

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 ВТ

ОТЧЕТ
по лабораторно-практическим работам № 7 и 8
«Построение отчетов в PDF- и HTML- форматах»
«Организация многопоточных приложений»
по дисциплине «Объектно - ориентированное
программирование на языке Java»

Выполнила Половникова А.С

Факультет КТИ

Группа № 3312

Подпись преподавателя _____

Санкт-Петербург

2024 г

Цель работы

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

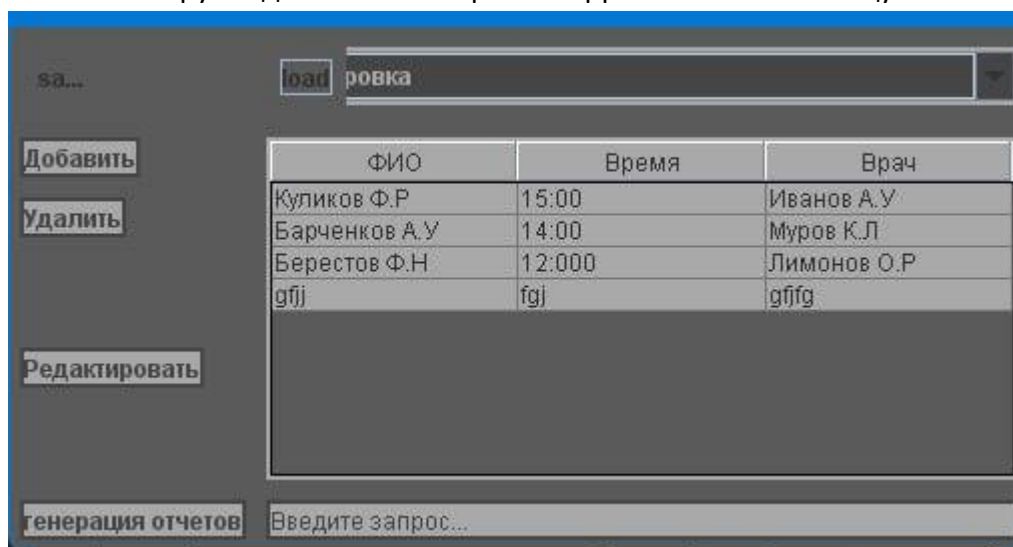
Описание задания

1. Создайте новый проект, который будет дублировать проект лабораторной работы № 7.
2. В новом проекте опишите 3 параллельных потока, один из которых будет загружать данные из XML-файла, второй \square редактировать данные и формировать XML-файл для отчета, а третий \square строить отчет в HTML-формате. Второй поток не должен формировать XML-файл для отчета, пока первый не загрузит данные в экранную форму, а третий поток не должен формировать отчет, пока второй поток редактирует данные и записывает их в XML-файл.
3. С помощью конструктора подготовьте шаблон для отчета.
4. Запустите приложение и убедитесь, что сформирован HTML-файл. Просмотрите его в браузере и проверьте правильность данных и формы.
5. Сгенерируйте документацию с помощью Javadoc и просмотрите ее в браузере.

Описание и проверка работоспособности приложения

При запуске программы пользователь должен будет нажать на поле меню «генерация отчетов», после чего произойдет следующие:

1. Начнется загрузка данных из xml файла “appointmens” в таблицу.



ФИО	Время	Врач
Куликов Ф.Р	15:00	Иванов А.У
Барченков А.У	14:00	Муров К.Л
Берестов Ф.Н	12:000	Лимонов О.Р
gfj	fgj	gfjfg

2. К данным, которые загрузились файла, будет добавлена ещё одна запись (в конце списка).

ФИО	Время	Врач
Куликов Ф.Р	15:00	Иванов А.У
Барченков А.У	14:00	Муров К.Л
Берестов Ф.Н	12:000	Лимонов О.Р
gfjj	fgj	gfjfg
фкурп	рпп	апов

Откроем файл, чтобы убедиться, что он не содержит последнюю строку

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Appointments>
  <Appointment>
    <ФИО>Куликов Ф.Р</ФИО>
    <Время>15:00</Время>
    <Врач>Иванов А.У</Врач>
  </Appointment>
  <Appointment>
    <ФИО>Барченков А.У</ФИО>
    <Время>14:00</Время>
    <Врач>Муров К.Л</Врач>
  </Appointment>
  <Appointment>
    <ФИО>Берестов Ф.Н</ФИО>
    <Время>12:000</Время>
    <Врач>Лимонов О.Р</Врач>
  </Appointment>
  <Appointment>
    <ФИО>gfjj</ФИО>
    <Время>fgj</Время>
    <Врач>gfjfg</Врач>
  </Appointment>
  <Appointment>
    <ФИО>фкурп</ФИО>
    <Время>рпп</Время>
    <Врач>апов</Врач>
  </Appointment>
</Appointments>
```

3. Вся данные из таблицы сохраняются в pdf файле и html файле

Dynamic Report

Куликов Ф.Р	15:00	Иванов А.У
Барченков А.У	14:00	Муров К.Л
Берестов Ф.Н	12:000	Лимонов О.Р
gfjj	fgj	gfjfg
фкурп	рппп	апов

Dynamic Report

.	15:00	.
.	14:00	.
.	12:000	.
gfjj	fgj	gfjfg
gfj	dgj	vcn

Текст документации Javadoc

The screenshot shows the Eclipse IDE with the Javadoc for the `Main` class in the `lib5` package. The left sidebar shows the package structure: `lib5` (Module) and `lib5` (Package). The main area displays the class `Main` with its package `lib5` and superclass `java.lang.Object`. The class is described as "открытый класс Main расширяет Object". The version is 2.0 and the author is "Половинкова Алиса 3312".

Краткое описание поля

Модификатор и тип	Поле	Описание
static Object [®] []	columns	
static Object [®] [][]	data	
static JFrame [®]	frame	
static DefaultTableModel [®]	model	
static int	quant	
static JTable [®]	table	

Краткое описание конструктора

Конструкторы	Описание
Конструктор	

The screenshot shows the Eclipse IDE with the Javadoc for the `Main` class, focusing on the methods section. The left sidebar is the same as the previous screenshot. The main area displays the "Краткое описание метода" (Brief description of the method) section. It includes tabs for "Все методы" (All methods), "Статические методы" (Static methods), "Методы экземпляра" (Instance methods), and "Конкретные методы" (Concrete methods). The "Все методы" tab is selected.

Модификатор и тип	Метод	Описание
static void	main(String[] args)	
void	show()	

Методы, унаследованные от класса java.lang.Object[®]

equals[®], getClass[®], hashCode[®], notify[®], notifyAll[®], toString[®], wait[®], wait[®], wait[®]

Подробная информация о поле

рамка

общедоступный статический JFrame[®] фрейм

Модель [®]

общедоступная статическая Таблица по умолчанию[®] Модель

таблица

общедоступный статический JTable[®] таблица

данные

общедоступный статический объект[®][][] данные

Фрагменты кода, отвечающие за организацию параллельной работы трех потоков.

```
package com.proghelp;
```

```
import net.sf.jasperreports.engine.*;
```

```
import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;
```

```
import javax.swing.JTable;
```

```
import javax.swing.table.DefaultTableModel;
```

```
import javax.swing.table.TableModel;
```

```
import net.sf.jasperreports.engine.*;
```

```
import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;
```

```
import javax.swing.*;
```

```
import javax.xml.parsers.DocumentBuilder;
```

```
import javax.xml.parsers.DocumentBuilderFactory;
```

```
import javax.xml.transform.*;
```

```
import javax.xml.transform.dom.DOMSource;
```

```
import javax.xml.transform.stream.StreamResult;
```

```
import javax.xml.parsers.ParserConfigurationException;
```

```
import java.io.File;
```

```
import java.io.FileOutputStream;
```

```
import java.util.*;
```

```
import java.util.concurrent.*;
```

```
import org.w3c.dom.Document;
```

```

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;


public class ReportGenerator {


    public static void generateReport(JTable table)
    {
        ExecutorService executor = Executors.newFixedThreadPool(3);

        CountDownLatch latch1 = new CountDownLatch(1);

        CountDownLatch latch2 = new CountDownLatch(1);


        executor.submit() -> {

            try

            {

                System.out.println("Loading data from XML...");

                loadXMLToTable(table, "aaa.xml", Main.model);

                System.out.println("Data loaded successfully.");


                latch1.countDown();

            }

            catch (Exception e)

            {

                e.printStackTrace();

            }

        });


        executor.submit() -> {

```

```
try
{
    latch1.await();

    System.out.println("Editing data and saving to XML...");
    saveTableToXML(table, "edited_data.xml");
    System.out.println("Data edited and saved successfully.");

    latch2.countDown();
}
catch (Exception e)
{
    e.printStackTrace();
}
});

executor.submit() -> {
    try
    {
        latch2.await();

        System.out.println("Generating HTML report...");
        generateHTMLReport(table);
        System.out.println("HTML report generated successfully.");
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
```


$$\} \\ \}));$$

```
executor.shutdown();
```

}

```
private static void loadXMLToTable(JTable table, String filePath, DefaultTableModel model)
throws Exception
```

 $\{$

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
```

```
DocumentBuilder builder = factory.newDocumentBuilder();
```

```
Document doc = builder.parse(filePath);
```

```
doc.getDocumentElement().normalize();
```

```
NodeList nodeList = doc.getElementsByTagName("Appointment");
```

```
model.setRowCount(0);
```

```
for (int i = 0; i < nodeList.getLength(); i++)
```

$$\{$$

```
Node node = nodeList.item(i);
```

```
if (node.getNodeType() == Node.ELEMENT_NODE)
```

 $\{$

Element element = (Element) node;

```
Object[] rowData = new Object[model.getColumnCount()];
```

```
for (int j = 0; j < model.getColumnCount(); j++)
```

 $\{$

```
String columnName = model.getColumnName(j);
```

```

        Node columnNode = element.getElementsByTagName(columnName).item(0);

        rowData[j] = (columnNode != null) ? columnNode.getTextContent() : "";
    }

    model.addRow(rowData);
}
}
}

```

private static void saveTableToXML(JTable table, String filePath) throws Exception

```

{
    DefaultTableModel model = (DefaultTableModel) table.getModel();

    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

    DocumentBuilder builder = factory.newDocumentBuilder();

    Document doc = builder.newDocument();

    Element root = doc.createElement("Table");
    doc.appendChild(root);

    for (int i = 0; i < model.getRowCount(); i++)
    {
        Element row = doc.createElement("Row");
        root.appendChild(row);

        for (int j = 0; j < model.getColumnCount(); j++)
        {
            Element column = doc.createElement(model洗getColumn洗Name(j));
            column.appendChild(doc.createTextNode(model洗.getValueAt(i, j).toString()));
        }
    }
}

```

```

        row.appendChild(column);
    }
}

TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(doc);

StreamResult result = new StreamResult(new FileOutputStream(filePath));

transformer.transform(source, result);
}

```

```

private static void generateHTMLReport(JTable table)
{
    try
    {
        TableModel model = table.getModel();

        int rowCount = model.getRowCount();

        int columnCount = model.getColumnCount();

        List<Map<String, Object>> dataList = new ArrayList<>();

        for (int i = 0; i < rowCount; i++)
        {
            Map<String, Object> rowData = new HashMap<>();

            for (int j = 0; j < columnCount; j++)
                rowData.put(model.getColumnName(j), model.getValueAt(i, j));

```

```

        dataList.add(rowData);
    }

    JRBeanCollectionDataSource dataSource = new JRBeanCollectionDataSource(dataList);

    String jrxmlFilePath = "dynamic_report.jrxml";
    JRXMLGenerator.generateJRXML(table, jrxmlFilePath);

    JasperReport jasperReport = JasperCompileManager.compileReport(jrxmlFilePath);

    Map<String, Object> parameters = new HashMap<>();
    parameters.put("ReportTitle", "Generated Report");

    JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, parameters,
dataSource);

    JasperExportManager.exportReportToHtmlFile(jasperPrint, "dynamic_report.html");
}
catch (JRException e)
{
    e.printStackTrace();
}
}
}

```

Вывод

Во время выполнения данной лабораторной работы, были получены навыки работы с технологией формирования отчетов с использованием конструктора Jaspersoft iReport Designert, библиотекой iTest и правилами построения параллельных приложений в языке Java. В пункте «Описание и проверки работоспособности приложения» ошибок выявлено не было.

Ссылки

https://github.com/MniAlice/lab_oop