**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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2024

**Содержание**

[Цель работы 3](#_Toc182600092)

[Распечатки XML-файлов до загрузки данных в экранную форму и после их выгрузки 3](#_Toc182600093)

[Распечатка сгенерированных отчетов 6](#_Toc182600094)

[Текст программы 8](#_Toc182600095)

[Приложение 18](#_Toc182600096)

# 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

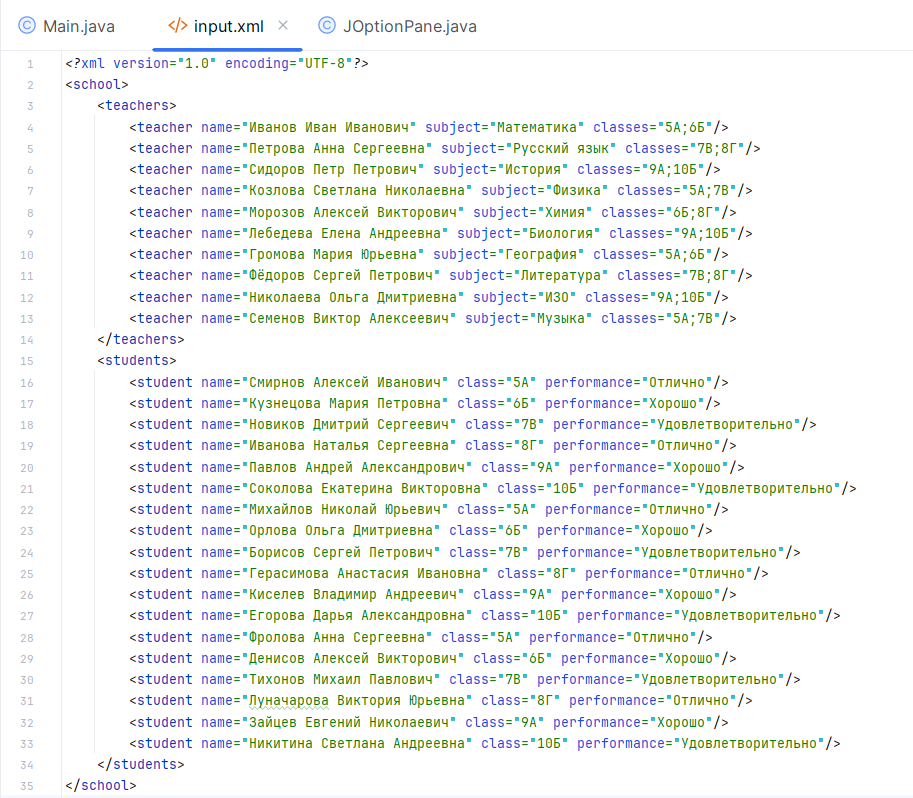


Рисунок 1 – Содержимое исходного XML-файла.



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

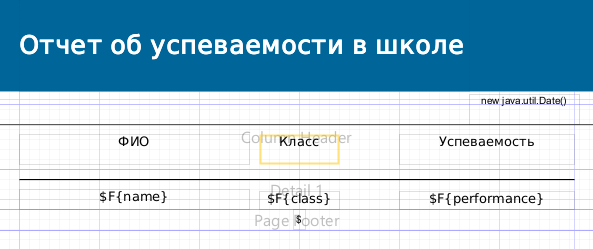
**Скриншоты, иллюстрирующие построение шаблона в дизайнере iReport **

Рисунок 3 – Создание макета.

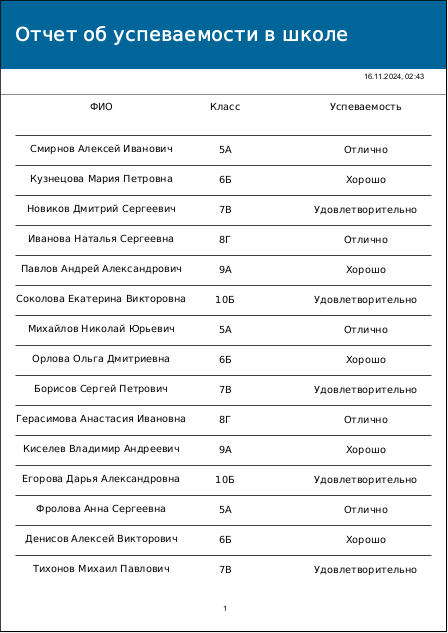
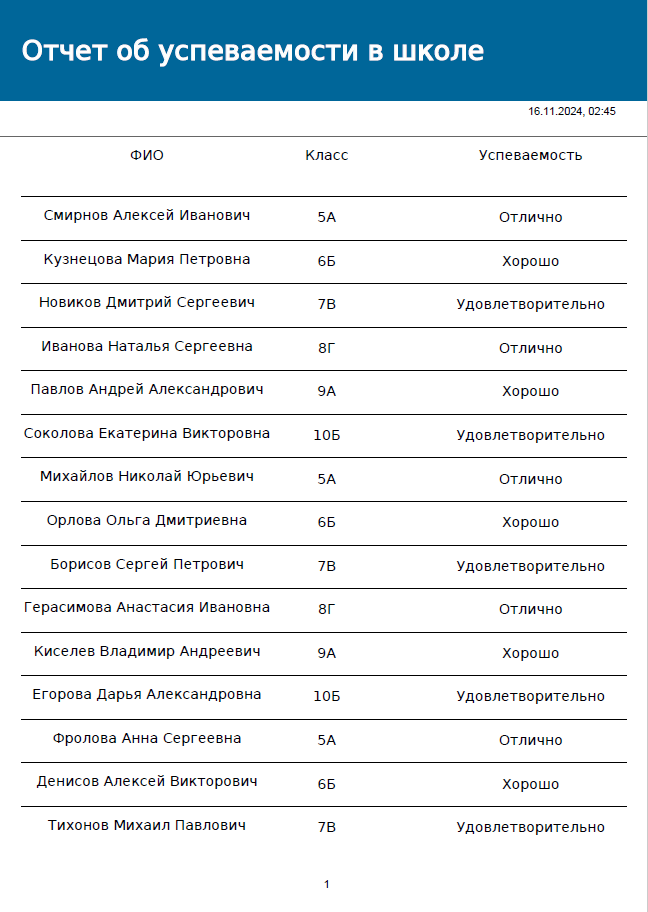


Рисунок 4 – Предварительный просмотр отчета, созданного по шаблону.

# Распечатка сгенерированных отчетов



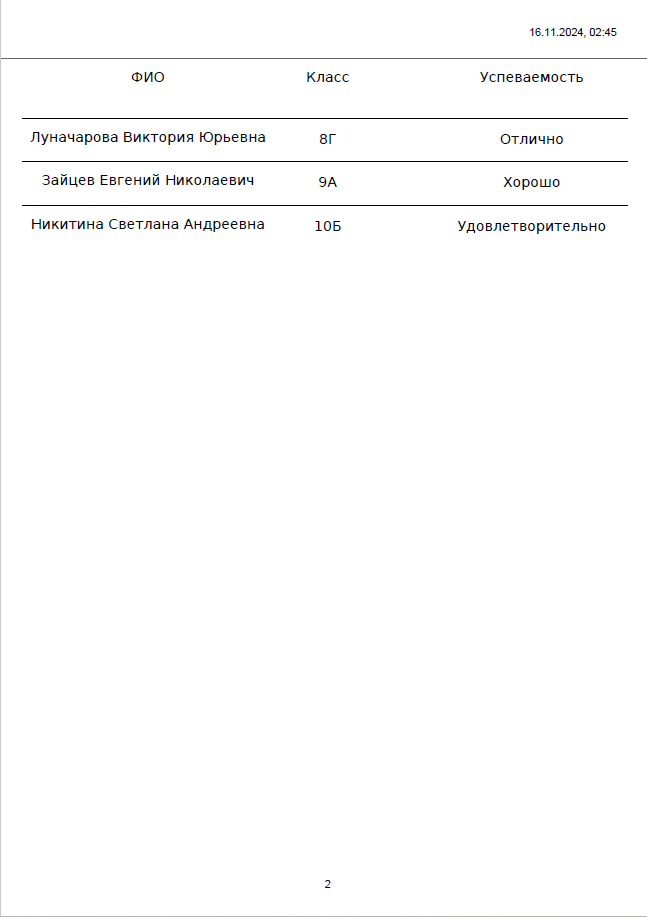


Рисунок 5 – Сгенерированный ответ в формате pdf.

# 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.List; import org.w3c.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 XML-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 может быть пустым.", "Ошибка", JOptionPane.ERROR\_MESSAGE);  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(), studentClass.trim(), performance.trim()};  studentTableModel.addRow(newStudent);  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 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 XML-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();   try {  // 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++) {  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);  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);  }   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 XML-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 XML  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 данных: " + e.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);  }  }   /\*\*  \* 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 XML-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 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("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("." + 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);  }   } catch (JRException e) {  e.printStackTrace();  JOptionPane.showMessageDialog(frame, "Ошибка при создании отчёта: " + e.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);  }  }    /\*\*  \* Метод для создания XML-файла с данными учеников.  \*  \* @return Путь к созданному XML-файлу или null в случае ошибки.  \*/  private String createStudentDataXML() {  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 <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;  }  }   /\*\*  \* 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();  }  });  } }** |

# Приложение

Репозиторий: https://github.com/PlatonBarchenkov/OOP\_lab\_07.git