

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

**ОТЧЕТ**  
**по лабораторно-практической работе № 10**  
**«Протоколирование работы приложения»**  
**по дисциплине «Объектно - ориентированное**  
**программирование на языке Java»**

Выполнил Сапронов К.Д.

Факультет КТИ

Группа № 3311

Подпись преподавателя \_\_\_\_\_

Санкт-Петербург

2024 г

## Цель работы

Знакомство с методами протоколирования работы приложения с использованием библиотеки Log4j.

## Описание задания

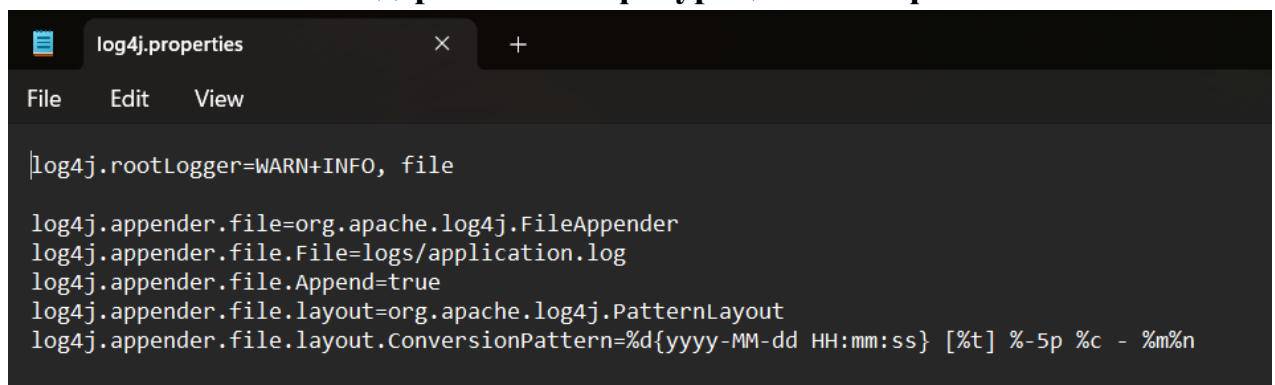
1. Проанализировать классы приложения из л/р №8 и определить, выполнение каких действий необходимо контролировать.
2. Описать конфигурационный файл, подключить библиотеку Log4j
3. Организовать вывод в лог файл сообщений

## Разработанные методы

Так как в работе класс GUI является основным, то будем работать с ним. Проконтролируем следующие действия:

1. Создание экранной формы – успешно создание экранной формы, для режима DEBUG также добавление слушателей
2. Загрузка данных из файла – нажатие пользователем кнопки Open, начало и конец загрузки данных
3. Сохранение данных в файл - нажатие пользователем кнопки Save, начало и конец сохранения данных
4. Также добавлено логирование других действий типа INFO и логирование исключительных ситуаций и ошибок типами WARN и ERROR

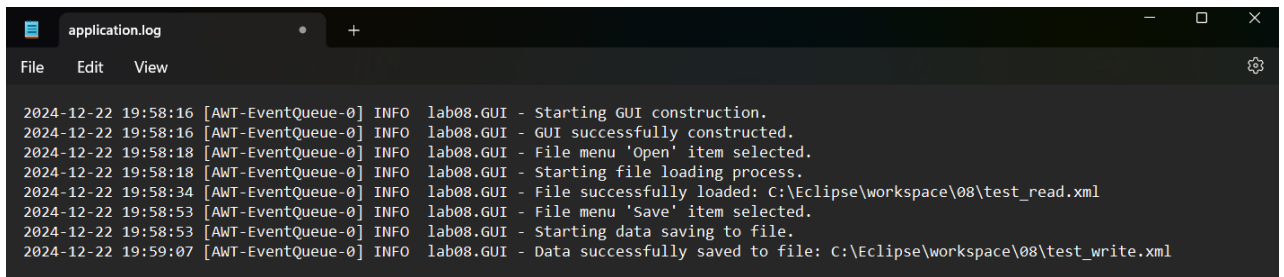
## Содержание конфигурационного файла

A screenshot of a text editor window titled 'log4j.properties'. The window has a menu bar with 'File', 'Edit', and 'View'. The text content is as follows:

```
log4j.rootLogger=WARN+INFO, file  
  
log4j.appender.file=org.apache.log4j.FileAppender  
log4j.appender.file.File=logs/application.log  
log4j.appender.file.Append=true  
log4j.appender.file.layout=org.apache.log4j.PatternLayout  
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} [%t] %-5p %c - %m%n
```

## Содержимое лог-файла

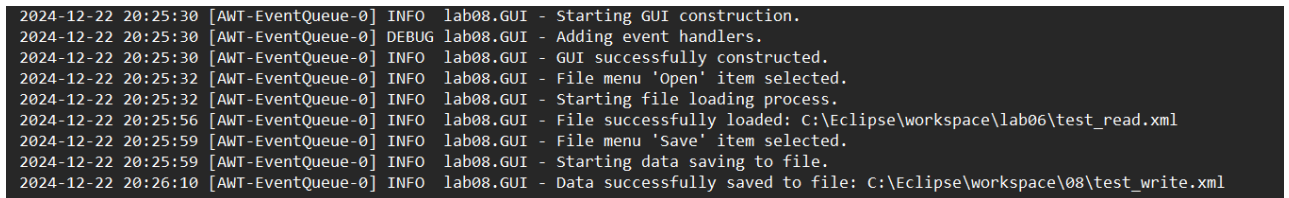
## 1. Режим WARN+INFO



```
application.log
File Edit View

2024-12-22 19:58:16 [AWT-EventQueue-0] INFO lab08.GUI - Starting GUI construction.
2024-12-22 19:58:16 [AWT-EventQueue-0] INFO lab08.GUI - GUI successfully constructed.
2024-12-22 19:58:18 [AWT-EventQueue-0] INFO lab08.GUI - File menu 'Open' item selected.
2024-12-22 19:58:18 [AWT-EventQueue-0] INFO lab08.GUI - Starting file loading process.
2024-12-22 19:58:34 [AWT-EventQueue-0] INFO lab08.GUI - File successfully loaded: C:\Eclipse\workspace\08\test_read.xml
2024-12-22 19:58:53 [AWT-EventQueue-0] INFO lab08.GUI - File menu 'Save' item selected.
2024-12-22 19:58:53 [AWT-EventQueue-0] INFO lab08.GUI - Starting data saving to file.
2024-12-22 19:59:07 [AWT-EventQueue-0] INFO lab08.GUI - Data successfully saved to file: C:\Eclipse\workspace\08\test_write.xml
```

## 2. Режим DEBUG



```
2024-12-22 20:25:30 [AWT-EventQueue-0] INFO lab08.GUI - Starting GUI construction.
2024-12-22 20:25:30 [AWT-EventQueue-0] DEBUG lab08.GUI - Adding event handlers.
2024-12-22 20:25:30 [AWT-EventQueue-0] INFO lab08.GUI - GUI successfully constructed.
2024-12-22 20:25:32 [AWT-EventQueue-0] INFO lab08.GUI - File menu 'Open' item selected.
2024-12-22 20:25:32 [AWT-EventQueue-0] INFO lab08.GUI - Starting file loading process.
2024-12-22 20:25:56 [AWT-EventQueue-0] INFO lab08.GUI - File successfully loaded: C:\Eclipse\workspace\lab06\test_read.xml
2024-12-22 20:25:59 [AWT-EventQueue-0] INFO lab08.GUI - File menu 'Save' item selected.
2024-12-22 20:25:59 [AWT-EventQueue-0] INFO lab08.GUI - Starting data saving to file.
2024-12-22 20:26:10 [AWT-EventQueue-0] INFO lab08.GUI - Data successfully saved to file: C:\Eclipse\workspace\08\test_write.xml
```

## Ссылки

[https://drive.google.com/drive/folders/1SkmitiMaArA7aWjd8Q5cThXVNP8ws\\_N4?usp=drive\\_link](https://drive.google.com/drive/folders/1SkmitiMaArA7aWjd8Q5cThXVNP8ws_N4?usp=drive_link)

В этой папке будут находиться все лабораторные работы

В папке lab10 находятся этот отчет, видеоотчет и папка 08, в которой находятся файлы проекта и документация javadoc.

## Текст класса

```
package lab08;

import org.apache.log4j.*;
import org.w3c.dom.*;
import javax.swing.*;
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 java.awt.*;
import java.io.File;
import java.io.FileWriter;

public class GUI {
    private static final Logger log = Logger.getLogger(GUI.class);

    private JFrame frame;
    private JMenuBar menuBar;
    private JMenu fileMenu, sortMenu;
    private JMenuItem openItem, saveItem, roomItem, nameItem;
    private JToolBar toolBar;
    private JButton addButton, deleteButton, searchButton;
    private JComboBox<String> searchType;
    private JTextField searchField;
    private JTable dataTable;
    private JScrollPane tableScrollPane;
    private DefaultTableModel tableModel;

    /**
     * Create and display the graphical interface of the application.
     */
    public void buildAndShowGUI() {
        log.info("Starting GUI construction.");
        frame = new JFrame("Hotel - Guest List");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 600);

        // Menu
        menuBar = new JMenuBar();
        fileMenu = new JMenu("File");
        openItem = new JMenuItem("Open");
        saveItem = new JMenuItem("Save");
        fileMenu.add(openItem);
        fileMenu.add(saveItem);
        menuBar.add(fileMenu);
```

```

// Sort menu
sortMenu = new JMenu("Sort by");
roomItem = new JMenuItem("Room");
nameItem = new JMenuItem("Name");
sortMenu.add(roomItem);
sortMenu.add(nameItem);
menuBar.add(sortMenu);

frame.setJMenuBar(menuBar);

// Toolbar
toolBar = new JToolBar();
addButton = new JButton("Add");
deleteButton = new JButton("Delete");
toolBar.add(addButton);
toolBar.add(deleteButton);
frame.add(toolBar, BorderLayout.NORTH);

// Search panel
JPanel searchPanel = new JPanel();
searchType = new JComboBox<>(new String[]{"Name", "Room", "Date"});
searchField = new JTextField(15);
searchButton = new JButton("Search");
searchPanel.add(new JLabel("Search by:"));
searchPanel.add(searchType);
searchPanel.add(searchField);
searchPanel.add(searchButton);
frame.add(searchPanel, BorderLayout.SOUTH);

// Table
String[] columns = {"Name", "Room", "Check-in Date", "Check-out Date"};
tableModel = new DefaultTableModel(columns, 0);
dataTable = new JTable(tableModel);
tableScrollPane = new JScrollPane(dataTable);
frame.add(tableScrollPane, BorderLayout.CENTER);

addListeners();
log.info("GUI successfully constructed.");
frame.setVisible(true);
}

/**
 * Add event handlers to GUI elements.
 */
private void addListeners() {
    log.debug("Adding event handlers.");
    saveItem.addActionListener(e -> {
        log.info("File menu 'Save' item selected.");
        saveDataToFile();
    });

    openItem.addActionListener(e -> {
        log.info("File menu 'Open' item selected.");
        loadDataFromFile();
    });

    addButton.addActionListener(e -> {
        log.debug("Add button clicked.");
        String name = JOptionPane.showInputDialog("Enter guest name:");
        String room = JOptionPane.showInputDialog("Enter room number:");
        String checkIn = JOptionPane.showInputDialog("Enter check-in date
(dd.mm.yyyy):");
        String checkOut = JOptionPane.showInputDialog("Enter check-out date
(dd.mm.yyyy):");

        try {

```

```

        validateGuestInput(name, room, checkIn, checkOut);
        tableModel.addRow(new Object[]{name, room, checkIn, checkOut});
        log.info(String.format("Added guest: %s, room: %s.", name, room));
    } catch (EmptyFieldException ex) {
        log.warn("Empty field encountered when adding guest.", ex);
        JOptionPane.showMessageDialog(frame, ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
});

deleteButton.addActionListener(e -> {
    log.debug("Delete button clicked.");
    int selectedRow = dataTable.getSelectedRow();
    if (selectedRow != -1) {
        log.info(String.format("Deleting row with index %d.", selectedRow));
        tableModel.removeRow(selectedRow);
    } else {
        log.warn("Attempted deletion without selecting a row.");
    }
});
}

/**
 * Save table data to XML file.
 */
private void saveDataToFile() {
    log.info("Starting data saving to file.");
    JFileChooser fileChooser = new JFileChooser();
    if (fileChooser.showSaveDialog(frame) == JFileChooser.APPROVE_OPTION) {
        File file = fileChooser.getSelectedFile();
        try {
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.newDocument();

            // Create XML
            Element rootElement = doc.createElement("Guests");
            doc.appendChild(rootElement);

            for (int i = 0; i < tableModel.getRowCount(); i++) {
                Element guest = doc.createElement("Guest");
                guest.setAttribute("Name", (String) tableModel.getValueAt(i, 0));
                guest.setAttribute("Room", (String) tableModel.getValueAt(i, 1));
                guest.setAttribute("CheckIn", (String) tableModel.getValueAt(i, 2));
                guest.setAttribute("CheckOut", (String) tableModel.getValueAt(i, 3));
                rootElement.appendChild(guest);
            }

            TransformerFactory transformerFactory = TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            transformer.setOutputProperty(OutputKeys.INDENT, "yes");
            DOMSource source = new DOMSource(doc);
            StreamResult result = new StreamResult(new FileWriter(file));
            transformer.transform(source, result);

            log.info("Data successfully saved to file: " + file.getAbsolutePath());
            JOptionPane.showMessageDialog(frame, "Data saved successfully!");
        } catch (Exception ex) {
            log.error("Error saving file.", ex);
            JOptionPane.showMessageDialog(frame, "Error saving file: " +
ex.getMessage());
        }
    }
}

/**

```

```

        * Load data from XML file into the table.
        */
private void loadDataFromFile() {
    log.info("Starting file loading process.");
    JFileChooser fileChooser = new JFileChooser();
    if (fileChooser.showOpenDialog(frame) == JFileChooser.APPROVE_OPTION) {
        File file = fileChooser.getSelectedFile();
        try {
            DocumentBuilder dBuilder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
            Document doc = dBuilder.parse(file);
            doc.getDocumentElement().normalize();

            tableModel.setRowCount(0);

            NodeList guestList = doc.getElementsByTagName("Guest");
            for (int i = 0; i < guestList.getLength(); i++) {
                Node guestNode = guestList.item(i);
                if (guestNode.getNodeType() == Node.ELEMENT_NODE) {
                    NamedNodeMap attributes = guestNode.getAttributes();
                    String name = attributes.getNamedItem("Name").getNodeValue();
                    String room = attributes.getNamedItem("Room").getNodeValue();
                    String checkIn =
attributes.getNamedItem("CheckIn").getNodeValue();
                    String checkOut =
attributes.getNamedItem("CheckOut").getNodeValue();

                    tableModel.addRow(new String[]{name, room, checkIn, checkOut});
                }
            }

            log.info("File successfully loaded: " + file.getAbsolutePath());
            JOptionPane.showMessageDialog(frame, "File loaded successfully!");

        } catch (Exception ex) {
            log.error("Error loading file.", ex);
            JOptionPane.showMessageDialog(frame, "Error opening file: " +
ex.getMessage());
        }
    }

    /**
     * Guest input validation.
     */
    private void validateGuestInput(String name, String room, String checkIn, String
checkOut) throws EmptyFieldException {
        if (name.isEmpty() || room.isEmpty() || checkIn.isEmpty() || checkOut.isEmpty())
        {
            log.warn("Empty field encountered in guest input.");
            throw new EmptyFieldException("All fields are required!");
        }
    }
}

```