**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

отчет

**по лабораторной работе №6**

**по дисциплине «Объектно-ориентированное программирование»**

Тема: «Обработка XML-документов»

|  |  |  |
| --- | --- | --- |
| Студент гр. 4311 |  | Левицкий М.Ю. |
| Преподаватель |  | Лазарев Э.А. |

Санкт-Петербург

2025

**ЦЕЛЬ**

1. С помощью текстового редактора создайте файл, в котором будет

описана структура XML-документа и данные для загрузки в экранную

форму, разработанную в лабораторной работе № 5. Проверьте корректность

XML-файла, открыв его в любом web-браузере (например, Internet Explorer

или Google Chrome).

2. Создайте новый проект, который будет дублировать проект

лабораторной работы № 5.

3. Напишите и замените в проекте обработчики кнопок загрузки

данных в XML-файл и выгрузки из него.

4. Загрузите данные в экранную форму из XML-файла. Убедитесь, что

данные в экранной форме соответствуют данным XML-файла.

5. Внесите изменения в данные экранной формы и сохраните их в

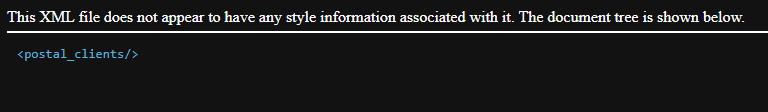
XML-файле.

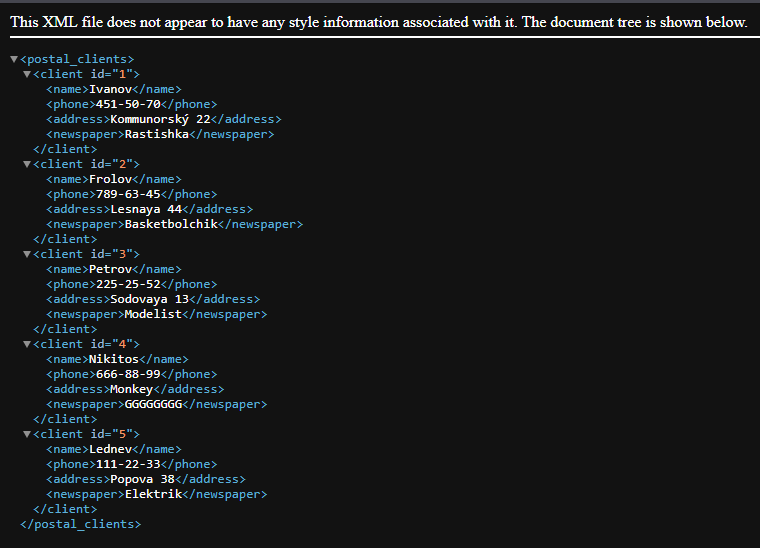
6. Просмотрите в браузере сохраненный XML-файл и убедитесь в

правильности работы приложения.

7. Сгенерируйте документацию с помощью Javadoc и просмотрите ее в

браузере.  
  
  
  
  
  
  
  
  
  
  
  
  
  
**Распечатки содержимого файлов с данными до и после внесения изменений.**

**  
Скриншот 1 (исходный файл)**

**Скриншот 2 (файл после загрузки данных в него)**

**СКРИНШОТЫ, ИЛЛЮСТРИРУЮЩИЕ ПРОЦЕСС ЗАГРУЗКИ ДАННЫХ В ФАЙЛ И ВЫГРУЗКИ ИЗ НЕГО**

|  |
| --- |
|  |

**Скриншот 1 (процесс выгрузки с файла)**

|  |
| --- |
|  |

**Скриншот 2 (процесс загрузки в файл)**

# ТЕКСТ ДОКУМЕНТАЦИИ

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

**Фрагменты кода, отвечающие за сохранение и чтение данных из файла.**

**1.Сохранение данных в файл (saveDataToXML()** **метод)**

|  |
| --- |
| /\*\*  \* Saves table data to XML file using DOM parser.  \* Creates structured XML document with client information.  \*/  private void saveDataToXML() {  FileDialog saveDialog = new FileDialog(mainFrame, "Save Client Data as XML", FileDialog.SAVE);  saveDialog.setFile("clients.xml");  saveDialog.setVisible(true);    String directory = saveDialog.getDirectory();  String fileName = saveDialog.getFile();    if (fileName == null) {  return; // User cancelled  }    String fullPath = directory + fileName;    try {  // Create document builder  DocumentBuilder builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();  Document doc = builder.newDocument();    // Create root element  Element rootElement = doc.createElement("postal\_clients");  doc.appendChild(rootElement);    // Add clients from table  for (int i = 0; i < model.getRowCount(); i++) {  Element client = doc.createElement("client");  client.setAttribute("id", String.valueOf(i + 1));    // Add name  Element name = doc.createElement("name");  name.appendChild(doc.createTextNode(model.getValueAt(i, 0).toString()));  client.appendChild(name);    // Add phone  Element phone = doc.createElement("phone");  phone.appendChild(doc.createTextNode(model.getValueAt(i, 1).toString()));  client.appendChild(phone);    // Add address  Element address = doc.createElement("address");  address.appendChild(doc.createTextNode(model.getValueAt(i, 2).toString()));  client.appendChild(address);    // Add newspaper  Element newspaper = doc.createElement("newspaper");  newspaper.appendChild(doc.createTextNode(model.getValueAt(i, 3).toString()));  client.appendChild(newspaper);    rootElement.appendChild(client);  }    // Write to file  Transformer transformer = TransformerFactory.newInstance().newTransformer();  transformer.setOutputProperty(OutputKeys.INDENT, "yes");  transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");  transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");    transformer.transform(new DOMSource(doc), new StreamResult(new File(fullPath)));    JOptionPane.showMessageDialog(mainFrame,  "Data successfully saved to XML: " + fileName,  "XML Save Complete",  JOptionPane.INFORMATION\_MESSAGE);    } catch (Exception ex) {  JOptionPane.showMessageDialog(mainFrame,  "Error saving XML file: " + ex.getMessage(),  "XML Save Error",  JOptionPane.ERROR\_MESSAGE);  ex.printStackTrace();  }  } |

**2.Чтение данных из файла (loadDataFromXML() метод)**

|  |
| --- |
| /\*\*  \* Loads table data from XML file using DOM parser.  \* Parses XML structure and populates the table with client data.  \*/  private void loadDataFromXML() {  FileDialog loadDialog = new FileDialog(mainFrame, "Load Client Data from XML", FileDialog.LOAD);  loadDialog.setFile("\*.xml");  loadDialog.setVisible(true);    String directory = loadDialog.getDirectory();  String fileName = loadDialog.getFile();    if (fileName == null) {  return; // User cancelled  }    String fullPath = directory + fileName;    try {  // Create document builder  DocumentBuilder builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();  Document doc = builder.parse(new File(fullPath));    // Normalize document  doc.getDocumentElement().normalize();    // Clear existing data  model.setRowCount(0);    // Get all client elements  NodeList clientList = doc.getElementsByTagName("client");    int loadedCount = 0;  for (int i = 0; i < clientList.getLength(); i++) {  Node clientNode = clientList.item(i);    if (clientNode.getNodeType() == Node.ELEMENT\_NODE) {  Element clientElement = (Element) clientNode;    // Extract data from elements  String name = getElementText(clientElement, "name");  String phone = getElementText(clientElement, "phone");  String address = getElementText(clientElement, "address");  String newspaper = getElementText(clientElement, "newspaper");    // If elements are empty, try attributes  if (name.isEmpty()) name = clientElement.getAttribute("name");  if (phone.isEmpty()) phone = clientElement.getAttribute("phone");  if (address.isEmpty()) address = clientElement.getAttribute("address");  if (newspaper.isEmpty()) newspaper = clientElement.getAttribute("newspaper");    // Add to table if we have basic data  if (!name.isEmpty() && !phone.isEmpty()) {  model.addRow(new Object[]{name, phone, address, newspaper});  loadedCount++;  }  }  }    updateSurnameComboBox(model, surnameComboBox);  clearInputFields();    JOptionPane.showMessageDialog(mainFrame,  "Data successfully loaded from XML: " + fileName + "\nLoaded " + loadedCount + " clients",  "XML Load Complete",  JOptionPane.INFORMATION\_MESSAGE);    } catch (FileNotFoundException ex) {  JOptionPane.showMessageDialog(mainFrame,  "XML file not found: " + fileName,  "XML Load Error",  JOptionPane.ERROR\_MESSAGE);  } catch (Exception ex) {  JOptionPane.showMessageDialog(mainFrame,  "Error loading XML file: " + ex.getMessage(),  "XML Load Error",  JOptionPane.ERROR\_MESSAGE);  ex.printStackTrace();  }  } |

# ВЫВОДЫ

В результате выполнения лабораторной работы я изучил механизм работы с XML-документами в Java. Освоил механику добавление данных в файл и выгрузки данных из него.