МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФРАСТРУКТУРИ ТА ТЕХНОЛОГІЙ
ІНСТИТУТ УПРАВЛІННЯ, ТЕХНОЛОГІЙ ТА ПРАВА
ФАКУЛЬТЕТ УПРАВЛІННЯ І ТЕХНОЛОГІЙ

Кафедра інформаційних технологій

Лабораторна робота №3

з дисципліни «Доменна інженерія»

з теми: «Apache Jena: Створення Java-застосунку для роботи з онтологіями»

Варіант 13

Виконав:
Студент групи
ІПЗд-23121 маг.
Петренко Д.М.
Перевірив:
Доцент кафедри ІТ
Ткаченко О.А.

Київ – 2023

Практична робота №3

Тема: Apache Jena: Створення Java-застосунку для роботи з онтологіями.

Завдання:

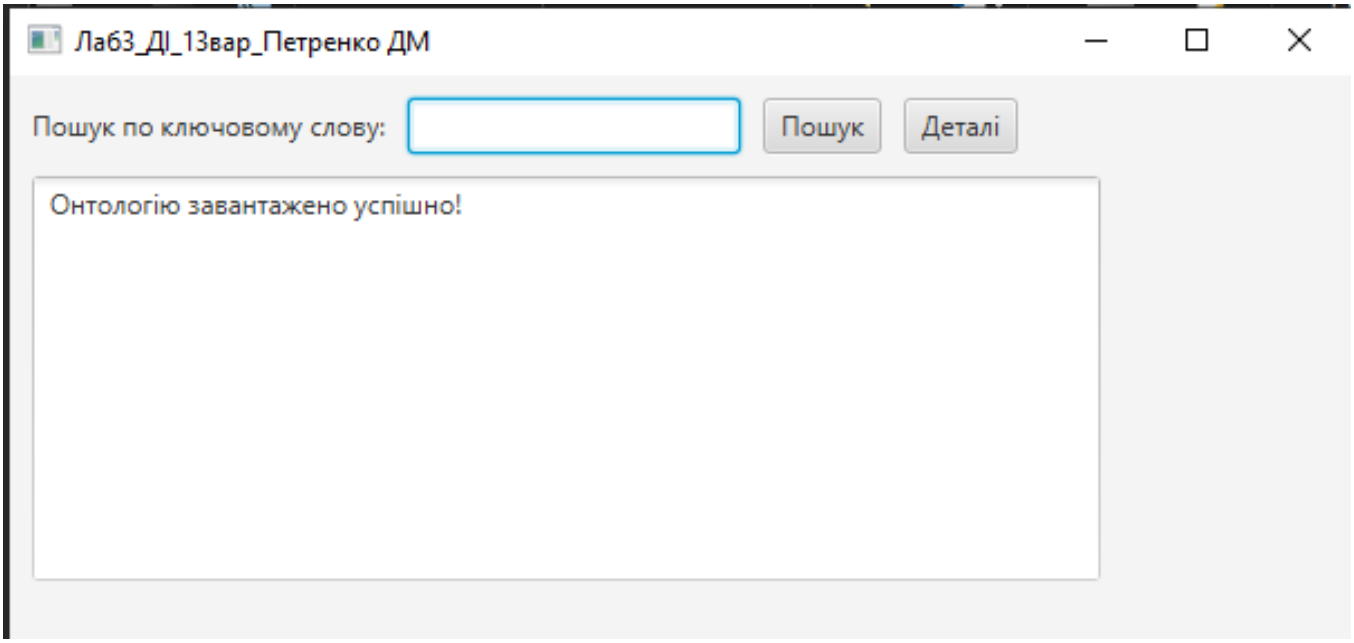
З використанням фреймворку Apache Jena API розробити Java-застосунок для роботи з онтологічною моделлю. Після поновлення моделі програмно використовувати різонер для перевірки онтології та отримання нових знань.

В текстовому полі інтерфейсу користувача ввести ключове слово або вираз для пошуку певного індивідуалу онтології. Після першого пошуку повинно виводитися декілька індивідуалів з цими ключовими словами. Далі ввести додаткові слова для звуження пошуку так, щоб в результаті залишилось тільки один індивідуал, про якого надати повну інформацію: до якого належить класу, які має властивості і їх значення.

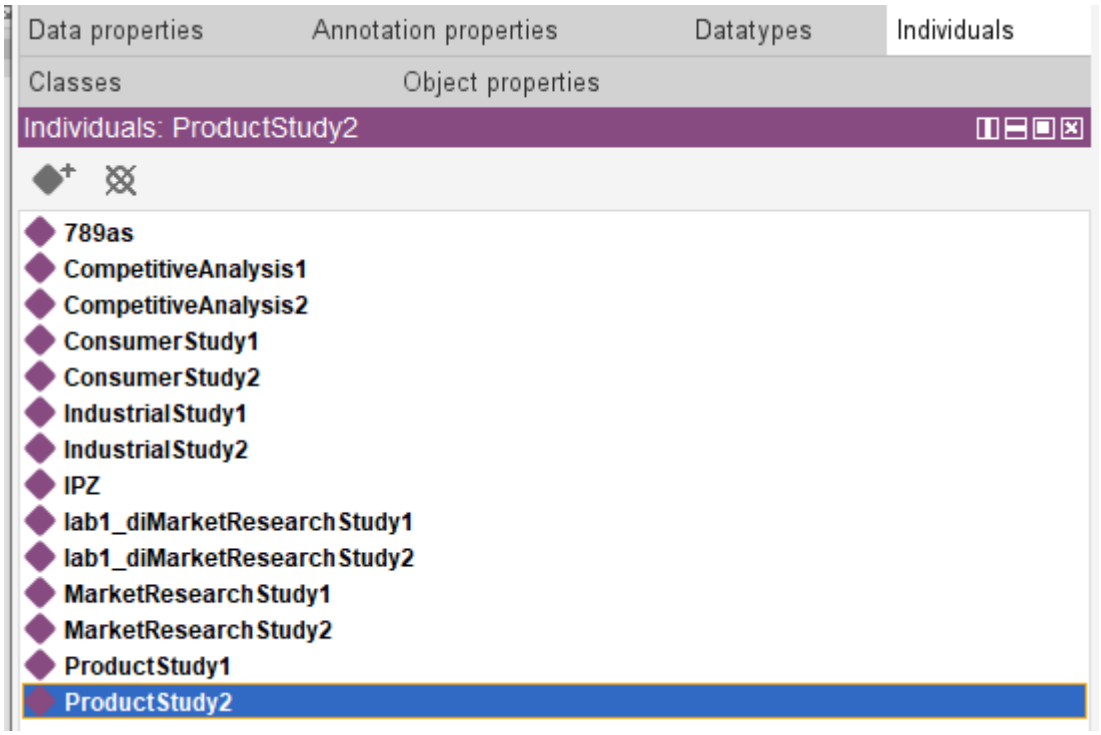
ХІД РОБОТИ:

Створимо Джава застосунок, котрий буде виконувати функції вище. Зайдемо в IntelliJ IDEA створимо дану програму.

Результатом даного коду є такий застосунок написаний на мові програмування Java:

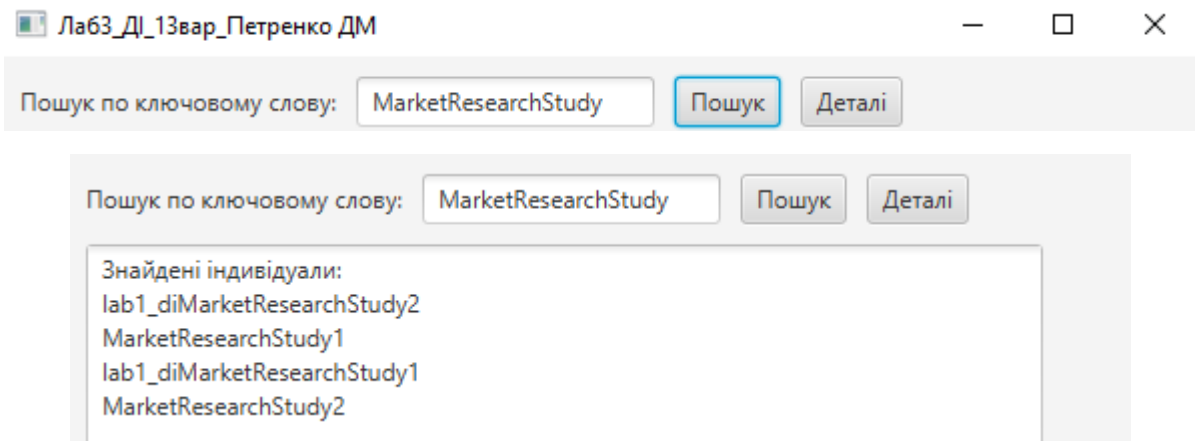


Покажемо список індивідуалів:

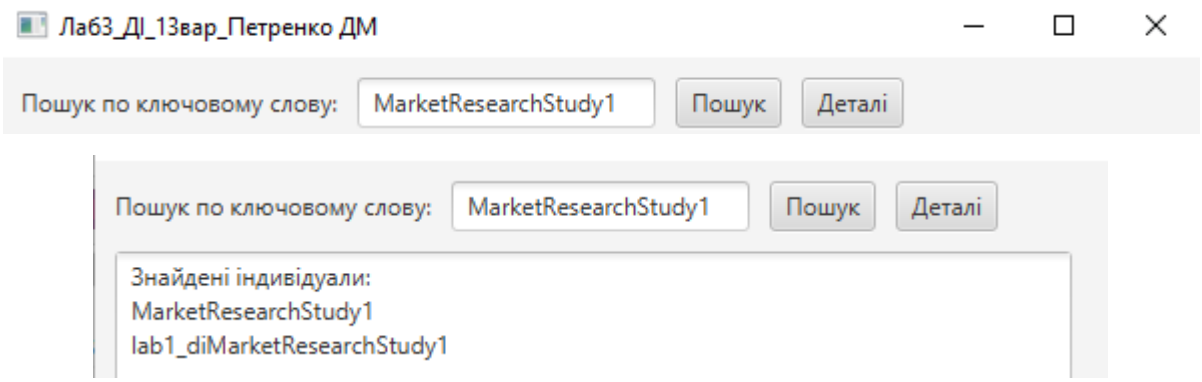


Покажемо принцип роботи програми:

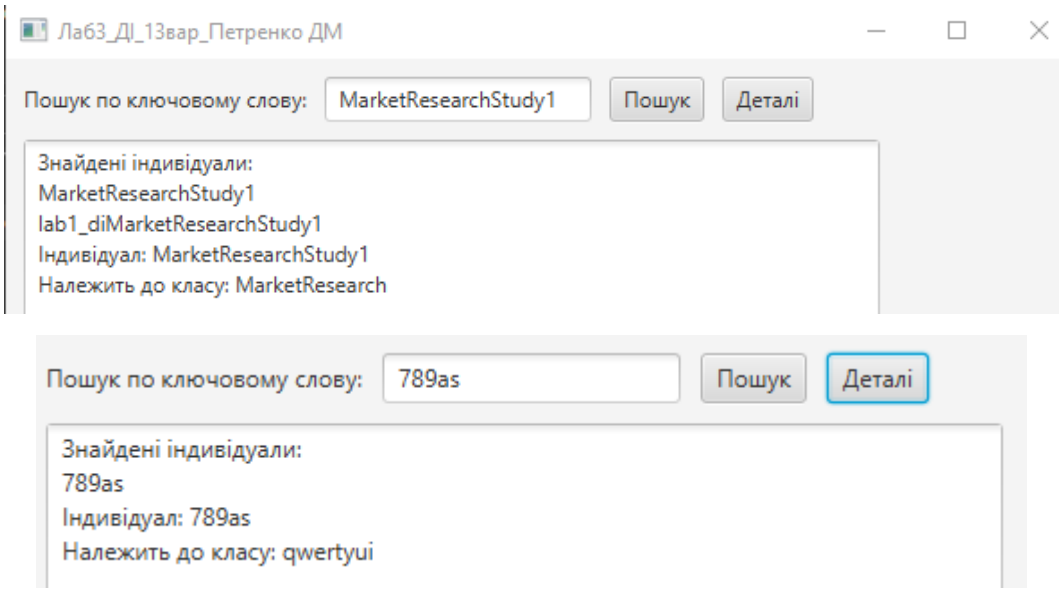
1. Вводимо в пошук ключове слово:



2. Звузимо пошук:



3. Отримуємо деталі про індивідуал:



ВИСНОВОК:

В ході виконання лабораторної роботи було створено застосунок на мові програмування Java з використанням фреймворку Apache Jena API для роботи з онтологічною моделлю. Було розроблено програму, яка в текстовому полі інтерфейсу користувача дозволяє ввести ключове слово або вираз для пошуку певного індивідуалу онтології. Також реалізовано виведення інформації про індивідуал, а саме: до якого належить класу, які має властивості і їх значення.

Лістинг програми:

```
import org.apache.jena.query.*;
import org.apache.jena.rdf.model.*;
import org.apache.jena.ontology.*;
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

import java.util.ArrayList;
import java.util.List;

public class Main extends Application {
    private TextArea outputTextArea;
```

```
    private TextField searchTextField;
    private OntModel model;


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

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Ontology Search Application");

        outputTextArea = new TextArea();
        outputTextArea.setEditable(false);

        searchTextField = new TextField();

        model = ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
        model.setNsPrefix("rdf", "http://www.w3.org/1999/02/22-rdf-syntax-ns#");
        model.setNsPrefix("rdfs", "http://www.w3.org/2000/01/rdf-schema#");
        model.setNsPrefix("owl", "http://www.w3.org/2002/07/owl#");

        loadOntology();

        Button searchButton = new Button("Search");
        searchButton.setOnAction(e -> searchAndDisplay());

        GridPane gridPane = new GridPane();
        gridPane.setHgap(10);
        gridPane.setVgap(10);
        gridPane.setPadding(new Insets(10, 10, 10, 10));
        gridPane.add(outputTextArea, 0, 0, 2, 1);
        gridPane.add(new Label("Search Keyword:"), 0, 1);
        gridPane.add(searchTextField, 1, 1);
        gridPane.add(searchButton, 2, 1);

        Scene scene = new Scene(gridPane, 600, 400);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    private void loadOntology() {
        String ontologyFilePath = "C:/LAB1.rdf";

        try {
            model.read(ontologyFilePath);
            outputTextArea.appendText("Ontology Loaded Successfully.\n");
        } catch (Exception e) {
            outputTextArea.appendText("Failed to load ontology.\n");
            e.printStackTrace();
        }
    }
```

```
    private List<Individual> findIndividualsByKeyword(String keyword) {
        List<Individual> matchingIndividuals = new ArrayList<>();

        String escapedKeyword = escapeRegex(keyword);

        // Search in ontology
        String queryString =
                "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>\n" +
                        "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n" +
                        "PREFIX owl: <http://www.w3.org/2002/07/owl#>\n" +
                        "SELECT ?individual WHERE {\n" +
                        "  ?individual rdf:type owl:NamedIndividual.\n" +
                        "  ?individual rdfs:label ?label.\n" +
                        "  FILTER (regex(?label, \"" + escapedKeyword + "\",
\"i\"))\n" +
                        "}";

        Query query = QueryFactory.create(queryString);
        QueryExecution queryExecution = QueryExecutionFactory.create(query, model);
        ResultSet resultSet = queryExecution.execSelect();

        while (resultSet.hasNext()) {
            QuerySolution solution = resultSet.nextSolution();
            RDFNode individual = solution.get("individual");
            if (individual != null) {
                matchingIndividuals.add(model.getIndividual(individual.toString()))
;
                outputTextArea.appendText("Individual: " + individual.toString() +
"\n");
            }
        }

        queryExecution.close();

        if (matchingIndividuals.isEmpty()) {
            outputTextArea.appendText("No matching individuals found for the
keyword.\n");
        }

        return matchingIndividuals;
    }


    private void searchAndDisplay() {
        String keyword = searchTextField.getText().trim();
        if (keyword.isEmpty()) {
            outputTextArea.appendText("Please enter a search keyword.\n");
            return;
        }

        outputTextArea.appendText("Search Results:\n");
```

```
        List<Individual> matchingIndividualsForKeyword =
findIndividualsByKeyword(keyword);
        if (!matchingIndividualsForKeyword.isEmpty()) {
            outputTextArea.appendText("Matching individuals for keyword:\n");
            for (Individual matchingIndividual : matchingIndividualsForKeyword) {
                getDetails(matchingIndividual);
            }
        } else {
            outputTextArea.appendText("No matching individuals found for the
keyword.\n");
        }


    private void getDetails(Individual selectedIndividual) {
        if (selectedIndividual != null) {
            String details = "Details:\n";
            details += "Local Name: " + selectedIndividual.getLocalName() + "\n";
            details += "URI: " + selectedIndividual.getURI() + "\n";

            outputTextArea.appendText(details);
        }
    }

    private String escapeRegex(String input) {
        return input.replaceAll("([\\\\.^$|?*+\\[\\](){}`])", "\\\\$1");
    }
}
```