Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Санкт-Петербургский государственный электротехнический университет “ЛЭТИ” им.В.И.Ульянова (Ленина) »

Кафедра МОЭВМ

ОТЧЕТ
по лабораторно-практической работе № 5
«Сохранение и загрузка данных из файла»
по дисциплине «Объектно - ориентированное программирование на языке Java»

Выполнил Сапронов К.Д.

Факультет КТИ

Группа № 3311

Подпись преподавателя _____

Санкт-Петербург

2024 г

Цель работы

Знакомство с организацией обмена данными между объектами экранной формы и файлом.

Описание задания

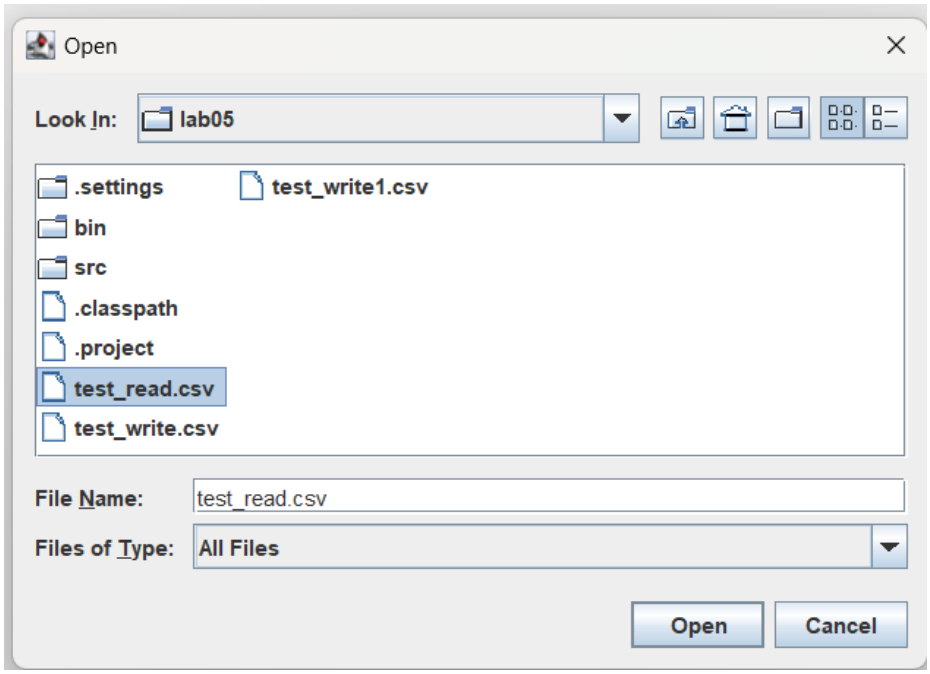
1. Проанализируйте разрабатываемое приложение и подготовьте в текстовом редакторе данные для его работы.
2. Напишите и добавьте в проект обработчики кнопок загрузки текста в файл и выгрузки из него.
3. Загрузить данные из файла в экранную форму приложения, изменить их и сохранить в файле, проверить содержимое файла

Работа с файлами

Для загрузки и сохранения данных в файл были созданы методы loadDataFromFile и saveDataFromFile соответственно. Эти методы работают с файлами .csv.

1. Чтение из файла

После нажатия File>Open пользователь выбирает файл через JFileChooser. После открытия файла данные оттуда считываются в буфер, откуда построчно переносятся в таблицу. Таблица перед этим очищается. Данные внутри одной строки разделяются по ячейкам через запятую.

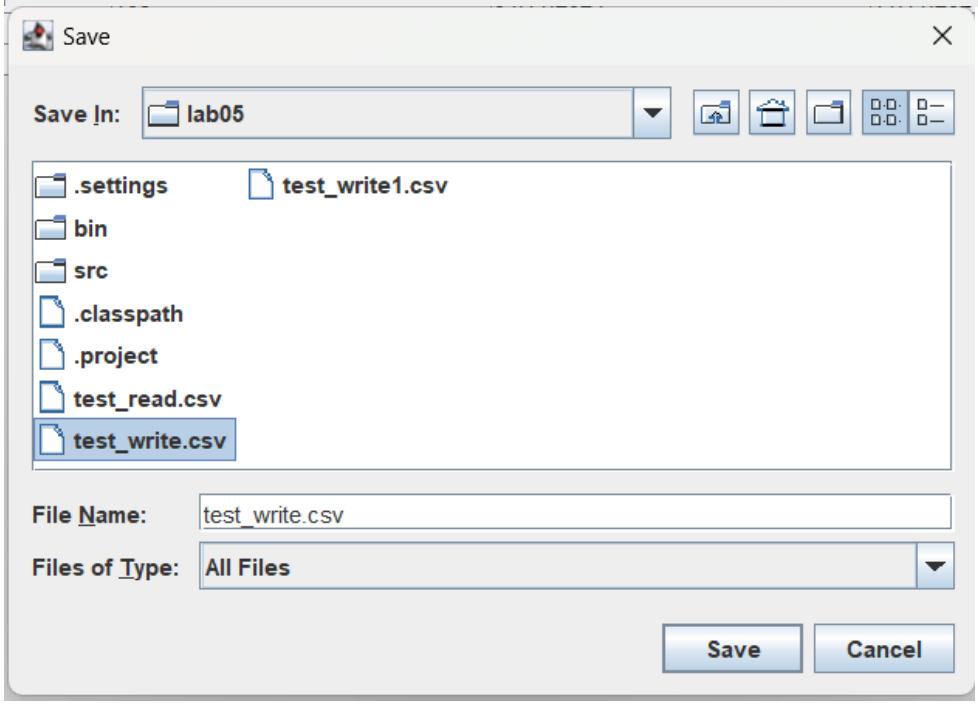


Name	Room	Check-in Date	Check-out Date
John Doe	106	04.11.2024	11.11.2024
Jane Doe	208	11.11.2024	18.11.2024

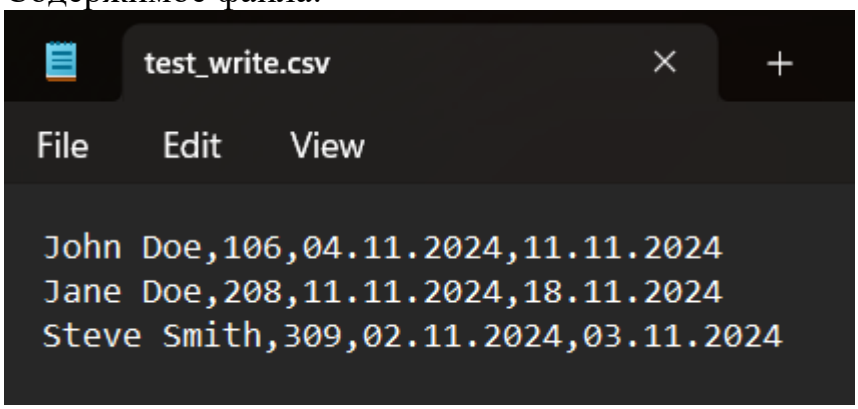
2. Запись в файл

После нажатия File>Save пользователь также выбирает файл через JFileChooser. После этого каждая строка записывается в буфер, разделяя содержимое ячеек запятыми, после чего происходит запись в файл.

Name	Room	Check-in Date	Check-out Date
John Doe	106	04.11.2024	11.11.2024
Jane Doe	208	11.11.2024	18.11.2024
Steve Smith	309	02.11.2024	03.11.2024



Содержимое файла:



Ссылки

https://drive.google.com/drive/folders/1SkmitiMaArA7aWjd8Q5cThXVNp8ws_N4?usp=drive_link

В этой папке будут находиться все лабораторные работы

В папке lab4 находятся этот отчет, видеоотчет и папка lab05, в которой находятся файлы проекта и документация javadoc.

Текст программы

```
package lab05;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;

/**
 * Custom exception that is thrown when an input field is empty.
 */
class EmptyFieldException extends Exception {
    public EmptyFieldException(String message) {
        super(message);
    }
}

/**
 * This class creates the GUI for the Hotel Guest List application.
 */
class GUI {
    private JFrame frame;
    private JMenuBar menuBar;
    private JMenu fileMenu, sortMenu;
    private JMenuItem openItem, saveItem, roomItem, nameItem;
    private JToolBar toolBar;
    private JButton saveButton, addButton, deleteButton;
    private JButton searchButton;
    private JComboBox<String> searchType;
    private JTextField searchField;
    private JTable dataTable;
    private JScrollPane tableScrollPane;
    private DefaultTableModel tableModel;

    /**
     * Sets up the GUI and displays it.
     */
    public void buildAndShowGUI() {
        frame = new JFrame("Hotel - Guest List");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 600);

        // Menu bar
        menuBar = new JMenuBar();
        fileMenu = new JMenu("File");
        openItem = new JMenuItem("Open");
        saveItem = new JMenuItem("Save");
```

```
        fileMenu.add(openItem);
        fileMenu.add(saveItem);
        menuBar.add(fileMenu);

        // Sort menu
        sortMenu = new JMenu("Sort by");
        roomItem = new JMenuItem("Room");
        nameItem = new JMenuItem("Name");
        sortMenu.add(nameItem);
        sortMenu.add(roomItem);
        menuBar.add(sortMenu);

        frame.setJMenuBar(menuBar);

        // Toolbar
        toolBar = new JToolBar();
        saveButton = new JButton("Save");
        addButton = new JButton("Add");
        deleteButton = new JButton("Delete");
        toolBar.add(saveButton);
        toolBar.add(addButton);
        toolBar.add(deleteButton);
        frame.add(toolBar, BorderLayout.NORTH);

        // Search panel
        JPanel searchPanel = new JPanel();
        searchType = new JComboBox<>(new String[]{"Name", "Room", "Date"});
        searchField = new JTextField(15);
        searchButton = new JButton("Search");
        searchPanel.add(new JLabel("Search by:"));
        searchPanel.add(searchType);
        searchPanel.add(searchField);
        searchPanel.add(searchButton);
        frame.add(searchPanel, BorderLayout.SOUTH);

        // Table setup
        String[] columns = {"Name", "Room", "Check-in Date", "Check-out Date"};
        tableModel = new DefaultTableModel(columns, 0);
        dataTable = new JTable(tableModel);
        tableScrollPane = new JScrollPane(dataTable);
        frame.add(tableScrollPane, BorderLayout.CENTER);

        addListeners();
        frame.setVisible(true);
    }

    /**
     * Adds event listeners to the GUI components.
     */
    private void addListeners() {
        saveButton.addActionListener(e -> saveDataToFile());
        saveItem.addActionListener(e -> saveDataToFile());
        openItem.addActionListener(e -> loadDataFromFile());

        addButton.addActionListener(e -> {
            try {
                String name = JOptionPane.showInputDialog("Enter guest name:");
                String room = JOptionPane.showInputDialog("Enter room number:");
                String checkIn = JOptionPane.showInputDialog("Enter check-in date
(dd.mm.yyyy):");
                String checkOut = JOptionPane.showInputDialog("Enter check-out date
(dd.mm.yyyy):");
                validateGuestInput(name, room, checkIn, checkOut);
                tableModel.addRow(new Object[]{name, room, checkIn, checkOut});
                JOptionPane.showMessageDialog(frame, "Guest added successfully!");
            } catch (EmptyFieldException ex) {
```

```
                JOptionPane.showMessageDialog(frame, ex.getMessage());
            }
        });

        deleteButton.addActionListener(e -> {
            int selectedRow = dataTable.getSelectedRow();
            if (selectedRow != -1) {
                int confirm = JOptionPane.showConfirmDialog(frame, "Are you sure you want
to delete the selected row?", "Delete", JOptionPane.YES_NO_OPTION);
                if (confirm == JOptionPane.YES_OPTION) {
                    tableModel.removeRow(selectedRow);
                }
            } else {
                JOptionPane.showMessageDialog(frame, "Please select a row to delete.");
            }
        });

        searchButton.addActionListener(e -> {
            try {
                String searchTerm = searchField.getText();
                if (searchTerm.isEmpty()) throw new EmptyFieldException("The search field
cannot be empty!");
                int selectedColumn = searchType.getSelectedIndex();
                for (int i = 0; i < dataTable.getRowCount(); i++) {
                    String value = dataTable.getValueAt(i, selectedColumn).toString();
                    if (value.toLowerCase().contains(searchTerm.toLowerCase())) {
                        dataTable.setRowSelectionInterval(i, i);
                        JOptionPane.showMessageDialog(frame, "Match found: " + value);
                        return;
                    }
                }
                JOptionPane.showMessageDialog(frame, "No matches found.");
            } catch (EmptyFieldException ex) {
                JOptionPane.showMessageDialog(frame, ex.getMessage());
            }
        });

        nameItem.addActionListener(e -> sortTable(0));
        roomItem.addActionListener(e -> sortTable(1));
    }

    /**
     * Sorts the table based on the specified column index.
     */
    private void sortTable(int columnIndex) {
        int rowCount = tableModel.getRowCount();
        Object[][] tableData = new Object[rowCount][tableModel.getColumnCount()];

        for (int i = 0; i < rowCount; i++) {
            for (int j = 0; j < tableModel.getColumnCount(); j++) {
                tableData[i][j] = tableModel.getValueAt(i, j);
            }
        }

        java.util.Arrays.sort(tableData, java.util.Comparator.comparing(o ->
o[columnIndex].toString()));

        tableModel.setRowCount(0);
        for (Object[] row : tableData) {
            tableModel.addRow(row);
        }

        JOptionPane.showMessageDialog(frame, "Table sorted by " +
tableModel.getColumnName(columnIndex));
    }
```

```
    /**
     * Saves table data to a file.
     */
    private void saveDataToFile() {
        JFileChooser fileChooser = new JFileChooser();
        if (fileChooser.showSaveDialog(frame) == JFileChooser.APPROVE_OPTION) {
            File file = fileChooser.getSelectedFile();
            try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {
                for (int i = 0; i < tableModel.getRowCount(); i++) {
                    for (int j = 0; j < tableModel.getColumnCount(); j++) {
                        writer.write(tableModel.getValueAt(i, j).toString());
                        if (j < tableModel.getColumnCount() - 1) writer.write(",");
                    }
                    writer.newLine();
                }
                JOptionPane.showMessageDialog(frame, "Data saved successfully!");
            } catch (IOException ex) {
                JOptionPane.showMessageDialog(frame, "Error saving file: " +
ex.getMessage());
            }
        }
    }

    /**
     * Loads table data from a file.
     */
    private void loadDataFromFile() {
        JFileChooser fileChooser = new JFileChooser();
        if (fileChooser.showOpenDialog(frame) == JFileChooser.APPROVE_OPTION) {
            File file = fileChooser.getSelectedFile();
            try (BufferedReader reader = new BufferedReader(new FileReader(file))) {
                tableModel.setRowCount(0);
                String line;
                while ((line = reader.readLine()) != null) {
                    String[] data = line.split(",");
                    tableModel.addRow(data);
                }
                JOptionPane.showMessageDialog(frame, "File loaded successfully!");
            } catch (IOException ex) {
                JOptionPane.showMessageDialog(frame, "Error opening file: " +
ex.getMessage());
            }
        }
    }

    /**
     * Validates guest input fields.
     */
    private void validateGuestInput(String name, String room, String checkIn, String
checkOut) throws EmptyFieldException {
        if (name == null || name.trim().isEmpty()) throw new EmptyFieldException("Guest
name cannot be empty.");
        if (room == null || room.trim().isEmpty()) throw new EmptyFieldException("Room
number cannot be empty.");
        if (checkIn == null || checkIn.trim().isEmpty()) throw new
EmptyFieldException("Check-in date cannot be empty.");
        if (checkOut == null || checkOut.trim().isEmpty()) throw new
EmptyFieldException("Check-out date cannot be empty.");
    }
}

/**
 * Main class to start the application.
 */
public class HotelGUI {
    public static void main(String[] args) {
```

```
        SwingUtilities.invokeLater(() -> new GUI().buildAndShowGUI());
    }
}
```