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 Васильев Т.В.

Факультет КТИ

Группа № 3312

Подпись преподавателя _____

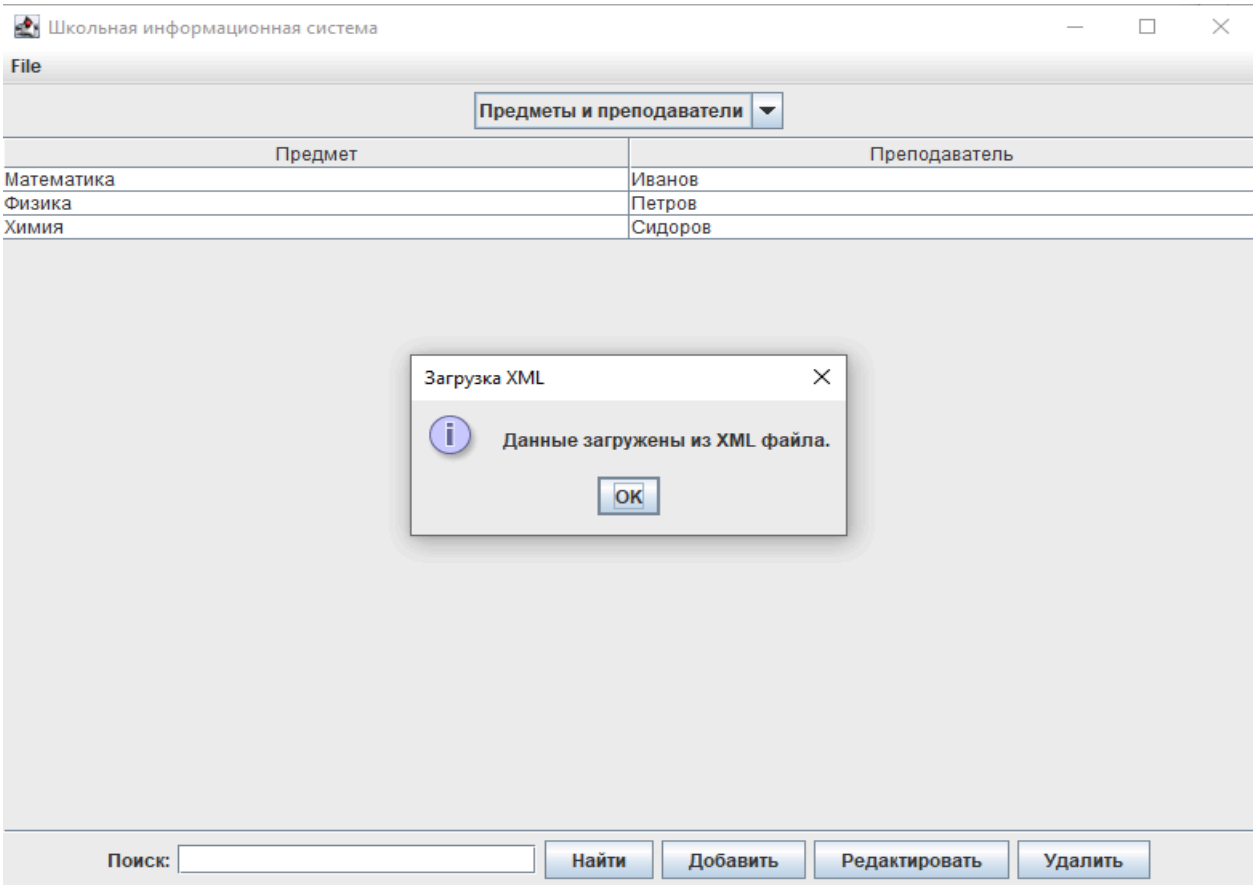
Санкт-Петербург

2024 г

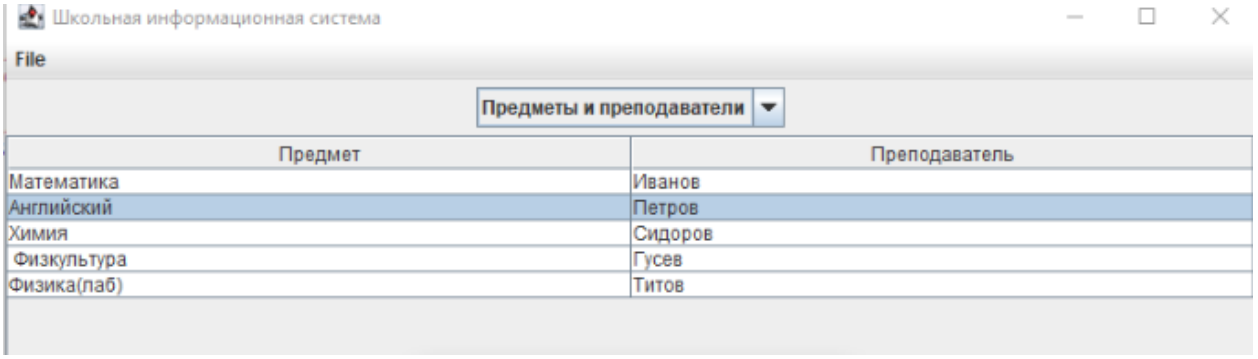
Цель работы

Знакомство с технологией обработки XML-документов и файлов

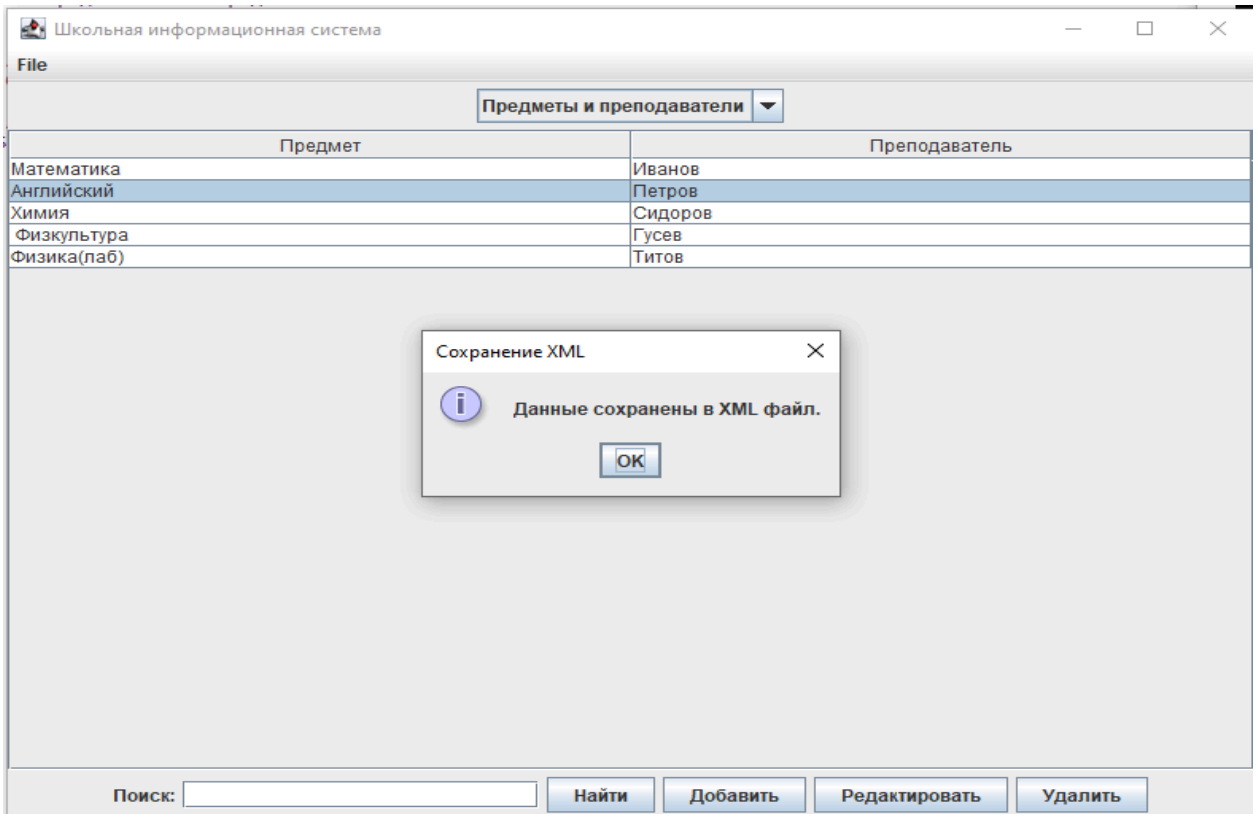
Пример работы программы



Пример 1 - Загрузка из Xml файла



Пример 2 - измененные данные



Пример 3 - Сохранение изменений в Xml файл

Содержимое файлов

```
<SchoolData>
  <Students>
    <Student>
      <Фамилия>Иванов</Фамилия>
      <Класс>5А</Класс>
      <Успеваемость>Отличник</Успеваемость>
    </Student>
    <Student>
      <Фамилия>Петров</Фамилия>
      <Класс>6Б</Класс>
      <Успеваемость>Двоечник</Успеваемость>
    </Student>
    <Student>
      <Фамилия>Сидоров</Фамилия>
      <Класс>7В</Класс>
      <Успеваемость>Хорошист</Успеваемость>
    </Student>
    <Student>
      <Фамилия>Кузнецов</Фамилия>
      <Класс>5А</Класс>
      <Успеваемость>Хорошист</Успеваемость>
    </Student>
    <Student>
      <Фамилия>Смирнов</Фамилия>
      <Класс>6Б</Класс>
      <Успеваемость>Отличник</Успеваемость>
    </Student>
    <Student>
      <Фамилия>Попов</Фамилия>
      <Класс>7В</Класс>
      <Успеваемость>Двоечник</Успеваемость>
    </Student>
  </Students>
  <Subjects>
    <Subject>
      <Предмет>Математика</Предмет>
      <Преподаватель>Иванов</Преподаватель>
    </Subject>
    <Subject>
      <Предмет>Физика</Предмет>
      <Преподаватель>Петров</Преподаватель>
    </Subject>
    <Subject>
      <Предмет>Химия</Предмет>
      <Преподаватель>Сидоров</Преподаватель>
    </Subject>
  </Subjects>
  <Classes>
    <Class>
      <Преподаватель>Иванов</Преподаватель>
      <Классы>5А,6Б</Классы>
    </Class>
    <Class>
      <Преподаватель>Петров</Преподаватель>
      <Классы>6Б,7В</Классы>
    </Class>
    <Class>
      <Преподаватель>Сидоров</Преподаватель>
      <Классы>7В,8А</Классы>
    </Class>
  </Classes>
</SchoolData>
```

Пример 4 - стартовые сохраненные данные

```
<SchoolData>
  <Students>
    ...
  </Students>
  <Subjects>
    <Subject>
      <Предмет>Математика</Предмет>
      <Преподаватель>Иванов</Преподаватель>
    </Subject>
    <Subject>
      <Предмет>Физика</Предмет>
      <Преподаватель>Петров</Преподаватель>
    </Subject>
    <Subject>
      <Предмет>Химия</Предмет>
      <Преподаватель>Сидоров</Преподаватель>
    </Subject>
    <Subject>
      <Предмет> Физкультура</Предмет>
      <Преподаватель>Гусев</Преподаватель>
    </Subject>
    <Subject>
      <Предмет>Физика(лаб)</Предмет>
      <Преподаватель>Титов</Преподаватель>
    </Subject>
  </Subjects>
  <Classes>
    ...
  </Classes>
</SchoolData>
```

Пример 5 - данные, сохраненные после изменения в программе

Текст программы

```
package lab6;
import javax.swing.*;
import lab6.FileSaveException;
import lab6.SearchException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
/**
* Класс создает экранную форму
* @version 1.1
*/
public class SchoolApp extends JFrame {

   private JTable subjectTable, classTable, studentTable;
   private JPanel tablePanel;
   private JComboBox<String> tableSelector, classSelector;
   private CardLayout cardLayout;
   private JTextField searchField;
   private JButton searchButton;
   private JButton addButton, editButton, deleteButton;
   private String[][] allStudentData = {
       {"Иванов", "5А", "Отличник"},
       {"Петров", "6Б", "Двоечник"},
       {"Сидоров", "7В", "Хорошист"},
       {"Кузнецов", "5А", "Хорошист"},
       {"Смирнов", "6Б", "Отличник"},
       {"Попов", "7В", "Двоечник"}
   };
   private String[] studentColumns = {"Фамилия", "Класс", "Успеваемость"};
   private String[][] originalStudentData;
   private String[][] subjectData = {
       {"Математика", "Иванов"},
       {"Физика", "Петров"},
       {"Химия", "Сидоров"}
```

```
    };
    private String[] subjectColumns = {"Предмет", "Преподаватель"};

    private String[][] classData = {
        {"Иванов", "5А,6Б"},
        {"Петров", "6Б,7В"},
        {"Сидоров", "7В,8А"}
    };
    private String[] classColumns = {"Преподаватель", "Классы"};
    public SchoolApp() {
        super("Школьная информационная система");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(800, 600);
        setLayout(new BorderLayout());
        originalStudentData = allStudentData.clone();
        subjectTable = new JTable(subjectData, subjectColumns);
        classTable = new JTable(classData, classColumns);
        studentTable = new JTable(allStudentData, studentColumns);
        cardLayout = new CardLayout();
        tablePanel = new JPanel(cardLayout);
        tablePanel.add(new JScrollPane(subjectTable), "Предметы и преподаватели");
        tablePanel.add(new JScrollPane(classTable), "Преподаватели и классы");
        tablePanel.add(new JScrollPane(studentTable), "Ученики");
        tableSelector = new JComboBox<>(new String[]{"Предметы и преподаватели",
"Преподаватели и классы", "Ученики"});
        tableSelector.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String selectedTable = (String) tableSelector.getSelectedItem();
                cardLayout.show(tablePanel, selectedTable);
                classSelector.setVisible(selectedTable.equals("Ученики"));
            }
        });
        classSelector = new JComboBox<>(new String[]{"Все классы", "5А", "6Б",
"7В"});
        classSelector.setVisible(false);
        JMenuBar menuBar = new JMenuBar();
        JMenu fileMenu = new JMenu("File");
        JMenuItem newItem = new JMenuItem("New");
        JMenuItem saveItem = new JMenuItem("Save file");
        JMenuItem importItem = new JMenuItem("Import");
        JMenuItem exportItem = new JMenuItem("Export");
        JMenuItem saveXmlItem = new JMenuItem("Save XML");
        JMenuItem loadXmlItem = new JMenuItem("Load XML");
        fileMenu.add(newItem);
        fileMenu.add(saveItem);
        fileMenu.add(importItem);
        fileMenu.add(exportItem);
        menuBar.add(fileMenu);
        fileMenu.add(saveXmlItem);
        fileMenu.add(loadXmlItem);
        setJMenuBar(menuBar);
        JPanel searchPanel = new JPanel();
        searchPanel.setLayout(new FlowLayout());
        JLabel searchLabel = new JLabel("Поиск:");
```

```
searchField = new JTextField(20);
searchButton = new JButton("Найти");
addButton = new JButton("Добавить");
editButton = new JButton("Редактировать");
deleteButton = new JButton("Удалить");
searchPanel.add(searchLabel);
searchPanel.add(searchField);
searchPanel.add(searchButton);
searchPanel.add(addButton);
searchPanel.add(editButton);
searchPanel.add(deleteButton);
JPanel topPanel = new JPanel();
topPanel.setLayout(new FlowLayout());
topPanel.add(tableSelector);
topPanel.add(classSelector);
add(topPanel, BorderLayout.NORTH);
add(tablePanel, BorderLayout.CENTER);
add(searchPanel, BorderLayout.SOUTH);
searchButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        handleSearch();
    }
});
addButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        handleAdd();
    }
});
editButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        handleEdit();
    }
});
deleteButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        handleDelete();
    }
});
saveItem.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        handleFileSave();
    }
});

importItem.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        handleImport();
    }
```

```
        });
        saveXmlItem.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JFileChooser fileChooser = new JFileChooser();
                int option = fileChooser.showSaveDialog(SchoolApp.this);
                if (option == JFileChooser.APPROVE_OPTION) {
                    File file = fileChooser.getSelectedFile();
                    try {
                        saveToXML(file);
                    } catch (Exception ex) {
                        JOptionPane.showMessageDialog(SchoolApp.this, "Ошибка при
сохранении XML файла.", "Ошибка", JOptionPane.ERROR_MESSAGE);
                    }
                }
            }
        });
        loadXmlItem.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JFileChooser fileChooser = new JFileChooser();
                int option = fileChooser.showOpenDialog(SchoolApp.this);
                if (option == JFileChooser.APPROVE_OPTION) {
                    File file = fileChooser.getSelectedFile();
                    try {
                        loadFromXML(file);
                    } catch (Exception ex) {
                        JOptionPane.showMessageDialog(SchoolApp.this, "Ошибка при
загрузке XML файла.", "Ошибка", JOptionPane.ERROR_MESSAGE);
                    }
                }
            }
        });

        classSelector.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                filterStudentsByClass((String) classSelector.getSelectedItem());
            }
        });
    }

        private String[][] originalSubjectData = subjectData.clone();
        private String[][] originalClassData = classData.clone();

        private void handleAdd() {
            String selectedTable = (String) tableSelector.getSelectedItem();
            String[] newRow = promptForRowData(selectedTable);
            if (newRow != null) {
                if (selectedTable.equals("Ученики")) {
                    // Добавляем нового ученика
                    allStudentData = addRowToTableData(allStudentData, newRow);
                    originalStudentData = addRowToTableData(originalStudentData,
newRow); // Обновляем оригинальные данные
```

```
                    // Обновляем селектор классов
                    updateClassSelector();
                    studentTable.setModel(new
javax.swing.table.DefaultTableModel(allStudentData, studentColumns));
                } else if (selectedTable.equals("Предметы и преподаватели")) {
                    subjectData = addRowToTableData(subjectData, newRow);
                    originalSubjectData = addRowToTableData(originalSubjectData,
newRow); // Обновляем оригинальные данные
                    subjectTable.setModel(new
javax.swing.table.DefaultTableModel(subjectData, subjectColumns));
                } else if (selectedTable.equals("Преподаватели и классы")) {
                    classData = addRowToTableData(classData, newRow);
                    originalClassData = addRowToTableData(originalClassData, newRow);
// Обновляем оригинальные данные
                    classTable.setModel(new
javax.swing.table.DefaultTableModel(classData, classColumns));
                }
            }
        }


        // Метод для обновления селектора классов
        private void updateClassSelector() {
            ArrayList<String> classList = new ArrayList<>();
            classList.add("Все классы"); // Добавляем опцию "Все классы"
            for (String[] row : allStudentData) {
                String studentClass = row[1]; // Предполагается, что класс находится во
втором столбце
                if (!classList.contains(studentClass)) {
                    classList.add(studentClass); // Добавляем уникальный класс
                }
            }
            // Обновляем модель JComboBox
            classSelector.setModel(new DefaultComboBoxModel<>(classList.toArray(new
String[0])));
        }

        private void filterStudentsByClass(String selectedClass) {
            if (selectedClass.equals("Все классы")) {
                // Если выбраны все классы, отображаем всех учеников
                studentTable.setModel(new
javax.swing.table.DefaultTableModel(allStudentData, studentColumns));
            } else {
                // Фильтруем учеников по выбранному классу
                ArrayList<String[]> filteredData = new ArrayList<>();
                for (String[] row : allStudentData) {
                    if (row[1].equals(selectedClass)) {
                        filteredData.add(row);
                    }
                }
                studentTable.setModel(new
javax.swing.table.DefaultTableModel(filteredData.toArray(new String[0][0]),
studentColumns));
            }
```

```
        }


    private void handleEdit() {
        String selectedTable = (String) tableSelector.getSelectedItem();
        JTable table = getSelectedTable(selectedTable);
        int selectedRow = table.getSelectedRow();
        if (selectedRow != -1) {
            String[] newRow = promptForRowData(selectedTable);
            if (newRow != null) {
                updateTableData(getSelectedTableData(selectedTable), selectedRow,
newRow);
                table.setModel(new
javax.swing.table.DefaultTableModel(getSelectedTableData(selectedTable),
getSelectedTableColumns(selectedTable)));
            }
        } else {
            JOptionPane.showMessageDialog(this, "Пожалуйста, выберите строку для
редактирования.");
        }
    }
    private void handleDelete() {
        String selectedTable = (String) tableSelector.getSelectedItem();
        JTable table = getSelectedTable(selectedTable);
        int selectedRow = table.getSelectedRow();
        if (selectedRow != -1) {
            if (selectedTable.equals("Ученики")) {
                allStudentData = deleteRowFromTableData(allStudentData, selectedRow);
                originalStudentData = deleteRowFromTableData(originalStudentData,
selectedRow); // Обновляем оригинальные данные
                studentTable.setModel(new
javax.swing.table.DefaultTableModel(allStudentData, studentColumns));
            } else if (selectedTable.equals("Предметы и преподаватели")) {
                subjectData = deleteRowFromTableData(subjectData, selectedRow);
                originalSubjectData = deleteRowFromTableData(originalSubjectData,
selectedRow); // Обновляем оригинальные данные
                subjectTable.setModel(new
javax.swing.table.DefaultTableModel(subjectData, subjectColumns));
            } else if (selectedTable.equals("Преподаватели и классы")) {
                classData = deleteRowFromTableData(classData, selectedRow);
                originalClassData = deleteRowFromTableData(originalClassData,
selectedRow); // Обновляем оригинальные данные
                classTable.setModel(new javax.swing.table.DefaultTableModel(classData,
classColumns));
            }
        } else {
            JOptionPane.showMessageDialog(this, "Пожалуйста, выберите строку для
удаления.");
        }
    }

    private String[] promptForRowData(String tableType) {
        String[] columns = getSelectedTableColumns(tableType);
        String[] newRow = new String[columns.length];
```

```
        for (int i = 0; i < columns.length; i++) {
            newRow[i] = JOptionPane.showInputDialog("Введите значение для " +
columns[i]);
            if (newRow[i] == null) {
                return null;
            }
        }
        return newRow;
    }
    private JTable getSelectedTable(String tableType) {
        switch (tableType) {
            case "Ученики":
                return studentTable;
            case "Предметы и преподаватели":
                return subjectTable;
            case "Преподаватели и классы":
                return classTable;
            default:
                return null;
        }
    }
    private String[][] getSelectedTableData(String tableType) {
        switch (tableType) {
            case "Ученики":
                return allStudentData;
            case "Предметы и преподаватели":
                return subjectData;
            case "Преподаватели и классы":
                return classData;
            default:
                return null;
        }
    }
    private String[] getSelectedTableColumns(String tableType) {
        switch (tableType) {
            case "Ученики":
                return studentColumns;
            case "Предметы и преподаватели":
                return subjectColumns;
            case "Преподаватели и классы":
                return classColumns;
            default:
                return null;
        }
    }
    private String[][] addRowToTableData(String[][] data, String[] newRow) {
        String[][] newData = new String[data.length + 1][];
        System.arraycopy(data, 0, newData, 0, data.length);
        newData[data.length] = newRow;
        return newData;
    }
    private void updateTableData(String[][] data, int rowIndex, String[] newRow) {
        System.arraycopy(newRow, 0, data[rowIndex], 0, newRow.length);
    }
```

```
    private String[][] deleteRowFromTableData(String[][] data, int rowIndex) {
        String[][] newData = new String[data.length - 1][];
        System.arraycopy(data, 0, newData, 0, rowIndex);
        System.arraycopy(data, rowIndex + 1, newData, rowIndex, data.length - rowIndex
- 1);
        return newData; // Возвращаем обновленный массив
    }
    private void handleSearch() {
        String query = searchField.getText().toLowerCase();
        String selectedTable = (String) tableSelector.getSelectedItem();

        try {
        if (selectedTable.equals("Ученики")) {
            searchInStudentTable(query);
        } else if (selectedTable.equals("Предметы и преподаватели")) {
            searchInSubjectTable(query);
        } else if (selectedTable.equals("Преподаватели и классы")) {
            searchInClassTable(query);
        }
        }catch (SearchException e) {
            JOptionPane.showMessageDialog(this, e.getMessage(), "Ошибка поиска",
JOptionPane.WARNING_MESSAGE);
        }
    }

    private void searchInStudentTable(String query) throws SearchException{
        ArrayList<String[]> filteredData = new ArrayList<>();
        for (String[] row : originalStudentData) { // Используем оригинальные данные
            for (String cell : row) {
                if (cell.toLowerCase().contains(query)) {
                    filteredData.add(row);
                    break;
                }
            }
        }
        allStudentData = filteredData.toArray(new String[0][0]);
        studentTable.setModel(new javax.swing.table.DefaultTableModel(allStudentData,
studentColumns));

        if (filteredData.isEmpty()) {
            throw new SearchException("Класс с заданным запросом не найден.");
        }
    }
    private void searchInSubjectTable(String query) throws SearchException {
        ArrayList<String[]> filteredData = new ArrayList<>();
        for (String[] row : originalSubjectData) { // Используем оригинальные данные
            for (String cell : row) {
                if (cell.toLowerCase().contains(query)) {
                    filteredData.add(row);
                    break;
                }
            }
        }
```

```
        subjectTable.setModel(new
javax.swing.table.DefaultTableModel(filteredData.toArray(new String[0][0]),
subjectColumns));

        if (filteredData.isEmpty()) {
            throw new SearchException("Класс с заданным запросом не найден.");
        }
    }
    private void searchInClassTable(String query) throws SearchException{
        ArrayList<String[]> filteredData = new ArrayList<>();
        for (String[] row : originalClassData) { // Используем оригинальные данные
            for (String cell : row) {
                if (cell.toLowerCase().contains(query)) {
                    filteredData.add(row);
                    break;
                }
            }
        }
        classTable.setModel(new
javax.swing.table.DefaultTableModel(filteredData.toArray(new String[0][0]),
classColumns));

        if (filteredData.isEmpty()) {
            throw new SearchException("Класс с заданным запросом не найден.");
        }
    }

    private void handleFileSave() {
        JFileChooser fileChooser = new JFileChooser();
        int option = fileChooser.showSaveDialog(SchoolApp.this);
        if (option == JFileChooser.APPROVE_OPTION) {
            File file = fileChooser.getSelectedFile();
            try {
                saveToFile(file);
            } catch (FileSaveException e) {
                JOptionPane.showMessageDialog(this, "Ошибка при сохранении файла.",
"Ошибка сохранения", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
    private void saveToFile(File file) throws FileSaveException {
        int columnWidth = 20;
        try (FileWriter writer = new FileWriter(file)) {
            // СОхранение учеников
            writer.write("Ученики:\n");
            for (String column : studentColumns) {
                writer.write(String.format("%-" + columnWidth + "s", column));
            }
            writer.write("\n");
            for (String[] row : allStudentData) {
                for (String data : row) {
                    writer.write(String.format("%-" + columnWidth + "s", data));
                }
                writer.write("\n");
```

```
            }

            // Сохранение предметов
            writer.write("Предметы и преподаватели:\n");
            for (String column : subjectColumns) {
                writer.write(String.format("%-" + columnWidth + "s", column));
            }
            writer.write("\n");
            for (String[] row : subjectData) {
                for (String data : row) {
                    writer.write(String.format("%-" + columnWidth + "s", data));
                }
                writer.write("\n");
            }

            // Сохранение преподавателей и классов
            writer.write("Преподаватели и классы:\n");
            for (String column : classColumns) {
                writer.write(String.format("%-" + columnWidth + "s", column));
            }
            writer.write("\n");
            for (String[] row : classData) {
                for (String data : row) {
                    writer.write(String.format("%-" + columnWidth + "s", data));
                }
                writer.write("\n");
            }
            JOptionPane.showMessageDialog(this, "Данные успешно сохранены.",
"Сохранение", JOptionPane.INFORMATION_MESSAGE);
        }     catch (IOException ex) {
            throw new FileSaveException("Ошибка при сохранении файла.");
        }
    }

    private void handleImport() {
        JFileChooser fileChooser = new JFileChooser();
        int option = fileChooser.showOpenDialog(SchoolApp.this);
        if (option == JFileChooser.APPROVE_OPTION) {
            File file = fileChooser.getSelectedFile();
            try {
                importFromFile(file);
            } catch (IOException e) {
                JOptionPane.showMessageDialog(this, "Ошибка при загрузке файла.",
"Ошибка загрузки", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
    private void importFromFile(File file) throws IOException {
        List<String[]> students = new ArrayList<>();
        List<String[]> subjects = new ArrayList<>();
        List<String[]> classes = new ArrayList<>();

        try (BufferedReader reader = new BufferedReader(new FileReader(file))) {
            String line;
```

```
            String currentTable = "";
            while ((line = reader.readLine()) != null )
            {
            if(!(line.contains("Преподаватель") || line.contains("Фамилия") ||
line.contains("Класс")))
            {
                if (line.startsWith("Ученики:") )
                {
                    currentTable = "students";
                    continue;
                } else if (line.startsWith("Предметы и преподаватели:") ) {
                    currentTable = "subjects";
                    continue;
                } else if (line.startsWith("Преподаватели и классы:")) {
                    currentTable = "classes";
                    continue;
                }
                // Используем разделитель по пробелам
                String[] data = line.trim().split("\\s+");

                if (currentTable.equals("students")) {
                    students.add(data);
                } else if (currentTable.equals("subjects")) {
                    subjects.add(data);
                } else if (currentTable.equals("classes")) {
                    classes.add(data);
                }
            }
            }
        }
        // Обновляем данные таблиц после импорта
        allStudentData = students.toArray(new String[0][0]);
        subjectData = subjects.toArray(new String[0][0]);
        classData = classes.toArray(new String[0][0]);
        // Также обновляем оригинальные данные
        originalStudentData = allStudentData.clone();
        originalSubjectData = subjectData.clone();
        originalClassData = classData.clone();
        // Обновляем модели таблиц
        studentTable.setModel(new javax.swing.table.DefaultTableModel(allStudentData,
studentColumns));
        subjectTable.setModel(new javax.swing.table.DefaultTableModel(subjectData,
subjectColumns));
        classTable.setModel(new javax.swing.table.DefaultTableModel(classData,
classColumns));
    }

    private void saveToXML(File file) throws Exception {
        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
        Document doc = dBuilder.newDocument();
        Element rootElement = doc.createElement("SchoolData");
        doc.appendChild(rootElement);
        // Сохранение данных учеников
```

```
        Element students = doc.createElement("Students");
        rootElement.appendChild(students);
        for (String[] row : allStudentData) {
            Element student = doc.createElement("Student");
            students.appendChild(student);
            for (int i = 0; i < studentColumns.length; i++) {
                Element field = doc.createElement(studentColumns[i]);
                field.appendChild(doc.createTextNode(row[i]));
                student.appendChild(field);
            }
        }
        // Сохранение данных предметов
        Element subjects = doc.createElement("Subjects");
        rootElement.appendChild(subjects);
        for (String[] row : subjectData) {
            Element subject = doc.createElement("Subject");
            subjects.appendChild(subject);
            for (int i = 0; i < subjectColumns.length; i++) {
                Element field = doc.createElement(subjectColumns[i]);
                field.appendChild(doc.createTextNode(row[i]));
                subject.appendChild(field);
            }
        }
        // Сохранение данных классов
        Element classes = doc.createElement("Classes");
        rootElement.appendChild(classes);
        for (String[] row : classData) {
            Element classElement = doc.createElement("Class");
            classes.appendChild(classElement);
            for (int i = 0; i < classColumns.length; i++) {
                Element field = doc.createElement(classColumns[i]);
                field.appendChild(doc.createTextNode(row[i]));
                classElement.appendChild(field);
            }
        }
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        DOMSource source = new DOMSource(doc);
        StreamResult result = new StreamResult(file);
        transformer.transform(source, result);

        JOptionPane.showMessageDialog(this, "Данные сохранены в XML файл.",
"Сохранение XML", JOptionPane.INFORMATION_MESSAGE);
    }

    private void loadFromXML(File file) throws Exception {
        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
        Document doc = dBuilder.parse(file);
        doc.getDocumentElement().normalize();
        List<String[]> students = new ArrayList<>();
        List<String[]> subjects = new ArrayList<>();
        List<String[]> classes = new ArrayList<>();
```

```
        // Загрузка данных учеников
        NodeList studentList = doc.getElementsByTagName("Student");
        for (int i = 0; i < studentList.getLength(); i++) {
            Node node = studentList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;
                String[] row = new String[studentColumns.length];
                for (int j = 0; j < studentColumns.length; j++) {
                    row[j] =
element.getElementsByTagName(studentColumns[j]).item(0).getTextContent();
                }
                students.add(row);
            }
        }
        // Загрузка данных предметов
        NodeList subjectList = doc.getElementsByTagName("Subject");
        for (int i = 0; i < subjectList.getLength(); i++) {
            Node node = subjectList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;
                String[] row = new String[subjectColumns.length];
                for (int j = 0; j < subjectColumns.length; j++) {
                    row[j] =
element.getElementsByTagName(subjectColumns[j]).item(0).getTextContent();
                }
                subjects.add(row);
            }
        }
        // Загрузка данных классов
        NodeList classList = doc.getElementsByTagName("Class");
        for (int i = 0; i < classList.getLength(); i++) {
            Node node = classList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;
                String[] row = new String[classColumns.length];
                for (int j = 0; j < classColumns.length; j++) {
                    row[j] =
element.getElementsByTagName(classColumns[j]).item(0).getTextContent();
                }
                classes.add(row);
            }
        }
        // Обновление данных таблиц
        allStudentData = students.toArray(new String[0][0]);
        subjectData = subjects.toArray(new String[0][0]);
        classData = classes.toArray(new String[0][0]);
        studentTable.setModel(new javax.swing.table.DefaultTableModel(allStudentData,
studentColumns));
        subjectTable.setModel(new javax.swing.table.DefaultTableModel(subjectData,
subjectColumns));
        classTable.setModel(new javax.swing.table.DefaultTableModel(classData,
classColumns));
        JOptionPane.showMessageDialog(this, "Данные загружены из XML файла.",
"Загрузка XML", JOptionPane.INFORMATION_MESSAGE);
```

```
    }

    public static void main(String[] args) {
        SchoolApp app = new SchoolApp();
        app.setVisible(true);
    }
}
```

https://github.com/lovushker/JavaLabs/tree/main/lab6