Лабораторна робота №16.

Розробка графічного інтерфейсу користувача

**Мета роботи:**

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

**1. Вимоги до лабораторної роботи**

* 1. **Розробник:**

*студент Литвин Ігнатій Ігоревич; КІТ-26А; Варіант №7*

* 1. **Загальне завдання**

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

* 1. **Прикладна задача**

Бюро знайомств. Запис про клієнта: стать; реєстраційний номер; дата реєстрації; відомості про себе (довільний набір властивостей: ім’я, зріст, колір очей, дата народження, хобі тощо); вимоги до партнера (довільний набір властивостей).

**2. Опис програми**

Програма має графічний інтерфейс користувача.

Основне призначення: демонстрація управління масивом domain-об’єктів.

Програма має наступний функціонал:

* Додавання клієнту
* Редагування клієнту
* Видалення клієнту
* Генерування клієнту
* Генерування клієнтів
* Очищення клієнтів
* Збереження клієнтів до фалу
* Зчитування клієнтів з файлу
* Методи обробки списку клієнтів з попередніх лабораторних робіт

**2.1 Ієрархія та структура класів**

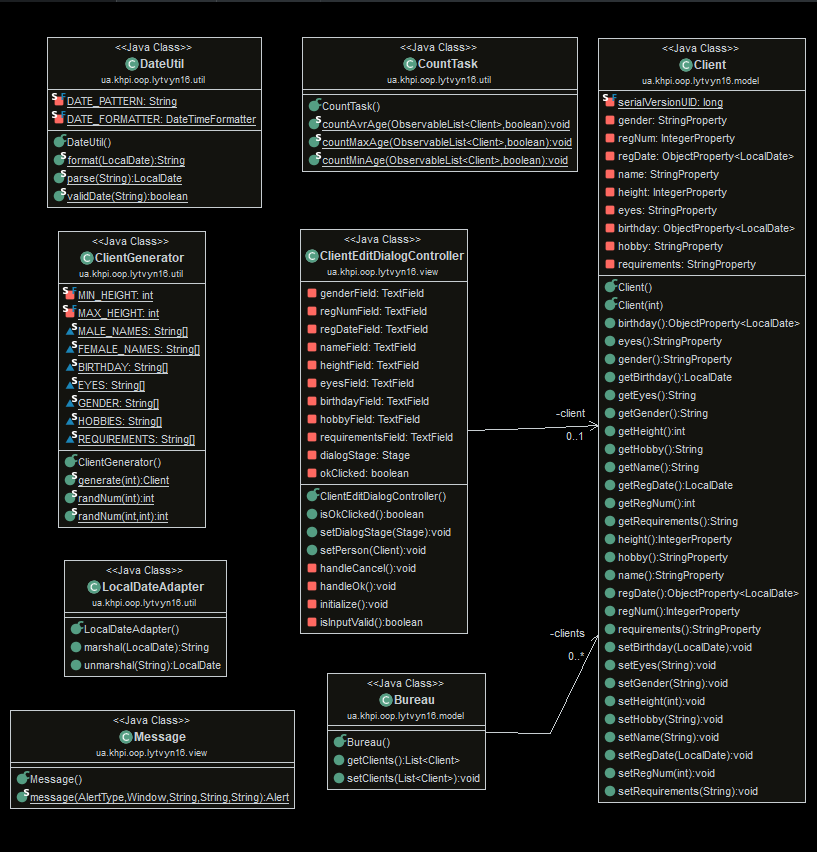
****

Рисунок 1 « Ієрархія та структура класів»

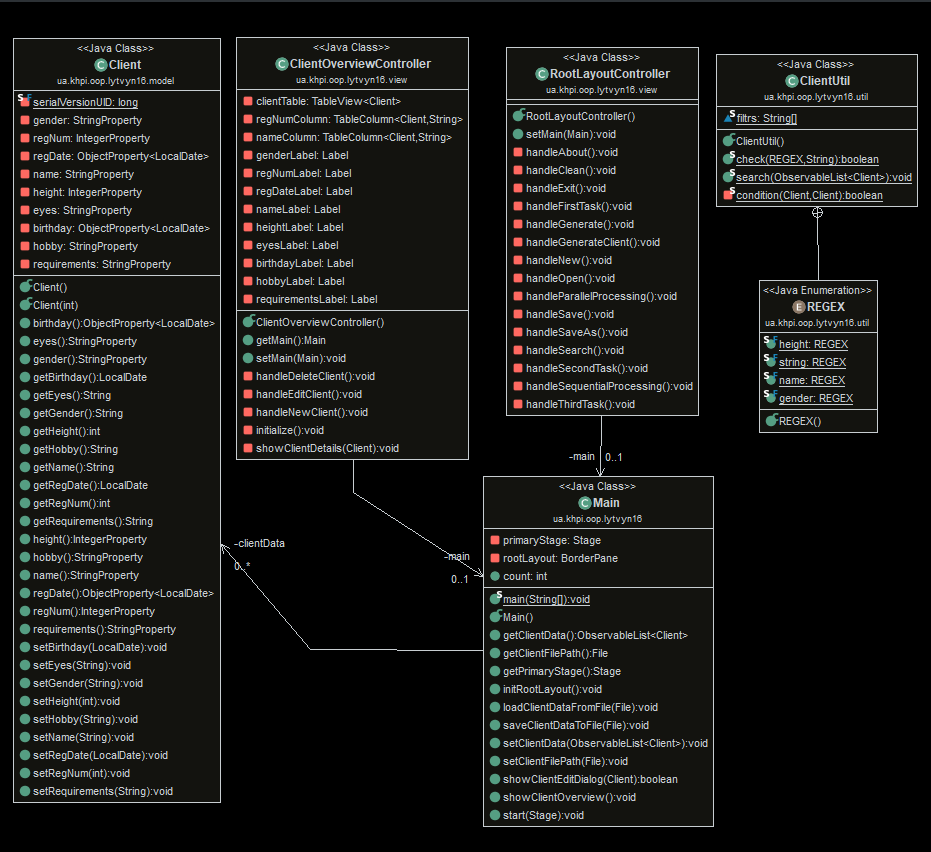


Рисунок 2 « Ієрархія та структура класів»

**2.2 Важливі фрагменти програми**

**/\*\***

**\* Відповідає за запуск роботи застосунку.**

**\***

**\* @author student Lytvyn I.I. KIT-26A**

**\*/**

**public class Main extends Application {**

**public static void main(String[] args) {**

**launch(args);**

**}**

**private Stage primaryStage;**

**private BorderPane rootLayout;**

**/\*\***

**\* Список клієнтів**

**\*/**

**private ObservableList<Client> clientData = FXCollections**

**.observableArrayList();**

**public int count = 0;**

**public Main() {**

**}**

**/\*\***

**\* @return the clientData**

**\*/**

**public ObservableList<Client> getClientData() {**

**return clientData;**

**}**

**/\*\***

**\* Returns the person file preference, i.e. the file that was last opened.**

**\* The preference is read from the OS specific registry. If no such**

**\* preference can be found, null is returned.**

**\***

**\* @return**

**\*/**

**public File getClientFilePath() {**

**final Preferences prefs = Preferences.userNodeForPackage(Main.class);**

**final String filePath = prefs.get("filePath", null);**

**if (filePath != null) {**

**return new File(filePath);**

**} else {**

**return null;**

**}**

**}**

**/\*\***

**\* Returns the main stage.**

**\***

**\* @return**

**\*/**

**public Stage getPrimaryStage() {**

**return primaryStage;**

**}**

**/\*\***

**\* Initializes the root layout and tries to load the last opened person**

**\* file.**

**\*/**

**public void initRootLayout() {**

**try {**

**// Load root layout from fxml file.**

**final FXMLLoader loader = new FXMLLoader();**

**loader.setLocation(**

**Main.class.getClassLoader().getResource(**

**"\\ua\\khpi\\oop\\lytvyn16\\view\\RootLayout.fxml"));**

**rootLayout = (BorderPane) loader.load();**

**// Show the scene containing the root layout.**

**final Scene scene = new Scene(rootLayout);**

**primaryStage.setScene(scene);**

**// Give the controller access to the main app.**

**final RootLayoutController controller = loader.getController();**

**controller.setMain(this);**

**primaryStage.show();**

**} catch (final IOException e) {**

**e.printStackTrace();**

**}**

**final File file = getClientFilePath();**

**if (file != null) {**

**loadClientDataFromFile(file);**

**}**

**}**

**public void loadClientDataFromFile(File file) {**

**try {**

**final JAXBContext context = JAXBContext**

**.newInstance(Bureau.class);**

**final Unmarshaller um = context.createUnmarshaller();**

**// Reading XML from the file and unmarshalling.**

**final Bureau bureau = (Bureau) um.unmarshal(file);**

**clientData.clear();**

**clientData.addAll(bureau.getClients());**

**count = bureau.getClients().size();**

**// Save the file path to the registry.**

**setClientFilePath(file);**

**} catch (final Exception e) { // catches ANY exception**

**final 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();**

**}**

**}**

**/\*\***

**\* Saves the current person data to the specified file.**

**\***

**\* @param file**

**\*/**

**public void saveClientDataToFile(File file) {**

**try {**

**final JAXBContext context = JAXBContext**

**.newInstance(Bureau.class);**

**final Marshaller m = context.createMarshaller();**

**m.setProperty(Marshaller.JAXB\_FORMATTED\_OUTPUT, true);**

**final Bureau wrapper = new Bureau();**

**wrapper.setClients(clientData);**

**m.marshal(wrapper, file);**

**setClientFilePath(file);**

**} catch (final Exception e) { // catches ANY exception**

**final Alert alert = Message.message(AlertType.ERROR, null, "Error",**

**"Could not save data",**

**"Could not save data to file:\n" + file.getPath());**

**alert.showAndWait();**

**}**

**}**

**/\*\***

**\* @param clientData**

**\* the clientData to set**

**\*/**

**public void setClientData(ObservableList<Client> clientData) {**

**this.clientData = clientData;**

**}**

**public void setClientFilePath(File file) {**

**final Preferences prefs = Preferences.userNodeForPackage(Main.class);**

**if (file != null) {**

**prefs.put("filePath", file.getPath());**

**// Update the stage title.**

**primaryStage.setTitle("AddressApp - " + file.getName());**

**} else {**

**prefs.remove("filePath");**

**// Update the stage title.**

**primaryStage.setTitle("AddressApp");**

**}**

**}**

**/\*\***

**\* Opens a dialog to edit details for the specified person. If the user**

**\* clicks OK, the changes are saved into the provided person object and true**

**\* is returned.**

**\***

**\* @param person**

**\* the person 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.**

**final FXMLLoader loader = new FXMLLoader();**

**loader.setLocation(**

**Main.class.getClassLoader().getResource(**

**"\\ua\\khpi\\oop\\lytvyn16\\view\\ClientEditDialog.fxml"));**

**final AnchorPane page = (AnchorPane) loader.load();**

**// Create the dialog Stage.**

**final Stage dialogStage = new Stage();**

**dialogStage.setTitle("Edit Client");**

**dialogStage.initModality(Modality.WINDOW\_MODAL);**

**dialogStage.initOwner(primaryStage);**

**final Scene scene = new Scene(page);**

**dialogStage.setScene(scene);**

**// Set the person into the controller.**

**final ClientEditDialogController controller = loader**

**.getController();**

**controller.setDialogStage(dialogStage);**

**controller.setPerson(client);**

**// Show the dialog and wait until the user closes it**

**dialogStage.showAndWait();**

**return controller.isOkClicked();**

**} catch (final IOException e) {**

**e.printStackTrace();**

**return false;**

**}**

**}**

**/\*\***

**\* Shows the client overview inside the root layout.**

**\*/**

**public void showClientOverview() {**

**try {**

**// Load person overview.**

**final FXMLLoader loader = new FXMLLoader();**

**loader.setLocation(Main.class.getClassLoader().getResource(**

**"\\ua\\khpi\\oop\\lytvyn16\\view\\ClientOverview.fxml"));**

**final AnchorPane personOverview = (AnchorPane) loader.load();**

**// Set person overview into the center of root layout.**

**rootLayout.setCenter(personOverview);**

**// Give the controller access to the main app.**

**final ClientOverviewController controller = loader.getController();**

**controller.setMain(this);**

**} catch (final IOException e) {**

**e.printStackTrace();**

**}**

**}**

**@Override**

**public void start(Stage primaryStage) {**

**this.primaryStage = primaryStage;**

**this.primaryStage.setTitle("Bureau of acquaintances");**

**// Set the application icon.**

**this.primaryStage.getIcons()**

**.add(new Image(**

**"\\ua\\khpi\\oop\\lytvyn16\\resources\\images\\ico\_32.png"));**

**initRootLayout();**

**showClientOverview();**

**}**

**}**

1. **Варіанти використання**

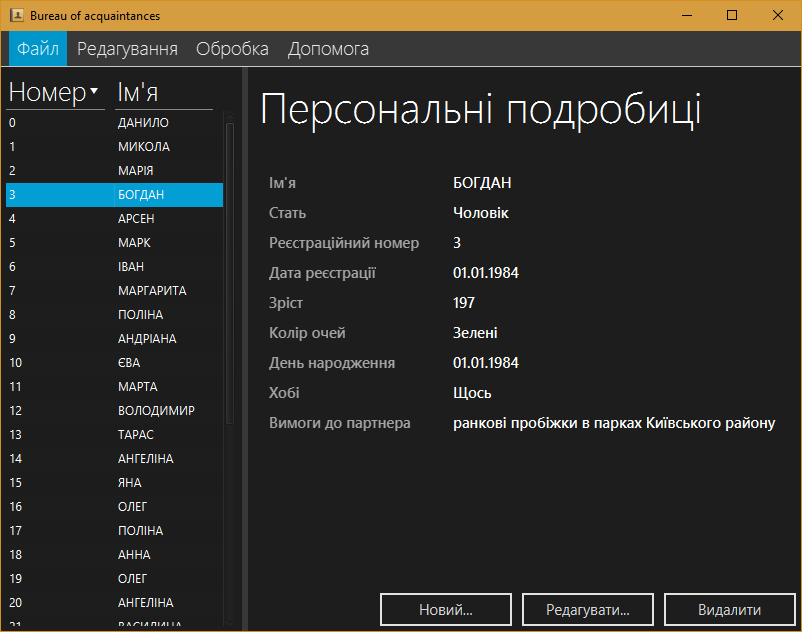
****

Рисунок 3 « Загальний вигляд програми»

Програма поділена на дві робочі частини.

Зліва знаходиться список записів клієнтів бюро знайомств. Кожний запис про клієнта містить його реєстраційний номер та ім’я.

У правій частині знаходиться детальна інформація про вибраного клієнта. Також внизу знаходяться кнопки для редагування списку клієнтів та окремого вибраного клієнту.

Зверху розташований меню-бар з набором наступних меню:

«Файл», «Редагування», «Обробка», «Допомога»

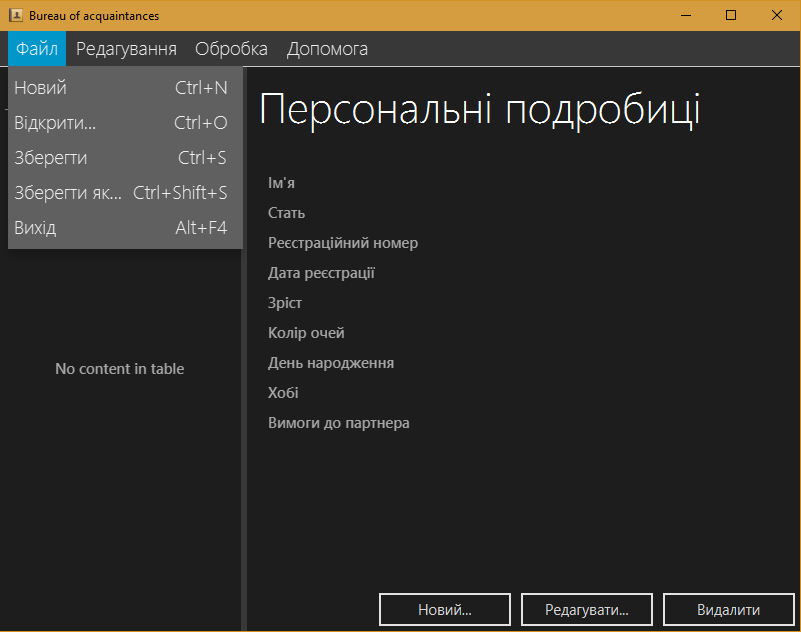
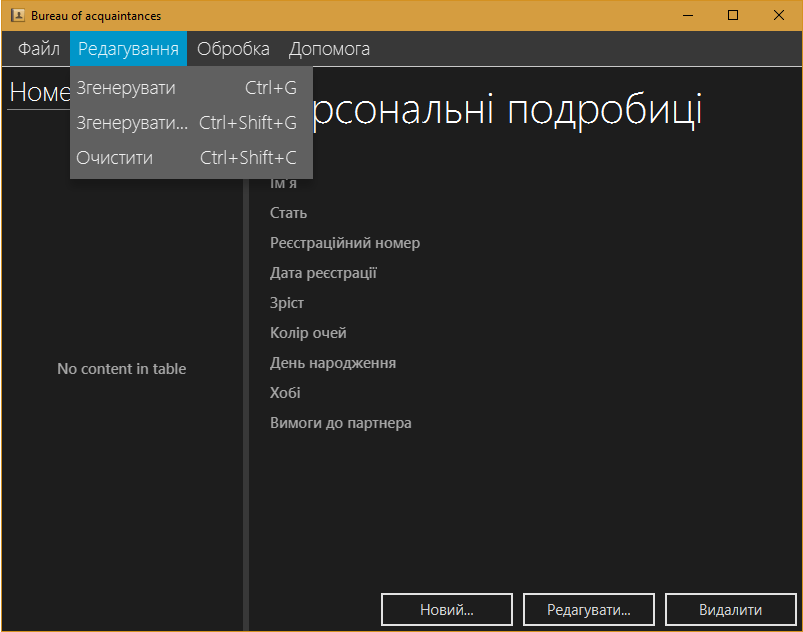
****

Рисунок 4 « Меню Файл»

* Новий – створює новий файл для збереження
* Відкрити… – відкриття файлу для зчитування з нього списку клієнтів
* Зберегти – збереження списку клієнтів за останнім обраним шляхом
* Зберегти як… – збереження списку клієнтів за обраним шляхом
* Вихід – завершення роботи програми

Рисунок 5 « Меню Редагування»

* Згенерувати – генерація нового клієнту
* Згенерувати… - генерація заданої кількості клієнтів
* Очистити – очищення списку клієнтів

****

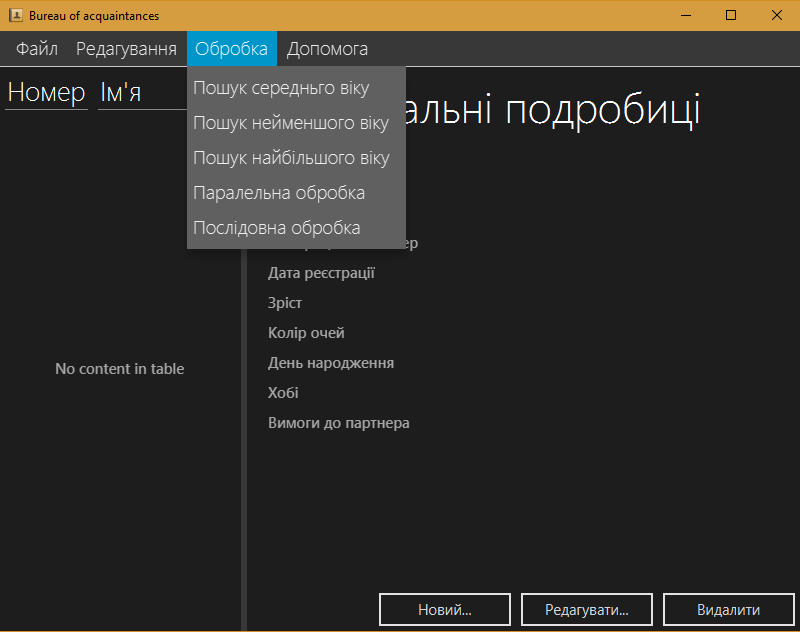
****

Рисунок 6 « Меню Обробка»

* Пошук середнього віку (серед клієнтів)
* Пошук найменшого віку (серед клієнтів)
* Пошук найбільшого віку (серед клієнтів)
* Паралельна обробка – вимір часу на паралельне виконання пошуків
* Послідовна обробка - вимір часу на послідовне виконання пошуків

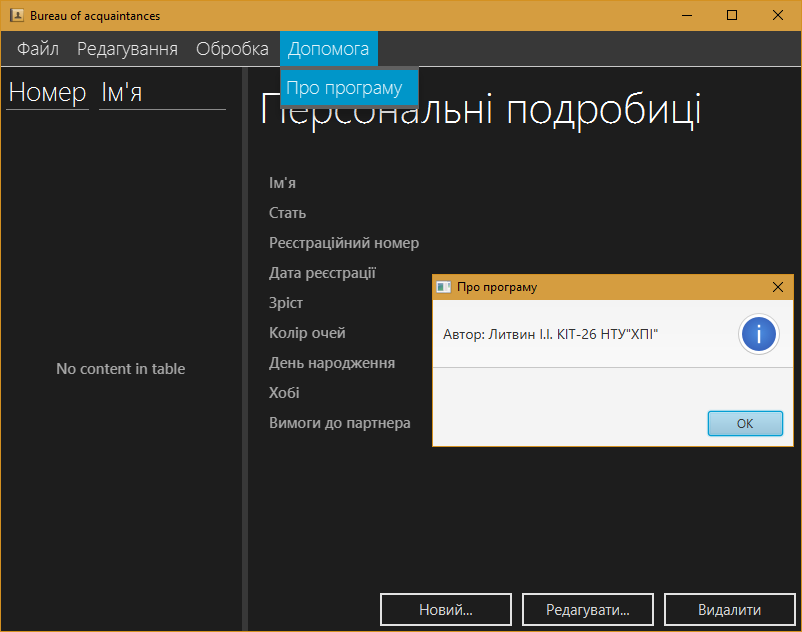
****

Рисунок 7 « Меню Допомога»

* Про програму – містить інформацію про автора

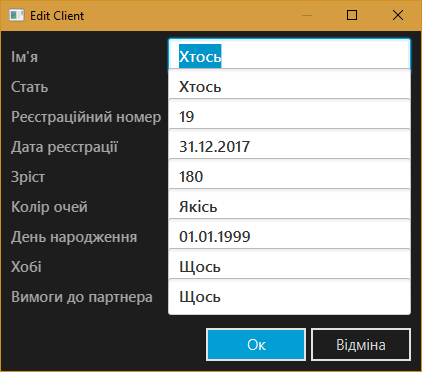
****

Рисунок 8 «Редагування запису про клієнта»

Вигляд вікна створення нового або редагування вже існуючого клієнту

**ВИСНОВКИ**

*Створено і налагоджено програму, що повністю виконую поставлене індивідуальне завдання та відповідає вимогам.*

*Було отримано і вдосконалено навички у використанні об’єктно-орієнтованого підходу для розробки об’єкта предметної (прикладної) галузі, та у розробці графічного інтерфейсу користувача.*