Лабораторна робота №16

Розробка графічного інтерфейсу користувача

Мета: Придбання навичок використання засобів клієнтських технологій (Client Technologies) платформи Java SE.

1 ВИМОГИ

Розробити графічний інтерфейс користувача для програми рішення попередньої лабораторної роботи з використанням засобів JavaFX.

1.1 Розробник

- П.І.Б: Заночкин Є. Д.
- Группа: КІТ-119а
- Варіант: 7

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП:

@FXML – анотація, необхідна для того, щоб fxml файл мав доступ до приватних полів та приватних методів;

Pattern pattern = Pattern.compile() – компілює регулярний вираз у шаблон;

Matcher matcher = pattern.matcher(data); – створює matcher, який буде відповідати даному вводу для цього шаблону.

2.2 Ієрархія та структура класів

Було створено файли розмітки ClientEditDialog (вікно редагування або додавання нового клієнта), ClientOverview (вікно огляду клієнтів), RootLayout (обгортка файлу ClientOverview, що містить меню) та класи MainApp (головний клас програми), Client (клас-модель, шо містить інформацію про клієнта), ClientListWrapper (клас-обгортка, потрібний для збереження колекції у файл XML), DateUtil (клас, що перетворює дату у рядок), LocalDateAdapter (клас, що перетворює дату у формат XML),

ClientEditDialogController (клас-контролер вікна редагування або додавання клієнта), ClientOverviewController (клас-контролер огляду клієнтів), RootLayoutController (клас-контролер поведінки меню).

2.3 Важливі фрагменти програми

Файл ClientOverview

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ButtonBar?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.SplitPane?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>

<AnchorPane prefHeight="500.0" prefWidth="800.0" xmlns="http://javafx.com/javafx/15.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="ua.khpi.oop.zanochkyn16.view.ClientOverviewController">
  <children>
    <SplitPane dividerPositions="0.20050125313283207" prefHeight="502.0" prefWidth="803.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
      <items>
        <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="160.0" prefWidth="100.0">
          <children>
            <TableView fx:id="clientListTable" layoutX="-12.0" layoutY="39.0" prefHeight="469.0" prefWidth="230.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
              <columns>
                <TableColumn fx:id="genderColumn" prefWidth="75.0" text="Gender" />
                  <TableColumn fx:id="ageColumn" prefWidth="75.0" text="Age" />
              </columns>
               <columnResizePolicy>
                 <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
               </columnResizePolicy>
            </TableView>
          </children>
        </AnchorPane>
        <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="300.0" prefWidth="600.0">
          <children>
            <Label layoutX="14.0" layoutY="6.0" text="Client Information" AnchorPane.leftAnchor="5.0" AnchorPane.topAnchor="5.0" />
            <GridPane layoutX="80.0" layoutY="42.0" AnchorPane.leftAnchor="5.0" AnchorPane.rightAnchor="5.0" AnchorPane.topAnchor="40.0">
              <columnConstraints>
                <ColumnConstraints hgrow="SOMETIMES" maxWidth="332.20001220703125" minWidth="10.0" prefWidth="169.4000244140625" />
                <ColumnConstraints hgrow="SOMETIMES" maxWidth="518.6000213623047" minWidth="10.0" prefWidth="455.79997558593755" />
              </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 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 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 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="Client gender" GridPane.rowIndex="2" />
  <Label fx:id="idLabel" text="Label" GridPane.columnIndex="1" />
  <Label text="ID" />
  <Label fx:id="registrationDateLabel" text="Label" GridPane.columnIndex="1" GridPane.rowIndex="1" />
  <Label fx:id="clientGenderLabel" text="Label" GridPane.columnIndex="1" GridPane.rowIndex="2" />
  <Label text="Registration date" GridPane.rowIndex="1" />
  <Label fx:id="clientNameLabel" text="Label" GridPane.columnIndex="1" GridPane.rowIndex="3" />
  <Label fx:id="clientAgeLabel" text="Label" GridPane.columnIndex="1" GridPane.rowIndex="4" />
  <Label fx:id="clientHeightLabel" text="Label" GridPane.columnIndex="1" GridPane.rowIndex="5" />
  <Label text="Client name" GridPane.rowIndex="3" />
  <Label text="Client age" GridPane.rowIndex="4" />
  <Label text="Client height" GridPane.rowIndex="5" />
  <Label text="Client eye colour" GridPane.rowIndex="6" />
  <Label text="Client hobbies" GridPane.rowIndex="7" />
  <Label text="Partner gender" GridPane.rowIndex="8" />
  <Label text="Partner min age" GridPane.rowIndex="9" />
  <Label text="Partner max age" GridPane.rowIndex="10" />
  <Label text="Partner hobbies" GridPane.rowIndex="11" />
  <Label fx:id="clientEyeColourLabel" text="Label" GridPane.columnIndex="1" GridPane.rowIndex="6" />
  <Label fx:id="clientHobbiesLabel" text="Label" GridPane.columnIndex="1" GridPane.rowIndex="7" />
  <Label fx:id="partnerGenderLabel" text="Label" GridPane.columnIndex="1" GridPane.rowIndex="8" />
  <Label fx:id="partnerMinAgeLabel" text="Label" GridPane.columnIndex="1" GridPane.rowIndex="9" />
  <Label fx:id="partnerMaxAgeLabel" text="Label" GridPane.columnIndex="1" GridPane.rowIndex="10" />
  <Label fx:id="partnerHobbiesLabel" text="Label" GridPane.columnIndex="1" GridPane.rowIndex="11" />
 </children>
</GridPane>
<ButtonBar buttonMinWidth="50.0" layoutX="209.0" layoutY="257.0" prefHeight="26.0" prefWidth="375.0" AnchorPane.bottomAnchor="10.0" AnchorPane.rightAnchor="10.0">
  <buttons>
   <Button mnemonicParsing="false" onAction="#handleNewClient" text="New..." />
    <Button mnemonicParsing="false" onAction="#handleEditClient" text="Edit..." />
    <Button mnemonicParsing="false" onAction="#handleDeleteClient" text="Delete" />
  </buttons>
</ButtonBar>
</children>
```

```
      </AnchorPane>
    </items>
  </SplitPane>
 </children>
</AnchorPane>
```

Файл ClientEditDialog

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ButtonBar?>
<?import javafx.scene.control.ComboBox?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>

<AnchorPane prefHeight="446.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/15.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="ua.khpi.oop.zanochkyn16.view.ClientEditDialogController">
   <children>
      <GridPane layoutX="100.0" layoutY="18.0" prefWidth="400.0" AnchorPane.leftAnchor="40.0" AnchorPane.rightAnchor="40.0" AnchorPane.topAnchor="20.0">
        <columnConstraints>
          <ColumnConstraints hgrow="SOMETIMES" maxWidth="255.20001220703125" minWidth="10.0" prefWidth="148.0" />
          <ColumnConstraints hgrow="SOMETIMES" maxWidth="377.60003662109375" minWidth="10.0" prefWidth="372.0" />
        </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 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 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 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="ID" />
            <Label text="Registration date" GridPane.rowIndex="1" />
            <Label text="Client gender" GridPane.rowIndex="2" />
            <Label text="Client age" GridPane.rowIndex="4" />
            <Label text="Client height" GridPane.rowIndex="5" />
            <Label text="Client eye colour" GridPane.rowIndex="6" />
            <Label text="Client hobbies" GridPane.rowIndex="7" />
            <Label text="Partner gender" GridPane.rowIndex="8" />
            <Label text="Partner min age" GridPane.rowIndex="9" />
            <Label text="Partner max age" GridPane.rowIndex="10" />
            <Label text="Partner hobbies" GridPane.rowIndex="11" />
            <Label text="Client name" GridPane.rowIndex="3" />
            <TextField fx:id="idField" GridPane.columnIndex="1" />
            <TextField fx:id="registrationDateField" GridPane.columnIndex="1" GridPane.rowIndex="1" />
```

```
                <TextField fx:id="clientNameField" GridPane.columnIndex="1" GridPane.rowIndex="3" />
                <TextField fx:id="clientAgeField" GridPane.columnIndex="1" GridPane.rowIndex="4" />
                <TextField fx:id="clientHeightField" GridPane.columnIndex="1" GridPane.rowIndex="5" />
                <TextField fx:id="clientEyeColourField" GridPane.columnIndex="1" GridPane.rowIndex="6" />
                <TextField fx:id="clientHobbiesField" GridPane.columnIndex="1" GridPane.rowIndex="7" />
                <TextField fx:id="partnerMinAgeField" GridPane.columnIndex="1" GridPane.rowIndex="9" />
                <TextField fx:id="partnerMaxAgeField" GridPane.columnIndex="1" GridPane.rowIndex="10" />
                <TextField fx:id="partnerHobbiesField" GridPane.columnIndex="1" GridPane.rowIndex="11" />
                <ComboBox fx:id="clientGenderComboBox" prefWidth="150.0" GridPane.columnIndex="1"
GridPane.rowIndex="2" />
                <ComboBox fx:id="partnerGenderComboBox" prefWidth="150.0" GridPane.columnIndex="1"
GridPane.rowIndex="8" />
              </children>
            </GridPane>
            <ButtonBar layoutX="406.0" layoutY="446.0" prefHeight="40.0" prefWidth="165.0"
AnchorPane.bottomAnchor="10.0" AnchorPane.rightAnchor="10.0">
              <buttons>
                <Button mnemonicParsing="false" onAction="#handleOk" text="OK" />
                  <Button mnemonicParsing="false" onAction="#handleCancel" text="Cancel" />
              </buttons>
            </ButtonBar>
          </children>
        </AnchorPane>
```

Файл RootLayout

```
        <?xml version="1.0" encoding="UTF-8"?>

        <?import javafx.scene.control.Menu?>
        <?import javafx.scene.control.MenuBar?>
        <?import javafx.scene.control.MenuItem?>
        <?import javafx.scene.input.KeyCodeCombination?>
        <?import javafx.scene.layout.BorderPane?>

        <BorderPane prefHeight="500.0" prefWidth="800.0" xmlns="http://javafx.com/javafx/15.0.1"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="ua.khpi.oop.zanochkyn16.view.RootLayoutController">
          <top>
            <MenuBar BorderPane.alignment="CENTER">
              <menus>
                <Menu mnemonicParsing="false" text="File">
                    <items>
                    <MenuItem mnemonicParsing="false" onAction="#handleNew" text="New list" />
                      <MenuItem mnemonicParsing="false" onAction="#handleSaveXMLFile" text="Save to XML
file">
                        <accelerator>
                          <KeyCodeCombination alt="UP" code="S" control="DOWN" meta="UP" shift="UP"
shortcut="UP" />
                        </accelerator></MenuItem>
                      <MenuItem mnemonicParsing="false" onAction="#handleOpenXMLFile" text="Open from
XML file">
                        <accelerator>
                          <KeyCodeCombination alt="UP" code="O" control="DOWN" meta="UP" shift="UP"
shortcut="UP" />
                        </accelerator></MenuItem>
                    </items>
                </Menu>
                <Menu mnemonicParsing="false" text="Help">
                  <items>
                    <MenuItem mnemonicParsing="false" onAction="#handleAbout" text="About" />
                  </items>
```

```
            </Menu>
          </menus>
        </MenuBar>
      </top>
    </BorderPane>
```

Клас MainApp

```
package ua.khpi.oop.zanochkyn16;

import java.io.File;
import java.io.IOException;
import java.time.LocalDate;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.Marshaller;
import javax.xml.bind.Unmarshaller;

import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.image.Image;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.BorderPane;
import javafx.stage.Modality;
import javafx.stage.Stage;
import ua.khpi.oop.zanochkyn16.model.Client;
import ua.khpi.oop.zanochkyn16.model.ClientListWrapper;
import ua.khpi.oop.zanochkyn16.view.ClientEditDialogController;
import ua.khpi.oop.zanochkyn16.view.ClientOverviewController;
import ua.khpi.oop.zanochkyn16.view.RootLayoutController;

public class MainApp extends Application
{
        private ObservableList<Client> clientList = FXCollections.observableArrayList();
    private Stage primaryStage;
    private BorderPane rootLayout;

    @Override
    public void start(Stage primaryStage)
    {
        this.primaryStage = primaryStage;
        this.primaryStage.setTitle("Dating Office");
        this.primaryStage.getIcons().add(new Image("file:D:\\Eclipse_Workspace\\zanochkyn-
yehor\\src\\ua\\khpi\\oop\\zanochkyn16\\thumbnail.jpg"));
        initRootLayout();
        showClientOverview();
    }

    public MainApp()
    {
        clientList.add(new Client("Male", 1, LocalDate.now(), "Yehor", 19, 185, "Blue", "Video games,
Music", "Female", 18, 25, "None"));
        clientList.add(new Client("Female", 2, LocalDate.now(), "Kate", 18, 170, "Green", "Art", "Male",
18, 25, "Music"));
    }
```

```
        /**
         * Returns the data as an observable list of client's.
         * @return
         */
        public ObservableList<Client> getClientList()
        {
                return clientList;
        }

/**
 * Initializes the root layout.
 */
public void initRootLayout()
{
   try
   {
      // Load root layout from fxml file.
      FXMLLoader loader = new FXMLLoader();
      loader.setLocation(MainApp.class.getResource("view/RootLayout.fxml"));
      rootLayout = (BorderPane) loader.load();
      Scene scene = new Scene(rootLayout);
            primaryStage.setScene(scene);
            RootLayoutController controller = loader.getController();
            controller.setMainApp(this);
            primaryStage.show();
   }
   catch (IOException e)
   {
      e.printStackTrace();
   }
}

/**
 * Shows the client overview inside the root layout.
 */
public void showClientOverview()
{
   try
   {
      // Load client overview.
      FXMLLoader loader = new FXMLLoader();
      loader.setLocation(MainApp.class.getResource("view/ClientOverview.fxml"));
      AnchorPane clientListOverview = (AnchorPane) loader.load();

      // Set client overview into the center of root layout.
      rootLayout.setCenter(clientListOverview);

      ClientOverviewController controller = loader.getController();
      controller.setMainApp(this);
   }
   catch (IOException e)
   {
      e.printStackTrace();
   }
}

/**
 * Opens a dialog to edit details for the specified client. If the user
 * clicks OK, the changes are saved into the provided person object and true
```

```
     * is returned.
     *
     * @param client the client object to be edited
     * @return true if the user clicked OK, false otherwise.
     */
    public boolean showClientEditDialog(Client client)
    {
        try
        {
            // Load the fxml file and create a new stage for the popup dialog.
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(MainApp.class.getResource("view/ClientEditDialog.fxml"));
            AnchorPane page = (AnchorPane) loader.load();
            // Create the dialog Stage.
            Stage dialogStage = new Stage();
            dialogStage.setTitle("Edit Client");
            dialogStage.initModality(Modality.WINDOW_MODAL);
            dialogStage.initOwner(primaryStage);
            Scene scene = new Scene(page);
            dialogStage.setScene(scene);
            // Set the client into the controller.
            ClientEditDialogController controller = loader.getController();
            controller.setDialogStage(dialogStage);
            controller.setClient(client);
            // Show the dialog and wait until the user closes it
            dialogStage.showAndWait();
            return controller.isOkClicked();
        }
        catch (IOException e)
        {
            e.printStackTrace();
            return false;
        }
    }

    /**
     * Returns the main stage.
     * @return
     */
    public Stage getPrimaryStage()
    {
        return primaryStage;
    }

    public static void main(String[] args)
    {
        launch(args);
    }

    /**
     * Saves the current client data to the specified file.
     *
     * @param file
     */
    public void saveClientsToXMLFile(File file)
    {
        try
        {
            JAXBContext context = JAXBContext.newInstance(ClientListWrapper.class);
            Marshaller m = context.createMarshaller();
```

```
            m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
            // Wrapping our person data.
            ClientListWrapper wrapper = new ClientListWrapper();
            wrapper.setClients(clientList);
            // Marshalling and saving XML to the file.
            m.marshal(wrapper, file);
        }
        catch (Exception e)
        {    // catches ANY exception
             e.printStackTrace();
            Alert alert = new Alert(AlertType.ERROR);
            alert.setTitle("Error");
            alert.setHeaderText("Could not save data");
            alert.setContentText("Could not save data to file:\n" + file.getPath());
            alert.showAndWait();
        }
    }

    /**
     * Loads client data from the specified file. The current client data will
     * be replaced.
     */
    public void loadClientsFromXMLFile(File file)
    {
        try
        {
            JAXBContext context = JAXBContext.newInstance(ClientListWrapper.class);
            Unmarshaller um = context.createUnmarshaller();
            // Reading XML from the file and unmarshalling.
            ClientListWrapper wrapper = (ClientListWrapper) um.unmarshal(file);
            clientList.clear();
            clientList.addAll(wrapper.getClients());
        }
        catch (Exception e)
        {    // catches ANY exception
             e.printStackTrace();
            Alert alert = new Alert(AlertType.ERROR);
            alert.setTitle("Error");
            alert.setHeaderText("Could not load data");
            alert.setContentText("Could not load data from file:\n" + file.getPath());
            alert.showAndWait();
        }
    }
}
```

Клас Client

```
package ua.khpi.oop.zanochkyn16.model;

import java.time.LocalDate;
import java.util.Objects;

import javax.xml.bind.annotation.adapters.XmlJavaTypeAdapter;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.ObjectProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleObjectProperty;
import javafx.beans.property.SimpleStringProperty;
```

```
import javafx.beans.property.StringProperty;
import ua.khpi.oop.zanochkyn16.util.LocalDateAdapter;

public class Client
{
	/*
	 * Client info
	 */
	private StringProperty clientGender;
	private IntegerProperty id;
	private ObjectProperty<LocalDate> registrationDate;
	private StringProperty name;
	private IntegerProperty age;
	private IntegerProperty height;
	private StringProperty eyeColour;
	private StringProperty clientHobbies;

	/*
	 * Partner requirements
	 */
	private StringProperty partnerGender;
	private IntegerProperty minAge;
	private IntegerProperty maxAge;
	private StringProperty partnerHobbies;

	/*
	 * Constructors
	 */
	public Client(String clientGender, int id, LocalDate date, String name, int age, int height, String eyeColour,
			String clientHobbies, String partnerGender, int minAge, int maxAge, String partnerHobbies)
	{
		this.clientGender = new SimpleStringProperty(clientGender);
		this.id = new SimpleIntegerProperty(id);
		this.registrationDate = new SimpleObjectProperty<LocalDate>(date);
		this.name = new SimpleStringProperty(name);
		this.age = new SimpleIntegerProperty(age);
		this.height = new SimpleIntegerProperty(height);
		this.eyeColour = new SimpleStringProperty(eyeColour);
		this.clientHobbies = new SimpleStringProperty(clientHobbies);
		this.partnerGender = new SimpleStringProperty(partnerGender);
		this.minAge = new SimpleIntegerProperty(minAge);
		this.maxAge = new SimpleIntegerProperty(maxAge);
		this.partnerHobbies = new SimpleStringProperty(partnerHobbies);
	}

	public Client()
	{
		this(null, 0, LocalDate.now(), null, 0, 0, null, null, null, 0, 0, null);
	}

	/*
	 * Getters and setters
	 */
	public String getClientGender()
	{
		return clientGender.get();
	}
```

```
public void setClientGender(String gender)
{
	this.clientGender.set(gender);
}

public int getId()
{
	return id.get();
}

public void setId(int id)
{
	this.id.set(id);
}

@XmlJavaTypeAdapter(LocalDateAdapter.class)
public LocalDate getRegistrationDate()
{
	return registrationDate.get();
}

public void setRegistrationDate(LocalDate date)
{
	this.registrationDate.set(date);
}

public String getName()
{
	return name.get();
}

public void setName(String name)
{
	this.name.set(name);
}

public int getAge()
{
	return age.get();
}

public void setAge(int age)
{
	this.age.set(age);
}

public int getHeight()
{
	return height.get();
}

public void setHeight(int height)
{
	this.height.set(height);
}

public String getEyeColour()
{
	return eyeColour.get();
}
```

```
public void setEyeColour(String eyeColour)
{
	this.eyeColour.set(eyeColour);
}

public String getClientHobbies()
{
	return clientHobbies.get();
}

public void setClientHobbies(String clientHobbies)
{
	this.clientHobbies.set(clientHobbies);
}

public String getPartnerGender()
{
	return partnerGender.get();
}

public void setPartnerGender(String partnerGender)
{
	this.partnerGender.set(partnerGender);
}

public int getMinAge()
{
	return minAge.get();
}

public void setMinAge(int minAge)
{
	this.minAge.set(minAge);
}

public int getMaxAge()
{
	return maxAge.get();
}

public void setMaxAge(int maxAge)
{
	this.maxAge.set(maxAge);
}

public String getPartnerHobbies()
{
	return partnerHobbies.get();
}

public void setPartnerHobbies(String partnerHobbies)
{
	this.partnerHobbies.set(partnerHobbies);
}

public StringProperty clientGenderProperty()
{
	return clientGender;
}
```

```
public IntegerProperty idProperty()
{
	return id;
}

public ObjectProperty<LocalDate> registrationDateProperty()
{
	return registrationDate;
}

public StringProperty nameProperty()
{
	return name;
}

public IntegerProperty ageProperty()
{
	return age;
}

public IntegerProperty heightProperty()
{
	return height;
}

public StringProperty eyeColourProperty()
{
	return eyeColour;
}

public StringProperty clientHobbiesProperty()
{
	return clientHobbies;
}

public StringProperty partnerGenderProperty()
{
	return partnerGender;
}

public IntegerProperty minAgeProperty()
{
	return minAge;
}

public IntegerProperty maxAgeProperty()
{
	return maxAge;
}

public StringProperty partnerHobbiesProperty()
{
	return partnerHobbies;
}

@Override
public boolean equals(Object o)
{
   if (this == o)
```

```
                return true;
            if (o == null || getClass() != o.getClass())
                return false;
            Client client = (Client) o;
            return Objects.equals(getClientGender(), client.getClientGender()) && Objects.equals(getId(),
client.getId()) &&
                    Objects.equals(getRegistrationDate(), client.getRegistrationDate()) &&
Objects.equals(getName(), client.getName()) &&
                    Objects.equals(getAge(), client.getAge()) && Objects.equals(getHeight(),
client.getHeight()) && Objects.equals(getEyeColour(), client.getEyeColour()) &&
                    Objects.equals(getClientHobbies(), client.getClientHobbies()) &&
Objects.equals(getPartnerGender(), client.getPartnerGender()) &&
                    Objects.equals(getMinAge(), client.getMinAge()) &&
Objects.equals(getMaxAge(), client.getMaxAge()) && Objects.equals(getPartnerHobbies(),
client.getPartnerHobbies());
        }
    }
```

Клас ClientListWrapper

```
package ua.khpi.oop.zanochkyn16.model;

import java.util.List;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

/**
 * Helper class to wrap a list of clients. This is used for saving the
 * list of persons to XML.
 */
@XmlRootElement(name = "clients")
public class ClientListWrapper
{
    private List<Client> clients;

    @XmlElement(name = "client")
    public List<Client> getClients()
    {
        return clients;
    }

    public void setClients(List<Client> clients)
    {
        this.clients = clients;
    }
}
```

Клас DateUtil

```
package ua.khpi.oop.zanochkyn16.util;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;

public class DateUtil
{
    private static final String DATE_PATTERN = "dd.MM.yyyy";
```

```
        private static final DateTimeFormatter DATE_FORMATTER = DateTimeFormatter.ofPattern(DATE_PATTERN);

        /**
     * Returns the given date as a well formatted String. The above defined
     * {@link DateUtil#DATE_PATTERN} is used.
     *
     * @param date the date to be returned as a string
     * @return formatted string
     */
    public static String format(LocalDate date)
    {
        if (date == null)
            return null;
        return DATE_FORMATTER.format(date);
    }

    /**
     * Converts a String in the format of the defined {@link DateUtil#DATE_PATTERN}
     * to a {@link LocalDate} object.
     *
     * Returns null if the String could not be converted.
     *
     * @param dateString the date as String
     * @return the date object or null if it could not be converted
     */
    public static LocalDate parse(String dateString)
    {
        try
        {
            return DATE_FORMATTER.parse(dateString, LocalDate::from);
        }
        catch (DateTimeParseException e)
        {
            return null;
        }
    }

    /**
     * Checks the String whether it is a valid date.
     *
     * @param dateString
     * @return true if the String is a valid date
     */
    public static boolean validDate(String dateString)
    {
            return DateUtil.parse(dateString) != null;
    }
}
```

Клас LocalDateAdapter

```
package ua.khpi.oop.zanochkyn16.util;

import java.time.LocalDate;

import javax.xml.bind.annotation.adapters.XmlAdapter;

public class LocalDateAdapter extends XmlAdapter<String, LocalDate>
{
        @Override
```

```
        public LocalDate unmarshal(String v) throws Exception
        {
                return LocalDate.parse(v);
        }

        @Override
        public String marshal(LocalDate v) throws Exception
        {
                return v.toString();
        }
}
```

Клас ClientEditDialogController

```
package ua.khpi.oop.zanochkyn16.view;

import java.time.LocalDate;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.ComboBox;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
import ua.khpi.oop.zanochkyn16.model.Client;
import ua.khpi.oop.zanochkyn16.util.DateUtil;

/**
 * Dialog to edit details of a client.
 */
public class ClientEditDialogController
{
    @FXML private TextField idField;
    @FXML private TextField registrationDateField;
    @FXML private TextField clientNameField;
    @FXML private TextField clientAgeField;
    @FXML private TextField clientHeightField;
    @FXML private TextField clientEyeColourField;
    @FXML private TextField clientHobbiesField;
    @FXML private TextField partnerMinAgeField;
    @FXML private TextField partnerMaxAgeField;
    @FXML private TextField partnerHobbiesField;
    @FXML private ComboBox<String> clientGenderComboBox;
    @FXML private ComboBox<String> partnerGenderComboBox;

    private Stage dialogStage;
    private Client client;
    private boolean okClicked = false;
    private static int ID = 3;

    /**
     * Initializes the controller class. This method is automatically called
     * after the fxml file has been loaded.
     */
    @FXML
    private void initialize()
    {
```

```
        ObservableList<String> langs = FXCollections.observableArrayList("Male", "Female");
    clientGenderComboBox.setItems(langs);
    partnerGenderComboBox.setItems(langs);
}

/**
 * Sets the stage of this dialog.
 *
 * @param dialogStage
 */
public void setDialogStage(Stage dialogStage) { this.dialogStage = dialogStage; }

/**
 * Sets the client to be edited in the dialog.
 *
 * @param client
 */
public void setClient(Client client)
{
    this.client = client;
    if(client.getClientGender() != null)
    {
                idField.setText(Integer.toString(client.getId()));
                registrationDateField.setText(DateUtil.format(client.getRegistrationDate()));
                registrationDateField.setPromptText("dd.mm.yyyy");
                clientGenderComboBox.getSelectionModel().select(client.getClientGender());
                clientNameField.setText(client.getName());
                clientAgeField.setText(Integer.toString(client.getAge()));
                clientHeightField.setText(Integer.toString(client.getHeight()));
                clientEyeColourField.setText(client.getEyeColour());
                clientHobbiesField.setText(client.getClientHobbies());
                partnerGenderComboBox.getSelectionModel().select(client.getPartnerGender());
            partnerMinAgeField.setText(Integer.toString(client.getMinAge()));
            partnerMaxAgeField.setText(Integer.toString(client.getMaxAge()));
            partnerHobbiesField.setText(client.getPartnerHobbies());
    }
    else
    {
        idField.setText(Integer.toString(ID++));
                registrationDateField.setText(DateUtil.format(LocalDate.now()));
                clientGenderComboBox.setValue("");
                clientNameField.setText("");
                clientAgeField.setText("");
                clientHeightField.setText("");
                clientEyeColourField.setText("");
                clientHobbiesField.setText("");
                partnerGenderComboBox.setValue("");
            partnerMinAgeField.setText("");
            partnerMaxAgeField.setText("");
            partnerHobbiesField.setText("");
    }
}

/**
 * Returns true if the user clicked OK, false otherwise.
 *
 * @return
 */
public boolean isOkClicked() { return okClicked; }
```

```
    /**
     * Called when the user clicks ok.
     */
    @FXML
    private void handleOk()
    {
       if (isInputValid())
       {
            client.setId(Integer.parseInt(idField.getText()));
            client.setRegistrationDate(DateUtil.parse(registrationDateField.getText()));
            client.setClientGender(clientGenderComboBox.getValue());
            client.setName(clientNameField.getText());
            client.setAge(Integer.parseInt(clientAgeField.getText()));
            client.setHeight(Integer.parseInt(clientHeightField.getText()));
            client.setEyeColour(clientEyeColourField.getText());
            client.setClientHobbies(clientHobbiesField.getText());
            client.setPartnerGender(partnerGenderComboBox.getValue());
            client.setMinAge(Integer.parseInt(partnerMinAgeField.getText()));
            client.setMaxAge(Integer.parseInt(partnerMaxAgeField.getText()));
            client.setPartnerHobbies(partnerHobbiesField.getText());
          okClicked = true;
          dialogStage.close();
       }
    }

    /**
     * Called when the user clicks cancel.
     */
    @FXML
    private void handleCancel() { dialogStage.close(); }

    /**
     * Validates the user input in the text fields.
     *
     * @return true if the input is valid
     */
    private boolean isInputValid()
    {
       String errorMessage = "";
       Pattern patternId = Pattern.compile("\\d+");
       Pattern patternRegistrationDay = Pattern.compile("(0[1-9]|[12][0-9]|3[01])\\.(0[1-9]|1[0-
2])\\.(19|20)\\d{2}");
       Pattern patternName = Pattern.compile("^([a-zA-Z]+)");
                Pattern patternAge = Pattern.compile("^(([1-9])|([1-9][0-9]))");
                Pattern patternHeight = Pattern.compile("^(([1-9])|([1-9][0-9])|([1-2][0-9][0-9]))");
                Pattern patternEyeColour = Pattern.compile("^([a-zA-Z]+)");
                Pattern patternHobby = Pattern.compile("^(([a-zA-Z]+|[a-zA-Z]+\\s[a-zA-Z]+)(,\\s([a-zA-
Z]+|[a-zA-Z]+\\s[a-zA-Z]+))*)");

                Matcher matcher = patternId.matcher(idField.getText());
       if (!matcher.matches())
          errorMessage += "No valid ID!\n";

       matcher = patternRegistrationDay.matcher(registrationDateField.getText());
       if (!matcher.matches())
          errorMessage += "No valid registration date!\n";

       matcher = patternName.matcher(clientNameField.getText());
       if (!matcher.matches())
          errorMessage += "No valid name!\n";
```

```
        matcher = patternAge.matcher(clientAgeField.getText());
        if (!matcher.matches())
            errorMessage += "No valid age!\n";

        matcher = patternHeight.matcher(clientHeightField.getText());
        if (!matcher.matches())
            errorMessage += "No valid height!\n";

        matcher = patternEyeColour.matcher(clientEyeColourField.getText());
        if (!matcher.matches())
            errorMessage += "No valid eye colour!\n";

        matcher = patternHobby.matcher(clientHobbiesField.getText());
        if (!matcher.matches())
            errorMessage += "No valid client hobbies!\n";

        matcher = patternAge.matcher(partnerMinAgeField.getText());
        if (!matcher.matches())
            errorMessage += "No valid min age!\n";

        matcher = patternAge.matcher(partnerMaxAgeField.getText());
        if (!matcher.matches())
            errorMessage += "No valid max age!\n";

        matcher = patternHobby.matcher(partnerHobbiesField.getText());
        if (!matcher.matches())
            errorMessage += "No valid partner hobbies!\n";

        if (errorMessage.length() == 0)
            return true;
        else
        {
            // Show the error message.
            Alert alert = new Alert(AlertType.ERROR);
            alert.initOwner(dialogStage);
            alert.setTitle("Invalid Fields");
            alert.setHeaderText("Please correct invalid fields");
            alert.setContentText(errorMessage);
            alert.showAndWait();
            return false;
        }
    }
}
```

Клас ClientOverviewController

```
package ua.khpi.oop.zanochkyn16.view;

import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import ua.khpi.oop.zanochkyn16.MainApp;
import ua.khpi.oop.zanochkyn16.model.Client;
import ua.khpi.oop.zanochkyn16.util.DateUtil;

public class ClientOverviewController
{
```

```
        @FXML private TableView<Client> clientListTable;
        @FXML private TableColumn<Client, String> genderColumn;
        @FXML private TableColumn<Client, Integer> ageColumn;
        @FXML private Label clientGenderLabel;
@FXML private Label idLabel;
@FXML private Label registrationDateLabel;
@FXML private Label clientNameLabel;
@FXML private Label clientAgeLabel;
@FXML private Label clientHeightLabel;
@FXML private Label clientEyeColourLabel;
@FXML private Label clientHobbiesLabel;
@FXML private Label partnerGenderLabel;
@FXML private Label partnerMinAgeLabel;
@FXML private Label partnerMaxAgeLabel;
@FXML private Label partnerHobbiesLabel;

private MainApp mainApp;

public ClientOverviewController() {}

@FXML
private void initialize()
{
        genderColumn.setCellValueFactory(cellData -> cellData.getValue().clientGenderProperty());
        ageColumn.setCellValueFactory(cellData -> cellData.getValue().ageProperty().asObject());

        // Clear client details.
    showClientDetails(null);

    // Listen for selection changes and show the client details when changed.
    clientListTable.getSelectionModel().selectedItemProperty().addListener(
          (observable, oldValue, newValue) -> showClientDetails(newValue));
}

/**
 * Is called by the main application to give a reference back to itself.
 *
 * @param mainApp
 */
public void setMainApp(MainApp mainApp)
{
    this.mainApp = mainApp;
    clientListTable.setItems(mainApp.getClientList());
}

/**
 * Fills all text fields to show details about the client.
 * If the specified client is null, all text fields are cleared.
 *
 * @param client the client or null
 */
private void showClientDetails(Client client)
{
    if (client != null)
    {
       // Fill the labels with info from the person object.
        clientGenderLabel.setText(client.getClientGender());
        idLabel.setText(Integer.toString(client.getId()));
        registrationDateLabel.setText(DateUtil.format(client.getRegistrationDate()));
        clientNameLabel.setText(client.getName());
```

```
            clientAgeLabel.setText(Integer.toString(client.getAge()));
            clientHeightLabel.setText(Integer.toString(client.getHeight()));
            clientEyeColourLabel.setText(client.getEyeColour());
            clientHobbiesLabel.setText(client.getClientHobbies());
            partnerGenderLabel.setText(client.getPartnerGender());
          partnerMinAgeLabel.setText(Integer.toString(client.getMinAge()));
          partnerMaxAgeLabel.setText(Integer.toString(client.getMaxAge()));
          partnerHobbiesLabel.setText(client.getPartnerHobbies());
      }
      else
      {
          // Person is null, remove all the text.
            clientGenderLabel.setText("");
            idLabel.setText("");
            registrationDateLabel.setText("");
            clientNameLabel.setText("");
            clientAgeLabel.setText("");
            clientHeightLabel.setText("");
            clientEyeColourLabel.setText("");
            clientHobbiesLabel.setText("");
            partnerGenderLabel.setText("");
          partnerMinAgeLabel.setText("");
          partnerMaxAgeLabel.setText("");
          partnerHobbiesLabel.setText("");
      }
  }

  /**
   * Called when the user clicks on the delete button.
   */
  @FXML
  private void handleDeleteClient()
  {
      int selectedIndex = clientListTable.getSelectionModel().getSelectedIndex();
      if (selectedIndex >= 0)
          clientListTable.getItems().remove(selectedIndex);
      else
      {
          // Nothing selected.
          Alert alert = new Alert(AlertType.WARNING);
          alert.initOwner(mainApp.getPrimaryStage());
          alert.setTitle("No Selection");
          alert.setHeaderText("No Client Selected");
          alert.setContentText("Please select a client in the table.");
          alert.showAndWait();
      }
  }

  /**
   * Called when the user clicks the new button. Opens a dialog to edit
   * details for a new client.
   */
  @FXML
  private void handleNewClient()
  {
      Client tempPerson = new Client();
      boolean result = false;
      boolean okClicked = mainApp.showClientEditDialog(tempPerson);
      if (okClicked)
      {
```

```
            for (Client element : mainApp.getClientList())
            {
                    result = tempPerson.equals(element);
                    if (result)
                            break;
            }
            if (!result)
                    mainApp.getClientList().add(tempPerson);
          else
          {
            Alert alert = new Alert(AlertType.INFORMATION);
            alert.initOwner(mainApp.getPrimaryStage());
              alert.setTitle("Result of search");
              alert.setHeaderText("This element is in the list");
              alert.showAndWait();
          }
       }
    }

    /**
     * Called when the user clicks the edit button. Opens a dialog to edit
     * details for the selected client.
     */
    @FXML
    private void handleEditClient()
    {
       Client selectedPerson = clientListTable.getSelectionModel().getSelectedItem();
       if (selectedPerson != null)
       {
          boolean okClicked = mainApp.showClientEditDialog(selectedPerson);
          if (okClicked)
             showClientDetails(selectedPerson);
       }
       else
       {
          // Nothing selected.
          Alert alert = new Alert(AlertType.WARNING);
          alert.initOwner(mainApp.getPrimaryStage());
          alert.setTitle("No Selection");
          alert.setHeaderText("No Client Selected");
          alert.setContentText("Please select a client in the table.");
          alert.showAndWait();
       }
    }
}
```

Клас RootLayoutController

```
package ua.khpi.oop.zanochkyn16.view;

import java.io.File;

import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.stage.FileChooser;
import ua.khpi.oop.zanochkyn16.MainApp;

/**
 * The controller for the root layout. The root layout provides the basic
 * application layout containing a menu bar and space where other JavaFX
```

```
 * elements can be placed.
 */
public class RootLayoutController
{
    private MainApp mainApp;

    /**
     * Is called by the main application to give a reference back to itself.
     *
     * @param mainApp
     */
    public void setMainApp(MainApp mainApp)
    {
        this.mainApp = mainApp;
    }

    /**
     * Creates an empty address book.
     */
    @FXML
    private void handleNew()
    {
        mainApp.getClientList().clear();
    }

    @FXML
    private void handleSaveXMLFile()
    {
        FileChooser fileChooser = new FileChooser();
        FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter("XML files (*.xml)",
"*.xml");
        fileChooser.getExtensionFilters().add(extFilter);
        File file = fileChooser.showSaveDialog(mainApp.getPrimaryStage());
        if (file != null)
        {
            if (!file.getPath().endsWith(".xml"))
                file = new File(file.getPath() + ".xml");
            mainApp.saveClientsToXMLFile(file);
        }
    }

    @FXML
    private void handleOpenXMLFile()
    {
        FileChooser fileChooser = new FileChooser();
        FileChooser.ExtensionFilter extFilter = new FileChooser.ExtensionFilter("XML files (*.xml)",
"*.xml");
        fileChooser.getExtensionFilters().add(extFilter);
        File file = fileChooser.showOpenDialog(mainApp.getPrimaryStage());
        if (file != null)
            mainApp.loadClientsFromXMLFile(file);
    }

    @FXML
    private void handleAbout()
    {
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("About window");
        alert.setHeaderText("About");
        alert.setContentText("Author: Zanochkyn Yehor\nGroup: KIT-119a");
```

```
        alert.showAndWait();
    }
}
```

3 РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ

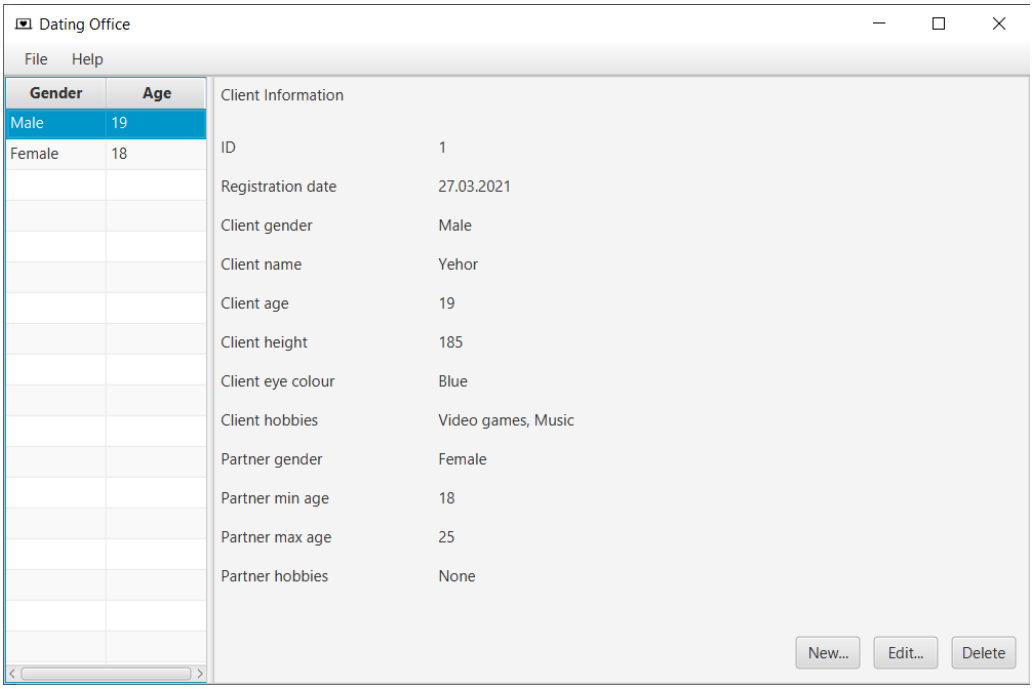

Рисунок 16.1 – Вікно огляду клієнтів

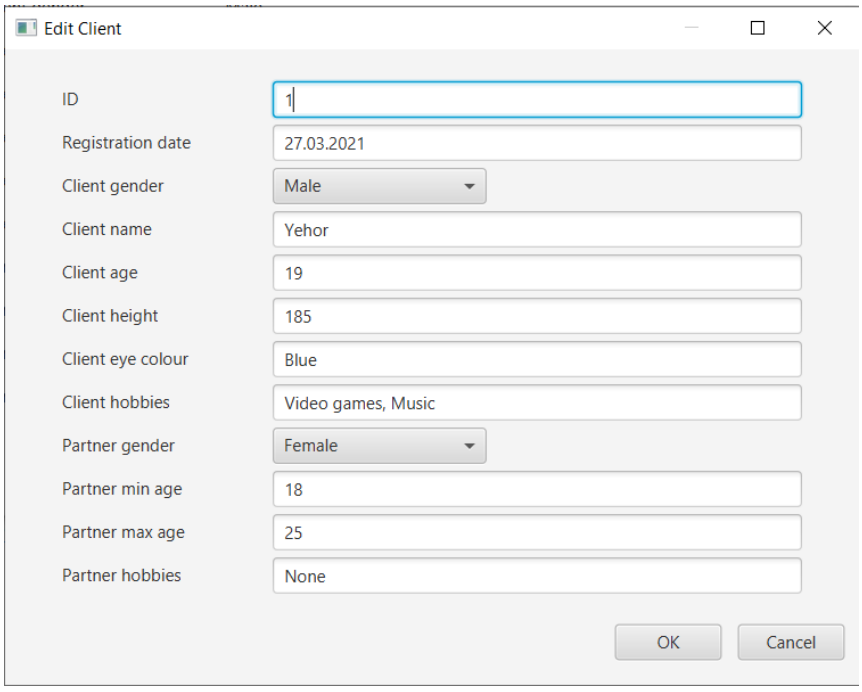

Рисунок 16.2 – Вікно редагування або додавання клієнта

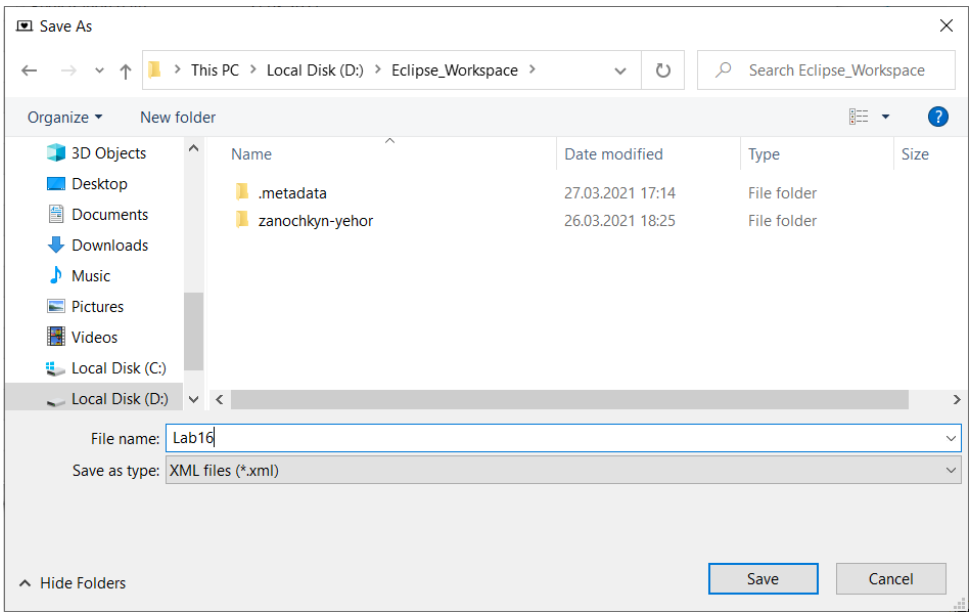

Рисунок 16.3 – Вікно збереження даних у файл XML

Висновок

Під час виконання лабораторної роботи було набуто навички використання засобів клієнтських технологій (Client Technologies) платформи Java SE та було розроблено графічний інтерфейс користувача з використанням засобів JavaFX.