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

ОТЧЕТ

по лабораторной работе № 10

по дисциплине «Объектно-ориентированное программирование»

Тема: Протоколирование работы приложения

Студентка гр. 2308 _____ Рымарь М.И.

Преподаватель _____ Павловский М.Г.

Санкт-Петербург

2023

Цель работы.

Познакомиться с методами протоколирования работы приложения с использованием библиотеки Log4j.

Задание.

Отчет по лабораторной работе должен содержать:

1. Перечень используемых типов сообщений, которые выводятся в лог-файл.

2. Конфигурационный файл log4j.properties.

3. Лог-файлы работы приложения в режимах WARN+INFO и DEBUG.

4. Исходные тексты классов, где осуществляется протоколирование работы приложения.

5. Текст документации, сгенерированный Javadoc.

Выполнение работы.

1. Перечень используемых типов сообщений, которые выводятся в лог-файл.

Программа записывает все события, происходящие в ней, как создание интерфейса, так и взаимодействие пользователя с ним. Это представлено на рисунке 1. Также программа уведомляет об ошибках, это показано на рисунке 2. Пример лог-файла представлен на рисунке 3.

```
/* Размещение окна с доп. информацией */
log.info("Размещение окна с доп. информацией");
priceList.add(additionalInformation, BorderLayout.EAST);
priceList.add(new JScrollPane(additionalInformation), BorderLayout.EAST);

/* Добавление слушателей */
log.info("Добавление слушателей");
```

Рисунок 1 – Создание интерфейса, взаимодействие пользователя с ним

```java
public class OpenButtonListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        try {
            openFile();
        } catch (FileNotFoundException FNFex) {
            log.error("Ошибка открытия файла");
            FNFex.printStackTrace();
        } catch (IOException IOex) {
            log.error("Ошибка открытия файла");
            IOex.printStackTrace();
        } catch (ParserConfigurationException PCEex) {
            log.error("Ошибка открытия файла");
            PCEex.printStackTrace();
        } catch (SAXException SAXex) {
            log.error("Ошибка открытия файла");
            SAXex.printStackTrace();
        }
    }
}
```

Рисунок 2 – Уведомление об ошибках

myproject.log - Notepad

File Edit Format View Help

```
22:06:21,944  INFO main class:main:12 - Запуск приложения
22:06:21,972  INFO main class:show:36 - Открытие экранной формы
22:06:21,972  INFO main class:creatingInterface:46 - Создание графического окна
22:06:22,344  INFO main class:creatingDataTable:142 - Создание таблицы данных
22:06:27,124  INFO AWT-EventQueue-0 class:lambda$openButtonAction$5:331 - Открытие файла
22:06:42,026 ERROR AWT-EventQueue-0 class:lambda$openButtonAction$5:353 - Ошибка открытия файла
java.io.FileNotFoundException: C:\Users\tsusa\IdeaProjects\EducationProjectOOP\csv\rrrrrrrr (Не удается найти указанный файл)
        at java.base/java.io.FileInputStream.open0(Native Method)
        at java.base/java.io.FileInputStream.open(FileInputStream.java:216)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:157)
        at java.base/java.io.FileInputStream.<init>(FileInputStream.java:111)
        at java.base/java.io.FileReader.<init>(FileReader.java:60)
        at Lab10.Action.lambda$openButtonAction$5(Action.java:341)
        at java.desktop/javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:1972)
        at java.desktop/javax.swing.AbstractButton$Handler.actionPerformed(AbstractButton.java:2313)
        at java.desktop/javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:405)
        at java.desktop/javax.swing.DefaultButtonModel.setPressed(DefaultButtonModel.java:262)
        at java.desktop/javax.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:279)
        at java.desktop/java.awt.AWTEventMulticaster.mouseReleased(AWTEventMulticaster.java:297)
        at java.desktop/java.awt.Component.processMouseEvent(Component.java:6626)
        at java.desktop/javax.swing.JComponent.processMouseEvent(JComponent.java:3389)
        at java.desktop/java.awt.Component.processEvent(Component.java:6391)
        at java.desktop/java.awt.Container.processEvent(Container.java:2266)
        at java.desktop/java.awt.Component.dispatchEventImpl(Component.java:5001)
        at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2324)
        at java.desktop/java.awt.Component.dispatchEvent(Component.java:4833)
        at java.desktop/java.awt.LightweightDispatcher.retargetMouseEvent(Container.java:4948)
        at java.desktop/java.awt.LightweightDispatcher.processMouseEvent(Container.java:4575)
        at java.desktop/java.awt.LightweightDispatcher.dispatchEvent(Container.java:4516)
```

Рисунок 3 – Пример лог-файла

2. Исходные тексты классов, где осуществляется контролирование приложения, и конфиг-файл представлены в приложении А. Исходный код всей программы приложен в папке.

3. Лог-файлы приложены в папке. Пример лог-файла находится в п.1.

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

Выводы.

В результате выполнения лабораторной работы были освоены методы протоколирования работы приложения с использованием библиотеки Log4j.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЕ ТЕКСТЫ КЛАССОВ, ОТВЕЧАЮЩИХ ЗА КОНТРОЛИРОВАНИЕ РАБОТЫ ПРИЛОЖЕНИЯ

```
package org.example;
/* Подключение библиотек */
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import net.sf.jasperreports.engine.*;
import net.sf.jasperreports.engine.data.JRXmlDataSource;
import net.sf.jasperreports.engine.export.JRPdfExporter;
import org.w3c.dom.*;
import org.xml.sax.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.HashMap;
import org.apache.log4j.Logger;

public class PriceList {
    private static final Logger log = Logger.getLogger("Interface.class");
    /* Объявления графических компонентов */
    private JFrame priceList;
    private DefaultTableModel priceListModel;
    private JButton saveButton;
    private JButton openButton;
    private JButton addButton;
    private JButton removeButton;
    private JButton editButton;
```

```
    private JLabel searchByLabel;
    private JComboBox searchBy;
    private JTextField searchInformation;
    private JButton searchButton;
    private JToolBar toolBar;
    private JTable GPUTable;
    private TableRowSorter GPUtableFilter;
    private JScrollPane tableScrollBar;
    private JTextArea additionalInformation;
    private boolean isTableEditable;
    public static final Object mutexPriceList = new Object();

    /**
     * Creates a JFrame window and sets its size
     * Creates save, open, add, remove, edit buttons and adds them to a toolbar
     * Creates search menu and puts adds it to a toolbar
     * Places a toolbar to the top of a window
     * Creates a table with the information about GPUs and puts it to the center of a window
     * Creates an additional information text field and puts it to the right of a window
     */
    public void show() {
        /* Создание окна */
        log.info("Создание окна");
        priceList = new JFrame("Price list");
        priceList.setSize(800, 600);
        priceList.setLocation(300, 300);
        priceList.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        /* Скейлинг изображений для правильного отображения в кнопках */
        ImageIcon imageIconSave = new ImageIcon(new
ImageIcon("src/main/resources/save.png").getImage().getScaledInstance(20, 20,
Image.SCALE_DEFAULT));
        ImageIcon imageIconOpen = new ImageIcon(new
ImageIcon("src/main/resources/open.png").getImage().getScaledInstance(20, 20,
Image.SCALE_DEFAULT));
```

```
        ImageIcon imageIconAdd = new ImageIcon(new
ImageIcon("src/main/resources/add.png").getImage().getScaledInstance(20, 20,
Image.SCALE_DEFAULT));
        ImageIcon imageIconRemove = new ImageIcon(new
ImageIcon("src/main/resources/remove.png").getImage().getScaledInstance(20, 20,
Image.SCALE_DEFAULT));
        ImageIcon imageIconEdit = new ImageIcon(new
ImageIcon("src/main/resources/edit.png").getImage().getScaledInstance(20, 20,
Image.SCALE_DEFAULT));

        /* Создание кнопок и прикрепление иконок */
        log.info("Создание кнопок");
        saveButton = new JButton(imageIconSave);
        openButton = new JButton(imageIconOpen);
        addButton = new JButton(imageIconAdd);
        removeButton = new JButton(imageIconRemove);
        editButton = new JButton(imageIconEdit);

        /* Настройка подсказок для кнопок */
        log.info("Настройка подсказок для кнопок");
        saveButton.setToolTipText("Save");
        openButton.setToolTipText("Open");
        addButton.setToolTipText("Add");
        removeButton.setToolTipText("Remove");
        editButton.setToolTipText("Edit");

        /* Настройка размера кнопок */
        log.info("Настройка размера кнопок");
        saveButton.setPreferredSize(new Dimension(25, 25));
        openButton.setPreferredSize(new Dimension(25, 25));
        addButton.setPreferredSize(new Dimension(25, 25));
        removeButton.setPreferredSize(new Dimension(25, 25));
        editButton.setPreferredSize(new Dimension(25, 25));

        /* Создание поиска */
```

```
log.info("Создание поиска");
searchByLabel = new JLabel("Search by");
searchBy = new JComboBox(new String[]{"manufacturer", "model"});
searchInformation = new JTextField("Enter information");
searchInformation.setForeground(Color.LIGHT_GRAY);
searchButton = new JButton("Search");

/* Добавление кнопок и поиска на панель инструментов */
log.info("Добавление кнопок и поиска на панель инструментов");
toolBar = new JToolBar("Toolbar");
toolBar.add(saveButton);
toolBar.add(openButton);
toolBar.add(addButton);
toolBar.add(removeButton);
toolBar.add(editButton);
toolBar.add(Box.createHorizontalStrut(12));
toolBar.add(new JSeparator(SwingConstants.VERTICAL));
toolBar.add(Box.createHorizontalStrut(12));
toolBar.add(searchByLabel);
toolBar.add(Box.createHorizontalStrut(3));
toolBar.add(searchBy);
toolBar.add(Box.createHorizontalStrut(3));
toolBar.add(searchInformation);
toolBar.add(Box.createHorizontalStrut(3));
toolBar.add(searchButton);

/* Размещение панели инструментов */
log.info("Размещение панели инструментов");
priceList.setLayout(new BorderLayout());
priceList.add(toolBar, BorderLayout.NORTH);



/* Создание таблицы с данными */
log.info("Создание таблицы с данными");
String[] Columns = {"Manufacturer", "Model", "Price, rub", "Availability"};
```

```
            String[][] Data = {/*{"MSI", "GeForce GTX 1080Ti", "30000", "Yes"},
{"GIGABYTE", "GeForce RTX 2060Ti", "40000", "Yes"},
                {"Palit", "GeForce RTX 2070", "50000", "Yes"}, {"Asus", "GeForce RTX
2070Ti", "55000", "No"},
                {"Asus ROG", "GeForce RTX 2080Ti", "60000", "No"}*/};
            isTableEditable = false;
            priceListModel = new DefaultTableModel(Data, Columns) {
               @Override
               public boolean isCellEditable(int row, int column) {
                  return isTableEditable;
               }
            };
            GPUTable = new JTable(priceListModel);
            GPUTable.setAutoCreateRowSorter(true);
            GPUtableFilter = new TableRowSorter<DefaultTableModel>(priceListModel);
            GPUTable.setRowSorter(GPUtableFilter);
            tableScrollBar = new JScrollPane(GPUTable);
            GPUTable.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

            /* Размещение таблицы с данными */
            log.info("Размещение таблицы с данными");
            priceList.add(tableScrollBar, BorderLayout.CENTER);

            /* Создание окна с доп. информацией */
            log.info("Создание окна с доп. информацией");
            additionalInformation = new JTextArea("Space for additional information");

            /* Размещение окна с доп. информацией */
            log.info("Размещение окна с доп. информацией");
            priceList.add(additionalInformation, BorderLayout.EAST);
            priceList.add(new JScrollPane(additionalInformation), BorderLayout.EAST);

            /* Добавление слушателей */
            log.info("Добавление слушателей");
            OpenButtonListener openButtonListener = new OpenButtonListener();
```

```
            openButton.addMouseListener(openButtonListener);

            SaveButtonListener saveButtonListener = new SaveButtonListener();
            saveButton.addMouseListener(saveButtonListener);

            EditButtonListener editButtonListener = new EditButtonListener();
            editButton.addMouseListener(editButtonListener);

            FocusListener searchInformationFocusListener = new
SearchInformationFocusListener();
            SearchInformationKeyListener searchInformationKeyListener = new
SearchInformationKeyListener();
            searchInformation.addFocusListener(searchInformationFocusListener);
            searchInformation.addKeyListener(searchInformationKeyListener);

            SearchButtonListener searchButtonListener = new SearchButtonListener();
            searchButton.addMouseListener(searchButtonListener);

            RemoveButtonListener removeButtonListener = new RemoveButtonListener();
            removeButton.addMouseListener(removeButtonListener);

            priceList.setVisible(true);
            Thread fillerThread = new Thread(new TableFillerThread("PriceList.xml",
priceListModel));
            fillerThread.start();
        }

        /**
         * Creates a frame for selecting a CSV file and writes GPUTable information to it.
         *
         * @throws IOException                  if there was a problem writing information.
         * @throws ParserConfigurationException if there was a problem configuring a parser.
         * @throws TransformerConfigurationException if there was a problem configuring a
transformer.
         * @throws TransformerException         if there was a problem with a transformer.
```

```
     */
    private void saveFile() throws IOException, ParserConfigurationException,
TransformerConfigurationException, TransformerException {
        log.info("Сохранение данных в файл");
        String fileFormat = "*.xml";
        FileDialog save = new FileDialog(priceList, "Open file", FileDialog.SAVE);
        save.setFile(fileFormat);
        save.setVisible(true);
        String fileName = save.getDirectory() + save.getFile();
        if (!fileName.endsWith(".xml")) {
            fileName += ".xml";
        }
        if (save.getFile() == null) return;
        Thread saveToXmlThread = new Thread(new SaveToFileThread(fileName,
priceListModel));
        saveToXmlThread.start();
        Thread generateReportThread = new Thread(new
CreateHTMLReportThread(fileName, "jasperReport.pdf"));
        try {
            saveToXmlThread.join();
        } catch (InterruptedException e) {
            log.error("Ошибка считывания данных");
            e.printStackTrace();
        }
        generateReportThread.start();
    }

    /**
     * Creates a frame for selecting a CSV file and writes its information to GPUTable.
     *
     * @throws ParserConfigurationException if there was a problem  configuring a parser.
     * @throws SAXException                 if there was a problem parsing a file.
     * @throws IOException                  if there was a problem reading a file.
     */
```

```
    private void openFile() throws ParserConfigurationException, SAXException,
IOException {
        log.info("Считывание данных из файла");
        String fileFormat = "*.xml";
        FileDialog load = new FileDialog(priceList, "Open file", FileDialog.LOAD);
        load.setFile(fileFormat);
        load.setDirectory(load.getDirectory());
        load.setVisible(true);
        String fileName = load.getDirectory() + load.getFile();
        if (load.getFile() == null) return;
        DocumentBuilder dBuilder =
                DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document document = dBuilder.parse(new File(fileName));
        document.getDocumentElement().normalize();
        NodeList nodeListGPUs = document.getElementsByTagName("GPU");
        int rows = priceListModel.getRowCount();
        for (int i = 0; i < rows; i++) priceListModel.removeRow(0);
        for (int i = 0; i < nodeListGPUs.getLength(); i++) {
            Node element = nodeListGPUs.item(i);
            NamedNodeMap attrs = element.getAttributes();
            String manufacturer = attrs.getNamedItem("manufacturer").getNodeValue();
            String model = attrs.getNamedItem("model").getNodeValue();
            String price = attrs.getNamedItem("price").getNodeValue();
            String availability = attrs.getNamedItem("availability").getNodeValue();
            priceListModel.addRow(new String[]{manufacturer, model, price, availability});
        }
    }

    /**
     * Removes a selected row
     *
     * @throws NoRowSelectedException if no row is selected
     */
    private void removeRow() throws NoRowSelectedException {
        log.info("Удаление строки");
```

```
            if (GPUTable.getSelectedRow() == -1) {
                throw new NoRowSelectedException();
            } else
priceListModel.removeRow(GPUtableFilter.convertRowIndexToModel(GPUTable.getSelectedRo
w()));
        }

        /**
         * Performs search in a table with DefaultTableModel in a column selected in
JComboBox searchBy ignoring letter register.
         *
         * @throws SearchFailedException if search did not yield results.
         */
        public void performSearch() throws SearchFailedException {
            log.info("Выполнение поиска");
            if (!searchInformation.getText().equals("Enter information")) {
                RowFilter<DefaultTableModel, Object> rowFilter = null;
                rowFilter = RowFilter.regexFilter("(?i)" + searchInformation.getText(),
searchBy.getSelectedIndex());
                GPUtableFilter.setRowFilter(rowFilter);
                if (GPUtableFilter.getViewRowCount() == 0) {
                    throw new SearchFailedException();
                }
            }
        }

        /* Класс слушателя для кнопки записи информации в файл */
        public class SaveButtonListener extends MouseAdapter {
            public void mousePressed(MouseEvent e) {
                try {
                    saveFile();
                } catch (IOException IOex) {
                    IOex.printStackTrace();
                } catch (ParserConfigurationException PCex) {
                    PCex.printStackTrace();
```

```
        } catch (TransformerConfigurationException TCex) {
            TCex.printStackTrace();
        } catch (TransformerException Tex) {
            Tex.printStackTrace();
        }
    }
}

/* Класс слушателя для кнопки открытия файла */
public class OpenButtonListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        try {
            openFile();
        } catch (FileNotFoundException FNFex) {
            log.error("Ошибка открытия файла");
            FNFex.printStackTrace();
        } catch (IOException IOex) {
            log.error("Ошибка открытия файла");
            IOex.printStackTrace();
        } catch (ParserConfigurationException PCEex) {
            log.error("Ошибка открытия файла");
            PCEex.printStackTrace();
        } catch (SAXException SAXex) {
            log.error("Ошибка открытия файла");
            SAXex.printStackTrace();
        }
    }
}

/* Класс слушателя для кнопки удаления строки */
public class RemoveButtonListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        try {
            removeRow();
        } catch (NoRowSelectedException NRWex) {
```

```
                log.error("Ошибка удаления строки");
                JOptionPane.showMessageDialog(null, NRWex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    /* Класс слушателя для кнопки изменения */
    public class EditButtonListener extends MouseAdapter {
        public void mousePressed(MouseEvent e) {
            ImageIcon editInactive = new ImageIcon(new
ImageIcon("src/main/resources/edit.png").getImage().getScaledInstance(20, 20,
Image.SCALE_DEFAULT));
            ImageIcon editActive = new ImageIcon(new
ImageIcon("src/main/resources/editActive.png").getImage().getScaledInstance(20, 20,
Image.SCALE_DEFAULT));
            isTableEditable = !isTableEditable;
            if (isTableEditable) {
                editButton.setBackground(Color.BLACK);
                editButton.setIcon(editActive);
            } else {
                editButton.setBackground(Color.WHITE);
                editButton.setIcon(editInactive);
            }
        }
    }


    /* Класс слушателя для строки поиска */
    private class SearchInformationFocusListener implements FocusListener {
        public void focusGained(FocusEvent e) {
            if (searchInformation.getText().equals("Enter information")) {
                searchInformation.setText("");
                searchInformation.setForeground(Color.BLACK);
            }
```

```
    }

    public void focusLost(FocusEvent e) {
        if (searchInformation.getText().length() == 0) {
            GPUtableFilter.setRowFilter(null);
            searchInformation.setText("Enter information");
            searchInformation.setForeground(Color.LIGHT_GRAY);
        }
    }
}

/* Класс слушателя нажатий клавиш для строки поиска */
private class SearchInformationKeyListener extends KeyAdapter {
    public void keyPressed(KeyEvent e) {
        if (e.getKeyCode() == KeyEvent.VK_ENTER) {
            try {
                performSearch();
            } catch (SearchFailedException SFex) {
                log.error("Ошибка поиска");
                handleSearchFailedException(SFex);
            }
        }
    }
}

/* Класс слушателя для кнопки поиска */
public class SearchButtonListener extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        try {
            performSearch();
        } catch (SearchFailedException SFex) {
            log.error("Ошибка поиска");
            handleSearchFailedException(SFex);
        }
    }
```

```
        }

        /* Класс исключения в случае нажатия кнопки, требующей выбранной строки, при отсутствии таковой */
        private class NoRowSelectedException extends Exception {
            public NoRowSelectedException() {
                super("Please, select a row");
            }
        }

        /**
         * Handles SearchFailedException
         *
         * @param SFex an object of SearchFailedException class
         */
        private void handleSearchFailedException(SearchFailedException SFex) {
            GPUtableFilter.setRowFilter(null);
            JOptionPane.showMessageDialog(null, SFex.getMessage(), "Search failed", JOptionPane.ERROR_MESSAGE);
        }

        public static void createJasperReport(String datasource, String xpath, String template, String resultpath) {
            try {
                log.info("Создание отчета");
    // Указание источника XML-данных
                JRXmlDataSource dataSource = new JRXmlDataSource(datasource, xpath);
    // Создание отчета на базе шаблона
                JasperReport jasperReport = JasperCompileManager.compileReport(template);
    // Заполнение отчета данными
                JasperPrint print = JasperFillManager.fillReport(jasperReport, new HashMap(), dataSource);
                JRExporter exporter = null;
                if (resultpath.toLowerCase().endsWith("pdf"))
                    exporter = new JRPdfExporter(); // Генерация отчета в формате PDF
```

```
// Задание имени файла для выгрузки отчета
            exporter.setParameter(JRExporterParameter.OUTPUT_FILE_NAME, resultpath);
// Подключение данных к отчету
            exporter.setParameter(JRExporterParameter.JASPER_PRINT, print);
// Выгрузка отчета в заданном формате
            exporter.exportReport();
        } catch (JRException e) {
            e.printStackTrace();
        }
    }

    public static class Util {
        public static boolean isEmpty(String value) {
            return value == null || "".equals(value);
        }

        public static int sum(int x, int y) {
            return x + y;
        }

        public static boolean checkName(String bName) {
            return bName == null || bName.equals("Test");
        }
    }
}
```