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-практической работе № 10**

**«Протоколирование работы приложения»**

**по дисциплине «Объектно - ориентированное программирование на языке Java»**

Выполнил Половникова А.С.

Факультет КТИ

Группа № 3312

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2024 г

**Цель работы**

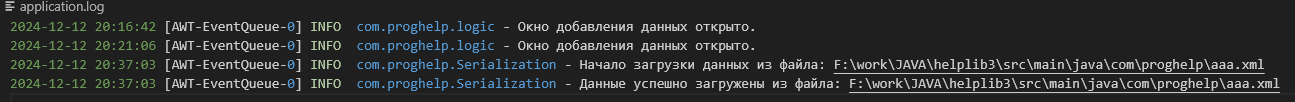
Знакомство c методами протоколирования работы приложения с использования библиотеки Log4j.

**Описание задания**

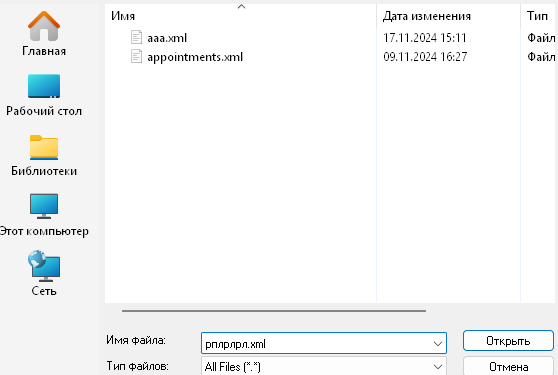
1. Перечень используемых типов сообщений, которые выводятся в логфайл.
2. Конфигурационный файл log4j.properties.
3. Исходные тексты классов, где осуществляется протоколирование работы приложения.
4. Текст документации, сгенерированный Javadoc
5. Фрагменты кода, содержащие WARN, INFO, ERROR и DEBUG

**Описание и проверки работоспособности приложения**

При запуске программы начинается автоматическое протоколирование. Записывается каждое действие пользователя с указанием времени и класса.



Если попробовать открыть не существующий файл.



То в протоколе будет внесено следующие сообщение об ошибки.



**Конфигурационный файл log4j.xml**

<?xml version="1.0" encoding="UTF-8"?>

<Configuration status="WARN">

    <Appenders>

        <File name="File" fileName="logs/application.log">

            <PatternLayout>

                <Pattern>%d{yyyy-MM-dd HH:mm:ss} [%t] %-5level %logger{36} - %msg%n</Pattern>

            </PatternLayout>

        </File>

    </Appenders>

    <Loggers>

        <Logger name="com.proghelp" level="info" additivity="false">

            <AppenderRef ref="File"/>

        </Logger>

        <Root level="debug">

            <AppenderRef ref="File"/>

        </Root>

    </Loggers>

</Configuration>

**Фрагменты кода, содержащие WARN, INFO, ERROR и DEBUG**

package com.proghelp;

import java.awt.\*;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import javax.swing.\*;

/\*\*

 \* @author Половникова Алиса 3312

 \* @version 2.0

 \*/

public class logic

{

    private static final Logger logger = LogManager.getLogger(logic.class);

    private static JFrame addWindow;

    private static void saving(JTextField nameP, JTextField time, JTextField nameD) throws EmptyFieldException

    {

        try

        {

            if (nameP.getText().isEmpty())

                throw new EmptyFieldException("Поле 'ФИО пациента' не может быть пустым.");

            if (time.getText().isEmpty())

                throw new EmptyFieldException("Поле 'Время' не может быть пустым.");

            if (nameD.getText().isEmpty())

                throw new EmptyFieldException("Поле 'Врач' не может быть пустым.");

            Main.model.addRow(new Object[] {nameP.getText(), time.getText(), nameD.getText()});

            Main.quant++;

            logger.info("Добавлена запись: " + nameP.getText() + ", " + time.getText() + ", " + nameD.getText());

        }

        catch (EmptyFieldException e)

        {

            logger.error("Ошибка при сохранении данных: " + e.getMessage(), e);

            JOptionPane.showMessageDialog(null, e.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);

        }

        catch (Exception e)

        {

            logger.error("Неизвестная ошибка: " + e.getMessage(), e);

            JOptionPane.showMessageDialog(null, e.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);

        }

        addWindow.dispose();

    }

    public static void reportGenerateClick()

    {

        ReportGenerator.generateReport(Main.table);

    }

    public static void b1Click()

    {

        addWindow = new JFrame("Добавление");

        addWindow.setResizable(false);

        addWindow.setSize(400, 200);

        addWindow.setLocation(600, 100);

        addWindow.getContentPane().setBackground(new java.awt.Color(90, 90, 90));

        logger.info("Окно добавления данных открыто.");

        JLabel lable1 = new JLabel("Имя пациента");

        lable1.setForeground(Color.WHITE);

        JLabel lable2 = new JLabel("Время");

        lable2.setForeground(Color.WHITE);

        JLabel lable3 = new JLabel("Имя врача");

        lable3.setForeground(Color.WHITE);

        JTextField nameP = new JTextField("");

        nameP.setPreferredSize(new Dimension(300, 25));

        nameP.setBackground(new java.awt.Color(169, 169, 169));

        nameP.setForeground(new java.awt.Color(69, 69, 69));

        nameP.setBorder(BorderFactory.createLineBorder(new java.awt.Color(69, 69, 69), 2));

        JTextField time = new JTextField("");

        time.setPreferredSize(new Dimension(300, 25));

        time.setBackground(new java.awt.Color(169, 169, 169));

        time.setForeground(new java.awt.Color(69, 69, 69));

        time.setBorder(BorderFactory.createLineBorder(new java.awt.Color(69, 69, 69), 2));

        JTextField nameD = new JTextField("");

        nameD.setPreferredSize(new Dimension(300, 25));

        nameD.setBackground(new java.awt.Color(169, 169, 169));

        nameD.setForeground(new java.awt.Color(69, 69, 69));

        nameD.setBorder(BorderFactory.createLineBorder(new java.awt.Color(69, 69, 69), 2));

        JButton save = new JButton("Сохранить");

        save.setPreferredSize(new Dimension(100, 30));

        save.setBackground(new java.awt.Color(169, 169, 169));

        save.setForeground(new java.awt.Color(69, 69, 69));

        save.setBorder(BorderFactory.createLineBorder(new java.awt.Color(69, 69, 69), 2));

        save.addActionListener(ae -> {

            try {

                logic.saving(nameP, time, nameD);

            } catch (EmptyFieldException e) {

                JOptionPane.showMessageDialog(null, e.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);

            }

        });

        addWindow.setLayout(new GridLayout(4, 2));

        addWindow.add(lable1);

        addWindow.add(nameP);

        addWindow.add(lable2);

        addWindow.add(time);

        addWindow.add(lable3);

        addWindow.add(nameD);

        addWindow.add(save);

        addWindow.setVisible(true);

        //GridBagConstraints mainGrid = new GridBagConstraints();

    }

    public static void b2Click(){

        try

        {

            int selectedRow = Main.table.getSelectedRow();

            if (selectedRow != -1)

            {

                Main.model.removeRow(selectedRow);

                JOptionPane.showMessageDialog(Main.frame, "Удаление прошло успешно");

                logger.info("Удалена запись: " + selectedRow);

            }

            else

                throw new Exception("Не выбрана строка для удаления");

        }

        catch (Exception e)

        {

            logger.error("Ошибка при удалении: " + e.getMessage(), e);

            JOptionPane.showMessageDialog(Main.frame, e.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);

        }

    }

}

package com.proghelp;

import java.awt.FileDialog;

import java.io.File;

import org.w3c.dom.\*;

import javax.swing.table.DefaultTableModel;

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.swing.JOptionPane;

import javax.swing.JTable;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

public class Serialization {

    private static final Logger logger = LogManager.getLogger(Serialization.class);

    public static void SaveButtonClick(JTable table)

    {

        FileDialog save = new FileDialog(Main.frame, "Сохранение данных", FileDialog.SAVE);

        save.setFile("\*.xml");

        save.setVisible(true);

        String fileName = save.getDirectory() + save.getFile();

        if(fileName == null)

        {

            logger.warn("Пользователь отменил сохранение файла.");

            return;

        }

        try

        {

            logger.info("Начало сохранения данных в файл: " + fileName);

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

            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

            DocumentBuilder builder = factory.newDocumentBuilder();

            Document doc = builder.newDocument();

            Element rootElement = doc.createElement("Appointments");

            doc.appendChild(rootElement);

            for (int i = 0; i < model.getRowCount(); i++)

            {

                Element appointment = doc.createElement("Appointment");

                rootElement.appendChild(appointment);

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

                {

                    Element column = doc.createElement(model.getColumnName(j));

                    column.appendChild(doc.createTextNode(model.getValueAt(i, j).toString()));

                    appointment.appendChild(column);

                }

            }

            TransformerFactory transformerFactory = TransformerFactory.newInstance();

            Transformer transformer = transformerFactory.newTransformer();

            transformer.setOutputProperty(OutputKeys.INDENT, "yes");

            DOMSource source = new DOMSource(doc);

            StreamResult result = new StreamResult(new File(fileName));

            transformer.transform(source, result);

            logger.info("Данные успешно сохранены в файл: " + fileName);

            JOptionPane.showMessageDialog(null, "Данные успешно сохранены в файл appointments.xml.");

        }

        catch (Exception e)

        {

            logger.error("Ошибка при сохранении данных: " + e.getMessage(), e);

            JOptionPane.showMessageDialog(null, "Ошибка при сохранении данных: " + e.getMessage());

        }

    }

    public static void LoadButtonClick(JTable table, DefaultTableModel model) {

        FileDialog load = new FileDialog(Main.frame, "Загрузка данных", FileDialog.LOAD);

        load.setFile("\*.xml");

        load.setVisible(true);

        String fileName = load.getDirectory() + load.getFile();

        if (fileName == null)

        {

            logger.warn("Пользователь отменил загрузку файла.");

            return;

        }

        try

        {

            logger.info("Начало загрузки данных из файла: " + fileName);

            File file = new File(fileName);

            if (!file.exists())

            {

                logger.error("Файл не найден: " + fileName);

                JOptionPane.showMessageDialog(null, "Файл не найден.");

                return;

            }

            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

            DocumentBuilder builder = factory.newDocumentBuilder();

            Document doc = builder.parse(file);

            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);

                }

            }

            logger.info("Данные успешно загружены из файла: " + fileName);

            JOptionPane.showMessageDialog(null, "Данные успешно загружены из файла appointments.xml.");

        }

        catch (Exception e)

        {

            logger.error("Ошибка при загрузке данных: " + e.getMessage(), e);

            JOptionPane.showMessageDialog(null, "Ошибка при загрузке данных: " + e.getMessage());

        }

    }

}

**Вывод**

Во время выполнения данной лабораторной работы, были получены навыки работы с методами протоколирования приложения с использования библиотеки Log4j. В пункте «Описание и проверки работоспособности приложения» ошибок выявлено не было.