МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

ОТЧЕТ

по лабораторной работе № 8

по дисциплине «Объектно-ориентированное программирование»

Тема: Организация многопоточных приложений

Студентка гр. 2308 _____ Рымарь М.И.

Преподаватель _____ Павловский М.Г.

Санкт-Петербург

2023

Цель работы.

Познакомиться с правилами и классами построения параллельных приложений в языке Java.

Задание.

Отчет по лабораторной работе должен содержать:

1. Исходный и отредактированный XML-файлы.
2. Скриншот построенного отчёта.
3. Текст документации, сгенерированный Javadoc.
4. Фрагменты кода, отвечающие за организацию параллельной работы трёх потоков.

Выполнение работы.

1. Исходный и отредактированный XML-файлы представлены на рисунках 1 и 2, соответственно.

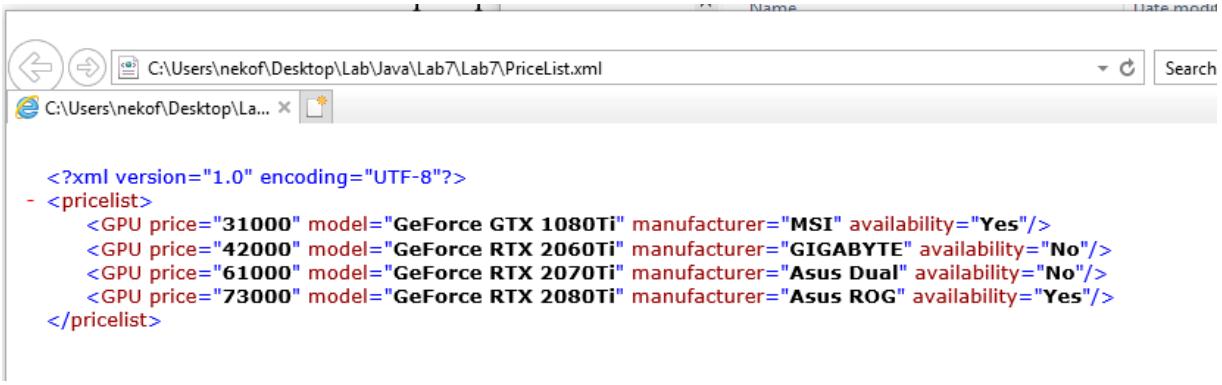


Рисунок 1 – Исходный XML-файл

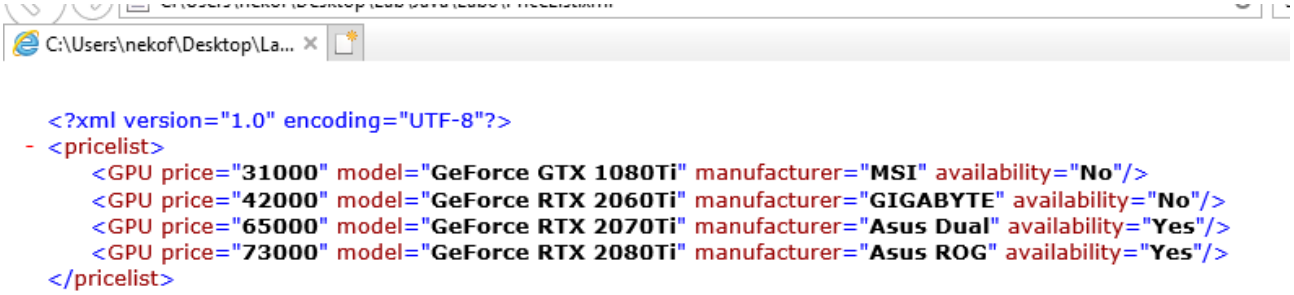
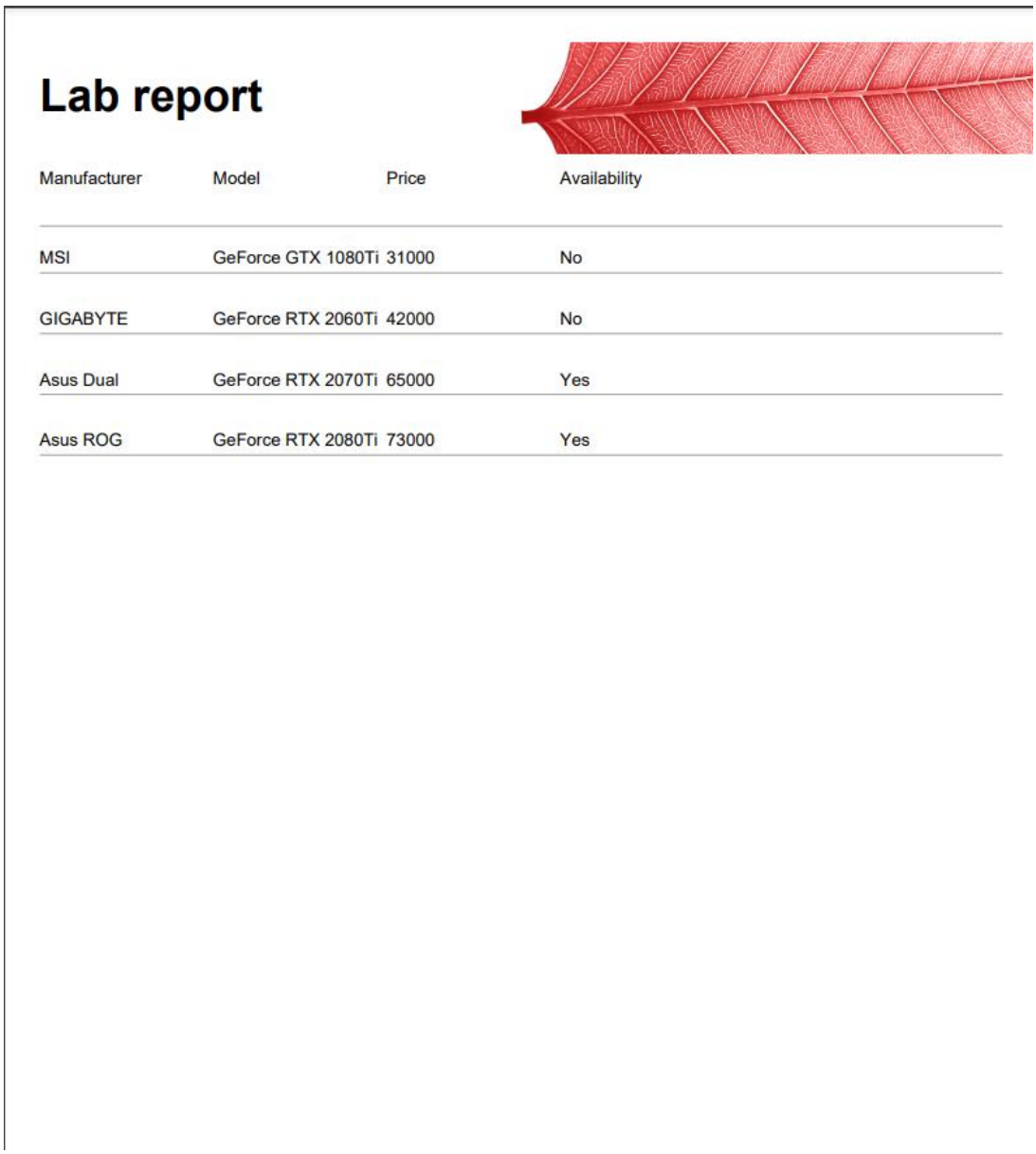


Рисунок 2 – Отредактированный XML-файл

2. Скриншот построенного отчёта представлен на рисунке 3.



Lab report

Manufacturer	Model	Price	Availability
MSI	GeForce GTX 1080Ti	31000	No
GIGABYTE	GeForce RTX 2060Ti	42000	No
Asus Dual	GeForce RTX 2070Ti	65000	Yes
Asus ROG	GeForce RTX 2080Ti	73000	Yes

Рисунок 3 – Построенный отчёт

3. Полный текст документации, сгенерированной Javadoc, приложен в папке с кодом.

4. Фрагменты кода, отвечающие за организацию параллельной работы, представлены в приложении А. Исходный код программы приложен в папке.

Выводы.

В результате выполнения лабораторной работы была освоена технология написания параллельных приложений на языке Java.

ПРИЛОЖЕНИЕ А
ФУНКЦИИ, ОТВЕЧАЮЩИЕ ЗА ОРГАНИЗАЦИЮ ПАРАЛЛЕЛЬНОЙ РАБОТЫ

```
public class TableFillerThread implements Runnable {
    private String filename;
    private DefaultTableModel model;

    public TableFillerThread(String filename, DefaultTableModel model) {
        if (!filename.endsWith(".xml")) {
            throw new RuntimeException("The .xml file expected!");
        }

        if (!Files.exists(Paths.get(filename))) {
            throw new RuntimeException("The xml file doesn't exists!");
        }

        this.filename = filename;
        this.model = model;
    }

    @Override
    public void run() {
        synchronized (PriceList.mutexPriceList) {
            try {
                Thread.sleep(10);
                System.out.println("TableFillerThread is running");
                DocumentBuilder builder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
                Document doc = builder.parse(new File(filename));
                doc.getDocumentElement().normalize();
                NodeList GPUList = doc.getElementsByTagName("GPU");
                for (int i = 0; i < GPUList.getLength(); ++i) {
                    Node element = GPUList.item(i);
                    NamedNodeMap attributes = element.getAttributes();
```

```
                    String[] rowsData = {
                        attributes.getNamedItem("manufacturer").getNodeValue(),
                        attributes.getNamedItem("model").getNodeValue(),
                        attributes.getNamedItem("price").getNodeValue(),
                        attributes.getNamedItem("availability").getNodeValue(),
                    };
                    model.addRow(rowsData);
                }
            } catch (ParserConfigurationException | SAXException | IOException |
InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    public class SaveToFileThread implements Runnable {
        private String xmlFilename;
        private DefaultTableModel model;

        public SaveToFileThread(String xmlFilename, DefaultTableModel model) {
            if (!xmlFilename.endsWith(".xml")) {
                throw new RuntimeException("The .xml file is expected!");
            }

            this.xmlFilename = xmlFilename;
            this.model = model;
        }

        @Override
        public void run() {
            synchronized (PriceList.mutexPriceList) {
                try {
                    Thread.sleep(10);
                    System.out.println("SaveToFileThread is running");
                    DocumentBuilder builder =
```

```
                    DocumentBuilderFactory.newInstance().newDocumentBuilder();
                Document document = builder.newDocument();
                Node pricelistNode = document.createElement("pricelist");
                document.appendChild(pricelistNode);
                for (int i = 0; i < model.getRowCount(); i++) {
                   Element GPU = document.createElement("GPU");
                   pricelistNode.appendChild(GPU);
                   GPU.setAttribute("manufacturer", (String) model.getValueAt(i, 0));
                   GPU.setAttribute("model", (String) model.getValueAt(i, 1));
                   GPU.setAttribute("price", (String) model.getValueAt(i, 2));
                   GPU.setAttribute("availability", (String) model.getValueAt(i, 3));
                }
                Transformer transformer = TransformerFactory.newInstance().newTransformer();
                java.io.FileWriter fileWriter = new FileWriter(xmlFilename);
                transformer.transform(new DOMSource(document), new
StreamResult(fileWriter));

             } catch (ParserConfigurationException | IOException | TransformerException |
InterruptedException e) {
                e.printStackTrace();
             }
          }
       }

       public class CreateHTMLReportThread implements Runnable {
          private ClassLoader classLoader = getClass().getClassLoader();
          private String xmlFilePath;
          private String htmlOutPath;

          public CreateHTMLReportThread(String xmlFilePath, String htmlOutPath) {
             this.xmlFilePath = xmlFilePath;
             this.htmlOutPath = htmlOutPath;
          }
```

```
            @Override
            public void run() {
                synchronized (PriceList.mutexPriceList) {
                    try {
                        Thread.sleep(2 * 1000); // imitates delay between joining mutex
                        System.out.println("CreateHTMLReportThread is running");
                        PriceList.createJasperReport(xmlFilePath, "/pricelist/GPU", "Lab7.jrxml",
htmlOutPath);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
```