

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

Студент гр. 3312

Сабиров Р. Д.

Преподаватель

Павловский М. Г.

Санкт-Петербург

2024

Содержание

Цель работы	3
Перечень используемых типов сообщений, которые выводятся в лог-файл	3
Конфигурационный файл log4j.properties.....	4
Лог-файлы работы приложения в режимах WARN+INFO и DEBUG.....	4
Исходные тексты классов, где осуществляется протоколирование работы приложения	5
Текст документации, сгенерированный Javadoc.....	9
Вывод.....	10

Цель работы

Знакомство с методами протоколирования работы приложения с использованием библиотеки Log4j в языке программирования Java.

Перечень используемых типов сообщений, которые выводятся в лог-файл

1. DEBUG (Отладочная информация):

- Используется для вывода подробной отладочной информации, которая полезна при разработке и отладке программы. Эти сообщения позволяют отслеживать выполнение кода на низком уровне и понимать, как работает приложение.
- Этот уровень используется для отслеживания шагов в методах, значений переменных и других технических подробностей, которые помогают в отладке.

2. INFO (Информационные сообщения):

- Используется для логирования основных событий, которые отражают нормальное функционирование приложения. Эти сообщения могут быть полезны для мониторинга работы приложения в реальном времени.
- Этот уровень используется для фиксации успешных завершений операций и других значимых, но не критичных событий.

3. WARN (Предупреждения):

- Используется для сообщений, которые указывают на потенциальные проблемы, которые могут возникнуть в будущем, но не мешают продолжению работы программы. Такие события не являются фатальными.
- Этот уровень используется для предупреждения о ситуации, которая требует внимания, но не прерывает работу приложения.

Конфигурационный файл log4j.properties

```
log4j.rootLogger=DEBUG, file

log4j.appender.file=org.apache.log4j.FileAppender
log4j.appender.file.file=myproject.log
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.conversionPattern=%d{ABSOLUTE} %5p %t
%c{1}:%M:%L - %m%n
```

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

Лог-файл myproject.log в режиме DEBUG

```
12:28:48,787 INFO main lab10:Bus:51 - Программа запущена
12:28:51,073 INFO Thread-0 lab10:lambda$startLoad$1:236 - Нажата кнопка
'Загрузить'
12:28:51,075 INFO Thread-0 lab10:lambda$startLoad$1:237 - Первый поток
начал работу
12:28:54,373 DEBUG Thread-0 lab10:loadFromXML:509 - Данные успешно
загружены из XML: drivers.xml
12:28:54,374 INFO Thread-1 lab10:lambda$startEditAndSave$6:258 - Второй
поток начал работу
12:28:56,346 INFO AWT-EventQueue-0 lab10:lambda$startEditAndSave$4:278 -
Нажата кнопка 'Удалить'
12:28:56,346 WARN AWT-EventQueue-0 lab10:lambda$startEditAndSave$4:282 -
Строка для удаления не была выбрана
12:29:01,410 INFO AWT-EventQueue-0 lab10:lambda$startEditAndSave$4:278 -
Нажата кнопка 'Удалить'
12:29:02,573 DEBUG AWT-EventQueue-0 lab10:lambda$startEditAndSave$4:289 -
Водитель удален:Смирнов Николай Александрович
12:29:03,346 INFO AWT-EventQueue-0 lab10:lambda$startEditAndSave$5:296 -
Нажата кнопка 'Сохранить'
12:29:09,421 DEBUG AWT-EventQueue-0 lab10:saveToXML:562 - Данные успешно
загружены из XML: abc.xml
12:29:09,421 INFO Thread-2 lab10:lambda$startReport$8:317 - Третий поток
начал работу
12:29:10,587 INFO AWT-EventQueue-0 lab10:lambda$startReport$7:320 -
Нажата кнопка 'Создать отчет'
12:29:13,246 DEBUG AWT-EventQueue-0 lab10:generateReport:640 - Отчет
сформирован в HTML: DriverReport.html
12:29:17,484 INFO AWT-EventQueue-0 lab10>windowClosing:117 - Программа
завершена
```

Лог-файл myproject1.log в режиме WARN+INFO

```
12:28:48,787 INFO main lab10:Bus:51 - Программа запущена
12:28:51,073 INFO Thread-0 lab10:lambda$startLoad$1:236 - Нажата кнопка
'Загрузить'
12:28:51,075 INFO Thread-0 lab10:lambda$startLoad$1:237 - Первый поток
начал работу
12:28:54,374 INFO Thread-1 lab10:lambda$startEditAndSave$6:258 - Второй
поток начал работу
12:28:56,346 INFO AWT-EventQueue-0 lab10:lambda$startEditAndSave$4:278 -
Нажата кнопка 'Удалить'
12:28:56,346 WARN AWT-EventQueue-0 lab10:lambda$startEditAndSave$4:282 -
Строка для удаления не была выбрана
12:29:01,410 INFO AWT-EventQueue-0 lab10:lambda$startEditAndSave$4:278 -
Нажата кнопка 'Удалить'
12:29:03,346 INFO AWT-EventQueue-0 lab10:lambda$startEditAndSave$5:296 -
Нажата кнопка 'Сохранить'
12:29:09,421 INFO Thread-2 lab10:lambda$startReport$8:317 - Третий поток
начал работу
12:29:10,587 INFO AWT-EventQueue-0 lab10:lambda$startReport$7:320 -
Нажата кнопка 'Создать отчет'
12:29:17,484 INFO AWT-EventQueue-0 lab10>windowClosing:117 - Программа
завершена
```

Исходные тексты классов, где осуществляется протоколирование работы приложения

```
/**
 * Первый поток для загрузки данных из XML
 */
private void startLoad() {
    Thread thread1 = new Thread(() -> {
        log.info("Нажата кнопка 'Загрузить'");
        log.info("Первый поток начал работу");
        System.out.println("First thread is running");

        loadFromXML();

        firstThreadLatch.countDown();
    });
    thread1.start();
}

/**
 * Второй поток для редактирования и сохранения в XML
 */
private void startEditAndSave() {
    Thread thread2 = new Thread(() -> {
        try {
            firstThreadLatch.await();
            System.out.println("Second thread is running");
            log.info("Второй поток начал работу");
        }
    });
    thread2.start();
}
```

```

        addButton.addActionListener(e -> {
            log.info("Нажата кнопка 'Добавить'");
            addDriverDialog();
        });
        editButton.addActionListener(e -> {
            log.info("Нажата кнопка 'Редактировать'");
            try {
                int selectedRow = driverTable.getSelectedRow();
                if (selectedRow == -1) {
                    log.warn("Строка для редактирования не была
выбрана");
                    throw new RowNotSelectedException("Пожалуйста,
выберите строку для редактирования");
                }
                editDriverDialog(selectedRow);
            } catch (RowNotSelectedException ex) {
                JOptionPane.showMessageDialog(frame,
ex.getMessage(), "Ошибка", JOptionPane.ERROR_MESSAGE);
            }
        });
        deleteButton.addActionListener(e -> {
            log.info("Нажата кнопка 'Удалить'");
            try {
                int selectedRow = driverTable.getSelectedRow();
                if (selectedRow == -1) {
                    log.warn("Строка для удаления не была
выбрана");
                    throw new RowNotSelectedException("Пожалуйста,
выберите строку для удаления");
                }
                tableModel.removeRow(selectedRow);

                String name = (String)
tableModel.getValueAt(selectedRow, 0);
                JOptionPane.showMessageDialog(frame, "Водитель
удален");

                log.debug("Водитель удален:" + name);
            } catch (RowNotSelectedException ex) {
                JOptionPane.showMessageDialog(frame,
ex.getMessage(), "Ошибка", JOptionPane.ERROR_MESSAGE);
            }
        });

        saveButton.addActionListener(e -> {
            log.info("Нажата кнопка 'Сохранить'");
            saveToXML();
            secondThreadLatch.countDown();
        });

        secondThreadLatch.await();
    } catch (InterruptedException ex) {
        ex.printStackTrace();
    }
});
thread2.start();
}

```

```

/**
 * Метод для отображения диалогового окна добавления водителя
 */
private void addDriverDialog() {
    JPanel panel = new JPanel(new GridLayout(3, 2));
    JTextField nameField = new JTextField(20);
    JTextField experienceField = new JTextField(20);
    JTextField categoryField = new JTextField(20);

    panel.add(new JLabel("ФИО: "));
    panel.add(nameField);
    panel.add(new JLabel("Стаж работы: "));
    panel.add(experienceField);
    panel.add(new JLabel("Класс: "));
    panel.add(categoryField);

    int result = JOptionPane.showConfirmDialog(frame, panel, "Добавить водителя", JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);

    if (result == JOptionPane.OK_OPTION) {
        try {
            String name = nameField.getText();
            String experience = experienceField.getText();
            String category = categoryField.getText();

            validateFields(name, experience, category);

            tableModel.addRow(new Object[]{name, experience, category});
            JOptionPane.showMessageDialog(frame, "Водитель добавлен!");
            log.debug("Добавлен новый водитель: " + name);

        } catch (InvalidFieldException ex) {
            JOptionPane.showMessageDialog(frame, ex.getMessage(), "Ошибка", JOptionPane.ERROR_MESSAGE);
        }
    }
}

/**
 * Метод для отображения диалогового окна редактирования водителя
 * @param row номер редактируемой строки
 */
private void editDriverDialog(int row) {
    JPanel panel = new JPanel(new GridLayout(3, 2));
    JTextField nameField = new JTextField((String) tableModel.getValueAt(row, 0), 20);
    JTextField experienceField = new JTextField((String) tableModel.getValueAt(row, 1), 20);
    JTextField categoryField = new JTextField((String) tableModel.getValueAt(row, 2), 20);

    panel.add(new JLabel("ФИО: "));
    panel.add(nameField);
    panel.add(new JLabel("Стаж работы: "));
    panel.add(experienceField);
    panel.add(new JLabel("Класс: "));
    panel.add(categoryField);
}

```

```

int result = JOptionPane.showConfirmDialog(frame, panel, "Редактировать
водителя", JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);

    if (result == JOptionPane.OK_OPTION) {
        try {
            String name = nameField.getText();
            String experience = experienceField.getText();
            String category = categoryField.getText();

            validateFields(name, experience, category);

            tableModel.setValueAt(name, row, 0);
            tableModel.setValueAt(experience, row, 1);
            tableModel.setValueAt(category, row, 2);
            JOptionPane.showMessageDialog(frame, "Данные водителя
обновлены!");
            log.debug("Изменен водитель: " + name);

        } catch (InvalidFieldException ex) {
            JOptionPane.showMessageDialog(frame, ex.getMessage(),
"Ошибка", JOptionPane.ERROR_MESSAGE);
        }
    }
}

/**
 * Метод для проверки корректности введенных данных
 * @param name      ФИО водителя
 * @param experience Стаж
 * @param category  Класс
 * @throws InvalidFieldException если хотя бы одно из полей пустое или
стаж работы не является целым числом
 */
private void validateFields(String name, String experience, String
category) throws InvalidFieldException {
    if (name.isEmpty() || experience.isEmpty() || category.isEmpty())
    {
        log.warn("Не все поля были заполнены");
        throw new InvalidFieldException("Все поля должны быть
заполнены!");
    }
    try {
        Integer.parseInt(experience);
    } catch (NumberFormatException ex) {
        log.warn("Стаж должен быть числом");
        throw new InvalidFieldException("Стаж работы должен быть
числом!");
    }
}

```


Текст документации, сгенерированный Javadoc

PACKAGE

CLASS

USE

TREE

INDEX

HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD

DETAIL: FIELD | CONSTR | METHOD

Package

org.example

Class lab10

java.lang.Object[Ⓜ]
org.example.lab10

public class lab10
extends Object[Ⓜ]

Version:
1.0

Author:
Сабиров Роман 3312

Constructor Summary

Constructors

Constructor	Description
lab10()	

Method Summary

All MethodsStatic MethodsInstance MethodsConcrete Methods

Modifier and Type	Method	Description
void	Bus()	
static void	main(String [Ⓜ] [] args)	

Methods inherited from class java.lang.Object[Ⓜ]

clone[Ⓜ], equals[Ⓜ], finalize[Ⓜ], getClass[Ⓜ], hashCode[Ⓜ], notify[Ⓜ], notifyAll[Ⓜ], toString[Ⓜ], wait[Ⓜ], wait[Ⓜ], wait[Ⓜ]

Constructor Details

lab10

public lab10()

Method Details

Bus

public void Bus()

main

public static void main(String[Ⓜ][] args)

Parameters:

args - аргументы командной строки (не используются)

Вывод

В ходе выполнения работы было разработано протоколирование приложения с использованием библиотеки Log4j на языке программирования Java.

Ссылка на репозиторий: https://github.com/sabiroma/OOP_labs/tree/main/lab10

Ссылка на видео: <https://disk.yandex.ru/i/CLjJOXvqI2AJdw>