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 № 6
# «Обработка XML-документов»
# 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 технологией обработки XML-документов и файлов.

## Описание задания

1. Заменить обработчики записи в файл и чтения из файла для работы с XML-файлами
2. Загрузить данные из файла, изменить их, сохранить в новый файл. Проверить корректность работы, открыв новый файл

## Работа с файлами

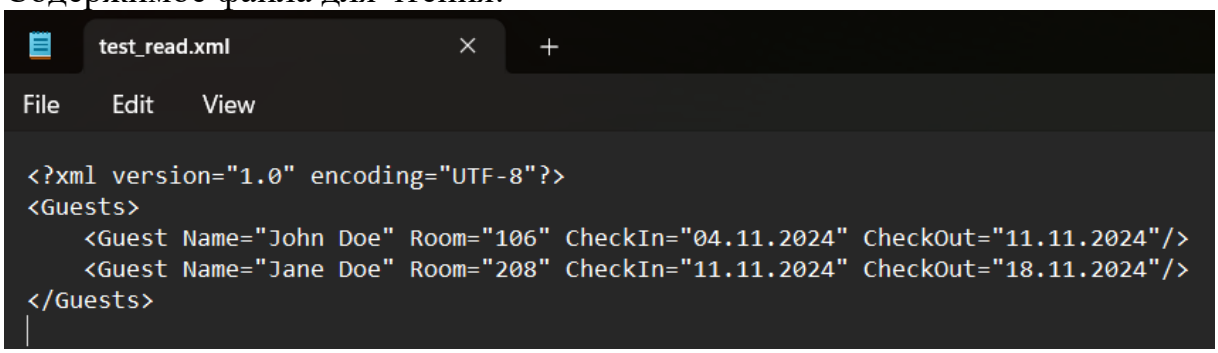
Методы loadDataFromFile и saveDataFromFile служат для загрузки и сохранения данных в файл соответственно. Они были изменены для работы с XML-документами.

### 1. Чтение из файла

После нажатия File>Open пользователь выбирает файл через JFileChooser. После выбора файла создается парсер, читающий документ, после чего полученный документ нормализуется и из него получаются все данные с тэгом Guest. После этого таблица очищается, и каждая ее строка заполняется соответствующими атрибутами каждого элемента.

| 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 |

Содержимое файла для чтения:

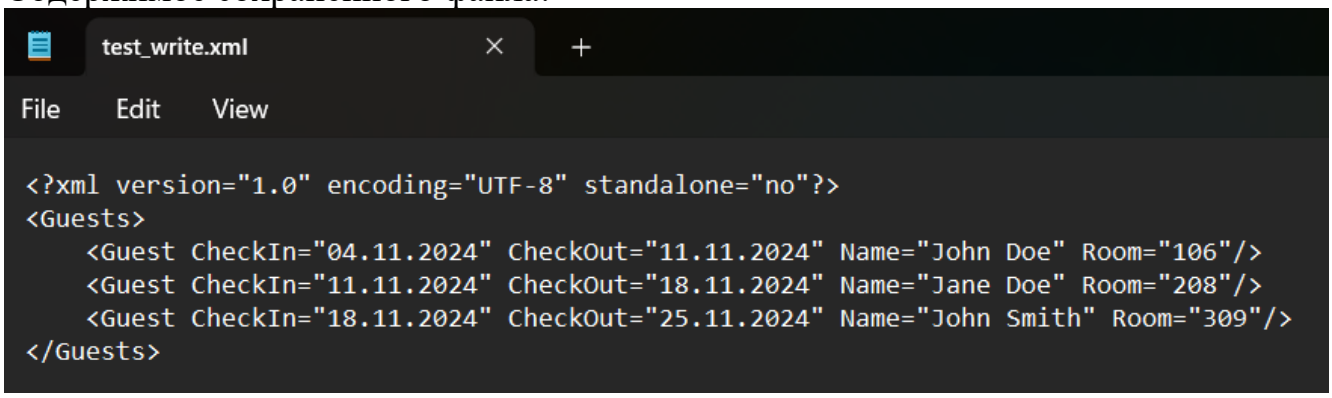



### 2. Запись в файл

После нажатия File>Save пользователь также выбирает файл через JFileChooser. После выбора создается новый документ с корневым элементом Guests, после чего каждая строка таблицы записывается в качестве элемента в документ, с атрибутами соответствующих ячеек таблицы. После этого с помощью преобразователя документ сохраняется как файл XML.

| 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 |
| John Smith | 309 | 18.11.2024 | 25.11.2024 |

Содержимое сохраненного файла:



## Ссылки

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

В этой папке будут находиться все лабораторные работы

В папке lab6 находятся этот отчет, видеоотчет и папка lab06, в которой находятся файлы проекта и документация javadoc.

## Текст программы

```
package lab06;

import org.w3c.dom.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import org.xml.sax.SAXException;

class EmptyFieldException extends Exception {
    public EmptyFieldException(String message) {
        super(message);
    }
}

class GUI {
    private JFrame frame;
    private JMenuBar menuBar;
    private JMenu fileMenu, sortMenu;
    private JMenuItem openItem, saveItem, roomItem, nameItem;
    private JToolBar toolBar;
    private JButton addButton, deleteButton, searchButton;
    private JComboBox<String> searchType;
    private JTextField searchField;
    private JTable dataTable;
    private JScrollPane tableScrollPane;
    private DefaultTableModel tableModel;

    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();
        addButton = new JButton("Add");
        deleteButton = new JButton("Delete");
        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);
    }

    private void addListeners() {
        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));
    }

    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));
    }

    private void saveDataToFile() {
        JFileChooser fileChooser = new JFileChooser();
        if (fileChooser.showSaveDialog(frame) == JFileChooser.APPROVE_OPTION) {
            File file = fileChooser.getSelectedFile();
            try {
                DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
                DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
                Document doc = dBuilder.newDocument();
```

```
                // Root element
                Element rootElement = doc.createElement("Guests");
                doc.appendChild(rootElement);

                for (int i = 0; i < tableModel.getRowCount(); i++) {
                    Element guest = doc.createElement("Guest");
                    guest.setAttribute("Name", (String) tableModel.getValueAt(i, 0));
                    guest.setAttribute("Room", (String) tableModel.getValueAt(i, 1));
                    guest.setAttribute("CheckIn", (String) tableModel.getValueAt(i, 2));
                    guest.setAttribute("CheckOut", (String) tableModel.getValueAt(i, 3));
                    rootElement.appendChild(guest);
                }

                TransformerFactory transformerFactory = TransformerFactory.newInstance();
                Transformer transformer = transformerFactory.newTransformer();
                transformer.setOutputProperty(OutputKeys.INDENT, "yes");
                DOMSource source = new DOMSource(doc);
                StreamResult result = new StreamResult(new FileWriter(file));
                transformer.transform(source, result);

                JOptionPane.showMessageDialog(frame, "Data saved successfully!");
            } catch (ParserConfigurationException | TransformerException | IOException
ex) {
                JOptionPane.showMessageDialog(frame, "Error saving file: " +
ex.getMessage());
            }
        }
    }

    private void loadDataFromFile() {
        JFileChooser fileChooser = new JFileChooser();
        if (fileChooser.showOpenDialog(frame) == JFileChooser.APPROVE_OPTION) {
            File file = fileChooser.getSelectedFile();
            try {
                DocumentBuilder dBuilder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
                Document doc = dBuilder.parse(file);
                doc.getDocumentElement().normalize();

                tableModel.setRowCount(0);

                NodeList guestList = doc.getElementsByTagName("Guest");
                for (int i = 0; i < guestList.getLength(); i++) {
                    Node guestNode = guestList.item(i);
                    if (guestNode.getNodeType() == Node.ELEMENT_NODE) {
                        NamedNodeMap attributes = guestNode.getAttributes();
                        String name = attributes.getNamedItem("Name").getNodeValue();
                        String room = attributes.getNamedItem("Room").getNodeValue();
                        String checkIn =
attributes.getNamedItem("CheckIn").getNodeValue();
                        String checkOut =
attributes.getNamedItem("CheckOut").getNodeValue();

                        tableModel.addRow(new String[]{name, room, checkIn, checkOut});
                    }
                }

                JOptionPane.showMessageDialog(frame, "File loaded successfully!");

            } catch (ParserConfigurationException | SAXException | IOException ex) {
                JOptionPane.showMessageDialog(frame, "Error opening file: " +
ex.getMessage());
            }
        }
    }
```

```
    private void validateGuestInput(String name, String room, String checkIn, String
checkOut) throws EmptyFieldException {
        if (name.isEmpty() || room.isEmpty() || checkIn.isEmpty() || checkOut.isEmpty())
{
            throw new EmptyFieldException("All fields are required!");
        }
    }
}

public class HotelGUI {
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new GUI().buildAndShowGUI());
    }
}
```