Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 3 з дисципліни
«Проектування алгоритмів»

„ **Проектування структур даних**”

Виконав(ла) *ІП-22 Андреєва Уляна Андріївна* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Ахаладзе Ілля Елдарійович* _____
(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – вивчити основні підходи проектування та обробки складних структур даних.

2 ЗАВДАННЯ

Відповідно до варіанту (таблиця 2.1), записати алгоритми пошуку, додавання, видалення і редагування запису в структурі даних за допомогою псевдокоду (чи іншого способу по вибору).

Записати часову складність пошуку в структурі в асимптотичних оцінках.

Виконати програмну реалізацію невеликої СУБД з графічним (не консольним) інтерфейсом користувача (дані БД мають зберігатися на ПЗП), з функціями пошуку (алгоритм пошуку у вузлі структури згідно варіанту таблиця 2.1, за необхідності), додавання, видалення та редагування записів (запис складається із ключа і даних, ключі унікальні і цілочисельні, даних може бути декілька полів для одного ключа, але достатньо одного рядка фіксованої довжини). Для зберігання даних використовувати структуру даних згідно варіанту (таблиця 2.1).

Заповнити базу випадковими значеннями до 10000 і зафіксувати середнє (із 10-15 пошуків) число порівнянь для знаходження запису по ключу.

Зробити висновок з лабораторної роботи.

Таблиця 2.1 – Варіанти алгоритмів

№	**Структура даних**
1	Файли з щільним індексом з перебудовою індексної області, бінарний пошук
2	Файли з щільним індексом з областю переповнення, бінарний пошук
3	Файли з не щільним індексом з перебудовою індексної області, бінарний пошук
4	Файли з не щільним індексом з областю переповнення, бінарний пошук
5	АВЛ-дерево

6	Червоно-чорне дерево
7	B-дерево t=10, бінарний пошук
8	B-дерево t=25, бінарний пошук
9	B-дерево t=50, бінарний пошук
10	B-дерево t=100, бінарний пошук
11	Файли з щільним індексом з перебудовою індексної області, однорідний бінарний пошук
12	Файли з щільним індексом з областю переповнення, однорідний бінарний пошук
13	Файли з не щільним індексом з перебудовою індексної області, однорідний бінарний пошук
14	Файли з не щільним індексом з областю переповнення, однорідний бінарний пошук
15	АВЛ-дерево
16	Червоно-чорне дерево
17	B-дерево t=10, однорідний бінарний пошук
18	B-дерево t=25, однорідний бінарний пошук
19	B-дерево t=50, однорідний бінарний пошук
20	B-дерево t=100, однорідний бінарний пошук
21	Файли з щільним індексом з перебудовою індексної області, метод Шарра
22	Файли з щільним індексом з областю переповнення, метод Шарра
23	Файли з не щільним індексом з перебудовою індексної області, метод Шарра
24	Файли з не щільним індексом з областю переповнення, метод Шарра
25	АВЛ-дерево
26	Червоно-чорне дерево
27	B-дерево t=10, метод Шарра
28	B-дерево t=25, метод Шарра

29	B-дерево t=50, метод Шарра
30	B-дерево t=100, метод Шарра
31	АВЛ-дерево
32	Червоно-чорне дерево
33	B-дерево t=250, бінарний пошук
34	B-дерево t=250, однорідний бінарний пошук
35	B-дерево t=250, метод Шарра

3 ВИКОНАННЯ

3.1 Псевдокод алгоритмів

```
addData():
    якщо filename не є нульовим:
        dialog = створити діалогове вікно для введення даних
        dialog.setTitle("Add Data")
        dialog.setHeaderText(null)
        dialog.setContentText("Enter Key and Value (separated by space):")

        result = показати діалогове вікно і отримати результат
        validInput = false

        поки not validInput і result присутній:
            input = результат введення
            parts = розбити вхід на частини, розділені пробілом
            якщо довжина parts дорівнює 2:
                спробувати:
                    key = перетворити першу частину в ціле число
                    якщо перша частина не дорівнює строковому представленню key:
                        викинути виняток NumberFormatException ("Key cannot have leading zeros")
                    value = друга частина
                    записи = прочитати дані з файлу
                    keyExists = чи є запис з таким ключем в записах
                    якщо keyExists:
                        показати повідомлення про те, що ключ вже існує
                    інакше:
                        записати дані в файл
                        очистити таблицю
```

```
                відобразити дані з файлу
                validInput = true
            виключення NumberFormatException:
                показати повідомлення про невірний ввід ключа
        інакше:
            показати повідомлення про невірний ввід даних

updateData():
    якщо filename не є нульовим:
        dialog = створити діалогове вікно для оновлення даних
        dialog.setTitle("Update Data")
        dialog.setHeaderText(null)
        dialog.setContentText("Enter Key and New Value (separated by space):")

        result = показати діалогове вікно і отримати результат
        result.ifPresent(input ->
            parts = розбити вхід на частини, розділені пробілом
            якщо довжина parts дорівнює 2:
                спробувати:
                    key = перетворити першу частину в ціле число
                    якщо перша частина не дорівнює строковому представленню key:
                        викинути виняток NumberFormatException ("Key cannot have leading zeros")
                    newValue = друга частина
                    записи = прочитати дані з файлу
                    updated = false

                    для кожного запису в записах:
                        якщо ключ запису дорівнює key:
                            оновити значення запису на newValue
```

```
                    оновити = true
                    зупинити цикл
                якщо updated:
                    записати всі дані в файл
                    очистити таблицю
                    відобразити дані з файлу
                інакше:
                    показати повідомлення про невдале оновлення
            виключення NumberFormatException:
                показати повідомлення про невірний ввід ключа
        інакше:
            показати повідомлення

deleteData():
    якщо filename не є нульовим:
        dialog = створити діалогове вікно для видалення даних
        dialog.setTitle("Delete Data")
        dialog.setHeaderText(null)
        dialog.setContentText("Enter Key to delete:")

        result = показати діалогове вікно і отримати результат
        validInput = false

        поки not validInput і result присутній:
            input = результат введення
            спробувати:
                keyToDelete = перетворити введений ключ в ціле число
                якщо введений ключ не дорівнює строковому представленню keyToDelete:
```

викинути виняток NumberFormatException ("Key cannot have leading zeros")

записи = прочитати дані з файлу

found = false

для кожного запису в записах:

якщо ключ запису дорівнює keyToDelete:

видалити запис зі списку

found = true

зупинити цикл

якщо found:

записати всі дані в файл

очистити таблицю

відобразити дані з файлу

validInput = true

інакше:

показати повідомлення про невдале видалення

виключення NumberFormatException:

показати повідомлення про невірний ввід ключа

3.2 Часова складність пошуку

Пошук записів для відображення у таблиці після відкриття файлу (displayData()):

Ваша програма зчитує всі записи з файлу і додає їх до таблиці та текстової області.

Цей пошук виконується лінійно, оскільки програма зчитує кожен запис у файлі.

Таким чином, часова складність цього операції - O(n), де n - кількість записів у файлі.

Пошук запису для оновлення (updateData()):

Ваша програма шукає запис із певним ключем у списку записів.

Цей пошук також виконується лінійно, оскільки програма перевіряє кожний запис у списку.

Таким чином, часова складність цього операції - O(n), де n - кількість записів у файлі.

Пошук запису для видалення (deleteData()):

Пошук запису з певним ключем у списку записів для видалення також виконується лінійно.

Програма перевіряє кожний запис у списку для пошуку необхідного запису.

Таким чином, часова складність цього операції - O(n), де n - кількість записів у файлі.

Всі інші операції також можуть мати лінійну часову складність, оскільки вони перевіряють або зчитують кожен запис у файлі.

3.3 Програмна реалізація

3.3.1 Вихідний код

```
package com.example.labwork3;
import javafx.application.Application;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.*;
import java.io.*;
import java.nio.file.*;
import java.util.*;

public class HelloApplication extends Application {
    public static String filename;
    private static TableView<DataRecord> table;
    public static TextArea fileContentsTextArea;
    private Label fileStatusLabel;

    private Label filePathLabel;

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

```
@Override
public void start(Stage primaryStage) {
    primaryStage.setTitle("Laboratory work 3");

    BorderPane root = new BorderPane();
    Scene scene = new Scene(root, 800, 600);

    MenuBar menuBar = new MenuBar();
    Menu fileMenu = new Menu("File");

    MenuItem openMenuItem = new MenuItem("Open");
    MenuItem clearMenuItem = new MenuItem("Clear File");
    MenuItem createFileMenuItem = new MenuItem("Create File");

    fileMenu.getItems().addAll(openMenuItem, clearMenuItem, createFileMenuItem);
    menuBar.getMenus().add(fileMenu);

    openMenuItem.setOnAction(event -> selectFile(primaryStage));
    clearMenuItem.setOnAction(event -> clearFile());
    createFileMenuItem.setOnAction(event -> createFile(primaryStage));

    VBox vbox = new VBox();
    vbox.setSpacing(10);

    fileStatusLabel = new Label("File is not selected");
    filePathLabel = new Label("");

    table = new TableView<>();
    TableColumn<DataRecord, Integer> keyColumn = new TableColumn<>("Key");
    TableColumn<DataRecord, String> valueColumn = new TableColumn<>("Value");
    keyColumn.setCellValueFactory(new PropertyValueFactory<>("key"));
    valueColumn.setCellValueFactory(new PropertyValueFactory<>("value"));
    table.getColumns().addAll(keyColumn, valueColumn);

    fileContentsTextArea = new TextArea();
    fileContentsTextArea.setWrapText(true);

    Button addDataButton = new Button("Add Data");
    Button updateDataButton = new Button("Update Data");
    Button deleteDataButton = new Button("Delete Data");

    addDataButton.setOnAction(event -> addData());
    updateDataButton.setOnAction(event -> updateData());
    deleteDataButton.setOnAction(event -> deleteData());
```

```
        vbox.getChildren().addAll(fileStatusLabel, filePathLabel, menuBar, addDataButton,
updateDataButton, deleteDataButton, table, fileContentsTextArea);

        root.setTop(vbox);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void setFilename(String filename) {
        HelloApplication.filename = filename;
    }

    public static String getFilename() {
        return filename;
    }

    private void createFile(Stage primaryStage) {
        FileChooser fileChooser = new FileChooser();
        fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("Data type",
"*.dat"));
        File file = fileChooser.showSaveDialog(primaryStage);
        if (file != null) {
            String filePath = file.getAbsolutePath();
            if (filePath.endsWith(".dat") && filePath.split("\\.").length == 2) {
                try {
                    Files.createFile(Paths.get(filePath));
                    setFilename(filePath);
                    fileStatusLabel.setText("Selected file:");
                    filePathLabel.setText(filePath);
                    table.getItems().clear();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            } else if (!filePath.contains(".")) {
                fileStatusLabel.setText("File is not selected");
            } else {
                System.out.println("Invalid file extension. Please choose a .dat file.");
            }
        }
    }

    private void selectFile(Stage primaryStage) {
        FileChooser fileChooser = new FileChooser();
```

```
        fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("Data type",
"*.dat"));
        File selectedFile = fileChooser.showOpenDialog(primaryStage);
        if (selectedFile != null) {
            String filePath = selectedFile.getAbsolutePath();
            if (Files.exists(Paths.get(filePath)) && filePath.endsWith(".dat") &&
filePath.split("\\.").length == 2) {
                filename = filePath;
                table.getItems().clear();
                displayData();
            } else if (!filePath.contains(".")) {
                System.out.println("File is not selected");
            } else {
                System.out.println("Can't find such file");
            }
        }
    }

    public static void clearFile() {
        if (filename != null) {
            try {
                Files.deleteIfExists(Paths.get(filename));
                filename = null;
                System.out.println("File cleared.");
                table.getItems().clear();
                fileContentsTextArea.clear();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    private void displayData() {
        if (filename != null) {
            List<DataRecord> records = readDataFromFile();
            table.getItems().addAll(records);

            StringBuilder fileContents = new StringBuilder();
            try {
                Scanner scanner = new Scanner(new File(filename));
                while (scanner.hasNextLine()) {
                    fileContents.append(scanner.nextLine()).append("\n");
                }
                scanner.close();
            } catch (IOException e) {
```

```
            e.printStackTrace();
        }
        fileContentsTextArea.setText(fileContents.toString());
    }
}

public static List<DataRecord> readDataFromFile() {
    List<DataRecord> records = new ArrayList<>();
    try {
        Scanner scanner = new Scanner(new File(filename));
        while (scanner.hasNext()) {
            int key = scanner.nextInt();
            String value = scanner.next();
            records.add(new DataRecord(key, value));
        }
        scanner.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return records;
}

private void addData() {
    if (filename != null) {
        TextInputDialog dialog = new TextInputDialog();
        dialog.setTitle("Add Data");
        dialog.setHeaderText(null);
        dialog.setContentText("Enter Key and Value (separated by space):");

        Optional<String> result;
        boolean validInput = false;

        do {
            result = dialog.showAndWait();
            if (result.isPresent()) {
                String input = result.get();
                String[] parts = input.split(" ");
                if (parts.length == 2) {
                    try {
                        int key = Integer.parseInt(parts[0]);
                        if (!parts[0].equals(String.valueOf(key))) {
                            throw new NumberFormatException("Key cannot have leading
zeros");
                        }
                        String value = parts[1];
```

```
                    List<DataRecord> records = readDataFromFile();
                    boolean keyExists = records.stream().anyMatch(record ->
record.getKey() == key);

                    if (keyExists) {
                        showAlert("Key Already Exists", "The key already exists. Please enter a
different key.");
                    } else {
                        writeDataToFile(key, value);
                        System.out.println("Data added successfully.");
                        table.getItems().clear();
                        displayData();
                        validInput = true;
                    }
                } catch (NumberFormatException e) {
                    showAlert("Invalid Input", "Key must be a non-zero-padded integer.
Please try again.");
                }
            } else {
                showAlert("Invalid Input", "Please enter Key and Value separated by
space.");
            }
        }
    } while (!validInput && result.isPresent());
  }
  }


  private void showAlert(String title, String content) {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setTitle(title);
    alert.setHeaderText(null);
    alert.setContentText(content);
    alert.showAndWait();
  }


  public static void writeDataToFile(int key, String value) {
    if (filename != null) {
      try (PrintWriter writer = new PrintWriter(new FileWriter(filename, true))) {
        writer.println(key + " " + value);
      } catch (IOException e) {
        e.printStackTrace();
      }
```

```
    }
}

private void updateData() {
    if (filename != null) {
        TextInputDialog dialog = new TextInputDialog();
        dialog.setTitle("Update Data");
        dialog.setHeaderText(null);
        dialog.setContentText("Enter Key and New Value (separated by space):");

        Optional<String> result = dialog.showAndWait();
        result.ifPresent(input -> {
            String[] parts = input.split(" ");
            if (parts.length == 2) {
                try {
                    int key = Integer.parseInt(parts[0]);
                    if (!parts[0].equals(String.valueOf(key))) {
                        throw new NumberFormatException("Key cannot have leading zeros");
                    }
                    String newValue = parts[1];

                    List<DataRecord> records = readDataFromFile();
                    boolean updated = false;

                    for (DataRecord record : records) {
                        if (record.getKey() == key) {
                            record.setValue(newValue);
                            updated = true;
                            break;
                        }
                    }

                    if (updated) {
                        writeAllDataToFile(records);
                        System.out.println("Data updated successfully.");
                        table.getItems().clear();
                        displayData();
                    } else {
                        showAlert("Update Failed", "Key not found. Data update failed.");
                    }
                } catch (NumberFormatException e) {
                    showAlert("Invalid Input", "Key must be a non-zero-padded integer.");
                }
            } else {
```

```
                showAlert("Invalid Input", "Please enter Key and New Value separated by
space.");
            }
        });
    }
}

public static void writeAllDataToFile(List<DataRecord> records) {
    if (filename != null) {
        try (PrintWriter writer = new PrintWriter(new FileWriter(filename, false))) {
            for (DataRecord record : records) {
                writer.println(record.getKey() + " " + record.getValue());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

private void deleteData() {
    if (filename != null) {
        TextInputDialog dialog = new TextInputDialog();
        dialog.setTitle("Delete Data");
        dialog.setHeaderText(null);
        dialog.setContentText("Enter Key to delete:");

        Optional<String> result;
        boolean validInput = false;

        do {
            result = dialog.showAndWait();
            if (result.isPresent()) {
                String input = result.get();
                try {
                    int keyToDelete = Integer.parseInt(input);
                    if (!input.equals(String.valueOf(keyToDelete))) {
                        throw new NumberFormatException("Key cannot have leading zeros");
                    }
                    List<DataRecord> records = readDataFromFile();
                    boolean found = false;

                    Iterator<DataRecord> iterator = records.iterator();
                    while (iterator.hasNext()) {
                        DataRecord record = iterator.next();
                        if (record.getKey() == keyToDelete) {
```

```
                    iterator.remove();
                    found = true;
                    break;
                }
            }

            if (found) {
                writeAllDataToFile(records);
                System.out.println("Data deleted successfully.");
                table.getItems().clear();
                displayData();
                validInput = true;
            } else {
                showAlert("Delete Failed", "Key not found. No data deleted.");
            }
        } catch (NumberFormatException e) {
            showAlert("Invalid Input", "Key must be a non-zero-padded integer.");
        }
      }
    } while (!validInput && result.isPresent());
  }
}




public static class DataRecord {
  private final int key;
  private String value;

  public DataRecord(int key, String value) {
    this.key = key;
    this.value = value;
  }

  public int getKey() {
    return key;
  }

  public String getValue() {
    return value;
  }

  public void setValue(String value) {
    this.value = value;
```

```
        }
    }
}



package com.example.labwork3;

import javafx.fxml.FXML;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.FileChooser;
import javafx.stage.Stage;

import java.io.*;
import java.nio.file.*;
import java.util.*;

import static com.example.labwork3.HelloApplication.*;

public class GUI {

    @FXML
    private Label fileStatusLabel;

    @FXML
    private Label filePathLabel;

    @FXML
    private TableView<HelloApplication.DataRecord> table;

    public GUI() {
    }

    @FXML
    private void initialize() {
        fileStatusLabel.setText("File is not selected");
        filePathLabel.setText("");
        table.getColumns().clear();
        TableColumn<HelloApplication.DataRecord, Integer> keyColumn = new
TableColumn<>("Key");
        TableColumn<HelloApplication.DataRecord, String> valueColumn = new
TableColumn<>("Value");
        keyColumn.setCellValueFactory(new PropertyValueFactory<>("key"));
        valueColumn.setCellValueFactory(new PropertyValueFactory <> ("value"));
```

```
        table.getColumns().addAll(keyColumn, valueColumn);
    }

    @FXML
    private void selectFile() {
        Stage stage = (Stage) fileStatusLabel.getScene().getWindow();
        FileChooser fileChooser = new FileChooser();
        fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("Data type",
"*.dat"));
        File selectedFile = fileChooser.showOpenDialog(stage);
        if (selectedFile != null) {
            String filePath = selectedFile.getAbsolutePath();
            if (Files.exists(Paths.get(filePath)) && filePath.endsWith(".dat") &&
filePath.split("\\.").length == 2) {
                HelloApplication.setFilename(filePath);
                fileStatusLabel.setText("Selected file:");
                filePathLabel.setText(filePath);
                updateDataTable();
            } else if (!filePath.contains(".")) {
                fileStatusLabel.setText("File is not selected");
                filePathLabel.setText("");
            } else {
                fileStatusLabel.setText("File is not selected");
                filePathLabel.setText("");
            }
        }
    }

    @FXML
    private void addData() {
        if (filename != null) {
            TextInputDialog dialog = new TextInputDialog();
            dialog.setTitle("Add Data");
            dialog.setHeaderText(null);
            dialog.setContentText("Enter Key and Value (separated by space):");

            Optional<String> result;
            boolean validInput = false;

            do {
                result = dialog.showAndWait();
                if (result.isPresent()) {
                    String input = result.get();
                    String[] parts = input.split(" ");
                    if (parts.length == 2) {
```

```
                try {
                    int key = Integer.parseInt(parts[0]);
                    String value = parts[1];
                    writeDataToFile(key, value);
                    System.out.println("Data added successfully.");
                    table.getItems().clear();
                    displayData();
                    validInput = true;
                } catch (NumberFormatException e) {
                    showAlert("Invalid Input", "Key must be an integer. Please try again.");
                }
            } else {
                showAlert("Invalid Input", "Please enter Key and Value separated by
space.");
            }
        }
    } while (!validInput && result.isPresent());
  }
  }
  private void showAlert(String title, String content) {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setTitle(title);
    alert.setHeaderText(null);
    alert.setContentText(content);
    alert.showAndWait();
  }

  @FXML
  private void updateData() {
    if (filename != null) {
      TextInputDialog dialog = new TextInputDialog();
      dialog.setTitle("Update Data");
      dialog.setHeaderText(null);
      dialog.setContentText("Enter Key and New Value (separated by space):");

      Optional<String> result = dialog.showAndWait();
      result.ifPresent(input -> {
        String[] parts = input.split(" ");
        if (parts.length == 2) {
          try {
            int key = Integer.parseInt(parts[0]);
            String newValue = parts[1];

            List<DataRecord> records = readDataFromFile();
            boolean updated = false;
```

```
                for (DataRecord record : records) {
                    if (record.getKey() == key) {
                        record.setValue(newValue);
                        updated = true;
                        break;
                    }
                }

                if (updated) {
                    writeAllDataToFile(records);
                    System.out.println("Data updated successfully.");
                    table.getItems().clear();
                    displayData();
                } else {
                    showAlert("Update Failed", "Key not found. Data update failed.");
                }
            } catch (NumberFormatException e) {
                showAlert("Invalid Input", "Key must be an integer.");
            }
        } else {
            showAlert("Invalid Input", "Please enter Key and New Value separated by
space.");
        }
    });
  }
}

@FXML
private void displayData() {
  if (filename != null) {
    List<DataRecord> records = readDataFromFile();
    table.getItems().addAll(records);

    StringBuilder fileContents = new StringBuilder();
    try {
      Scanner scanner = new Scanner(new File(filename));
      while (scanner.hasNextLine()) {
        fileContents.append(scanner.nextLine()).append("\n");
      }
      scanner.close();
    } catch (IOException e) {
      e.printStackTrace();
    }
    fileContentsTextArea.setText(fileContents.toString());
```

```
    }
}

@FXML
private void deleteData() {
    if (filename != null) {
        TextInputDialog dialog = new TextInputDialog();
        dialog.setTitle("Delete Data");
        dialog.setHeaderText(null);
        dialog.setContentText("Enter Key to delete:");

        Optional<String> result = dialog.showAndWait();
        result.ifPresent(input -> {
            try {
                int keyToDelete = Integer.parseInt(input);
                List<DataRecord> records = readDataFromFile();
                boolean found = false;

                Iterator<DataRecord> iterator = records.iterator();
                while (iterator.hasNext()) {
                    DataRecord record = iterator.next();
                    if (record.getKey() == keyToDelete) {
                        iterator.remove();
                        found = true;
                        break;
                    }
                }

                if (found) {
                    writeAllDataToFile(records);
                    System.out.println("Data deleted successfully.");
                    table.getItems().clear();
                    displayData();
                } else {
                    showAlert("Delete Failed", "Key not found. No data deleted.");
                }
            } catch (NumberFormatException e) {
                showAlert("Invalid Input", "Key must be an integer.");
            }
        });
    }
}

@FXML
private void clearFile() {
```

```
        if (HelloApplication.getFilename() != null) {
            try {
                Files.deleteIfExists(Paths.get(HelloApplication.getFilename()));
                HelloApplication.setFilename(null);
                System.out.println("File cleared.");
                table.getItems().clear();
                fileContentsTextArea.clear();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    @FXML
    private void createFile() {
        FileChooser fileChooser = new FileChooser();
        fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("Data type",
"*.dat"));
        File file = fileChooser.showSaveDialog(null);
        if (file != null) {
            String filePath = file.getAbsolutePath();
            if (filePath.endsWith(".dat") && filePath.split("\\.").length == 2) {
                try {
                    Files.createFile(Paths.get(filePath));
                    HelloApplication.setFilename(filePath);
                    fileStatusLabel.setText("Selected file:");
                    filePathLabel.setText(filePath);
                    updateDataTable();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            } else if (!filePath.contains(".")) {
                fileStatusLabel.setText("File is not selected");
                filePathLabel.setText("");
            } else {
                System.out.println("Invalid file extension. Please choose a .dat file.");
            }
        }
    }

    private void updateDataTable() {
        if (HelloApplication.getFilename() != null) {
            List<HelloApplication.DataRecord> data = HelloApplication.readDataFromFile();
            table.getItems().clear();
            table.getItems().addAll(data);
        }
```

```
    }
        }
```

```
import com.example.labwork3.HelloApplication;
import javafx.application.Platform;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.io.TempDir;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import static org.junit.jupiter.api.Assertions.*;

class DataRecordTest {

    private static final String TEST_DATA = "1 testValue";

    @TempDir
    static Path sharedTempDir;

    @BeforeEach
    void setUp() {
        HelloApplication.setFilename(sharedTempDir.resolve("test.dat").toString());
    }
    @Test
    void testCreateDataRecord() {
        HelloApplication.DataRecord record = new HelloApplication.DataRecord (1,
"value");
        assertNotNull(record);
        assertEquals(1, record.getKey());
        assertEquals("value", record.getValue());
    }

    @Test
    void testSetValue() {
        HelloApplication.DataRecord record = new HelloApplication.DataRecord (1,
"initialValue");
        record.setValue("newValue");
        assertEquals("newValue", record.getValue());
```

```
    }

    @Test
    void testWriteDataToFile() throws Exception {
        HelloApplication.writeDataToFile(1, "testValue");

        List <String> lines = Files.readAllLines( Paths.get(HelloApplication.getFilename()));
        assertTrue(lines.contains(TEST_DATA));
    }

    @Test
    void testWriteAllData() throws Exception {
        List<HelloApplication.DataRecord> records = new ArrayList <> ();
        records.add(new HelloApplication.DataRecord(1, "testValue"));

        HelloApplication.writeAllDataToFile(records);

        List<String> lines = Files.readAllLines(Paths.get(HelloApplication.getFilename()));
        assertEquals(1, lines.size());
        assertTrue(lines.contains(TEST_DATA));
    }

    @Test
    void testReadDataFromFile() throws Exception {
        Files.write(Paths.get(HelloApplication.getFilename()),
Collections.singletonList(TEST_DATA));

        List<HelloApplication.DataRecord> records = HelloApplication.readDataFromFile();

        assertFalse(records.isEmpty());
        HelloApplication.DataRecord record = records.get(0);
        assertEquals(1, record.getKey());
        assertEquals("testValue", record.getValue());
    }
    @Test
    public void testSetAndGetFilename() {
        HelloApplication.setFilename("test.dat");
        assertEquals("test.dat", HelloApplication.getFilename());
    }

}

<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
```

```
<?import javafx.scene.control.cell.PropertyValueFactory?>

<BorderPane xmlns="http://javafx.com/javafx" xmlns:fx="http://javafx.com/fxml"
fx:controller="com.example.labwork3.GUI" style="-fx-background-color: #f4f4f4;">
  <top>
    <VBox spacing="15" style="-fx-background-color: #ffffff; -fx-padding: 20;">
      <Label fx:id="fileStatusLabel" text="File is not selected" style="-fx-font-size: 18px;
-fx-font-weight: bold;"/>
      <Label fx:id="filePathLabel" style="-fx-font-size: 14px; -fx-text-fill: #666;"/>
      <MenuBar>
        <Menu text="File">
          <MenuItem text="Open" onAction="#selectFile"/>
          <MenuItem text="Clear File" onAction="#clearFile"/>
          <MenuItem text="Create File" onAction="#createFile"/>
        </Menu>
      </MenuBar>
      <HBox spacing="15">
        <Button text="Add Data" onAction="#addData" style="-fx-background-color:
#5cb85c; -fx-text-fill: white; -fx-font-weight: bold; -fx-padding: 10 20;"/>
        <Button text="Update Data" onAction="#updateData" style="-fx-background-
color: #f0ad4e; -fx-text-fill: white; -fx-font-weight: bold; -fx-padding: 10 20;"/>
        <Button text="Delete Data" onAction="#deleteData" style="-fx-background-
color: #d9534f; -fx-text-fill: white; -fx-font-weight: bold; -fx-padding: 10 20;"/>
      </HBox>
    </VBox>
  </top>

  <center>
    <TableView fx:id="table" style="-fx-background-color: #ffffff; -fx-table-cell-border-
color: transparent; -fx-base: #ffffff; -fx-padding: 5;">
      <columns>
        <TableColumn text="Key" minWidth="200" style="-fx-font-weight: bold;">
          <cellValueFactory>
            <PropertyValueFactory property="key"/>
          </cellValueFactory>
        </TableColumn>
        <TableColumn text="Value" minWidth="200" style="-fx-font-weight: bold;">
          <cellValueFactory>
            <PropertyValueFactory property="value"/>
          </cellValueFactory>
        </TableColumn>
      </columns>
    </TableView>
  </center>
</BorderPane>
```

3.3.2 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми для додавання і пошуку запису.

Рисунок 3.1 –Додавання запису

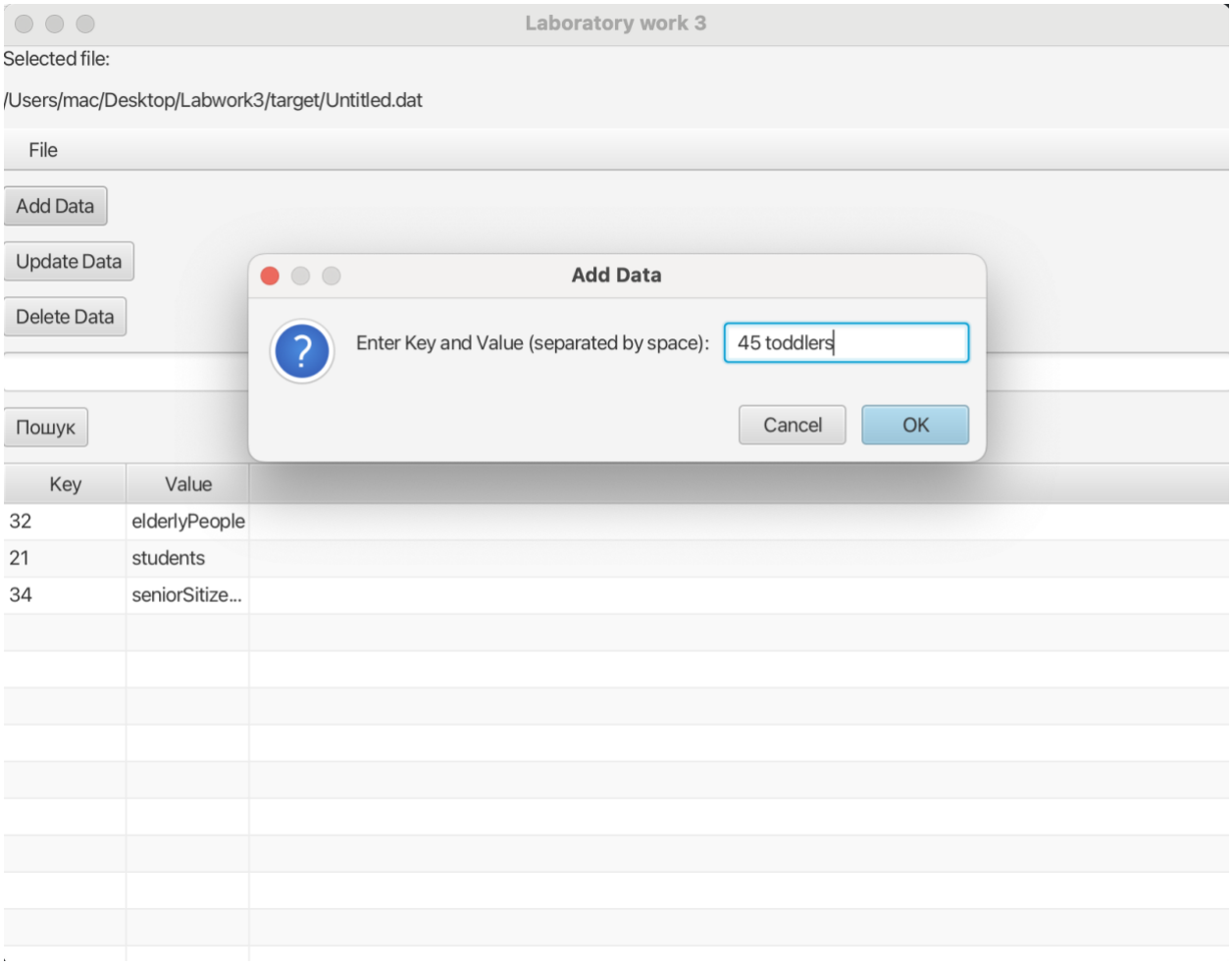
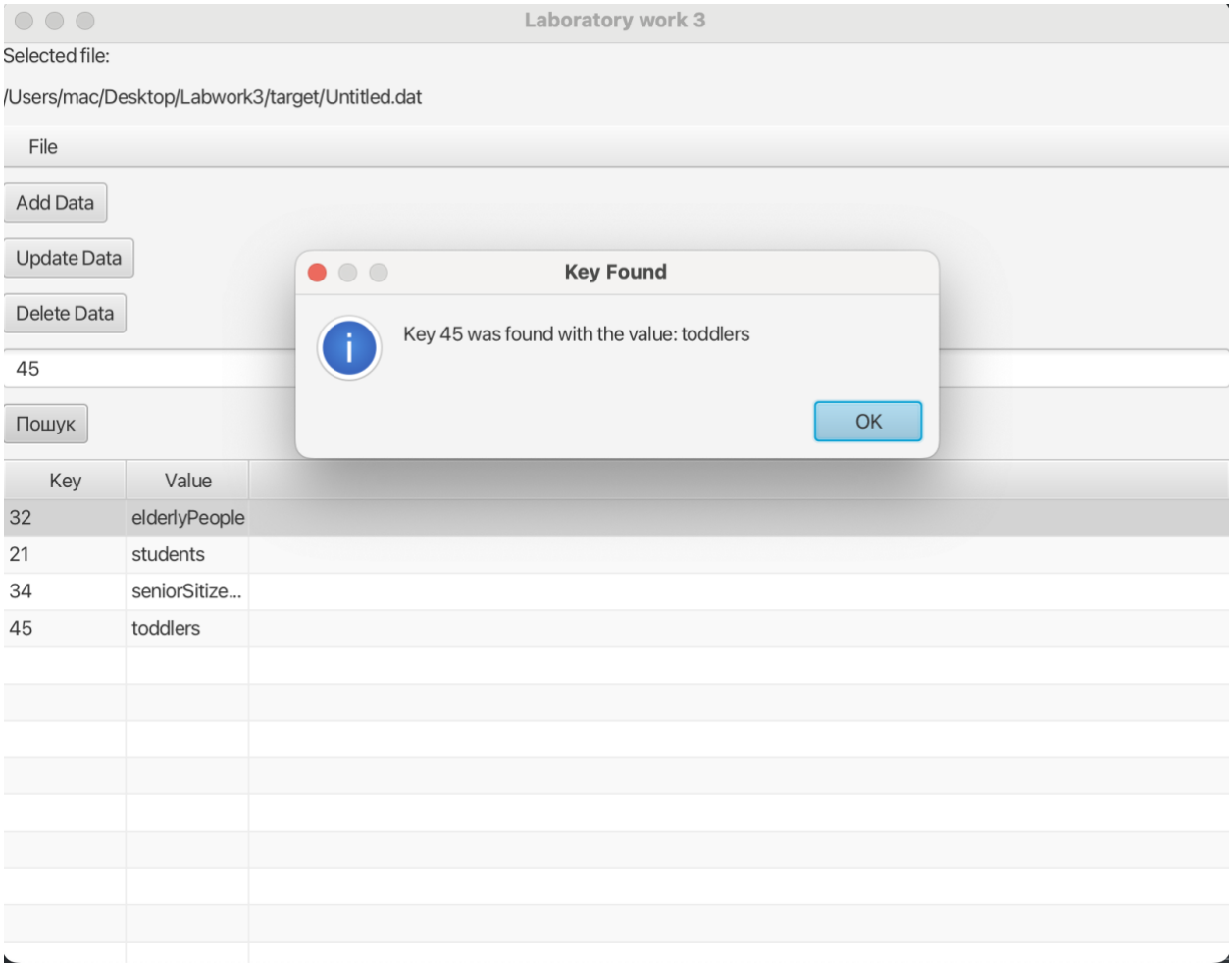


Рисунок 3.2 – Пошук запису



3.4 Тестування алгоритму

3.4.1 Часові характеристики оцінювання

В таблиці 3.1 наведено кількість порівнянь для 15 спроб пошуку запису по ключу.

Таблиця 3.1 – Число порівнянь при спробі пошуку запису по ключу

Номер спроби пошуку	Число порівнянь
1	8
2	12
3	23
4	34
5	63

ВИСНОВОК

В рамках лабораторної роботи було створено JavaFX додаток "HelloApplication", який дозволяє користувачеві працювати з файлами, вводити, оновлювати, видаляти та відображати дані. Додаток має графічний інтерфейс, який включає в себе можливість вибору та створення файлів, відображення таблиці з даними та текстову область для перегляду вмісту файлу. Користувач може додавати нові записи, оновлювати існуючі та видаляти записи з файлу.

КРИТЕРІЇ ОЦІНЮВАННЯ

За умови здачі лабораторної роботи до 13.11.2022 включно максимальний бал дорівнює – 5. Після 13.11.2022 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

- псевдокод алгоритму – 15%;
- аналіз часової складності – 5%;
- програмна реалізація алгоритму – 65%;
- тестування алгоритму – 10%;
- висновок – 5%.

+1 додатковий бал можна отримати за реалізацію графічного зображення структури ключів.