Минобрнауки России

Федеральное государственное автономное образовательное учреждение

высшего образования «Санкт-Петербургский государственный

электротехнический университет «ЛЭТИ»

им В. И. Ульянова (Ленина)»

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

**Лабораторная работа № 10**

**«Разработка интерфейса пользователя» по дисциплине**

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

Выполнил Авдеев Н. А.

Факультет КТИ

Группа № 3312

Санкт-Петербург

2024

**Цель работы:**

Знакомство с правилами и классами построения параллельных приложений в языке Java.

**Перечень используемых типов сообщений, которые выводятся в лог файл:**

* + *INFO*: Успешные операции, например, "Файл открыт".
  + *DEBUG*: Детальная информация, например, "Список строк: ...".
  + *WARN*: Предупреждения, например, "Файл не найден".

**Конфигурационный файл log4j.properties:**

# Root logger

log4j.rootLogger=INFO, Console, File

# Console appender

log4j.appender.Console=org.apache.log4j.ConsoleAppender

log4j.appender.Console.layout=org.apache.log4j.PatternLayout

log4j.appender.Console.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p [%t] %c{1}:%L - %m%n

# File appender

log4j.appender.File=org.apache.log4j.FileAppender

log4j.appender.File.File=logs/myproject.log

log4j.appender.File.layout=org.apache.log4j.PatternLayout

log4j.appender.File.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p [%t] %c{1}:%L - %m%n

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

|  |
| --- |
| 2024-12-14 16:18:25,094 [AWT-EventQueue-0] INFO Processor - Открытие файла:  2024-12-14 16:18:36,835 [AWT-EventQueue-0] INFO UserInterface - Удаление1 строк: [5]  2024-12-14 16:18:41,938 [AWT-EventQueue-0] INFO UserInterface - Удаление3 строк: [5, 4, 3]  2024-12-14 16:18:43,270 [AWT-EventQueue-0] INFO UserInterface - Добавлена новая строка.  2024-12-14 16:18:48,695 [AWT-EventQueue-0] WARN UserInterface - Попытка поиска без ввода текста.  2024-12-14 16:19:02,690 [AWT-EventQueue-0] INFO UserInterface - Поиск текста: Мигрень  2024-12-14 16:19:15,562 [AWT-EventQueue-0] INFO UserInterface - Поиск текста: Волков  2024-12-14 16:19:26,297 [AWT-EventQueue-0] INFO Processor - Экспорт данных в формат: htm**l** |

|  |
| --- |
| 2024-12-14 16:26:44,990 [AWT-EventQueue-0] INFO Processor - Открытие файла:  2024-12-14 16:26:51,010 [AWT-EventQueue-0] DEBUG Processor - Данные успешно загружены из файла: /Users/daniilmohno/Library/Mobile Documents/com~apple~CloudDocs/Student-staff/ооп/Лабы/com.study\_oop.Laba\_10/data.xml  2024-12-14 16:26:56,362 [AWT-EventQueue-0] WARN UserInterface - Попытка удалить строку без выбора.  2024-12-14 16:27:08,614 [AWT-EventQueue-0] INFO UserInterface - Удаление2 строк: [1, 0]  2024-12-14 16:27:09,310 [AWT-EventQueue-0] INFO UserInterface - Добавлена новая строка.  2024-12-14 16:27:09,310 [AWT-EventQueue-0] DEBUG UserInterface - Общее количество строк после добавления: 9  2024-12-14 16:27:17,578 [AWT-EventQueue-0] WARN UserInterface - Попытка поиска без ввода текста. |

**Исходные тексты классов, где осуществляется протоколирование работы приложения:**

private static final Logger log = Logger.getLogger(App.class);

    public static void printReportMultithreaded(String xmlFilePath, String templatePath, String outputPath, String format) {

        log.info("Запуск многопоточной обработки для генерации отчета.");

        CountDownLatch dataLoaded = new CountDownLatch(1);

        CountDownLatch dataEdited = new CountDownLatch(1);

        ExecutorService executor = Executors.newFixedThreadPool(3);

        // Поток для загрузки данных

        executor.submit(() -> {

            log.info("Начало загрузки данных из файла: " + xmlFilePath);

            try {

                JRXmlDataSource xmlDataSource = new JRXmlDataSource(new File(xmlFilePath));

                log.info("Данные успешно загружены из файла: " + xmlFilePath);

                dataLoaded.countDown();

            } catch (JRException e) {

                log.severe("Ошибка при загрузке данных: " + e.getMessage());

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

            }

        });

        // Поток для редактирования данных

        executor.submit(() -> {

            try {

                log.info("Ожидание завершения загрузки данных.");

                dataLoaded.await();

                log.info("Загрузка данных завершена. Начало редактирования данных.");

                // Реализуйте редактирование данных здесь

                log.info("Редактирование данных завершено.");

                dataEdited.countDown();

            } catch (InterruptedException ex) {

                log.warn("Редактирование данных прервано: " + ex.getMessage());

            }

        });

        // Поток для генерации отчета

        executor.submit(() -> {

            try {

                log.info("Ожидание завершения редактирования данных.");

                dataEdited.await();

                log.info("Редактирование данных завершено. Начало генерации отчета.");

                printReport(xmlFilePath, templatePath, outputPath, format);

                log.info("Отчет успешно сгенерирован. Выходной файл: " + outputPath);

            } catch (InterruptedException ex) {

                log.warn("Генерация отчета прервана: " + ex.getMessage());

            }

        });

        executor.shutdown();

        try {

            if (executor.awaitTermination(Long.MAX\_VALUE, TimeUnit.NANOSECONDS)) {

                log.info("Все задачи завершены.");

            }

        } catch (InterruptedException ex) {

            log.warn("Ожидание завершения задач было прервано: " + ex.getMessage());

        }

    }

**Вывод:**

Реализовано многопоточное приложение для генерации отчетов, обеспечивающее эффективное выполнение задач за счёт параллельной обработки. Приложение разделяет процесс на три этапа: загрузку данных из XML-файла, редактирование данных и формирование отчета в заданном формате (PDF, HTML, Excel).

Для синхронизации между потоками используется механизм CountDownLatch, который гарантирует строгий порядок выполнения операций, предотвращая ситуации, когда этап обработки начинается до завершения предыдущего.

Логирование на каждом этапе позволяет отслеживать статус выполнения задач, выявлять ошибки и упрощать диагностику.

**Ссылка на репозиторий**:

https://github.com/G3rb3r228/Labs\_OOP\_AvdeevNA-3312/tree/main/Lab10\_Avdeev3312