# 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

Студентка гр. 2308	 Рымарь М.И.
Преподаватель	 Павловский М.Г.

Санкт-Петербург

2023

## Цель работы.

Познакомиться с технологией обработки XML-документов и файлов.

### Задание.

Отчет по лабораторной работе должен содержать:

- 1. Распечатки XML-файлов до загрузки данных в экранную форму и после их выгрузки.
- 2. Скриншоты, иллюстрирующие процесс загрузки данных в XML-файл и выгрузки из него.
  - 3. Текст документации, сгенерированный Javadoc.
- 4. Фрагменты кода, отвечающие за сохранение и чтение данных из XMLфайла.

## Выполнение работы.

1. На рисунке 1 представлен XML-файл до загрузки данных в экранную форму, на рисунке 2 — после выгрузки.

Рисунок 1 - XML-файл до загрузки данных в экранную форму

Рисунок 2 - ХМС-файл после выгрузки данных из экранной формы

2. На рисунках 3 и 4 представлены экранные формы до и после изменения данных в экранной форме, соответственно.

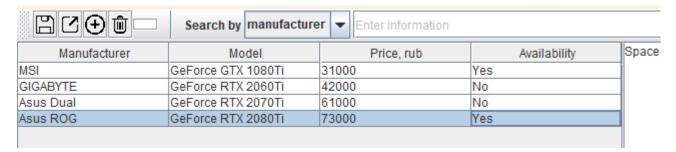


Рисунок 3 - Экранная форма после загрузки ХМС-файла

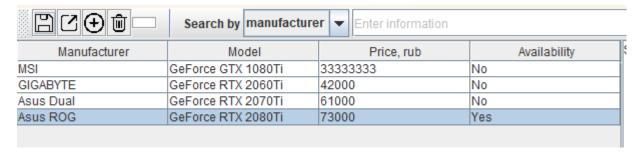


Рисунок 4 – Экранная форма после изменения данных в ней

3. Полный текст документации, сгенерированной Javadoc, приложен в папке с кодом. Часть документации представлена на рисунке 5.

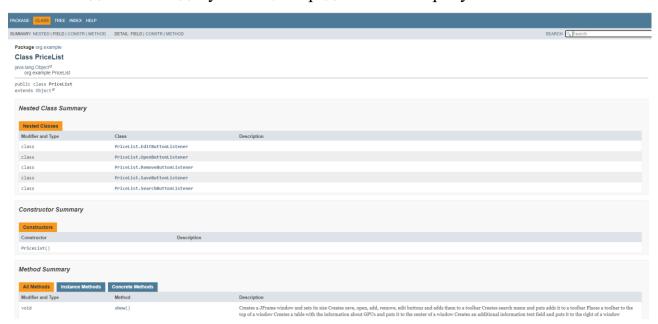


Рисунок 5 – Часть документации Javadoc

4. Фрагменты кода, отвечающие за сохранение и чтение данных из XML-файла представлены в приложении A.

Исходный код всей программы приложен в папке.

# Выводы.

В результате выполнения лабораторной работы была освоена технология обработки XML-документов.

#### ПРИЛОЖЕНИЕ А

# ИСХОДНЫЙ КОД ФУНКЦИЙ, ОТВЕЧАЮЩИХ ЗА СОХРАНЕНИЕ И ЧТЕНИЕ ДАННЫХ ИЗ XML-ФАЙЛА

```
/**
          * Creates a frame for selecting a CSV file and writes GPUTable information to it.
          * @throws IOException if there was a problem writing information.
          * @throws ParserConfigurationException if there was a problem configuring a parser.
          * @throws TransformerConfigurationException if there was a problem configuring a
transformer.
          * @throws TransformerException if there was a problem with a transformer.
          */
         private void saveFile() throws IOException, ParserConfigurationException,
TransformerConfigurationException, TransformerException {
            String fileFormat = "*.xml";
           FileDialog save = new FileDialog(priceList, "Open file", FileDialog.SAVE);
           save.setFile(fileFormat);
           save.setVisible(true);
           String fileName = save.getDirectory() + save.getFile();
           if (!fileName.endsWith(".xml")){
              fileName += ".xml";
           if(save.getFile() == null) return;
           DocumentBuilder builder =
                DocumentBuilderFactory.newInstance().newDocumentBuilder();
           Document document = builder.newDocument();
           Node pricelistNode = document.createElement("pricelist");
            document.appendChild(pricelistNode);
           for (int i = 0; i < priceListModel.getRowCount(); i++)
            {
              Element GPU = document.createElement("GPU");
              pricelistNode.appendChild(GPU);
              GPU.setAttribute("manufacturer", (String)priceListModel.getValueAt(i, 0));
              GPU.setAttribute("model", (String)priceListModel.getValueAt(i, 1));
```

```
GPU.setAttribute("price", (String)priceListModel.getValueAt(i, 2));
              GPU.setAttribute("availability", (String)priceListModel.getValueAt(i, 3));
            }
           Transformer = TransformerFactory.newInstance().newTransformer();
           java.io.FileWriter fileWriter = new FileWriter(fileName);
           transformer.transform(new DOMSource(document), new StreamResult(fileWriter));
         }
         /**
          * Creates a frame for selecting a CSV file and writes its information to GPUTable.
          * @throws ParserConfigurationException if there was a problem configuring a parser.
          * @throws SAXException if there was a problem parsing a file.
          * @throws IOException if there was a problem reading a file.
         private void openFile() throws ParserConfigurationException, SAXException,
IOException {
           String fileFormat = "*.xml";
           FileDialog load = new FileDialog(priceList, "Open file", FileDialog.LOAD);
           load.setFile(fileFormat);
           load.setDirectory(load.getDirectory());
           load.setVisible(true);
           String fileName = load.getDirectory() + load.getFile();
           if(load.getFile() == null) return;
           DocumentBuilder dBuilder =
                DocumentBuilderFactory.newInstance().newDocumentBuilder();
           Document document = dBuilder.parse(new File(fileName));
           document.getDocumentElement().normalize();
           NodeList nodeListGPUs = document.getElementsByTagName("GPU");
           int rows = priceListModel.getRowCount();
           for (int i = 0; i < rows; i++) priceListModel.removeRow(0);
           for (int i = 0; i < nodeListGPUs.getLength(); <math>i++) {
              Node element = nodeListGPUs.item(i);
              NamedNodeMap attrs = element.getAttributes();
              String manufacturer = attrs.getNamedItem("manufacturer").getNodeValue();
```

```
String model = attrs.getNamedItem("model").getNodeValue();
    String price = attrs.getNamedItem("price").getNodeValue();
    String availability = attrs.getNamedItem("availability").getNodeValue();
    priceListModel.addRow(new String[]{manufacturer, model, price, availability});
  }
}
/* Класс слушателя для кнопки открытия файла */
public class OpenButtonListener extends MouseAdapter {
  public void mousePressed(MouseEvent e) {
    try {
       openFile();
    catch (FileNotFoundException FNFex) {
       FNFex.printStackTrace();
    catch (IOException IOex) {
       IOex.printStackTrace();
     }
    catch (ParserConfigurationException PCEex){
       PCEex.printStackTrace();
     }
    catch (SAXException SAXex) {
       SAXex.printStackTrace();
    }
  }
}
```