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 МОЭВМ

Лабораторная работа по ООП №10 по теме “Протоколирование работы приложения”

Выполнил: Цацкис Артём гр. 3311

Проверил: Павловский М. Г.

Подпись преподавателя: _____

Санкт-Петербург

2024

Цель работы:

знакомство с методами протоколирования работы при ложения с использованием библиотеки Log4j.

Описание задания:

1. Создайте новый проект, который будет дублировать проект лабораторной работы № 8.

2. Проанализируйте методы в различных потоках приложения и определите основные действия, которые необходимо контролировать. На основе этого анализа опишите конфигурационный файл. 3. Подключите библиотеку Log4j и настройте вывод в лог-файл. 59

4. Организуйте вывод в лог-файл сообщений типа WARN, INFO и DEBUG. В код классов должны быть вставлены комментарии документации, поясняющие смысл выводимой информации.

5. Запустите приложение в различных режимах протоколирования. 6. Сгенерируйте документацию с помощью Javadoc и просмотрите ее в браузере.

Код программы:

```
package lab10;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class OOPlab10 {
    private static final Logger logger = LogManager.getLogger(OOPlab10.class);

    private JFrame bookList;
    private DefaultTableModel model;
    private JButton save, add, edit, delete, load;
    private JScrollPane scroll;
```

```
    private JTable books;

    public void show() {
        Logger.info("Инициализация GUI");
        bookList = new JFrame("Информация о книгах");
        bookList.setSize(600, 400);
        bookList.setLocation(100, 100);
        bookList.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        String[] columns = {"Название книги", "Автор", "Шрифт", "Закреплена?"};
        String[][] data = {
            {"Война и мир", "Лев Толстой", "Arial", "Нет"},
            {"1984", "Джордж Оруэлл", "Calibri", "Да"},
            {"Прощай оружие!", "Эрнест Хемингуэй", "Garamond", "Нет"},
            {"Убить пересмешника", "Харпер Ли", "Fraktur", "Да"},
            {"На дороге", "Джек Керуак", "Papyrus", "Нет"}
        };
        model = new DefaultTableModel(data, columns);
        books = new JTable(model);
        books.setAutoCreateRowSorter(true);
        scroll = new JScrollPane(books);
        bookList.getContentPane().add(scroll, BorderLayout.CENTER);

        JPanel buttonPanel = new JPanel();
        save = new JButton("Сохранить");
        load = new JButton("Загрузить");
        add = new JButton("Добавить");
        edit = new JButton("Редактировать");
        delete = new JButton("Удалить");

        buttonPanel.add(save);
        buttonPanel.add(load);
        buttonPanel.add(add);
        buttonPanel.add(edit);
        buttonPanel.add(delete);
        bookList.getContentPane().add(buttonPanel, BorderLayout.SOUTH);

        save.addActionListener(e -> {
            logger.info("Сохранение данных в XML");
            saveToXMLFile();
            JOptionPane.showMessageDialog(bookList, "Данные сохранены.");
        });

        load.addActionListener(e -> {
            logger.info("Загрузка данных из XML");
            loadFromXMLFile();
            JOptionPane.showMessageDialog(bookList, "Данные загружены.");
        });

        add.addActionListener(e -> {
            JTextField titleField = new JTextField();
            JTextField authorField = new JTextField();
            JTextField fontField = new JTextField();
            String[] options = {"Да", "Нет"};
            JComboBox<String> pinnedField = new JComboBox<>(options);

            JPanel panel = new JPanel(new GridLayout(0, 1));
```

```
            panel.add(new JLabel("Название книги:"));
            panel.add(titleField);
            panel.add(new JLabel("Автор:"));
            panel.add(authorField);
            panel.add(new JLabel("Шрифт:"));
            panel.add(fontField);
            panel.add(new JLabel("Закреплена?"));
            panel.add(pinnedField);

            int result = JOptionPane.showConfirmDialog(bookList, panel, "Добавить
новую книгу",
                    JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);
            if (result == JOptionPane.OK_OPTION) {
                String title = titleField.getText().trim();
                String author = authorField.getText().trim();
                String font = fontField.getText().trim();
                String pinned = (String) pinnedField.getSelectedItem();

                if (!title.isEmpty() && !author.isEmpty() && !font.isEmpty()) {
                    model.addRow(new Object[]{title, author, font, pinned});
                    logger.info("Книга добавлена: " + title);
                }
            }
        });

        edit.addActionListener(e -> {
            int rowIndex = books.getSelectedRow();
            if (rowIndex == -1) {
                JOptionPane.showMessageDialog(bookList, "Выберите книгу для
редактирования.");
                return;
            }
            String title = (String) model.getValueAt(rowIndex, 0);
            String author = (String) model.getValueAt(rowIndex, 1);
            String font = (String) model.getValueAt(rowIndex, 2);
            String pinned = (String) model.getValueAt(rowIndex, 3);

            JTextField titleField = new JTextField(title);
            JTextField authorField = new JTextField(author);
            JTextField fontField = new JTextField(font);
            JComboBox<String> pinnedField = new JComboBox<>(new String[]{"Да",
"Нет"});
            pinnedField.setSelectedItem(pinned);

            JPanel panel = new JPanel(new GridLayout(0, 1));
            panel.add(new JLabel("Название книги:"));
            panel.add(titleField);
            panel.add(new JLabel("Автор:"));
            panel.add(authorField);
            panel.add(new JLabel("Шрифт:"));
            panel.add(fontField);
            panel.add(new JLabel("Закреплена?"));
            panel.add(pinnedField);

            int result = JOptionPane.showConfirmDialog(bookList, panel,
"Редактировать книгу",
                    JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);
```

```
            if (result == JOptionPane.OK_OPTION) {
                model.setValueAt(titleField.getText().trim(), rowIndex, 0);
                model.setValueAt(authorField.getText().trim(), rowIndex, 1);
                model.setValueAt(fontField.getText().trim(), rowIndex, 2);
                model.setValueAt(pinnedField.getSelectedItem(), rowIndex, 3);
                logger.info("Книга отредактирована: " + titleField.getText());
            }
        });

        delete.addActionListener(e -> {
            int rowIndex = books.getSelectedRow();
            if (rowIndex == -1) {
                JOptionPane.showMessageDialog(bookList, "Выберите книгу для
удаления.");
                return;
            }
            String title = (String) model.getValueAt(rowIndex, 0);
            model.removeRow(rowIndex);
            logger.info("Книга удалена: " + title);
        });

        bookList.setVisible(true);
    }

    private void saveToXMLFile() {
        // Логика сохранения в XML
    }

    private void loadFromXMLFile() {
        // Логика загрузки из XML
    }

    public static void main(String[] args) {
        new OOPlab10().show();
    }
}
```

Dataeditor:

```
package lab10;

import java.io.File;
import java.util.concurrent.CountDownLatch;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.w3c.dom.*;

public class DataEditor implements Runnable {
    private static final Logger logger = LogManager.getLogger(DataEditor.class);
    private CountDownLatch latch1, latch2;

    public DataEditor(CountDownLatch latch1, CountDownLatch latch2) {
```

```
        this.latch1 = latch1;
        this.latch2 = latch2;
    }

    @Override
    public void run() {
        Logger.info("Начато редактирование данных");
        try {
            latch1.await();
            File xmlFile = new File("books.xml");
            Document doc =
DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(xmlFile);
            doc.getDocumentElement().normalize();

            Element root = doc.getDocumentElement();
            Element newBook = doc.createElement("Book");

            Element title = doc.createElement("Title");
            title.appendChild(doc.createTextNode("Новая книга"));
            newBook.appendChild(title);

            root.appendChild(newBook);

            Transformer transformer =
TransformerFactory.newInstance().newTransformer();
            transformer.setOutputProperty(OutputKeys.INDENT, "yes");
            transformer.transform(new DOMSource(doc), new StreamResult(xmlFile));

            Logger.info("Данные успешно отредактированы");
            latch2.countDown();
        } catch (Exception e) {
            Logger.error("Ошибка при редактировании данных", e);
        }
    }
}
```

dataloader:

```
package lab10;

import java.io.File;
import java.util.concurrent.CountDownLatch;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.w3c.dom.*;

public class DataLoader implements Runnable {
    private static final Logger logger = LogManager.getLogger(DataLoader.class);
    private CountDownLatch latch;

    public DataLoader(CountDownLatch latch) {
        this.latch = latch;
    }

    @Override
```

```
    public void run() {
        Logger.info("Начата загрузка данных из XML");
        try {
            File xmlFile = new File("books.xml");
            if (!xmlFile.exists()) {
                Logger.warn("Файл XML не найден");
                latch.countDown();
                return;
            }

            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document doc = builder.parse(xmlFile);

            doc.getDocumentElement().normalize();
            NodeList nodeList = doc.getElementsByTagName("Book");

            for (int i = 0; i < nodeList.getLength(); i++) {
                Node node = nodeList.item(i);
                if (node.getNodeType() == Node.ELEMENT_NODE) {
                    Element element = (Element) node;
                    Logger.debug("Загружена книга: " +
element.getElementsByTagName("Title").item(0).getTextContent());
                }
            }
            latch.countDown();
        } catch (Exception e) {
            Logger.error("Ошибка при загрузке данных", e);
        }
    }
}
```

Ссылка на репозиторий Git-hub:

TsatskisArtem/For-OOP10

OOPlab10.java-код программы

Doc- папка с документацией сгенерированная javadocs

Tsatskis-lab10.pdf-отчет по лабораторной работе