# Практическая работа 1

**Задание 1**

Был создан массив из 10000 случайных чисел.

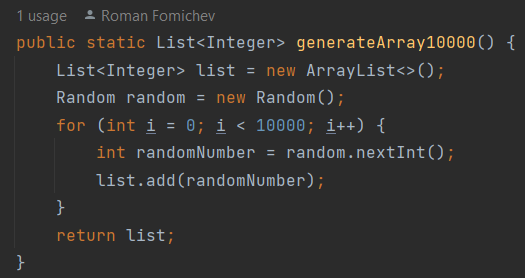


Рисунок 1 – Функция создания списка из 10000 элементов

В созданном массиве был начат поиск наименьшего элемента последовательно, многопоточно и с использованием ForkJoin.

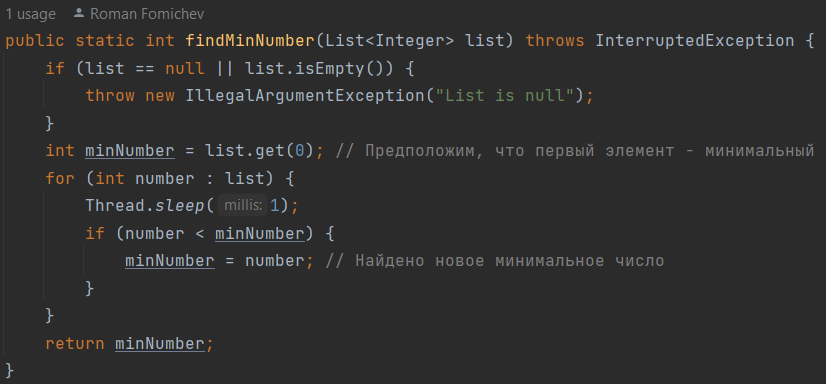


Рисунок 2 – Последовательный способ

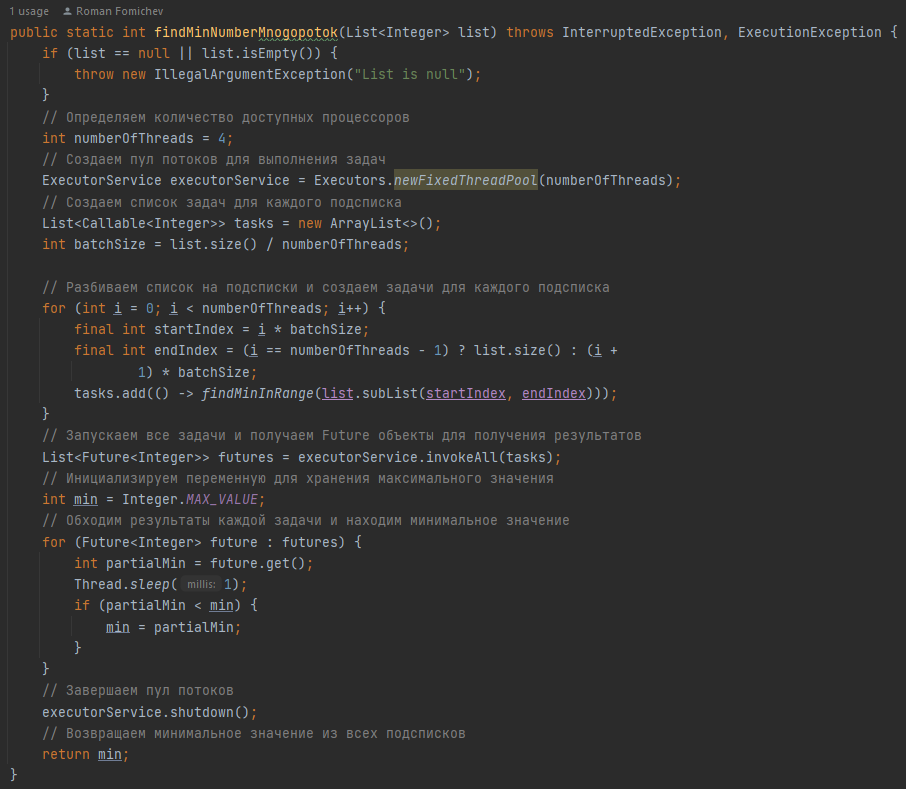


Рисунок 3 – Многопоточный способ

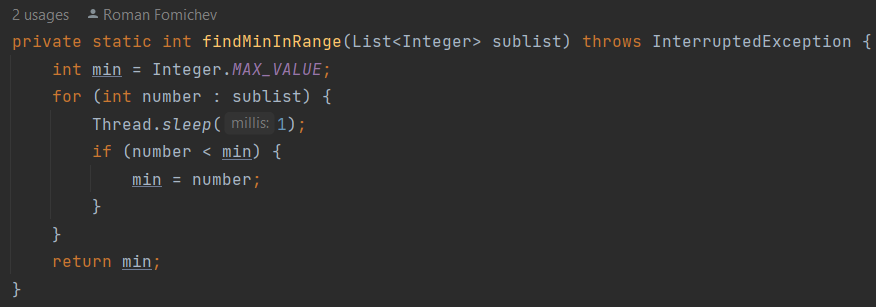


Рисунок 4 – Fork

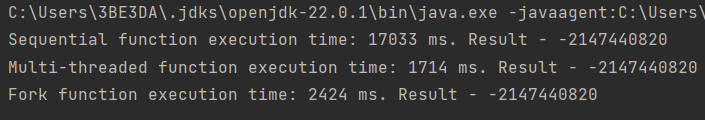


Рисунок 5 – Результат

Последовательный поиск наименьшего элемента оказался самым медленным.

**Задание 2**

Программа запрашивает у пользователя на вход число. Программа имитирует обработку запроса пользователя в виде задержки от 1 до 5 секунд выводит результат: число, возведенное в квадрат. В момент выполнения запроса пользователь имеет возможность отправить новый запрос. Реализовать с использованием Future.

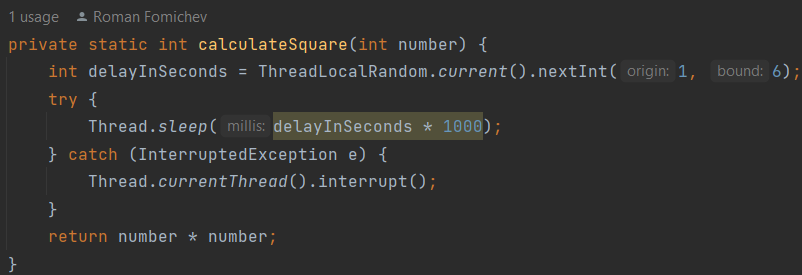


Рисунок 6 – Функция, высчитывающая квадрат



Рисунок 7 – Функция, обрабатывающая значения пользователя

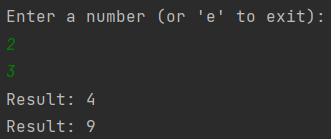


Рисунок 8 – Результат

В момент выполнения запроса пользователь имеет возможность отправить новый запрос.

**Задание 3**

Класс Файл имеет следующие характеристики:

1. Тип файла (например XML, JSON, XLS).

2. Размер файла — целочисленное значение от 10 до 100.

Генератор файлов — генерирует файлы с задержкой от 100 до 1000 мс. Очередь — получает файлы из генератора. Вместимость очереди — 5 файлов.

Обработчик файлов — получает файл из очереди. Каждый обработчик имеет параметр — тип файла, который он может обработать. Время обработки файла: «Размер файла\*7мс».

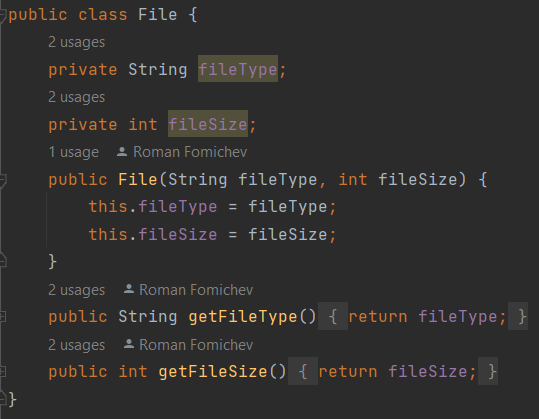


Рисунок 9 – Класс File

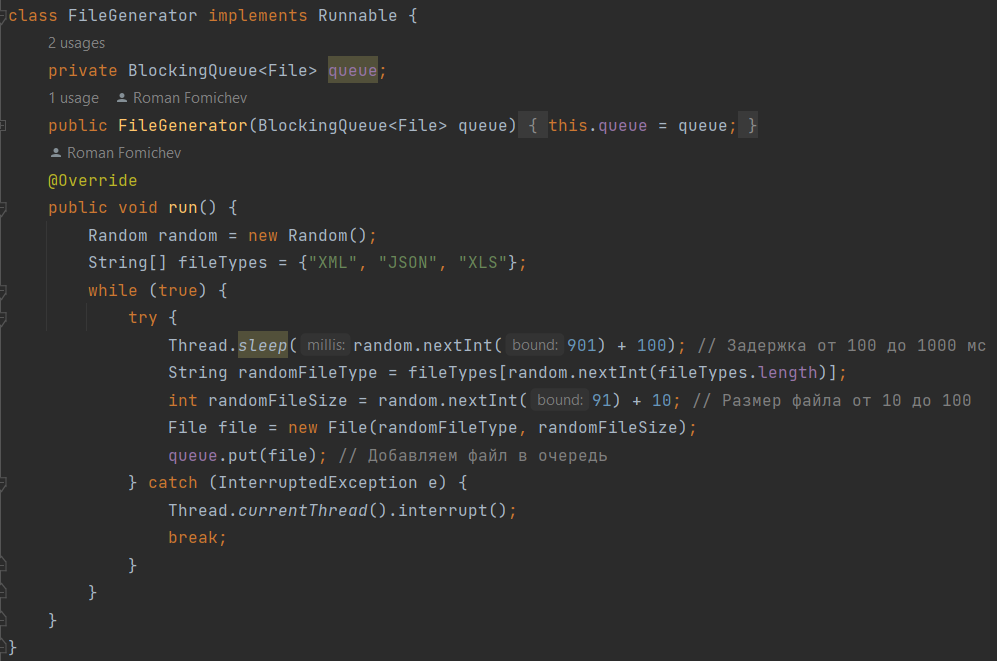


Рисунок 10 – Класс FileGenerator

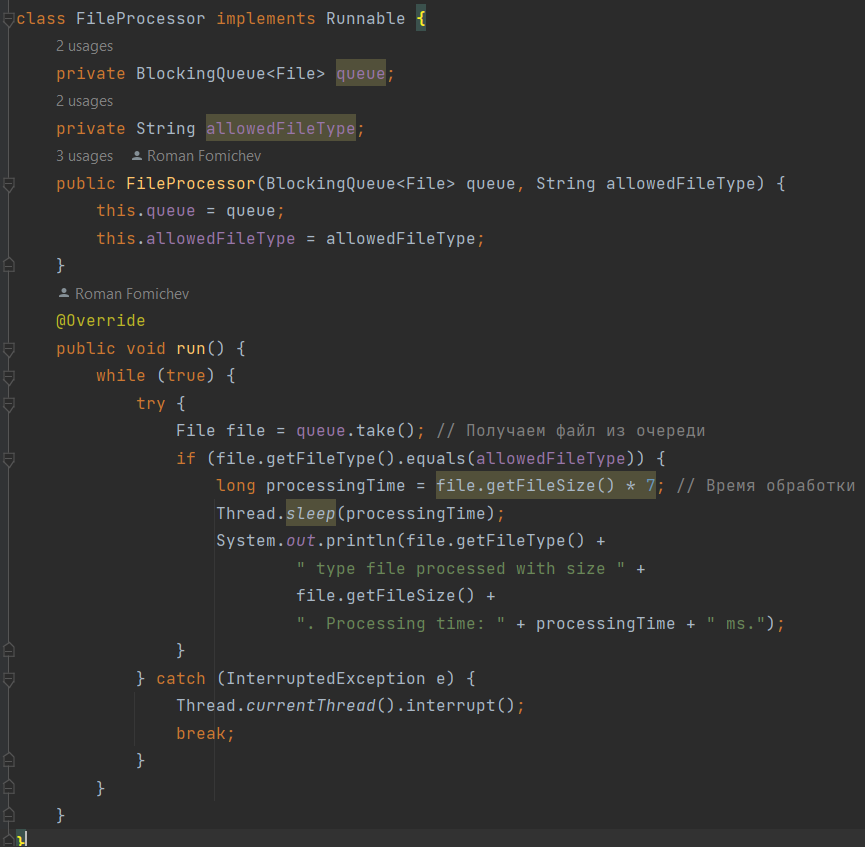


Рисунок 11 – Класс FileProcessor

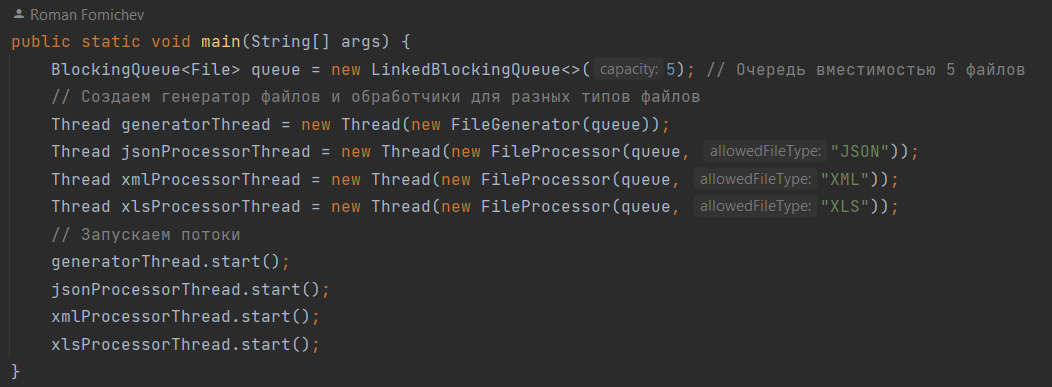


Рисунок 12 – Запуск потоков

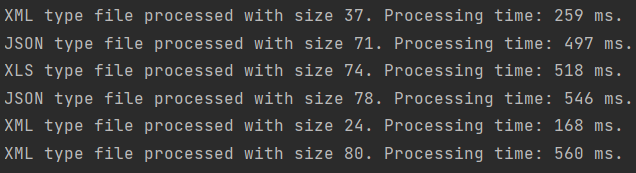


Рисунок 13 – Результат

В результате, обеспечена потокобезопасность, работа генератора не зависит от работы обработчиков, потоки не блокируют другие потоки при отсутствии задач

# Практическая работа 2

**Задание 1**

Для выполнения первого задания были созданы функции для создания файла и записи информации в него, а также чтения. Функции представлены на рисунке 14.

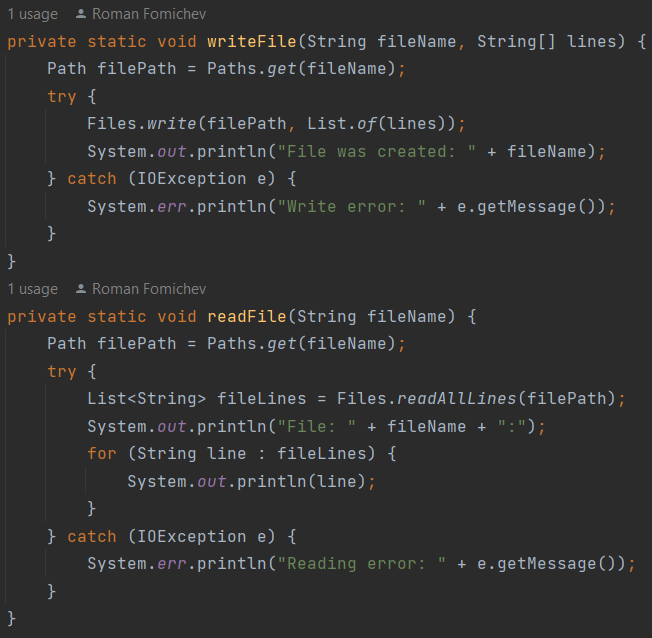


Рисунок 14 – Функции для чтения и записи

В файл записываются строки, а затем считываются. Создание файла и его чтение представлено на рисунке 15.

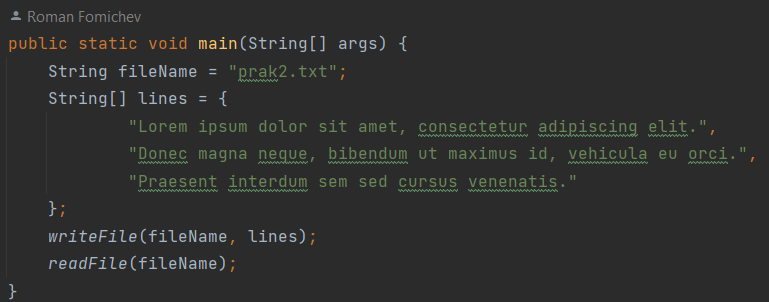


Рисунок 15 – Создание и чтение файла

Результат работы программы представлен на рисунке 16.

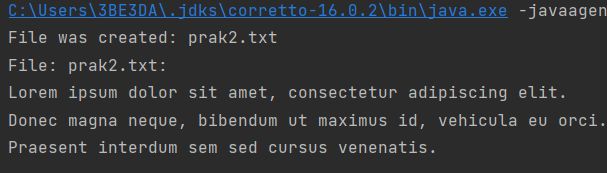


Рисунок 16 – Результат работы

**Задание 2**

Для выполнения второго задания были созданы функции для копирования файла разными методами, а именно: stream, FileChannel, Apache Common IO, Files class. Данные методы представлены на рисунке 17.

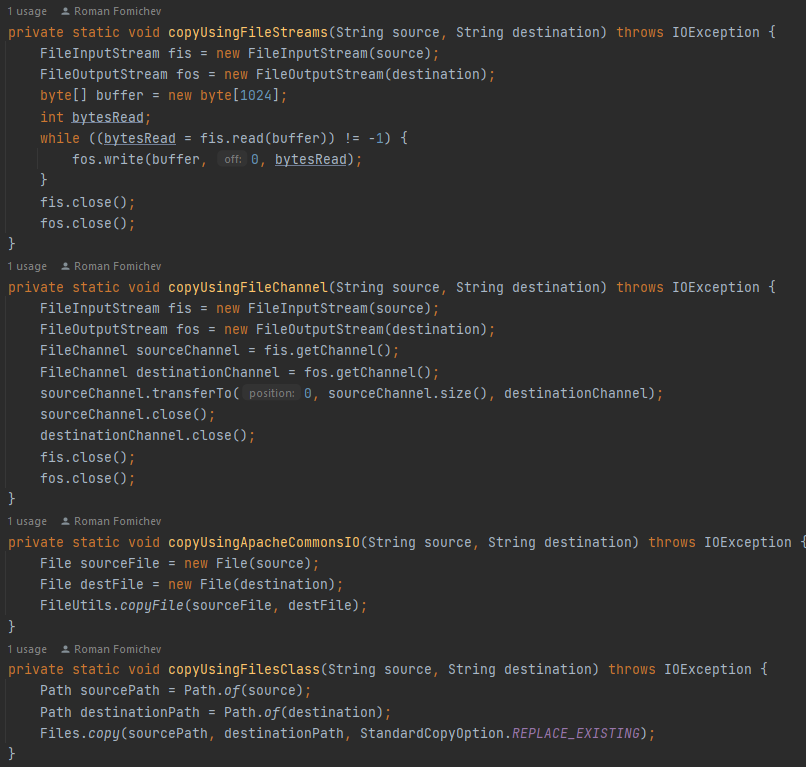


Рисунок 17 – Методы копирования файла

Эти методы последовательно вызываются из функции main(). Вызов функций представлен на рисунке 18.

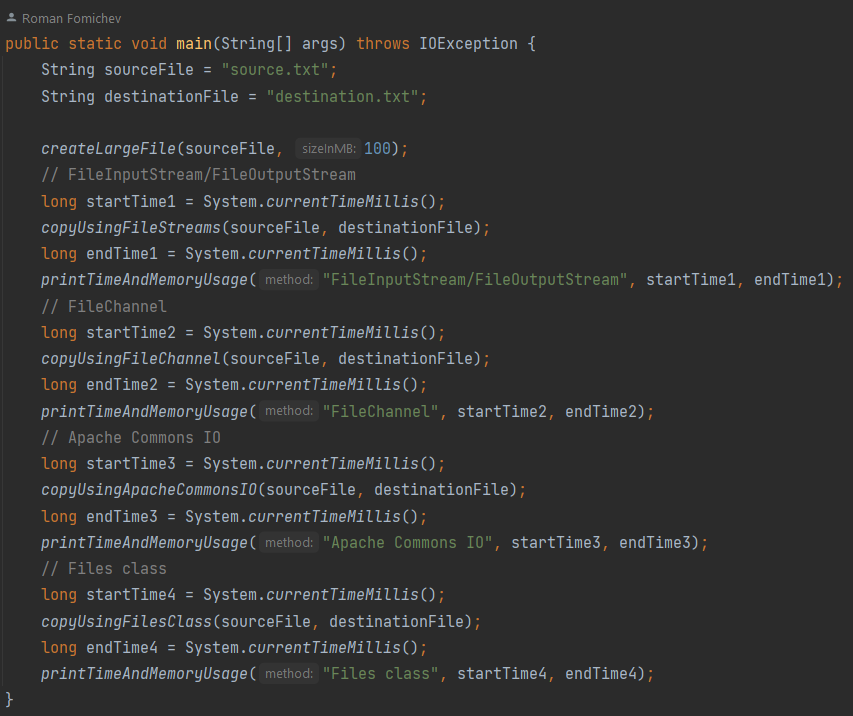


Рисунок 18 – Вызов функций копирования

Результат работы программы представлен на рисунке 19.

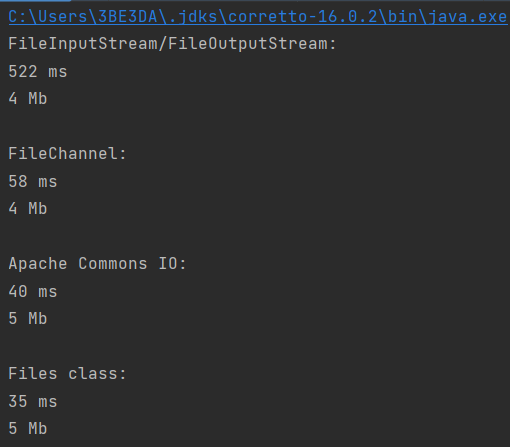


Рисунок 19 – Результат работы

**Задание 3**

Для выполнения третьего задания была создана функция подсчёта контрольной суммы файла. Код функции представлен на рисунке 20.

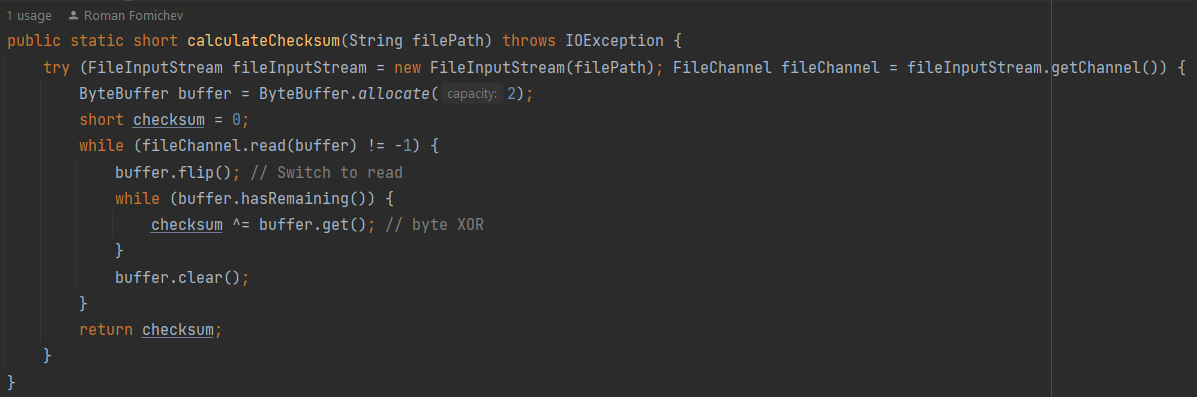


Рисунок 20 – Функция подсчёта контрольной суммы

Результат работы функции представлен на рисунке 21.

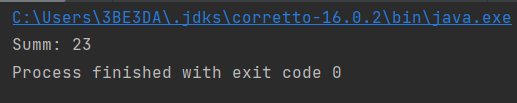


Рисунок 21 – Результат работы

**Задание 4**

Для четвертого задания были созданы функции для отслеживания изменений и чтения файла. Данные функции представлены на рисунке 22.

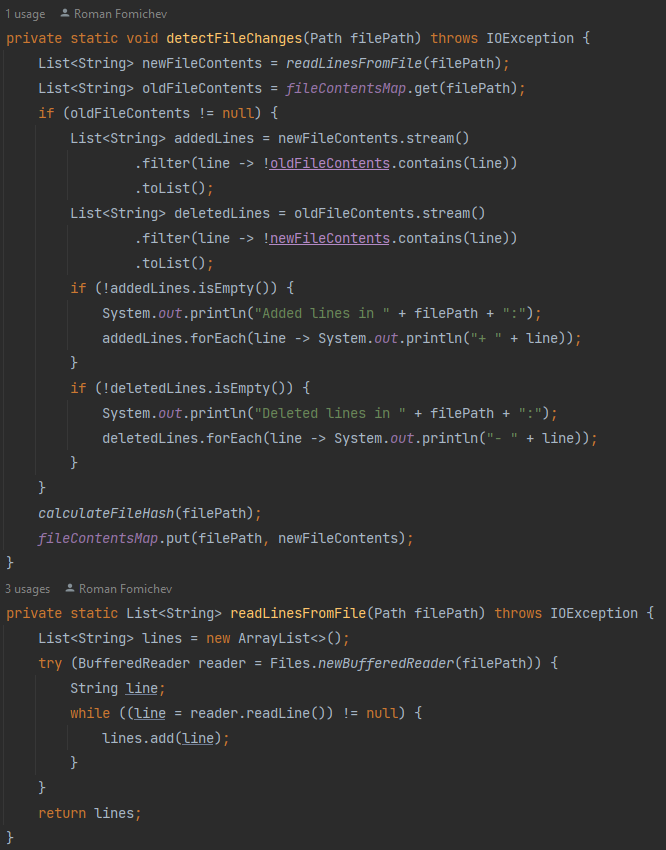


Рисунок 22 – Функции отслеживания изменений и чтения файла

Также была создана функция для подсчёта хэша, представленная на рисунке 23.



Рисунок 23 – Функция для подсчёта хэша

В функции main() происходит отслеживание изменений указанной директории, а также вызов соответствующих методов. Функция представлена на рисунке 24.

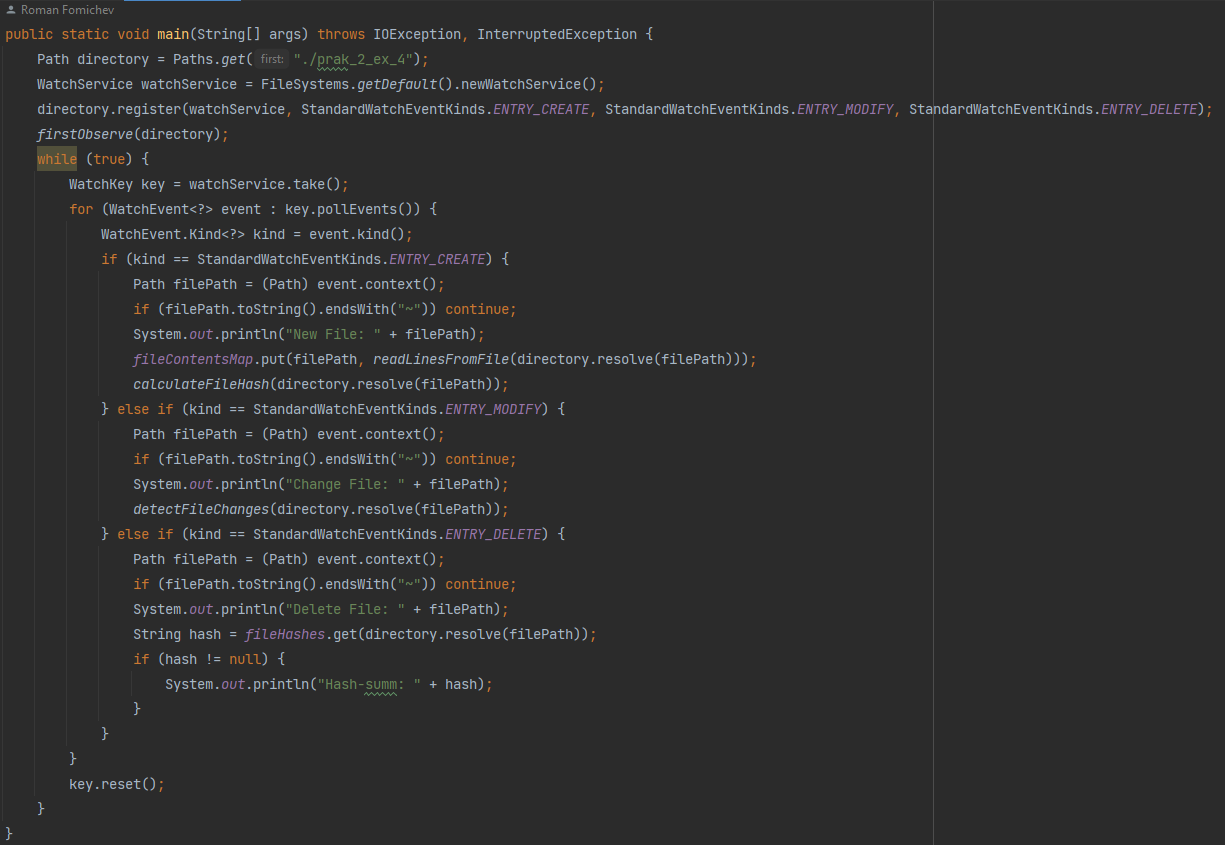


Рисунок 24 – Функция main

Результат работы программы представлен на рисунке 25.

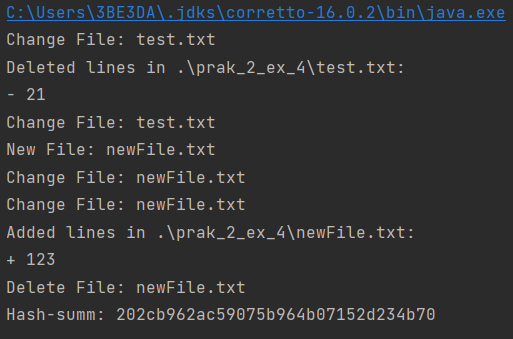


Рисунок 25 – Результат работы