **В чому полягає відмінність роботи JavaFX та AWT в контексті потоків виконання додатку Java?**

У javafx існує JavaFX Application Thread, що обробляє усі завдання, пов'язані з графічним інтерфейсом. Так, наприклад, слід оновлювати всі компоненти інтерфейсу користувача в цьому потоці. Тривалі завдання, такі як Thread.sleep, ніколи не повинні виконуватися в цьому потоці, оскільки вікна програми зависають, а графічний інтерфейс буде заморожений.

У AWT існує event dispatch thread - це спеціальний потік, яким керує AWT. В основному, це потік, який працює в нескінченному циклі, обробляючи події.

*Відмінність:* JavaFx немає спеціального потоку відправлення. Просто JavaFX Application Thread.

1. **Назвіть основні модулі JavaFX та їх функціонал.**

Згідно https://openjfx.io/javadoc/11/:

* javafx.base - визначає базові API для набору інструментів JavaFX UI, включаючи API для прив'язок, властивостей, колекцій та подій.
* javafx.controls - визначає елементи керування, діаграми та обкладинки інтерфейсу, доступні для набору інструментів інтерфейсу JavaFX.
* javafx.fxml - визначає API FXML для набору інструментів JavaFX UI.
* javafx.graphics - Визначає основні API сценарію для набору інструментів JavaFX UI (наприклад, контейнери макета, життєвий цикл програми, фігури, перетворення, полотно, введення, малювання, обробка зображень та ефекти), а також API для анімації, css, паралельності , геометрія, друк та вікна.
* javafx.media - визначає API для відтворення мультимедіа та аудіовмісту, як частину набору інструментів JavaFX UI, включаючи MediaView та MediaPlayer.
* javafx.swing - Визначає API для підтримки взаємодії JavaFX / Swing, що входить до набору інструментів JavaFX UI, включаючи SwingNode (для вбудування Swing всередині програми JavaFX) та JFXPanel (для вбудовування JavaFX всередину програми Swing)
* javafx.web - Визначає API для функціональних можливостей WebView, що містяться в наборі інструментів JavaFX UI.

1. **Який механізм надається у JavaFX для можливостей реагування на дії користувача, системи та сигнали від зовнішніх пристроїв?**

Обробка подій (Event Handling) - це механізм, який контролює подію та вирішує, що має статися, якщо подія трапиться. Цей механізм має код, який відомий як обробник подій, який виконується при настанні події.

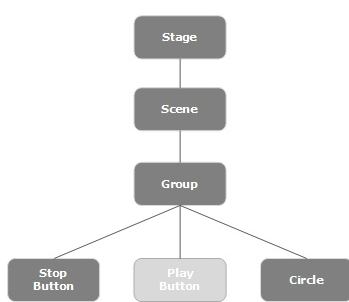
JavaFX надає обробники та фільтри для обробки подій. У JavaFX кожна подія має:

* Target - вузол, на якому відбулася подія. Ціль може бути вікном, сценою та вузлом.
* Source - джерелом, з якого генерується подія, буде джерелом події. У наведеному вище сценарії миша є джерелом події.
* Type - тип події, що сталася; у разі події миші - натискання миші, відпускання миші - це тип подій.

JavaFX надає підтримку для обробки різноманітних подій. Клас з назвою Event пакета javafx.event є базовим класом для події, таких як, наприклад, mouse event, key event, drag event, etc.

З етапів обробки подій у JavaFX можна виділити:

1. Route Construction - щоразу, коли генерується подія, початковий маршрут події визначається побудовою ланцюжка відправки подій. Це шлях від етапу до вихідного Вузла.



1. Event Capturing Phase - після побудови ланцюжка відправлення подій кореневий вузол програми відправляє подію. Ця подія переходить до всіх вузлів ланцюга відправлення (зверху вниз). Якщо на будь-якому з цих вузлів є зареєстрований фільтр для згенерованої події, він буде виконаний.
2. Event Bubbling Phase - події подія переходить від цільового вузла до вузла сцени (знизу вгору). Якщо на будь-якому з вузлів у ланцюжку відправлення подій є обробник, зареєстрований для згенерованої події, він буде виконаний. Якщо жоден з цих вузлів не має обробників для обробки події, тоді подія досягає кореневого вузла і нарешті процес буде завершено.
3. Event Handlers and Filters - фільтри та обробники подій, що містять логіку програми для обробки події. Вузол може зареєструватися більше ніж одним обробником / фільтром.
4. **Використання якого підходу дозволяє налаштовувати зовнішній вигляд додатку JavaFX?**

Це відбувається за допомогою FXML. FXML - це формат XML, який дозволяє створювати графічні інтерфейси JavaFX таким чином, як, наприклад, створюються веб-графічні інтерфейси в HTML. Таким чином, FXML дозволяє відокремити код макета JavaFX від решти коду програми. Це очищає як код верстки, так і решту коду програми.

1. **Перелічить види діаграм, які надає можливість будувати пакет javafx.scene.chart.**

Javafx chart підтримує наступні діаграми:

* Pie Chart - описує діаграму, яка представляє дані у формі кола, розділеного на трикутні клини, звані зрізами.
* Line Chart - описує лінійну діаграму, тип двовісної діаграми, що представляє дані як ряд точок, з’єднаних прямими лініями
* Area Chart - описує діаграму площ, яка представляє дані як область між рядом точок, з’єднаних прямими лініями, та віссю.
* Bubble Chart - описує бульбашкову діаграму, двовісну діаграму, яка наносить бульбашки для точок даних у серії.
* Scatter Chart - описує діаграму розсіювання, двовісну діаграму, яка представляє свої дані як набір точок.
* Bar Chart – описує гістограму, двовісну діаграму, яка представляє дискретні дані за допомогою прямокутних стовпчиків.

1. **Опишіть послідовність дій при побудові діаграми за допомогою ScatterChart.**

Для створення діаграмми ScatterChart, у загальному випадку (не зважаючи на типи <Number, Number>, etc), потрібно:

1. Створити 2 потрібних об’єкта Axis.
2. Створити екземпляр классу ScatterChart, у який передати об’єкти Axis з кроку 1.
3. Закріпити ScatterChart з кроку 2 у інтерфейсі користувача.
4. Свторити серії даних для графіку XYChart.Series, та заповнити ії точками XYChart.Data.
5. Додати серії даних з кроку 4 до ScatterChart з кроку 2.
6. Запустити програму.
7. **Надайте визначення FXML і Scene Builder та опишіть зв’язок між ними.**

FXML - це формат XML, який дозволяє створювати графічні інтерфейси JavaFX таким чином, як, наприклад, створюються веб-графічні інтерфейси в HTML. Таким чином, FXML дозволяє відокремити код макета JavaFX від решти коду програми. Це очищає як код верстки, так і решту коду програми.

JavaFX Scene Builder - це інструмент візуального макетування, який дозволяє користувачам швидко розробляти користувальницькі інтерфейси програм JavaFX.

*Зв’язок:*

За допомогою Scene Builder можливо створювати файли FXML, та не будувати інтерфейс «вручну» у коді. Таким чином, користувачі можуть перетягувати компоненти інтерфейсу користувача в робочу область, змінювати їх властивості, застосовувати таблиці стилів, а код FXML для макета, який вони створюють, автоматично створюється у фоновому режимі. Результат - файл FXML, який потім можна поєднати з проектом Java, прив’язавши інтерфейс користувача до логіки програми.

1. **Поясніть, як використовуються анотації @FXML.**

За допомогою аннотацій FXML можна зв’язати view (сутності у FXML файлах) з методами у контроллері, та опрацьовувати onClick, onDrag подіі, та не тільки.