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 № 7
«Построение отчетов в PDF- и HTML- форматах»
по дисциплине «Объектно - ориентированное программирование на языке Java»

Выполнил: Барченков П. А.

Факультет: КТИ

Группа: №3312

Подпись преподавателя:

Содержание

Цель работы	3
Распечатки XML-файлов до загрузки данных в экранную форму	и после их
выгрузки	3
Распечатка сгенерированных отчетов	5
Текст программы	8
Приложение	18

Цель работы

Знакомство со способами формирования отчетов на языке программирования Java с использованием конструктора Jaspersoft iReport Designer.

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

```
✓> input.xml ×
                                    OptionPane.java
     <?xml version="1.0" encoding="UTF-8"?>
     <school>
         <teachers>
            <teacher name="Иванов Иван Иванович" subject="Математика" classes="5A;6Б"/>
            <teacher name="Петрова Анна Сергеевна" subject="Русский язык" classes="7B;8Г"/>
            <teacher name="Сидоров Петр Петрович" subject="История" classes="9A;10Б"/>
            <teacher name="Козлова Светлана Николаевна" subject="Физика" classes="5A;7B"/>
            <teacher name="Морозов Алексей Викторович" subject="Химия" classes="6Б;8Г"/>
9
            <teacher name="Лебедева Елена Андреевна" subject="Биология" classes="9A;10Б"/>
            <teacher name="Громова Мария Юрьевна" subject="География" classes="5A;6Б"/>
             <teacher name="Фёдоров Сергей Петрович" subject="Литература" classes="7B;8Г"/>
             <teacher name="Николаева Ольга Дмитриевна" subject="ИЗО" classes="9A;10Б"/>
             <teacher name="Семенов Виктор Алексеевич" subject="Музыка" classes="5A;7B"/>
         </teachers>
         <students>
            <student name="Смирнов Алексей Иванович" class="5A" performance="Отлично"/>
             <student name="Кузнецова Мария Петровна" class="6Б" performance="Хорошо"/>
            <student name="Новиков Дмитрий Сергеевич" class="7B" performance="Удовлетворительно"/>
18
            <student name="Иванова Наталья Сергеевна" class="8Г" performance="Отлично"/>
            <student name="Павлов Андрей Александрович" class="9A" performance="Хорошо"/>
            <student name="Соколова Екатерина Викторовна" class="105" performance="Удовлетворительно"/>
            <student name="Михайлов Николай Юрьевич" class="5A" performance="Отлично"/>
            <student name="Орлова Ольга Дмитриевна" class="6Б" performance="Хорошо"/>
            <student name="Борисов Сергей Петрович" class="7B" performance="Удовлетворительно"/>
            <student name="Герасимова Анастасия Ивановна" class="8Г" performance="Отлично"/>
            <student name="Киселев Владимир Андреевич" class="9A" performance="Хорошо"/>
            <student name="Егорова Дарья Александровна" class="106" performance="Удовлетворительно"/>
            <student name="Фролова Анна Сергеевна" class="5A" performance="Отлично"/>
28
            <student name="Денисов Алексей Викторович" class="66" performance="Хорошо"/>
            <student name="Тихонов Михаил Павлович" class="7B" performance="Удовлетворительно"/>
             <student name="Луначарова Виктория Юрьевна" class="8Г" performance="Отлично"/>
             <student name="Зайцев Евгений Николаевич" class="9A" performance="Хорошо"/>
             <student name="Никитина Светлана Андреевна" class="105" performance="Удовлетворительно"/>
         </students>
     </school>
```

Рисунок 1 – Содержимое исходного ХМL-файла.

```
Main.java
                 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
     <school>
         <teachers>
            <teacher classes="5A;6Б" name="Иванов Иван Иванович" subject="Математика"/>
             <teacher classes="7B;8Г" name="Петрова Анна Сергеевна" subject="Русский язык"/>
             <teacher classes="9A;10Б" name="Сидоров Петр Петрович" subject="История"/>
             <teacher classes="5A;7B" name="Козлова Светлана Николаевна" subject="Физика"/>
             <teacher classes="6Б;8Г" name="Морозов Алексей Викторович" subject="Химия"/>
             <teacher classes="9A;10Б" name="Лебедева Елена Андреевна" subject="Биология"/>
         </teachers>
         <students>
            <student class="5A" name="Смирнов Алексей Иванович" performance="Отлично"/>
             <student class="6Б" name="Кузнецова Мария Петровна" performance="Хорошо"/>
            <student class="7B" name="Новиков Дмитрий Сергеевич" performance="Удовлетворительно"/>
            <student class="8Г" name="Иванова Наталья Сергеевна" performance="Отлично"/>
            <student class="9A" name="Павлов Андрей Александрович" performance="Хорошо"/>
            <student class="106" name="Соколова Екатерина Викторовна" performance="Удовлетворительно"/>
             <student class="5A" name="Михайлов Николай Юрьевич" performance="Отлично"/>
             <student class="6Б" name="Орлова Ольга Дмитриевна" performance="Хорошо"/>
         </students>
     </school>
```

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

Скриншоты, иллюстрирующие построение шаблона в дизайнере iReport

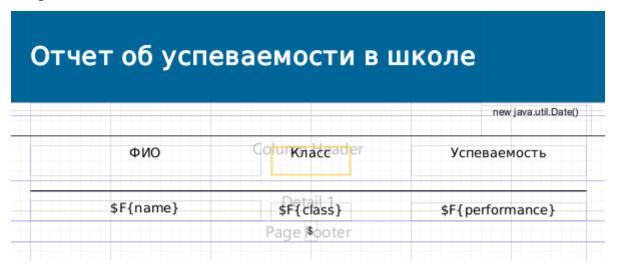


Рисунок 3 – Создание макета.

		16.11.2024, 02:43
ФИО	Класс	Успеваемость
Смирнов Алексей Иванович	5A	Отлично
Кузнецова Мария Петровна	66	Хорошо
Новиков Дмитрий Сергеевич	7B	Удовлетворительно
Иванова Наталья Сергеевна	8Г	Отлично
Павлов Андрей Александрович	9A	Хорошо
Соколова Екатерина Викторовна	105	Удовлетворительно
Михайлов Николай Юрьевич	5A	Отлично
Орлова Ольга Дмитриевна	65	Хорошо
Борисов Сергей Петрович	7B	Удовлетворительно
Герасимова Анастасия Ивановна	8Г	Отлично
Киселев Владимир Андреевич	9A	Хорошо
Егорова Дарья Александровна	105	Удовлетворительно
Фролова Анна Сергеевна	5A	Отлично
Денисов Алексей Викторович	65	Хорошо
Тихонов Михаил Павлович	7B	Удовлетворительно

Рисунок 4 – Предварительный просмотр отчета, созданного по шаблону.

Распечатка сгенерированных отчетов

Отчет об успеваемости в школе

16.11.2024, 02:45

ФИО	Класс	Успеваемость
Смирнов Алексей Иванович	5A	Отлично
Кузнецова Мария Петровна	6Б	Хорошо
Новиков Дмитрий Сергеевич	7B	Удовлетворительно
Иванова Наталья Сергеевна	8Γ	Отлично
Павлов Андрей Александрович	9 A	Хорошо
Соколова Екатерина Викторовна	105	Удовлетворительно
Михайлов Николай Юрьевич	5A	Отлично
Орлова Ольга Дмитриевна	6Б	Хорошо
Борисов Сергей Петрович	7B	Удовлетворительно
Герасимова Анастасия Ивановна	8Γ	Отлично
Киселев Владимир Андреевич	9 A	Хорошо
Егорова Дарья Александровна	105	Удовлетворительно
Фролова Анна Сергеевна	5A	Отлично
Денисов Алексей Викторович	6Б	Хорошо
Тихонов Михаил Павлович	7B	Удовлетворительно

1

ОИФ	Класс	Успеваемость
Луначарова Виктория Юрьевна	8Г	Отлично
Зайцев Евгений Николаевич	9A	Хорошо
Никитина Светлана Андреевна	105	Удовлетворительно

Текст программы

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;
import javax.swing.RowFilter;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Collections;
import java.util.List;
import org.w3c.dom.*;
import bly.wse.dom.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.xml.sax.SAXException;
// Импорты JasperReports
import net.sf.jasperreports.engine.*;
import net.sf.jasperreports.engine.data.JRXmlDataSource;
import java.util.HashMap;
 * Программа управления школой с обработкой ХМL-документов.
  * @author Барченков Платон 3312
 * @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;
      /**
       * 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("School Management System");
           frame.setSize(1000, 700); // Увеличиваем размер окна для двух таблиц
           frame.setDefaultCloseOperation(JFrame.EXIT ON CLOSE);
           frame.setLayout(new BorderLayout());
             / Создание панели инструментов с кнопками действий
           JToolBar actionPanel = new JToolBar("Toolbar");
           // Кнопки для учителей и учеников
           addTeacherButton = new JButton("Добавить учителя");
           addStudentButton = new JButton("Добавить ученика");
           deleteTeacherButton = new JButton("Удалить учителей");
           deleteStudentButton = new JButton("Удалить учеников");
           generateReportButton = new JButton("Создать отчет");
           // 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);
         // 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);
```

```
// 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);
        // 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();
        // Действие при нажатии кнопки "Поиск"
        searchButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                 String criterion = (String) searchCriteria.getSelectedItem();
String value = searchField.getText().trim();
                 searchTable(criterion, value);
             }
        });
        // Действие при нажатии кнопки "Сбросить"
        resetButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                 resetTable();
        });
        // Действие при нажатии кнопки "Добавить учителя"
        addTeacherButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                 String teacherName = JOptionPane.showInputDialog(frame,
"Введите ФИО учителя:");
                 if (teacherName == null || teacherName.trim().isEmpty()) {
                     JOptionPane.showMessageDialog(frame, "ФИО учителя не
может быть пустым.", "Ошибка", JOptionPane.ERROR_MESSAGE);
                     return:
                 String subject = JOptionPane.showInputDialog(frame, "Введите
предмет:");
                 if (subject == null || subject.trim().isEmpty()) {
                     JOptionPane.showMessageDialog(frame, "Предмет не может
быть пустым.",
                "Ошибка", JOptionPane.ERROR_MESSAGE);
                     return;
                 }
                 String classes = JOptionPane.showInputDialog(frame, "Введите
классы (разделенные точкой с запятой ';'):");

if (classes == null || classes.trim().isEmpty()) {
                     JOptionPane.showMessageDialog(frame, "Классы не могут
быть пустыми.",
                 "Ошибка", JOptionPane.ERROR_MESSAGE);
                     return;
                 String[] newTeacher = {teacherName.trim(), subject.trim(),
classes.trim();
                 teacherTableModel.addRow(newTeacher);
                 originalTeacherData.add(newTeacher);
             }
```

```
});
        // Действие при нажатии кнопки "Удалить учителей"
        deleteTeacherButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                 int[] selectedRows = teacherTable.getSelectedRows();
                 if (selectedRows.length > 0) {
                     // Создаем массив опций с русскими надписями Object[] options = {"Да", "Нет"};
                     int confirm = JOptionPane.showOptionDialog(
                             frame.
                             "Вы уверены, что хотите удалить выбранных
учителей?",
                              "Подтверждение удаления"
                             JOptionPane.YES NO OPTION,
                             JOptionPane.QUESTION MESSAGE,
                             null,
                             options
                             options[1]
                     );
                     if (confirm == JOptionPane.YES OPTION) {
                          // Преобразуем индексы с учёта сортировки и сортируем
в обратном порядке
                         List<Integer> rows = new ArrayList<>();
                         for (int row : selectedRows) {
rows.add(teacherTable.convertRowIndexToModel(row));
                         Collections.sort(rows, Collections.reverseOrder());
                         for (int row : rows) {
                              teacherTableModel.removeRow(row);
                             originalTeacherData.remove(row);
                         JOptionPane.showMessageDialog(frame, "Учителя успешно
удалены.", "Успех",
                     JOptionPane.INFORMATION MESSAGE);
                 } else
JOptionPane.showMessageDialog(frame, "Пожалуйста, выберите учителей для удаления.", "Ошибка", JOptionPane.ERROR_MESSAGE);
             }
        });
        // Действие при нажатии кнопки "Добавить ученика"
        addStudentButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                 String studentName = JOptionPane.showInputDialog(frame,
"Введите ФИО ученика:");
                 if (studentName == null || studentName.trim().isEmpty()) {
                     JOptionPane.showMessageDialog(frame, "ФИО ученика не "Ошибка", JOptionPane.ERROR MESSAGE);
может быть пустым.",
                     return;
                 }
                 String studentClass = JOptionPane.showInputDialog(frame,
"Введите класс:");
                 if
                    (studentClass == null || studentClass.trim().isEmpty()) {
                     JOptionPane.showMessageDialog(frame, "Класс не может быть
пустым.", "Ошибка",
                     JOptionPane.ERROR MESSAGE);
                     return;
                 }
String performance = JOptionPane.showInputDialog(frame, "Введите успеваемость:");
                 if (performance == null || performance.trim().isEmpty()) {
                     JOptionPane.showMessageDialog(frame, "Успеваемость не
может быть пустой.", "Ошибка", JOptionPane.ERROR MESSAGE);
                     return:
                 String[] newStudent = {studentName.trim(),
originalStudentData.add(newStudent);
        });
```

```
// Действие при нажатии кнопки "Удалить учеников"
        deleteStudentButton.addActionListener(new ActionListener() {
             public void actionPerformed(ActionEvent e) {
                 int[] selectedRows = studentTable.getSelectedRows();
                 if (selectedRows.length > 0) {
                     // Создаем массив опций с русскими надписями Object[] options = {"Да", "Нет"};
                     int confirm = JOptionPane.showOptionDialog(
                              frame,
                              "Вы уверены, что хотите удалить выбранных
учеников?",
                              "Подтверждение удаления",
JOptionPane.YES NO OPTION,
                              JOptionPane.QUESTION MESSAGE,
                              null,
                              options,
                              options[1]
                     );
                      if (confirm == JOptionPane.YES OPTION) {
                          // Преобразуем индексы с учёта сортировки и сортируем
в обратном порядке
                          List<Integer> rows = new ArrayList<>();
                          for (int row : selectedRows) {
rows.add(studentTable.convertRowIndexToModel(row));
                          Collections.sort(rows, Collections.reverseOrder());
                          for (int row : rows) {
    studentTableModel.removeRow(row);
                              originalStudentData.remove(row);
                          JOptionPane.showMessageDialog(frame, "Ученики успешно
удалены.", "Успех",
                     JOptionPane.INFORMATION MESSAGE);
                   else {
                     JOptionPane.showMessageDialog(frame, "Пожалуйста,
выберите учеников для удаления.", "Ошибка", JOptionPane.ERROR MESSAGE);
                 }
        });
         // Действие при нажатии кнопки "Загрузить данные"
        loadButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                 loadDataFromXML();
             }
        });
         // Действие при нажатии кнопки "Сохранить данные"
        saveButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                 saveDataToXML();
        });
        // Действие при нажатии кнопки "Создать отчет" generateReportButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                 generateReport();
        });
         // Делаем главное окно видимым
        frame.setVisible(true);
    }
     * 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
         "Ошибка", JOptionPane.ERROR_MESSAGE);
пустым. ",
            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 ХМL-файла.
    private void loadDataFromXML() {
```

```
JFileChooser fileChooser = new JFileChooser();
        fileChooser.setDialogTitle("Выберите XML-файл для загрузки данных");
         // Установка текущей директории на папку проекта
        File projectFolder = new File(System.getProperty("user.dir"));
        File dataFolder = new File(projectFolder, "data");
           Если папка data не существует, создаём её
        if (!dataFolder.exists()) {
             dataFolder.mkdir();
        fileChooser.setCurrentDirectory(dataFolder);
        fileChooser.setFileFilter(new
javax.swing.filechooser.FileNameExtensionFilter("XML файлы", "xml"));
        int userSelection = fileChooser.showOpenDialog(frame);
        if (userSelection == JFileChooser.APPROVE OPTION) {
             File xmlFile = fileChooser.getSelectedFile();
                 // Создание парсера и загрузка документа
                 DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
                 DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
                 Document doc = dBuilder.parse(xmlFile);
                 // Нормализация документа
                 doc.getDocumentElement().normalize();
                 // Очистка текущих данных в таблицах и исходных списках
                 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++) {</pre>
                     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};
                     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);</pre>
                     String name = studentElement.getAttribute("name");
                     String studentClass =
studentElement.getAttribute("class");
                     String performance =
studentElement.getAttribute("performance");
                     String[] student = {name, studentClass, performance};
                     studentTableModel.addRow(student);
                     originalStudentData.add(student);
                 1
                 JOptionPane.showMessageDialog(frame, "Данные успешно
загружены из XML-файла.", "Успех", JOptionPane.INFORMATION MESSAGE);
             } catch (ParserConfigurationException | SAXException |
IOException e) {
                 e.printStackTrace();
                 JOptionPane.showMessageDialog(frame, "Ошибка при загрузке
данных: "
          + e.getMessage(), "Ошибка", JOptionPane.ERROR_MESSAGE);
    }
    /**
```

```
* Метод для сохранения данных в ХМL-файл.
    private void saveDataToXML() {
         try {
              // Создание фабрики и построителя 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]);
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 ХМL
              transformer.setOutputProperty(OutputKeys.INDENT, "yes");
transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount",
"4");
              DOMSource source = new DOMSource(doc);
              // Создание папки output, если не существует
              File outputDir = new File("output");
              if (!outputDir.exists()) {
                   outputDir.mkdir();
              // Настройка JFileChooser
              JFileChooser fileChooser = new JFileChooser();
              fileChooser.setDialogTitle("Сохранить данные в XML-файл");
              fileChooser.setCurrentDirectory(outputDir);
              fileChooser.setFileFilter(new
javax.swing.filechooser.FileNameExtensionFilter("XML файлы", "xml"));
              int userSelection = fileChooser.showSaveDialog(frame);
              if (userSelection == JFileChooser.APPROVE OPTION) {
                   File xmlFile = fileChooser.getSelectedFile();
                     / Добавляем расширение .xml, если оно отсутствует
                   if (!xmlFile.getName().toLowerCase().endsWith(".xml")) {
    xmlFile = new File(xmlFile.getParentFile(),
xmlFile.getName() + ".xml");
                   StreamResult result = new StreamResult(xmlFile);
```

```
transformer.transform(source, result);
JOptionPane.showMessageDialog(frame, "Данные успешно
сохранены в XML-файл.", "Успех", JOptionPane.INFORMATION MESSAGE);
        } catch (ParserConfigurationException | TransformerException e) {
             e.printStackTrace();
             JOptionPane.showMessageDialog(frame, "Ошибка при сохранении
        " + e.getMessage(), "Ошибка", JOptionPane.ERROR_MESSAGE);
ланных:
    }
    /**
     * Метод для генерации отчёта.
    private void generateReport() {
        try {
             // Путь к шаблону отчёта
String reportPath = "lab_07.jrxml"; // Убедитесь, что путь
правильный
             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 ХМL-файла с данными учеников для отчёта
             String xmlDataPath = createStudentDataXML();
             if (xmlDataPath == null) {
                 JOptionPane.showMessageDialog(frame, "Не удалось создать XML-
файл с данными для отчёта.", "Ошибка", JOptionPane.ERROR MESSAGE);
                 return:
             }
             // Создание источника данных из ХМL-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("output"));
             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("." +
if (choice == 0) { // PDF
                     JasperExportManager.exportReportToPdfFile(jasperPrint,
filePath);
                 } else if (choice == 1) { // HTML
                     JasperExportManager.exportReportToHtmlFile(jasperPrint,
filePath):
                 JOptionPane.showMessageDialog(frame, "Отчёт успешно сохранён:
" + filePath,
               "Ycπex", JOptionPane.INFORMATION_MESSAGE);
        } catch (JRException e) {
             e.printStackTrace();
             JOptionPane.showMessageDialog(frame, "Ошибка при создании отчёта:
 + e.getMessage(), "Ошибка", JOptionPane.ERROR MESSAGE);
    }
    /**
     * Метод для создания ХМL-файла с данными учеников.
     * @return Путь к созданному XML-файлу или null в случае ошибки.
    private String createStudentDataXML() {
        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 ХМL
             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();
```

Приложение

Репозиторий: https://github.com/PlatonBarchenkov/OOP lab 07.git