Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

**ЛАБОРАТОРНА РОБОТА № 6**

з дисципліни «Математичні методи та технології тестування та верифікації програмного забезпечення»

Тема: «Фреймворк для логування log4j»

Виконав студент 2 курсу

групи КС-21

Безрук Юрій Русланович

Перевірив:

Доцент Нарєжній О. П.

Харків – 2020

Целью данной работы является изучение изучение фреймворка записи логирования log4j.

Log4J – фреймверк для скрытия реализации рутинных операций по логированию некоторых событий, которые происходят во время работы приложения, которое написано на Java и других языков программирования.

# ХОД РАБОТЫ

Для обучения работы с фреймворком log4j, его нужно скачать с официального сайта и подключить к проектам Java и Maven.

Для начала опробуем подключение к обычному Java-проекту. После его создания, в папке srs нужно создать файл log4j2.xml и добавляем туда минимальные настройки.

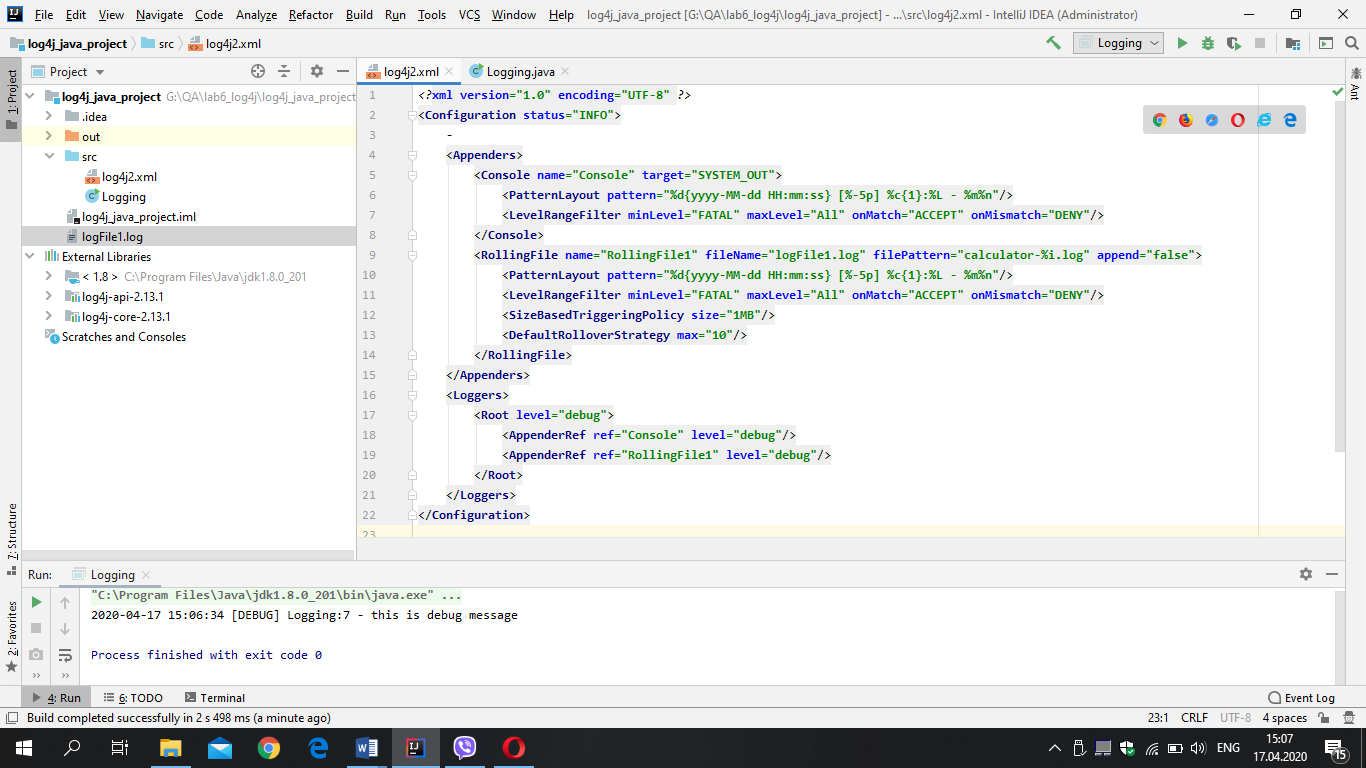


Рисунок 1 - файл log4j2.xml

Данный файл управляет конфигурациями логирования в проекте. Далее нужно подключить библиотеки log4j к проекту (файлы .jar в External libraries).

В папке srs создаем класс Logging в котором проведем тест возможностей класса Logger и его основных методов.

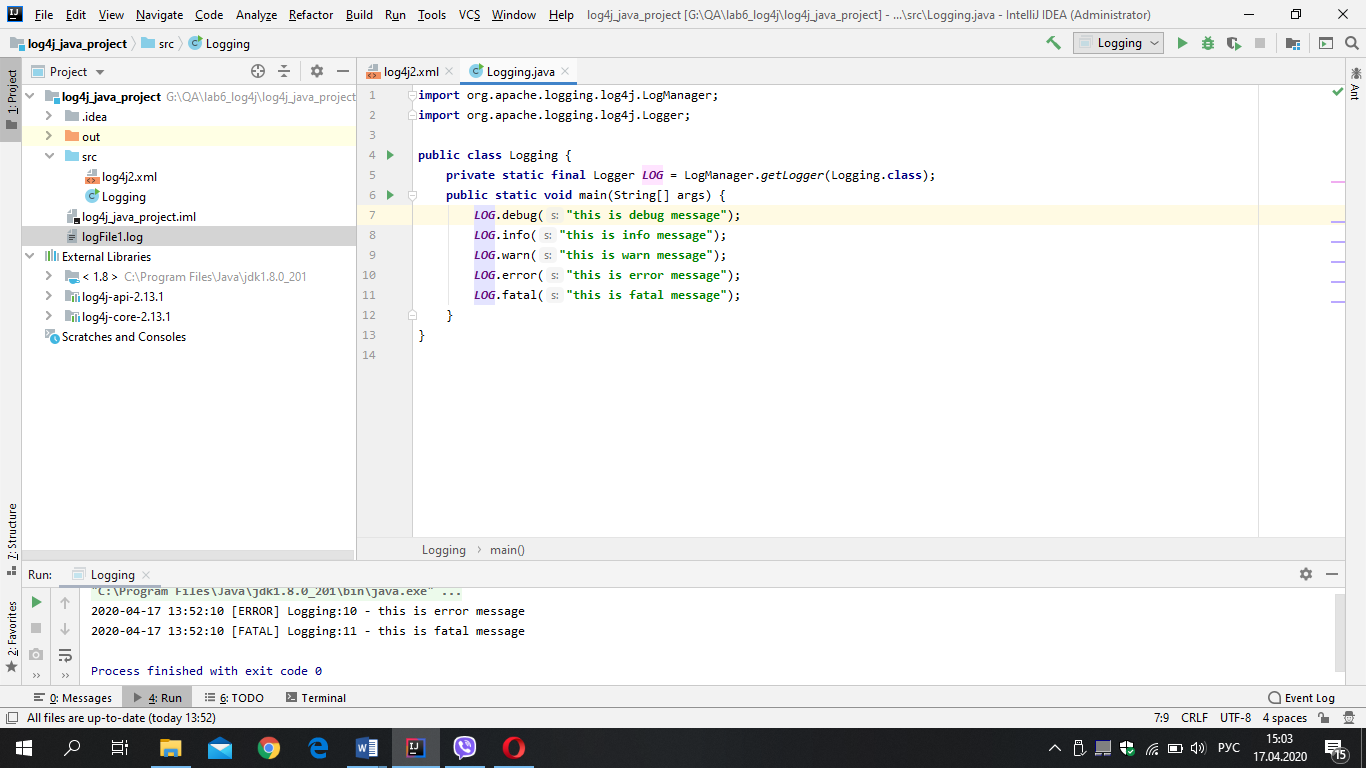


Рисунок 2 - класс Logging

После чего запускаем проект, и получаем вывод лога в консоль по всем уровням сообщений.

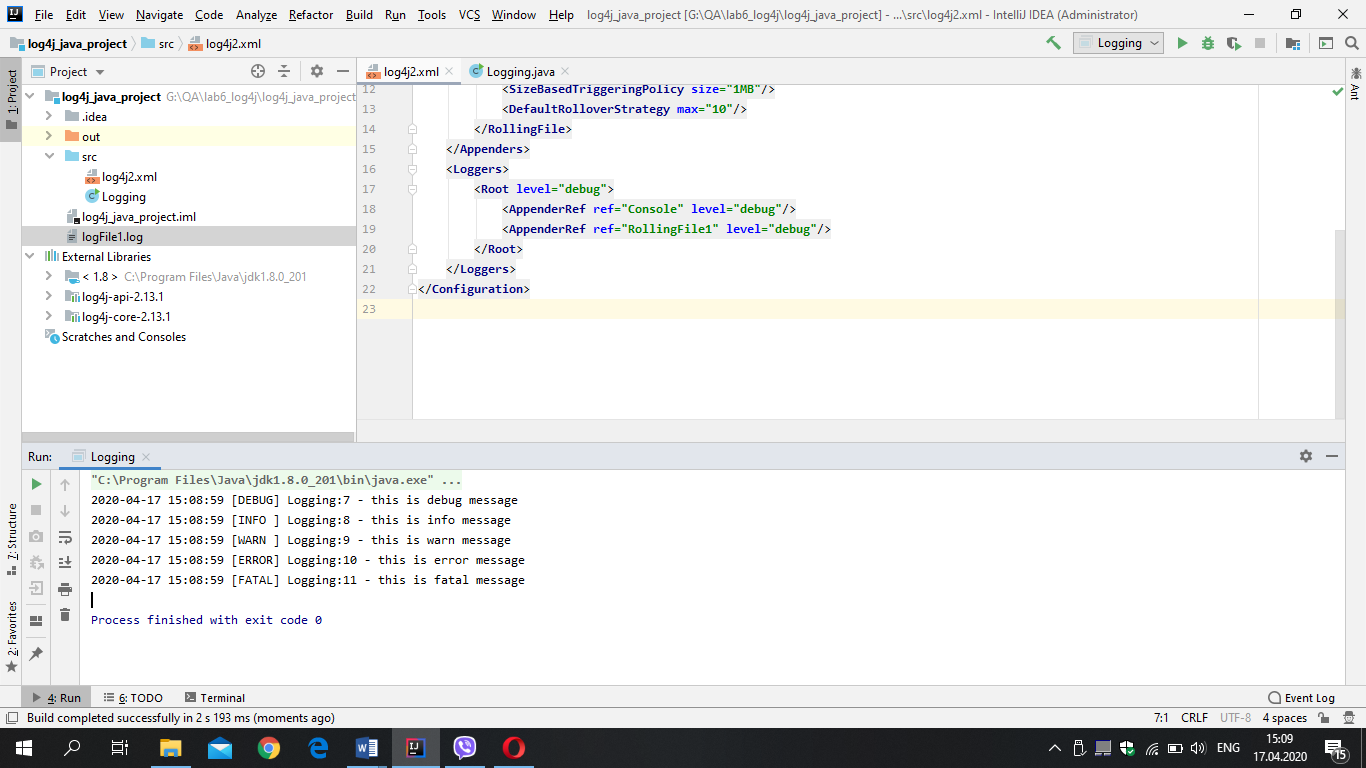


Рисунок 3 - вывод лога в консоль

Кроме того, также эти сообщения автоматически записываются в файл logFile1.log, указанный в файле конфигураций логирования.

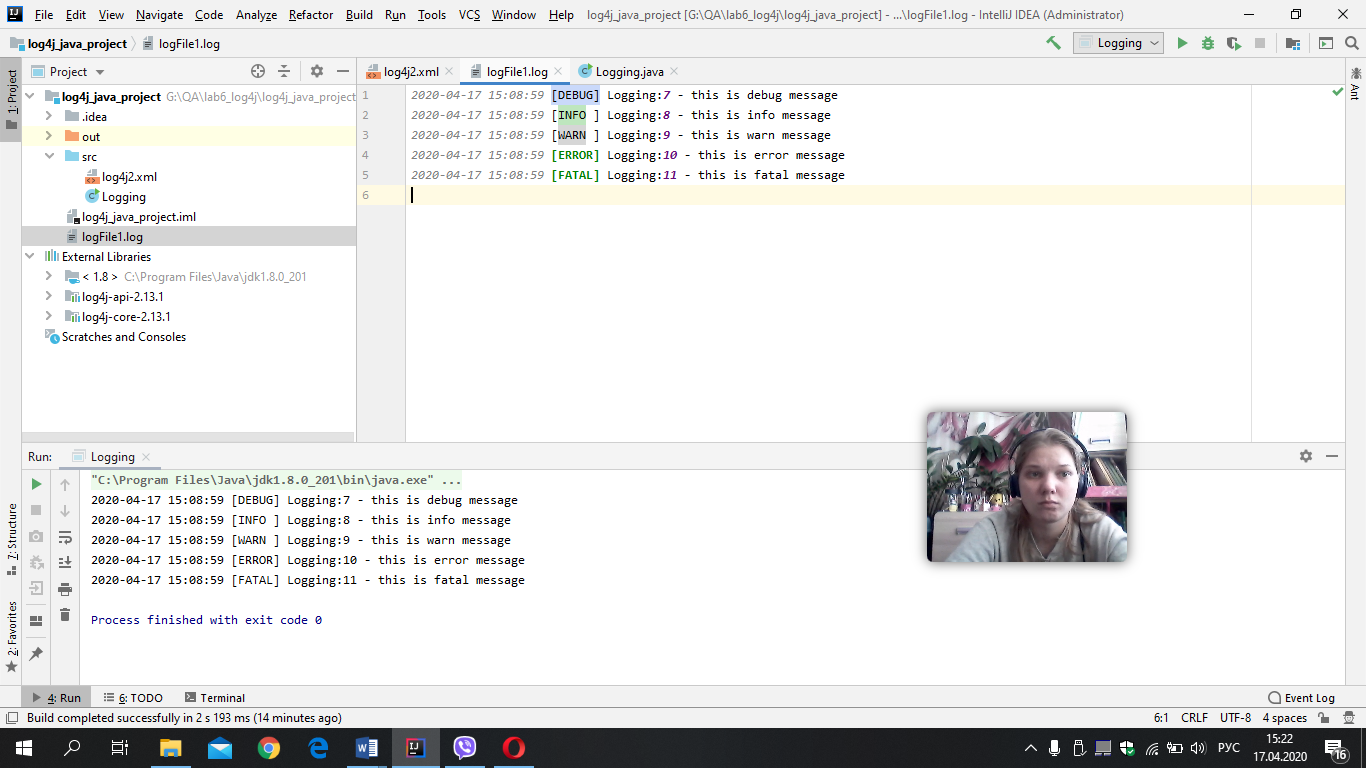


Рисунок 4 - запись в первом файле

Для того, чтобы записать в файл лога только сообщения уровней ERROR и FATAL, необходимо изменить значение атрибута maxlevel в теге LevelRangeFilter в конфигурациях лога в файл (тег RollingFile) на ERROR. Также изменим название самого выходного файла для лога на logFile2.log, чтобы сохранить результат в другом файле.

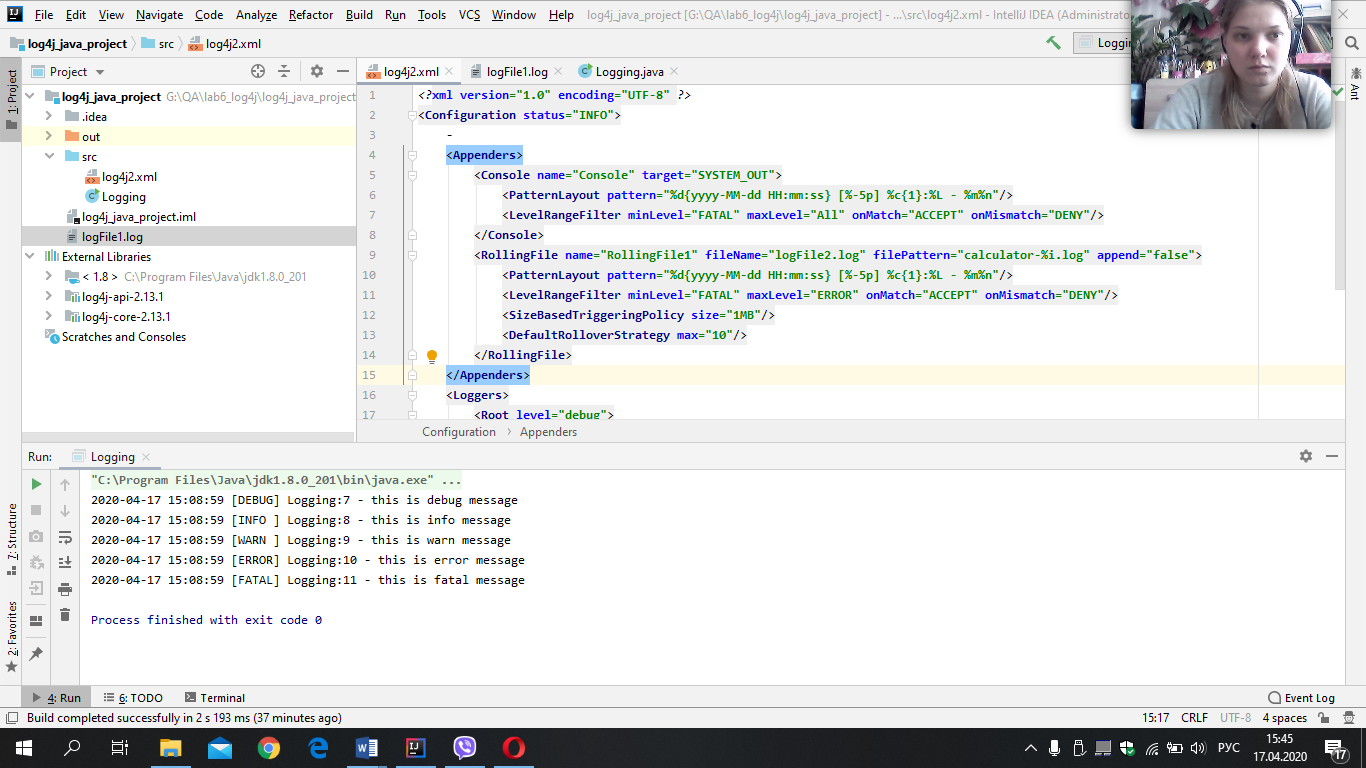


Рисунок 5 - изменения в файле конфигурации

После чего запускаем класс и результат выполнения лога с новыми параметрами запишется в новый файл.

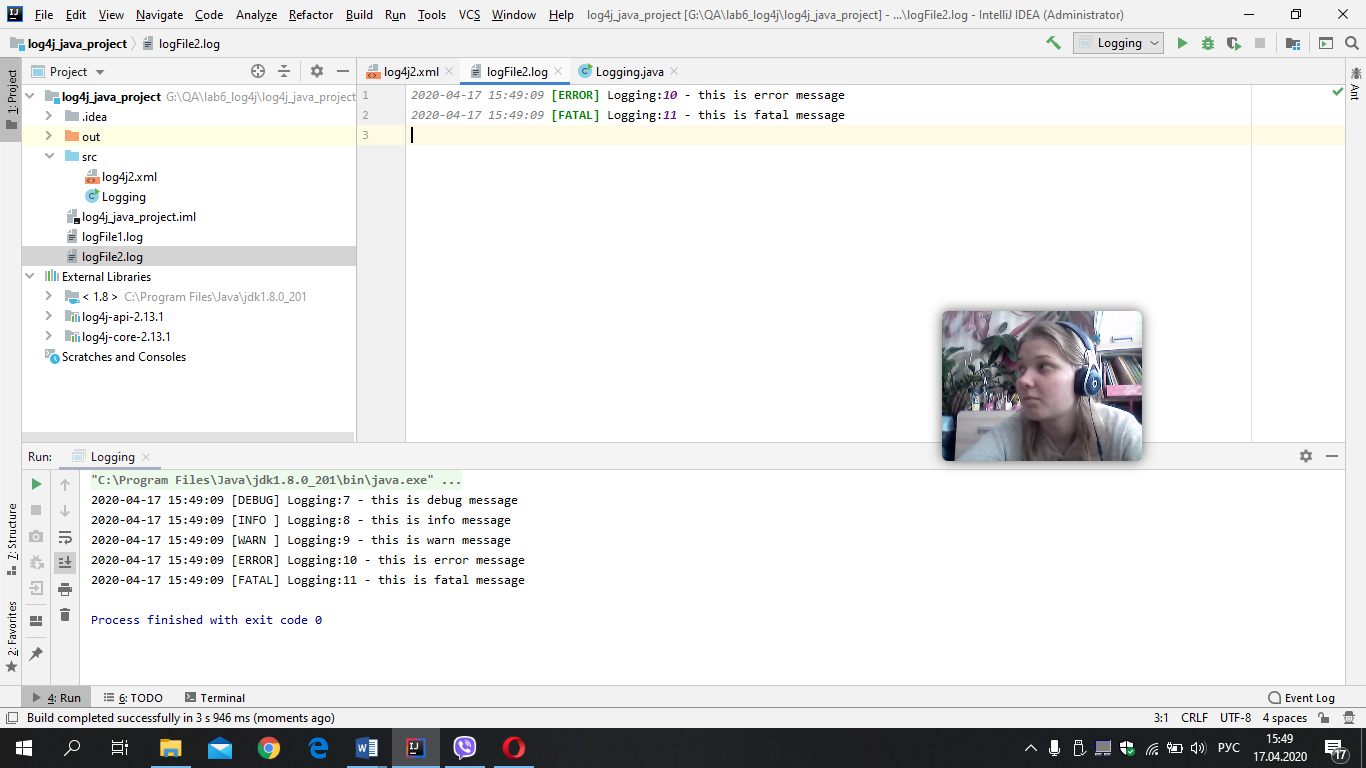


Рисунок 6 – запись во втором файле

Для того, чтобы подключить log4j к проекту Maven, достаточно добавить в файл pom.xml необходимые зависимости и нажать кнопку «import changes».

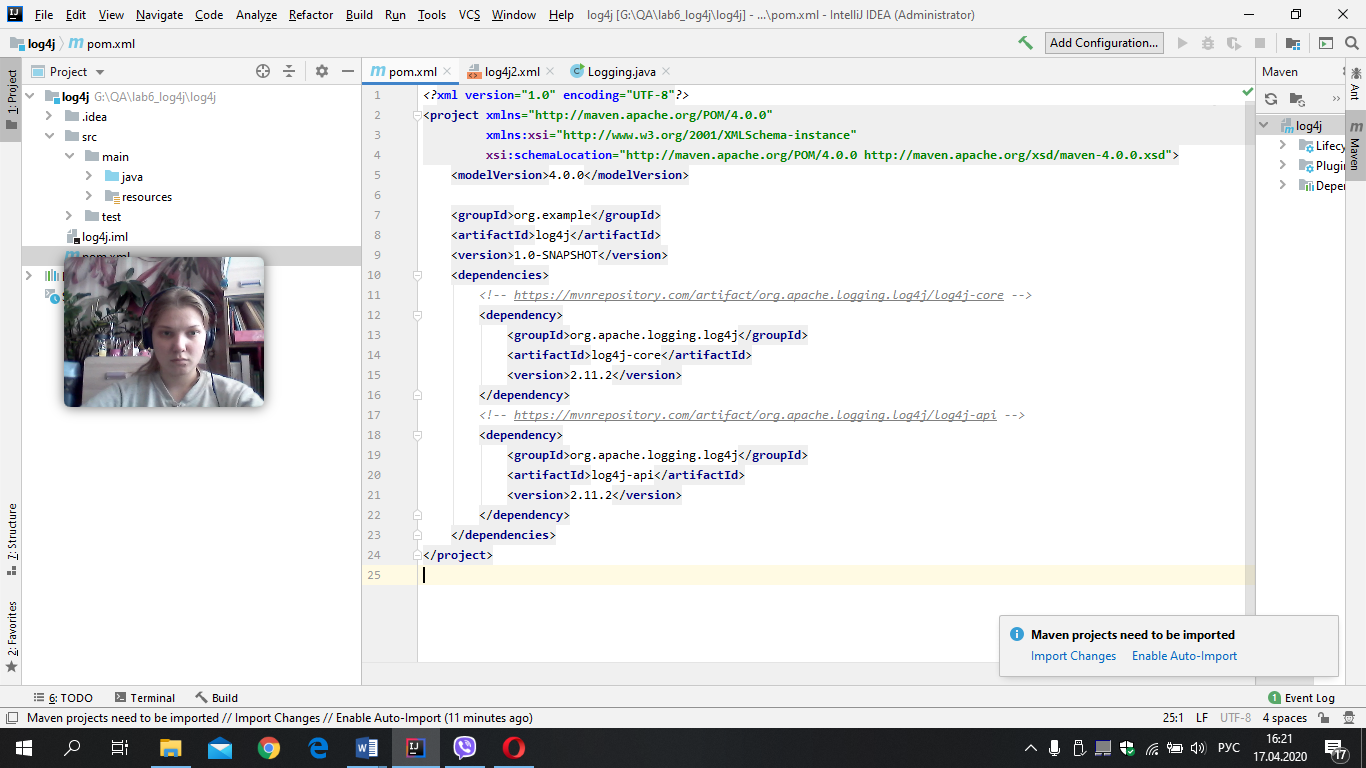


Рисунок 7 - файл pom.xml с необходимыми зависимостями

После чего, как и в обычном Java-проекте, необходимо создать файл log4j2.xml с необходимыми конфигурациями логирования, который, на этот раз нужно добавить на вкладку resourses проекта Maven. Класс Logging добавляем в папку main\java.

Для того, чтобы провести логирование проекта пятой лабораторной работы, копируем в проект класс WebTesting и дописываем в файл pom.xml соответствующие зависимости на JUnit и Selenium WebDriver.

Логирование будет производиться так: в методе с аннотацией @BeforeClass запишем в лог информацию о начале тестирования, @AfterClass – об окончании. Кроме того, в каждом методе тестирования выведем информацию о начале теста, а в конце – лог результата теста (passed / not passed). Также можно добавить лог ошибки на процесс создания драйвера.

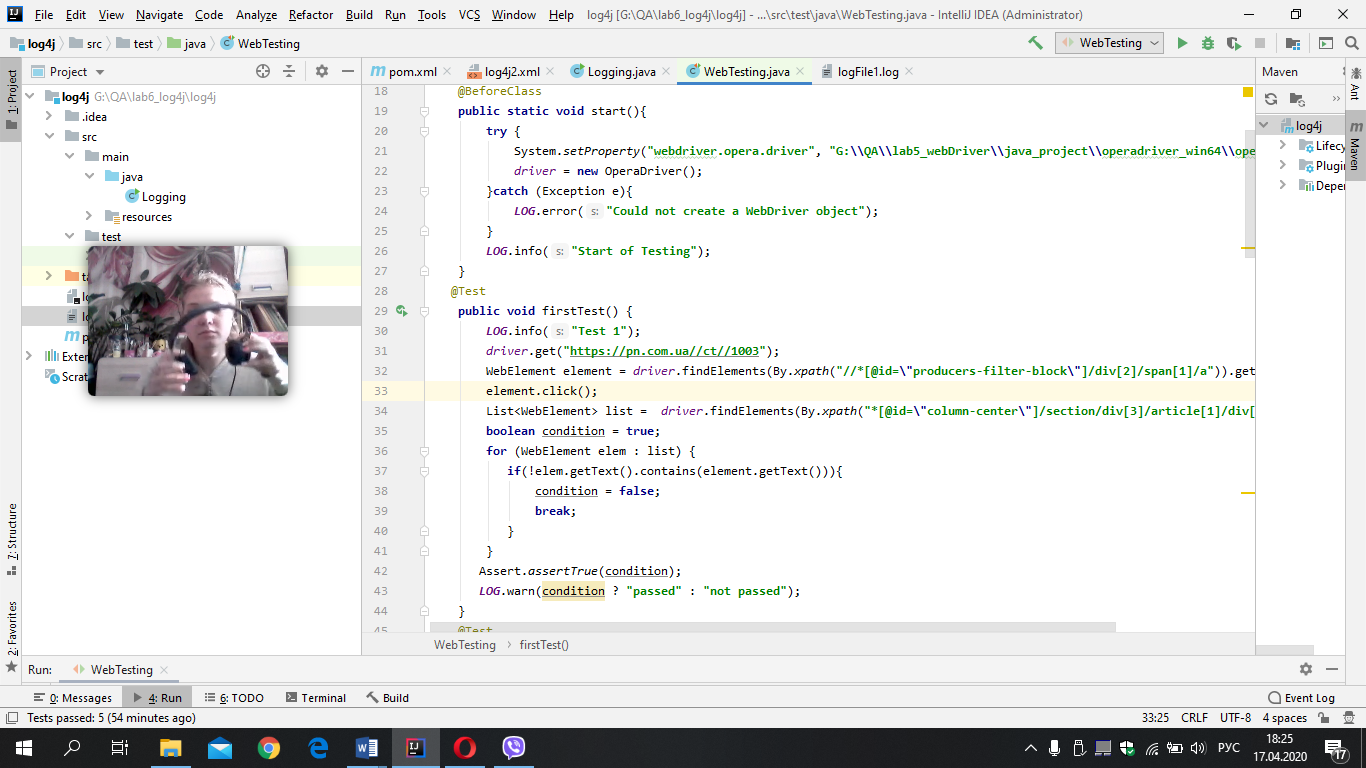


Рисунок 8 - пример внесенных изменений

После чего запустим класс с тестами, и проверен выходной лог-файл.

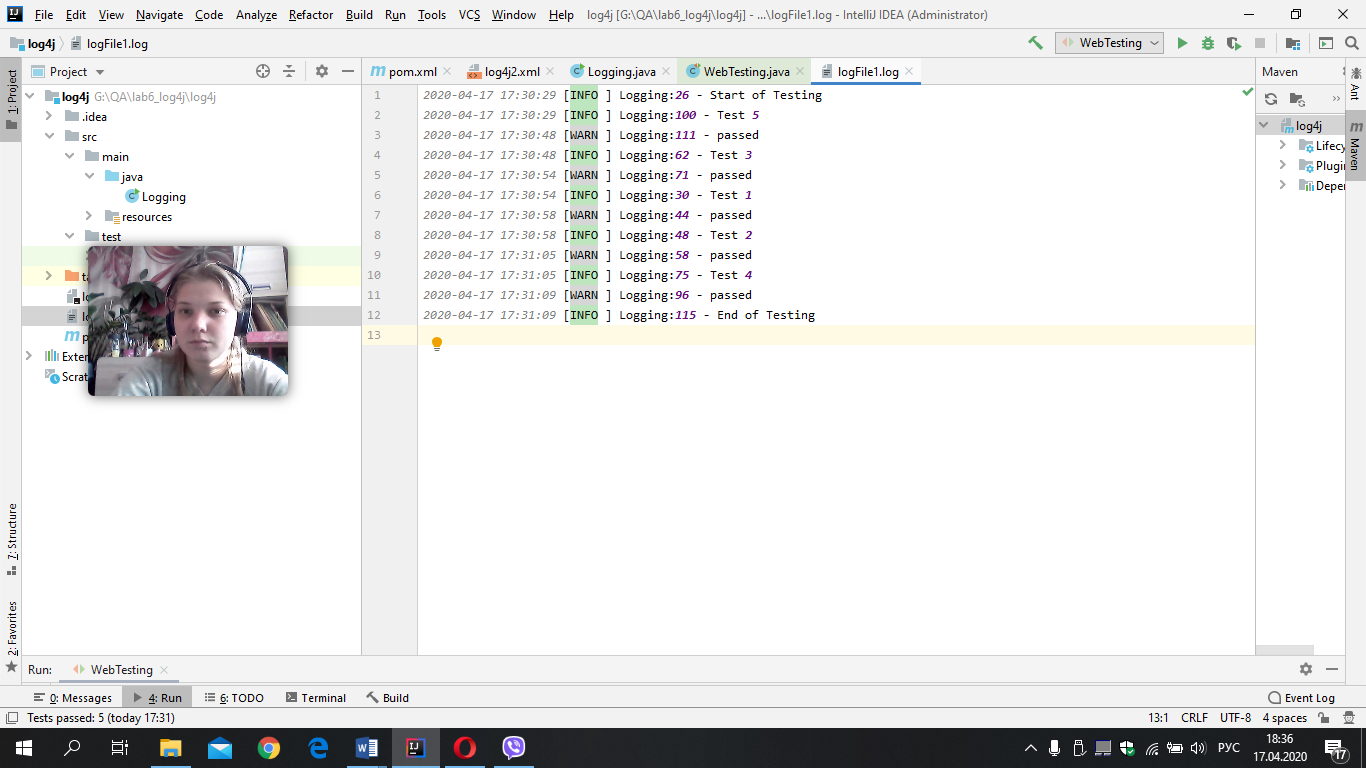


Рисунок 9 - выходной лог-файл проекта пятой лабораторной

# ВЫВОДЫ

Таким образом, в ходе данной лабораторной работы был рассмотрен процесс логирования на примере фреймворка Log4J.

Во время логирования пользователь решает следующие вопросы: - выбор хранилища (консоль, файл) - конфигурация хранилища - наименование хранилища, объем хранилища, путь, сохранение конфигурации; - форматирование записей журнала - дата, время, класс, метод и т.д.

В процессе выполнения работы были созданы два проекта, в которых реализованы классы и файлы, позволяющие проводить процесс логирования. В результате выполнения обеих программ, было получено корректные исходные данные, как на консоли, так и в файлах.