

**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 № 8  
«Организация многопоточных приложений»  
по дисциплине «Объектно - ориентированное программирование на  
языке Java»**

Выполнил: Барченков П. А.

Факультет: КТИ

Группа: №3312

Подпись преподавателя: \_\_\_\_\_

Санкт-Петербург

2024

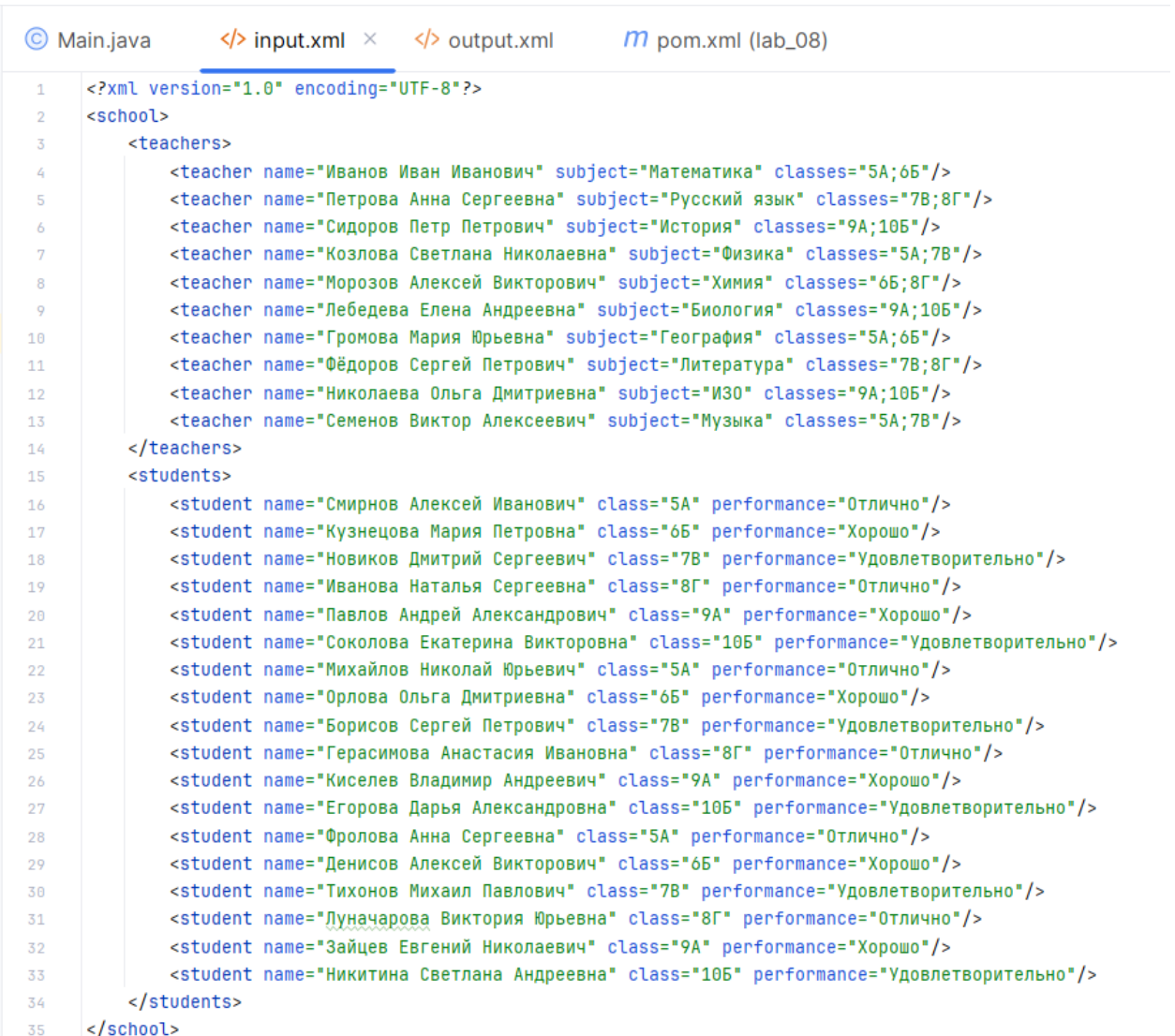
## Содержание

Цель работы .....	3
Распечатки XML-файлов до загрузки данных в экранную форму и после их выгрузки .....	3
Многопоточность .....	4
Распечатка сгенерированных отчетов.....	7
Текст программы.....	8
Приложение .....	22

## Цель работы

Знакомство с правилами и классами построения параллельных приложений в языке программирования Java.

## Распечатки XML-файлов до загрузки данных в экранную форму и после их выгрузки



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <school>
3   <teachers>
4     <teacher name="Иванов Иван Иванович" subject="Математика" classes="5А;6Б"/>
5     <teacher name="Петрова Анна Сергеевна" subject="Русский язык" classes="7В;8Г"/>
6     <teacher name="Сидоров Петр Петрович" subject="История" classes="9А;10Б"/>
7     <teacher name="Козлова Светлана Николаевна" subject="Физика" classes="5А;7В"/>
8     <teacher name="Морозов Алексей Викторович" subject="Химия" classes="6Б;8Г"/>
9     <teacher name="Лебедева Елена Андреевна" subject="Биология" classes="9А;10Б"/>
10    <teacher name="Громова Мария Юрьевна" subject="География" classes="5А;6Б"/>
11    <teacher name="Фёдоров Сергей Петрович" subject="Литература" classes="7В;8Г"/>
12    <teacher name="Николаева Ольга Дмитриевна" subject="ИЗО" classes="9А;10Б"/>
13    <teacher name="Семенов Виктор Алексеевич" subject="Музыка" classes="5А;7В"/>
14  </teachers>
15  <students>
16    <student name="Смирнов Алексей Иванович" class="5А" performance="Отлично"/>
17    <student name="Кузнецова Мария Петровна" class="6Б" performance="Хорошо"/>
18    <student name="Новиков Дмитрий Сергеевич" class="7В" performance="Удовлетворительно"/>
19    <student name="Иванова Наталья Сергеевна" class="8Г" performance="Отлично"/>
20    <student name="Павлов Андрей Александрович" class="9А" performance="Хорошо"/>
21    <student name="Соколова Екатерина Викторовна" class="10Б" performance="Удовлетворительно"/>
22    <student name="Михайлов Николай Юрьевич" class="5А" performance="Отлично"/>
23    <student name="Орлова Ольга Дмитриевна" class="6Б" performance="Хорошо"/>
24    <student name="Борисов Сергей Петрович" class="7В" performance="Удовлетворительно"/>
25    <student name="Герасимова Анастасия Ивановна" class="8Г" performance="Отлично"/>
26    <student name="Киселев Владимир Андреевич" class="9А" performance="Хорошо"/>
27    <student name="Егорова Дарья Александровна" class="10Б" performance="Удовлетворительно"/>
28    <student name="Фролова Анна Сергеевна" class="5А" performance="Отлично"/>
29    <student name="Денисов Алексей Викторович" class="6Б" performance="Хорошо"/>
30    <student name="Тихонов Михаил Павлович" class="7В" performance="Удовлетворительно"/>
31    <student name="Луначарова Виктория Юрьевна" class="8Г" performance="Отлично"/>
32    <student name="Зайцев Евгений Николаевич" class="9А" performance="Хорошо"/>
33    <student name="Никитина Светлана Андреевна" class="10Б" performance="Удовлетворительно"/>
34  </students>
35 </school>
```

Рисунок 1 – Содержимое исходного XML-файла.

```

© Main.java  </> output.xml  ×  © JOptionPane.java
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <school>
3      <teachers>
4          <teacher classes="5A;6Б" name="Иванов Иван Иванович" subject="Математика"/>
5          <teacher classes="7B;8Г" name="Петрова Анна Сергеевна" subject="Русский язык"/>
6          <teacher classes="9A;10Б" name="Сидоров Петр Петрович" subject="История"/>
7          <teacher classes="5A;7B" name="Козлова Светлана Николаевна" subject="Физика"/>
8          <teacher classes="6Б;8Г" name="Морозов Алексей Викторович" subject="Химия"/>
9          <teacher classes="9A;10Б" name="Лебедева Елена Андреевна" subject="Биология"/>
10     </teachers>
11     <students>
12         <student class="5A" name="Смирнов Алексей Иванович" performance="Отлично"/>
13         <student class="6Б" name="Кузнецова Мария Петровна" performance="Хорошо"/>
14         <student class="7B" name="Новиков Дмитрий Сергеевич" performance="Удовлетворительно"/>
15         <student class="8Г" name="Иванова Наталья Сергеевна" performance="Отлично"/>
16         <student class="9A" name="Павлов Андрей Александрович" performance="Хорошо"/>
17         <student class="10Б" name="Соколова Екатерина Викторовна" performance="Удовлетворительно"/>
18         <student class="5A" name="Михайлов Николай Юрьевич" performance="Отлично"/>
19         <student class="6Б" name="Орлова Ольга Дмитриевна" performance="Хорошо"/>
20     </students>
21 </school>

```

Рисунок 2 – Содержимое XML-файла с данными после изменений.

## Многопоточность

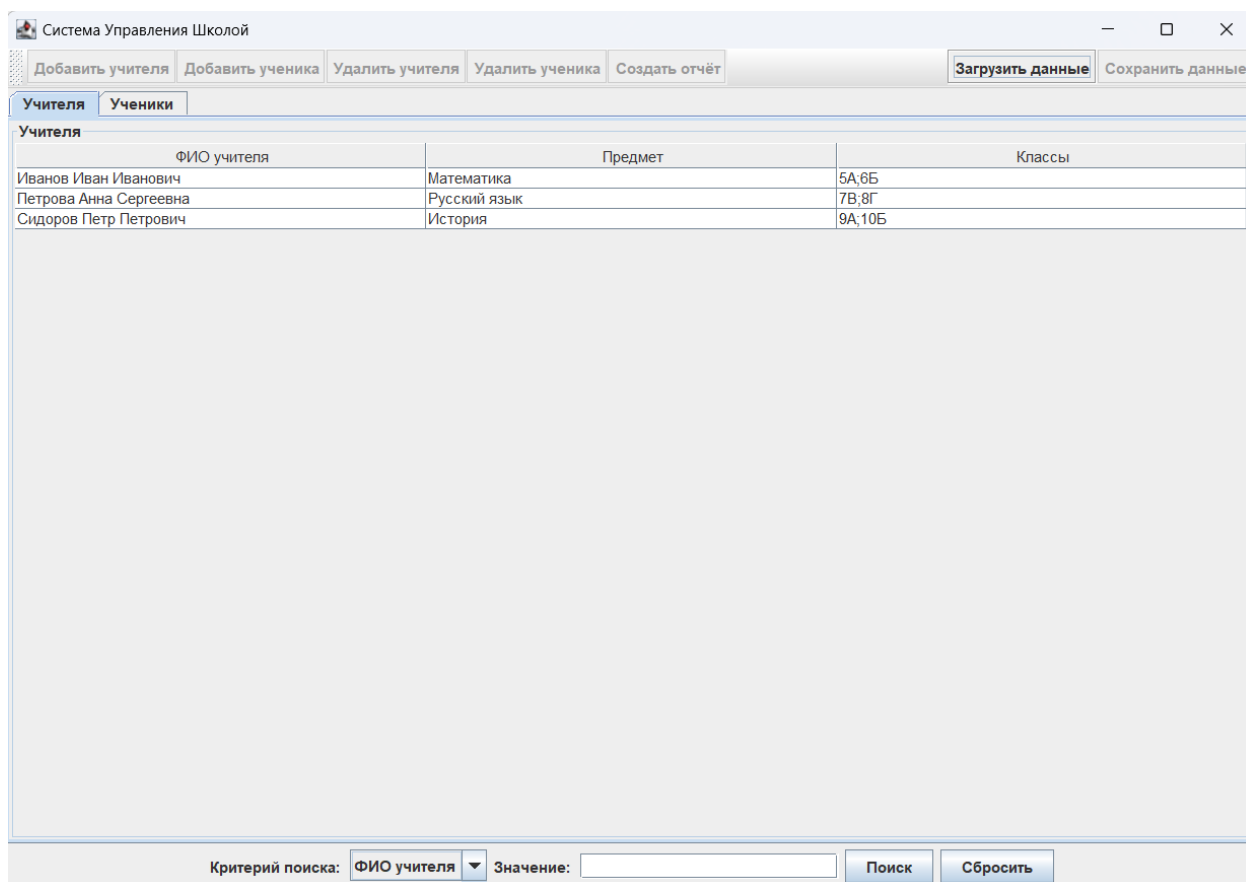


Рисунок 3 – Недоступность редактирования до загрузки данных.

Система Управления Школой

Добавить учителя   
 Добавить ученика   
 Удалить учителя   
 Удалить ученика   
 Создать отчёт   
 Загрузить данные   
 Сохранить данные

Учителя    Ученики

Учителя

ФИО учителя	Предмет	Классы
Иванов Иван Иванович	Математика	5А, 6Б
Петрова Анна Сергеевна	Русский язык	7В, 8Г
Сидоров Петр Петрович	История	9А, 10Б
Козлова Светлана Николаевна	Физика	5А, 7В
Морозов Алексей Викторович	Химия	6Б, 8Г
Лебедева Елена Андреевна	Биология	9А, 10Б
Громова Мария Юрьевна	География	5А, 6Б
Федоров Сергей Петрович	Литература	7В, 8Г
Николаева Ольга Дмитриевна	ИЗО	9А, 10Б
Семенов Виктор Алексеевич	Музыка	5А, 7В

Критерий поиска: ФИО учителя    Значение:    Поиск    Сбросить

Рисунок 4 – Недоступность создания отчета до того, как данные будут сохранены.

Система Управления Школой

Добавить учителя | Добавить ученика | Удалить учителя | Удалить ученика | Создать отчёт | Загрузить данные | Сохранить данные

Учителя | Ученики

ФИО учителя	Предмет	Классы
Иванов Иван Иванович	Математика	5А, 6Б
Петрова Анна Сергеевна	Русский язык	7В, 8Г
Сидоров Петр Петрович	История	9А, 10Б
Козлова Светлана Николаевна	Физика	5А, 7В
Морозов Алексей Викторович	Химия	6Б, 8Г
Лебедева Елена Андреевна	Биология	9А, 10Б
Громова Мария Юрьевна	География	5А, 6Б

Критерий поиска: ФИО учителя ▼ Значение:  Поиск Сбросить

Рисунок 5 – Доступность кнопки создания отчета.

## Отчет об успеваемости в школе

17.11.2024, 17:54

ФИО	Класс	Успеваемость
Смирнов Алексей Иванович	5А	Отлично
Орлова Ольга Дмитриевна	6Б	Хорошо
Борисов Сергей Петрович	7В	Удовлетворительно
Герасимова Анастасия Ивановна	8Г	Отлично
Киселев Владимир Андреевич	9А	Хорошо
Егорова Дарья Александровна	10Б	Удовлетворительно
Фролова Анна Сергеевна	5А	Отлично
Денисов Алексей Викторович	6Б	Хорошо
Тихонов Михаил Павлович	7В	Удовлетворительно
Луначарова Виктория Юрьевна	8Г	Отлично
Зайцев Евгений Николаевич	9А	Хорошо
Никитина Светлана Андреевна	10Б	Удовлетворительно

## Текст программы

```
package org.example;

import javax.swing.*.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;
import javax.swing.RowFilter;
import javax.swing.event.ChangeListener;
import javax.swing.event.ChangeEvent;
import java.awt.*.*;
import java.awt.event.*.*;
import java.io.*.*;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CountDownLatch;

// Импорты для работы с XML
import org.w3c.dom.*.*;
import javax.xml.parsers.*.*;
import org.xml.sax.SAXException;
import javax.xml.transform.*.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

// Импорты для JasperReports
import net.sf.jasperreports.engine.*.*;
import net.sf.jasperreports.engine.data.JRXmlDataSource;
import net.sf.jasperreports.view.JasperViewer;
import java.util.HashMap;

/**
 * Программа для управления данными учителей и учеников в системе управления
 * школой.
 * Содержит функции добавления, удаления учителей и учеников, а также поиска,
 * фильтрации,
 * загрузки и сохранения данных из/в XML-файлы и генерации отчётов.
 *
 * @version 1.0
 */
public class Main {
    private JFrame frame;
    private JTable teacherTable, studentTable;
    private DefaultTableModel teacherTableModel, studentTableModel;
    private JPanel filterPanel;
    private JButton addTeacherButton, addStudentButton, deleteTeacherButton,
deleteStudentButton, generateReportButton;
    private JButton searchButton, resetButton, loadButton, saveButton;
    private JComboBox<String> searchCriteria;
    private JTextField searchField;
    private JScrollPane teacherScrollPane, studentScrollPane;
    private JTabbedPane tabbedPane;
    private List<String[]> originalTeacherData; // Исходные данные учителей
    private List<String[]> originalStudentData; // Исходные данные учеников
    private TableRowSorter<DefaultTableModel> teacherSorter, studentSorter;

    // CountDownLatch для синхронизации потоков
    private final CountDownLatch loadLatch = new CountDownLatch(1);
    private final CountDownLatch saveLatch = new CountDownLatch(1);

    /**
     * Метод для создания и отображения основного окна программы.
     */
    public void SchoolManagementSystem() {
        // Инициализация исходных данных
        originalTeacherData = new ArrayList<>();
        originalStudentData = new ArrayList<>();

        // Создание главного окна программы
        frame = new JFrame("Система Управления Школой");
        frame.setSize(1000, 700); // Увеличиваем размер окна для двух таблиц
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Закрытие
        // окна завершает программу
        frame.setLayout(new BorderLayout()); // Устанавливаем BorderLayout
        // для главного окна

        // Создание панели инструментов с кнопками действий
        JToolBar actionPanel = new JToolBar("Панель инструментов");
```



```

// Кнопки для учителей и учеников
addTeacherButton = new JButton("Добавить учителя");
addStudentButton = new JButton("Добавить ученика");
deleteTeacherButton = new JButton("Удалить учителя");
deleteStudentButton = new JButton("Удалить ученика");
generateReportButton = new JButton("Создать отчёт");

// Кнопки загрузки и сохранения данных
loadButton = new JButton("Загрузить данные");
saveButton = new JButton("Сохранить данные");

// Добавляем кнопки на панель инструментов слева
actionPanel.add(addTeacherButton);
actionPanel.add(addStudentButton);
actionPanel.add(deleteTeacherButton);
actionPanel.add(deleteStudentButton);
actionPanel.add(generateReportButton);

// Добавляем гибкое пространство, чтобы следующие кнопки были справа
actionPanel.add(Box.createHorizontalGlue());

// Добавляем кнопки загрузки и сохранения данных справа
actionPanel.add(loadButton);
actionPanel.add(saveButton);

frame.add(actionPanel, BorderLayout.NORTH); // Размещаем панель
инструментов сверху

// Определяем столбцы таблицы учителей
String[] teacherColumns = {"ФИО учителя", "Предмет", "Классы"};
// Исходные данные для таблицы учителей
String[][] initialTeachers = {
    {"Иванов Иван Иванович", "Математика", "5А;6Б"},
    {"Петрова Анна Сергеевна", "Русский язык", "7В;8Г"},
    {"Сидоров Петр Петрович", "История", "9А;10Б"}
};
for (String[] teacher : initialTeachers) {
    originalTeacherData.add(teacher);
}

// Инициализация модели таблицы учителей
teacherTableModel = new DefaultTableModel(teacherColumns, 0);
for (String[] teacher : originalTeacherData) {
    teacherTableModel.addRow(teacher);
}
teacherTable = new JTable(teacherTableModel);

teacherTable.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
// Разрешаем множественный выбор
teacherScrollPane = new JScrollPane(teacherTable);

teacherScrollPane.setBorder(BorderFactory.createTitledBorder("Учителя"));

// Создание сортировщика для таблицы учителей
teacherSorter = new TableRowSorter<>(teacherTableModel);
teacherTable.setRowSorter(teacherSorter);

// Определяем столбцы таблицы учеников
String[] studentColumns = {"ФИО ученика", "Класс", "Успеваемость"};
// Исходные данные для таблицы учеников
String[][] initialStudents = {
    {"Смирнов Алексей Иванович", "5А", "Отлично"},
    {"Кузнецова Мария Петровна", "6Б", "Хорошо"},
    {"Новиков Дмитрий Сергеевич", "7В", "Удовлетворительно"}
};
for (String[] student : initialStudents) {
    originalStudentData.add(student);
}

// Инициализация модели таблицы учеников
studentTableModel = new DefaultTableModel(studentColumns, 0);
for (String[] student : originalStudentData) {
    studentTableModel.addRow(student);
}
studentTable = new JTable(studentTableModel);

studentTable.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION)

```

```

; // Разрешаем множественный выбор
    studentScrollPane = new JScrollPane(studentTable);

studentScrollPane.setBorder(BorderFactory.createTitledBorder("Ученики"));

// Создание сортировщика для таблицы учеников
studentSorter = new TableRowSorter<>(studentTableModel);
studentTable.setRowSorter(studentSorter);

// Создание вкладок для таблиц
tabbedPane = new JTabbedPane();
tabbedPane.addTab("Учителя", teacherScrollPane);
tabbedPane.addTab("Ученики", studentScrollPane);
frame.add(tabbedPane, BorderLayout.CENTER); // Размещаем вкладки в
центре

// Создание компонентов для панели поиска и фильтрации данных
searchCriteria = new JComboBox<>(new String[]{
    "ФИО учителя", "Предмет", "Классы",
    "ФИО ученика", "Класс ученика", "Успеваемость"
});
searchField = new JTextField(20);
searchButton = new JButton("Поиск");
resetButton = new JButton("Сбросить");

// Панель фильтрации
filterPanel = new JPanel();
filterPanel.add(new JLabel("Критерий поиска: "));
filterPanel.add(searchCriteria);
filterPanel.add(new JLabel("Значение: "));
filterPanel.add(searchField);
filterPanel.add(searchButton);
filterPanel.add(resetButton);
frame.add(filterPanel, BorderLayout.SOUTH); // Размещаем панель
фильтрации снизу

// Действие при переключении вкладок для обновления критериев поиска
tabbedPane.addChangeListener(new ChangeListener() {
    @Override
    public void stateChanged(ChangeEvent e) {
        updateSearchCriteria();
    }
});

// Инициализация критериев поиска по текущей вкладке
updateSearchCriteria();

// Добавление слушателей к кнопкам
addListeners();

// Установка начального состояния кнопок
setInitialButtonStates();

// Делаем главное окно видимым
frame.setVisible(true);
}

/**
 * Метод для установки начального состояния кнопок.
 * Только кнопка "Загрузить данные" активна.
 */
private void setInitialButtonStates() {
    addTeacherButton.setEnabled(false);
    addStudentButton.setEnabled(false);
    deleteTeacherButton.setEnabled(false);
    deleteStudentButton.setEnabled(false);
    saveButton.setEnabled(false);
    generateReportButton.setEnabled(false);
}

/**
 * Метод для добавления слушателей к кнопкам
 */
private void addListeners() {
    // Слушатель для кнопки "Поиск"
    searchButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {

```

```

        String criterion = (String) searchCriteria.getSelectedItemAt();
        String value = searchField.getText().trim();
        searchTable(criterion, value);
    }
});

// Слушатель для кнопки "Сбросить"
resetButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        resetTable();
    }
});

// Слушатель для кнопки "Добавить учителя"
addTeacherButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        addTeacherDialog();
    }
});

// Слушатель для кнопки "Удалить учителя"
deleteTeacherButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        deleteSelectedTeachers();
    }
});

// Слушатель для кнопки "Добавить ученика"
addStudentButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        addStudentDialog();
    }
});

// Слушатель для кнопки "Удалить ученика"
deleteStudentButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        deleteSelectedStudents();
    }
});

// Слушатель для кнопки "Загрузить данные"
loadButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Отключаем все кнопки кроме загрузки
        setButtonStatesDuringLoad(true);
        // Запускаем поток загрузки данных
        Thread loadThread = new Thread(new LoadDataThread(Main.this,
loadLatch), "LoadDataThread");
        loadThread.start();
    }
});

// Слушатель для кнопки "Сохранить данные"
saveButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Отключаем кнопку создания отчёта до завершения сохранения
        generateReportButton.setEnabled(false);
        // Запускаем поток сохранения данных
        Thread saveThread = new Thread(new SaveDataThread(Main.this,
loadLatch, saveLatch), "SaveDataThread");
        saveThread.start();
    }
});

// Слушатель для кнопки "Создать отчёт"
generateReportButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Запускаем поток генерации отчёта
        Thread reportThread = new Thread(new
GenerateReportThread(Main.this, saveLatch), "GenerateReportThread");
        reportThread.start();
    }
});

}

/**

```

```

    * Метод для управления состоянием кнопок во время загрузки данных.
    *
    * @param isLoading true, если данные загружаются; false иначе.
    */
private void setButtonStatesDuringLoad(boolean isLoading) {
    addTeacherButton.setEnabled(!isLoading && loadLatch.getCount() == 0);
    addStudentButton.setEnabled(!isLoading && loadLatch.getCount() == 0);
    deleteTeacherButton.setEnabled(!isLoading && loadLatch.getCount() ==
0);
    deleteStudentButton.setEnabled(!isLoading && loadLatch.getCount() ==
0);
    saveButton.setEnabled(!isLoading && loadLatch.getCount() == 0);
    generateReportButton.setEnabled(false);
    loadButton.setEnabled(!isLoading); // Разрешаем повторную загрузку
}

/**
 * Метод для управления состоянием кнопок после загрузки данных.
 *
 * @param isLoading true, если данные успешно загружены; false иначе.
 */
private void setButtonStatesAfterLoad(boolean isLoading) {
    if (isLoading) {
        addTeacherButton.setEnabled(true);
        addStudentButton.setEnabled(true);
        deleteTeacherButton.setEnabled(true);
        deleteStudentButton.setEnabled(true);
        saveButton.setEnabled(true);
    } else {
        addTeacherButton.setEnabled(false);
        addStudentButton.setEnabled(false);
        deleteTeacherButton.setEnabled(false);
        deleteStudentButton.setEnabled(false);
        saveButton.setEnabled(false);
    }
    // Кнопка создания отчёта остаётся отключённой до сохранения
    generateReportButton.setEnabled(false);
}

/**
 * Обновляет критерии поиска в зависимости от выбранной вкладки.
 */
private void updateSearchCriteria() {
    int selectedIndex = tabbedPane.getSelectedIndex();
    searchCriteria.removeAllItems();

    if (selectedIndex == 0) { // Учителя
        searchCriteria.addItem("ФИО учителя");
        searchCriteria.addItem("Предмет");
        searchCriteria.addItem("Классы");
    } else if (selectedIndex == 1) { // Ученики
        searchCriteria.addItem("ФИО ученика");
        searchCriteria.addItem("Класс ученика");
        searchCriteria.addItem("Успеваемость");
    }
}

/**
 * Метод для фильтрации данных в таблице на основе критерия и значения
 * поиска.
 *
 * @param criterion Критерий поиска.
 * @param value Значение для поиска.
 */
private void searchTable(String criterion, String value) {
    if (value.isEmpty()) {
        JOptionPane.showMessageDialog(frame, "Поле поиска не может быть
пустым.", "Ошибка", JOptionPane.ERROR_MESSAGE);
        return;
    }

    int selectedIndex = tabbedPane.getSelectedIndex();

    if (selectedIndex == 0) { // Учителя
        int columnIndex = -1;
        switch (criterion) {
            case "ФИО учителя":
                columnIndex = 0;

```

```

        break;
    case "Предмет":
        columnIndex = 1;
        break;
    case "Классы":
        columnIndex = 2;
        break;
    }

    if (columnIndex != -1) {
        teacherSorter.setRowFilter(RowFilter.regexFilter("(?i)" +
value, columnIndex));
    }
} else if (selectedIndex == 1) { // Ученики
    int columnIndex = -1;
    switch (criterion) {
        case "ФИО ученика":
            columnIndex = 0;
            break;
        case "Класс ученика":
            columnIndex = 1;
            break;
        case "Успеваемость":
            columnIndex = 2;
            break;
    }

    if (columnIndex != -1) {
        studentSorter.setRowFilter(RowFilter.regexFilter("(?i)" +
value, columnIndex));
    }
}

}

/**
 * Метод для сброса фильтров и восстановления исходных данных.
 */
private void resetTable() {
    // Сброс фильтра для учителей
    teacherSorter.setRowFilter(null);
    // Сброс фильтра для учеников
    studentSorter.setRowFilter(null);
    // Очистка поля поиска
    searchField.setText("");
}

/**
 * Метод для загрузки данных из файла, выбранного пользователем.
 *
 * @throws DataLoadException Если возникает ошибка при загрузке данных.
 */
private void loadDataFromFile() throws DataLoadException {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Выберите XML-файл для загрузки данных");
    int userSelection = fileChooser.showOpenDialog(frame);

    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File xmlFile = fileChooser.getSelectedFile();
        try {
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document doc = builder.parse(xmlFile);

            // Нормализация документа
            doc.getDocumentElement().normalize();

            // Очистка текущих данных в таблицах и исходных списках
            SwingUtilities.invokeLater(() -> {
                teacherTableModel.setRowCount(0);
                studentTableModel.setRowCount(0);
                originalTeacherData.clear();
                originalStudentData.clear();
            });

            // Загрузка учителей
            NodeList teacherList = doc.getElementsByTagName("teacher");
            for (int i = 0; i < teacherList.getLength(); i++) {

```

```

        Element teacherElement = (Element) teacherList.item(i);
        String name = teacherElement.getAttribute("name");
        String subject = teacherElement.getAttribute("subject");
        String classes = teacherElement.getAttribute("classes");

        String[] teacher = {name, subject, classes};
        SwingUtilities.invokeLater(() -> {
            teacherTableModel.addRow(teacher);
            originalTeacherData.add(teacher);
        });
    }

    // Загрузка учеников
    NodeList studentList = doc.getElementsByTagName("student");
    for (int i = 0; i < studentList.getLength(); i++) {
        Element studentElement = (Element) studentList.item(i);
        String name = studentElement.getAttribute("name");
        String studentClass =
studentElement.getAttribute("class");
        String performance =
studentElement.getAttribute("performance");

        String[] student = {name, studentClass, performance};
        SwingUtilities.invokeLater(() -> {
            studentTableModel.addRow(student);
            originalStudentData.add(student);
        });
    }

    SwingUtilities.invokeLater(() -> {
        JOptionPane.showMessageDialog(frame, "Данные успешно
загружены из XML-файла.", "Успех", JOptionPane.INFORMATION_MESSAGE);
        // Включаем остальные кнопки после загрузки
        setButtonStatesAfterLoad(true);
    });
} catch (ParserConfigurationException | SAXException |
IOException e) {
    throw new DataLoadException("Ошибка при загрузке данных: " +
e.getMessage());
}
}

/**
 * Метод для сохранения данных в файл, выбранный пользователем.
 *
 * @throws DataSaveException Если возникает ошибка при сохранении данных.
 */
private void saveDataToFile() throws DataSaveException {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Сохраните данные в XML-файл");
    int userSelection = fileChooser.showSaveDialog(frame);

    if (userSelection == JFileChooser.APPROVE_OPTION) {
        File xmlFile = fileChooser.getSelectedFile();
        try {
            // Создание фабрики и строителя документов
            DocumentBuilderFactory docFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder docBuilder = docFactory.newDocumentBuilder();

            // Создание нового документа
            Document doc = docBuilder.newDocument();

            // Создание корневого элемента <school>
            Element rootElement = doc.createElement("school");
            doc.appendChild(rootElement);

            // Создание элемента <teachers>
            Element teachersElement = doc.createElement("teachers");
            rootElement.appendChild(teachersElement);

            // Добавление каждого учителя как элемента <teacher>
            for (String[] teacher : originalTeacherData) {
                Element teacherElement = doc.createElement("teacher");
                teacherElement.setAttribute("name", teacher[0]);
                teacherElement.setAttribute("subject", teacher[1]);
            }
        } catch (Exception e) {
            throw new DataSaveException("Ошибка при сохранении данных: " +
e.getMessage());
        }
    }
}

```

```

        teacherElement.setAttribute("classes", teacher[2]);
        teachersElement.appendChild(teacherElement);
    }

    // Создание элемента <students>
    Element studentsElement = doc.createElement("students");
    rootElement.appendChild(studentsElement);

    // Добавление каждого ученика как элемента <student>
    for (String[] student : originalStudentData) {
        Element studentElement = doc.createElement("student");
        studentElement.setAttribute("name", student[0]);
        studentElement.setAttribute("class", student[1]);
        studentElement.setAttribute("performance", student[2]);
        studentsElement.appendChild(studentElement);
    }

    // Создание преобразователя и запись документа в файл
    TransformerFactory transformerFactory =
TransformerFactory.newInstance();
    Transformer transformer =
transformerFactory.newTransformer();
    // Для красивого форматирования XML
    transformer.setOutputProperty(OutputKeys.INDENT, "yes");

    transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount",
"4");

    DOMSource source = new DOMSource(doc);

    StreamResult result = new StreamResult(xmlFile);
    transformer.transform(source, result);

    SwingUtilities.invokeLater(() -> {
        JOptionPane.showMessageDialog(frame, "Данные успешно
сохранены в XML-файл.", "Успех", JOptionPane.INFORMATION_MESSAGE);
        // Включаем кнопку создания отчёта после сохранения
        generateReportButton.setEnabled(true);
    });

    } catch (ParserConfigurationException | TransformerException e) {
        throw new DataSaveException("Ошибка при сохранении данных: "
+ e.getMessage());
    }
}

/**
 * Метод для отображения диалогового окна добавления учителя
 */
private void addTeacherDialog() {
    JPanel panel = new JPanel(new GridLayout(3, 2));
    JTextField nameField = new JTextField(20);
    JTextField subjectField = new JTextField(20);
    JTextField classesField = new JTextField(20);

    panel.add(new JLabel("ФИО учителя: "));
    panel.add(nameField);
    panel.add(new JLabel("Предмет: "));
    panel.add(subjectField);
    panel.add(new JLabel("Классы: "));
    panel.add(classesField);

    int result = JOptionPane.showConfirmDialog(frame, panel, "Добавить
учителя", JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);

    if (result == JOptionPane.OK_OPTION) {
        try {
            String name = nameField.getText();
            String subject = subjectField.getText();
            String classes = classesField.getText();

            validateFields(name, subject, classes);

            // Добавление в таблицу и исходные данные
            String[] newTeacher = {name, subject, classes};
            teacherTableModel.addRow(newTeacher);
            originalTeacherData.add(newTeacher);
        }
    }
}

```



```

        JOptionPane.showMessageDialog(frame, "Учитель добавлен!",
"Добавление", JOptionPane.INFORMATION_MESSAGE);
    } catch (InvalidInputException ex) {
        JOptionPane.showMessageDialog(frame, ex.getMessage(),
"Ошибка", JOptionPane.ERROR_MESSAGE);
    }
}

/**
 * Метод для отображения диалогового окна добавления ученика
 */
private void addStudentDialog() {
    JPanel panel = new JPanel(new GridLayout(3, 2));
    JTextField nameField = new JTextField(20);
    JTextField classField = new JTextField(20);
    JTextField performanceField = new JTextField(20);

    panel.add(new JLabel("ФИО ученика: "));
    panel.add(nameField);
    panel.add(new JLabel("Класс: "));
    panel.add(classField);
    panel.add(new JLabel("Успеваемость: "));
    panel.add(performanceField);

    int result = JOptionPane.showConfirmDialog(frame, panel, "Добавить
ученика", JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);

    if (result == JOptionPane.OK_OPTION) {
        try {
            String name = nameField.getText();
            String className = classField.getText();
            String performance = performanceField.getText();

            validateFields(name, className, performance);

            // Добавление в таблицу и исходные данные
            String[] newStudent = {name, className, performance};
            studentTableModel.addRow(newStudent);
            originalStudentData.add(newStudent);

            JOptionPane.showMessageDialog(frame, "Ученик добавлен!",
"Добавление", JOptionPane.INFORMATION_MESSAGE);
        } catch (InvalidInputException ex) {
            JOptionPane.showMessageDialog(frame, ex.getMessage(),
"Ошибка", JOptionPane.ERROR_MESSAGE);
        }
    }
}

/**
 * Метод для проверки корректности введенных данных
 *
 * @param name ФИО учителя или ученика
 * @param field1 Предмет учителя или класс ученика
 * @param field2 Классы учителя или успеваемость ученика
 * @throws InvalidInputException если хотя бы одно из полей пустое
 */
private void validateFields(String name, String field1, String field2)
throws InvalidInputException {
    if (name.isEmpty() || field1.isEmpty() || field2.isEmpty()) {
        throw new InvalidInputException("Все поля должны быть
заполнены!");
    }
    // Дополнительная валидация может быть добавлена здесь, например,
    проверка формата классов
}

/**
 * Метод для генерации отчёта.
 */
private void generateReport() {
    try {
        // Путь к шаблону отчёта
        String reportPath = "lab 08.jrxml"; // Убедитесь, что путь
правильный и файл существует
        File reportFile = new File(reportPath);
        if (!reportFile.exists()) {

```



```

        JOptionPane.showMessageDialog(frame, "Файл шаблона отчёта не
найден: " + reportPath, "Ошибка", JOptionPane.ERROR MESSAGE);
        return;
    }

    // Компиляция шаблона отчёта
    JasperReport jasperReport =
JasperCompileManager.compileReport(reportPath);

    // Создание XML-файла с данными для отчёта
    String xmlDataPath = createDataXML();

    if (xmlDataPath == null) {
        JOptionPane.showMessageDialog(frame, "Не удалось создать XML-
файл с данными для отчёта.", "Ошибка", JOptionPane.ERROR MESSAGE);
        return;
    }

    // Создание источника данных из XML-файла
    JRXmlDataSource xmlDataSource = new JRXmlDataSource(xmlDataPath,
"/school/students/student");

    // Заполнение отчёта данными
    JasperPrint jasperPrint =
JasperFillManager.fillReport(jasperReport, new HashMap<>(), xmlDataSource);

    // Настройка диалога для выбора формата и места сохранения отчёта
    String[] options = {"PDF", "HTML"};
    int choice = JOptionPane.showOptionDialog(
        frame,
        "Выберите формат отчёта для сохранения:",
        "Выбор формата отчёта",
        JOptionPane.DEFAULT OPTION,
        JOptionPane.INFORMATION MESSAGE,
        null,
        options,
        options[0]
    );

    if (choice == JOptionPane.CLOSED OPTION) {
        return; // Пользователь закрыл диалог без выбора
    }

    // Настройка JFileChooser для выбора места сохранения
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setDialogTitle("Сохранить отчёт");
    fileChooser.setFileFilter(new
javax.swing.filechooser.FileNameExtensionFilter(options[choice] + " файлы",
options[choice].toLowerCase()));
    fileChooser.setCurrentDirectory(new
File(System.getProperty("user.home")));

    int userSelection = fileChooser.showSaveDialog(frame);

    if (userSelection == JFileChooser.APPROVE OPTION) {
        File saveFile = fileChooser.getSelectedFile();
        String filePath = saveFile.getAbsolutePath();

        // Добавляем расширение, если оно отсутствует
        if (!filePath.toLowerCase().endsWith(".") +
options[choice].toLowerCase()) {
            filePath += "." + options[choice].toLowerCase();
        }

        if (choice == 0) { // PDF
            JasperExportManager.exportReportToPdfFile(jasperPrint,
filePath);
        } else if (choice == 1) { // HTML
            JasperExportManager.exportReportToHtmlFile(jasperPrint,
filePath);
        }

        JOptionPane.showMessageDialog(frame, "Отчёт успешно сохранён:
" + filePath, "Успех", JOptionPane.INFORMATION MESSAGE);
        JasperViewer.viewReport(jasperPrint, false);
    }

} catch (JRException e) {

```

```

        JOptionPane.showMessageDialog(frame, "Ошибка при создании отчёта:
" + e.getMessage(), "Ошибка", JOptionPane.ERROR_MESSAGE);
        e.printStackTrace();
    }
}

/**
 * Метод для создания XML-файла с данными для отчёта.
 * @return Путь к созданному XML-файлу или null в случае ошибки.
 */
private String createDataXML() {
    try {
        // Создание фабрики и строителя документов
        DocumentBuilderFactory docFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder docBuilder = docFactory.newDocumentBuilder();

        // Создание нового документа
        Document doc = docBuilder.newDocument();

        // Создание корневого элемента <school>
        Element rootElement = doc.createElement("school");
        doc.appendChild(rootElement);

        // Создание элемента <students>
        Element studentsElement = doc.createElement("students");
        rootElement.appendChild(studentsElement);

        // Добавление каждого ученика как элемента <student>
        for (String[] student : originalStudentData) {
            Element studentElement = doc.createElement("student");
            studentElement.setAttribute("name", student[0]);
            studentElement.setAttribute("class", student[1]);
            studentElement.setAttribute("performance", student[2]);
            studentsElement.appendChild(studentElement);
        }

        // Создание временного файла для данных отчёта
        File tempFile = File.createTempFile("student data ", ".xml");
        tempFile.deleteOnExit(); // Удаление файла при завершении
программы

        // Запись документа в файл
        TransformerFactory transformerFactory =
TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        // Для красивого форматирования XML
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");

        transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount",
"4");

        DOMSource source = new DOMSource(doc);
        StreamResult result = new StreamResult(tempFile);
        transformer.transform(source, result);

        return tempFile.getAbsolutePath();
    } catch (ParserConfigurationException | TransformerException |
IOException e) {
        e.printStackTrace();
        return null;
    }
}

/**
 * Метод для запуска потоков в правильной последовательности.
 */
private void runThreadsSequentially() {
    Thread loadDataThread = new Thread(new LoadDataThread(this,
loadLatch), "LoadDataThread");
    Thread saveDataThread = new Thread(new SaveDataThread(this,
loadLatch, saveLatch), "SaveDataThread");
    Thread generateReportThread = new Thread(new
GenerateReportThread(this, saveLatch), "GenerateReportThread");

    loadDataThread.start();
    saveDataThread.start();
}

```

```

        generateReportThread.start();
    }

    /**
     * Класс для загрузки данных из XML-файла.
     */
    class LoadDataThread implements Runnable {
        private Main mainApp;
        private CountDownLatch latch;

        public LoadDataThread(Main mainApp, CountDownLatch latch) {
            this.mainApp = mainApp;
            this.latch = latch;
        }

        @Override
        public void run() {
            System.out.println(Thread.currentThread().getName() + ": Начало
загрузки данных.");
            try {
                mainApp.loadDataFromFile();
                System.out.println(Thread.currentThread().getName() + ":
Завершена загрузка данных.");
                latch.countDown(); // Уведомляем следующий поток о завершении
загрузки
            } catch (DataLoadException e) {
                SwingUtilities.invokeLater(() -> {
                    JOptionPane.showMessageDialog(frame, "Ошибка в потоке
загрузки данных: " + e.getMessage(), "Ошибка", JOptionPane.ERROR MESSAGE);
                });
                e.printStackTrace();
            }
        }
    }

    /**
     * Класс для сохранения данных в XML-файл.
     */
    class SaveDataThread implements Runnable {
        private Main mainApp;
        private CountDownLatch loadLatch;
        private CountDownLatch saveLatch;

        public SaveDataThread(Main mainApp, CountDownLatch loadLatch,
CountDownLatch saveLatch) {
            this.mainApp = mainApp;
            this.loadLatch = loadLatch;
            this.saveLatch = saveLatch;
        }

        @Override
        public void run() {
            try {
                System.out.println(Thread.currentThread().getName() + ":
Ожидание завершения загрузки данных.");
                loadLatch.await(); // Ждем завершения загрузки данных
                System.out.println(Thread.currentThread().getName() + ":
Начало сохранения данных.");

                mainApp.saveDataToFile();

                System.out.println(Thread.currentThread().getName() + ":
Завершено сохранение данных.");
                saveLatch.countDown(); // Уведомляем следующий поток о
завершении сохранения
            } catch (InterruptedException e) {
                SwingUtilities.invokeLater(() -> {
                    JOptionPane.showMessageDialog(frame, "Поток сохранения
данных был прерван.", "Ошибка", JOptionPane.ERROR MESSAGE);
                });
                e.printStackTrace();
            } catch (DataSaveException e) {
                SwingUtilities.invokeLater(() -> {
                    JOptionPane.showMessageDialog(frame, "Ошибка сохранения
данных: " + e.getMessage(), "Ошибка", JOptionPane.ERROR MESSAGE);
                });
                e.printStackTrace();
            }
        }
    }

```

```

    }
}

/**
 * Класс для генерации отчёта.
 */
class GenerateReportThread implements Runnable {
    private Main mainApp;
    private CountDownLatch latch;

    public GenerateReportThread(Main mainApp, CountDownLatch latch) {
        this.mainApp = mainApp;
        this.latch = latch;
    }

    @Override
    public void run() {
        try {
            System.out.println(Thread.currentThread().getName() + ":
Ожидание завершения сохранения данных.");
            latch.await(); // Ждем завершения сохранения данных
            System.out.println(Thread.currentThread().getName() + ":
Начало генерации отчёта.");

            mainApp.generateReport();

            System.out.println(Thread.currentThread().getName() + ":
Завершена генерация отчёта.");
        } catch (InterruptedException e) {
            SwingUtilities.invokeLater(() -> {
                JOptionPane.showMessageDialog(frame, "Поток генерации
отчёта был прерван.", "Ошибка", JOptionPane.ERROR_MESSAGE);
            });
            e.printStackTrace();
        } catch (Exception e) {
            SwingUtilities.invokeLater(() -> {
                JOptionPane.showMessageDialog(frame, "Ошибка генерации
отчёта: " + e.getMessage(), "Ошибка", JOptionPane.ERROR_MESSAGE);
            });
            e.printStackTrace();
        }
    }
}

/**
 * Метод для удаления выбранных учителей с подтверждением.
 */
private void deleteSelectedTeachers() {
    int[] selectedRows = teacherTable.getSelectedRows();
    if (selectedRows.length == 0) {
        JOptionPane.showMessageDialog(frame, "Пожалуйста, выберите
учителей для удаления.", "Ошибка удаления", JOptionPane.ERROR_MESSAGE);
        return;
    }

    // Получение имен выбранных учителей
    StringBuilder names = new StringBuilder();
    for (int row : selectedRows) {
        int modelRow = teacherTable.convertRowIndexToModel(row);
        String teacherName = (String)
teacherTableModel.getValueAt(modelRow, 0);
        names.append(teacherName).append("\n");
    }

    // Определяем тексты кнопок на русском
    String[] options = {"Да", "Нет"};

    // Показываем диалог подтверждения с русскими кнопками
    int confirm = JOptionPane.showOptionDialog(
        frame,
        "Вы уверены, что хотите уволить следующих учеников?\n" +
names.toString(),
        "Подтверждение удаления",
        JOptionPane.YES_NO_OPTION,
        JOptionPane.WARNING_MESSAGE,
        null,
        options,
        options[0]
    );
}

```

```

    );

    if (confirm == JOptionPane.YES_OPTION) {
        // Сортировка выбранных строк по убыванию индексов
        int[] sortedRows = selectedRows.clone();
        java.util.Arrays.sort(sortedRows);
        for (int i = sortedRows.length - 1; i >= 0; i--) {
            int modelRow =
teacherTable.convertRowIndexToModel(sortedRows[i]);
            teacherTableModel.removeRow(modelRow);
            originalTeacherData.remove(modelRow);
        }
        JOptionPane.showMessageDialog(frame, "Учителя удалены.",
"Удаление", JOptionPane.INFORMATION_MESSAGE);
    }

    /**
     * Метод для удаления выбранных учеников с подтверждением.
     */
    private void deleteSelectedStudents() {
        int[] selectedRows = studentTable.getSelectedRows();
        if (selectedRows.length == 0) {
            JOptionPane.showMessageDialog(
                frame,
                "Пожалуйста, выберите учеников для удаления.",
                "Ошибка удаления",
                JOptionPane.ERROR_MESSAGE
            );
            return;
        }

        // Получение имен выбранных учеников
        StringBuilder names = new StringBuilder();
        for (int row : selectedRows) {
            int modelRow = studentTable.convertRowIndexToModel(row);
            String studentName = (String)
studentTableModel.getValueAt(modelRow, 0);
            names.append(studentName).append("\n");
        }

        // Определяем тексты кнопок на русском
        String[] options = {"Да", "Нет"};

        // Показываем диалог подтверждения с русскими кнопками
        int confirm = JOptionPane.showOptionDialog(
            frame,
            "Вы уверены, что хотите удалить следующих учеников?\n" +
names.toString(),
            "Подтверждение удаления",
            JOptionPane.YES_NO_OPTION,
            JOptionPane.WARNING_MESSAGE,
            null,
            options,
            options[0]
        );

        if (confirm == JOptionPane.YES_OPTION || confirm == 0) { // Проверяем
оба варианта
            // Сортировка выбранных строк по убыванию индексов
            int[] sortedRows = selectedRows.clone();
            java.util.Arrays.sort(sortedRows);
            for (int i = sortedRows.length - 1; i >= 0; i--) {
                int modelRow =
studentTable.convertRowIndexToModel(sortedRows[i]);
                studentTableModel.removeRow(modelRow);
                originalStudentData.remove(modelRow);
            }
            JOptionPane.showMessageDialog(
                frame,
                "Ученики удалены.",
                "Удаление",
                JOptionPane.INFORMATION_MESSAGE
            );
        }
    }

    /**

```

```

    * Точка входа в программу. Запуск приложения.
    *
    * @param args Аргументы командной строки (не используются).
    */
    public static void main(String[] args) {
        // Запуск интерфейса в потоке обработки событий Swing
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new Main().SchoolManagementSystem();
            }
        });
    }

    /**
     * Исключение, выбрасываемое при неверном вводе данных.
     */
    class InvalidInputException extends Exception {
        public InvalidInputException(String message) {
            super(message);
        }
    }

    /**
     * Исключение, выбрасываемое если строка таблицы не выбрана.
     */
    class RowNotSelectedException extends Exception {
        public RowNotSelectedException(String message) {
            super(message);
        }
    }

    /**
     * Исключение, выбрасываемое при ошибках загрузки данных из файла.
     */
    class DataLoadException extends Exception {
        public DataLoadException(String message) {
            super(message);
        }
    }

    /**
     * Исключение, выбрасываемое при ошибках сохранения данных в файл.
     */
    class DataSaveException extends Exception {
        public DataSaveException(String message) {
            super(message);
        }
    }
}

```

## Приложение

Репозиторий: [https://github.com/PlatonBarchenkov/OOP\\_lab\\_08.git](https://github.com/PlatonBarchenkov/OOP_lab_08.git)