|  |
| --- |
| https://lh6.googleusercontent.com/QcftzNtI05T0Y6fjdSh1Rr2rt8oqZ1IvnLvbn1jLJ7CCyteVir3k-xBLv4SL1wAgWJsRhmmJSR0UW-RP63_GQenE4vVWv05BRoZTsmIcBccVTnfxwmsnNMvjg599x9SqZd8E3dkd |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** |

Институт информационных технологий (ИТ)

Кафедра инструментального и прикладного программного обеспечения (ИиППО)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ** | | | | | |
| **по дисциплине «Разработка клиент-серверных приложений»** | | | | | |
|  | | | | | |
|  | | | | | |
| Выполнил студент группы ИКБО-20-19 | | | | Московка А.А. | |
|  | | | |  | |
| Принял  *Ассистент* | | Зарипов Е.А. | | |
| Практические работы выполнены | «\_\_\_»\_\_\_\_\_\_\_2022 г. | | (подпись студента) | | |
| «Зачтено» | «\_\_\_»\_\_\_\_\_\_\_2022 г. | | (подпись руководителя) | | |
|  |  | |  | | |

Москва 2022

**Практическая работа №1: Примитивы синхронизации и их использование при создании клиент серверных приложений**

**Ход работы:** для решения первой практической работы были написаны класс Main (Рисунок 1.1), три класса для функций (Рисунки 1.2-1.5), классы для реализации с помощью многопоточности (Рисунок 1.6) и с использованием метода ForkJoin (Рисунок 1.7), класс с используемыми инструментами Tools (Рисунок 1.8).

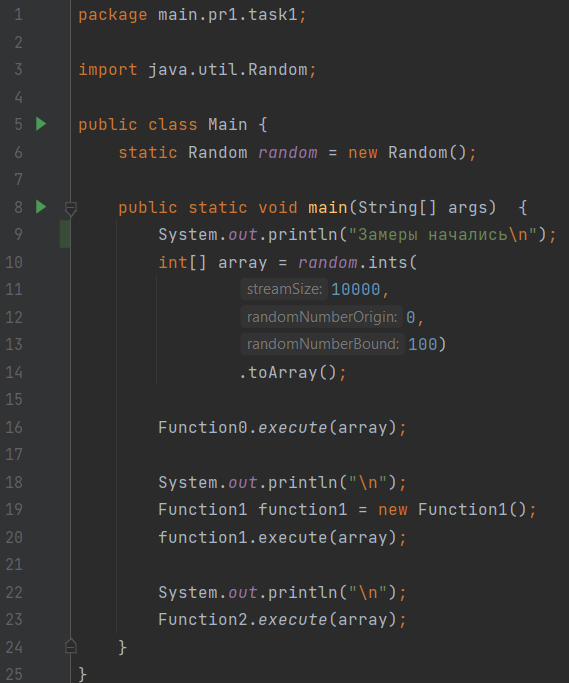


Рисунок 1.1 – Скриншот кода класса Main

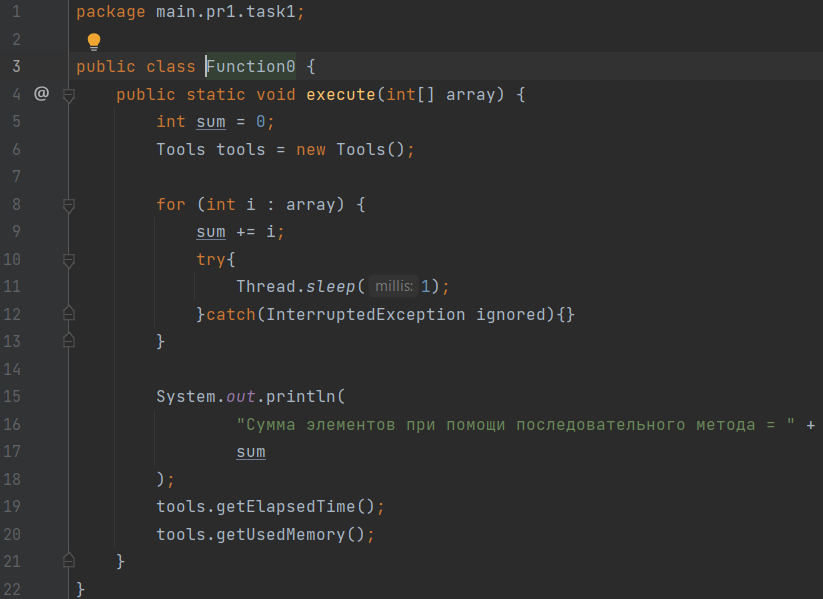


Рисунок 1.2 – Скриншот кода класса императивного решения задачи

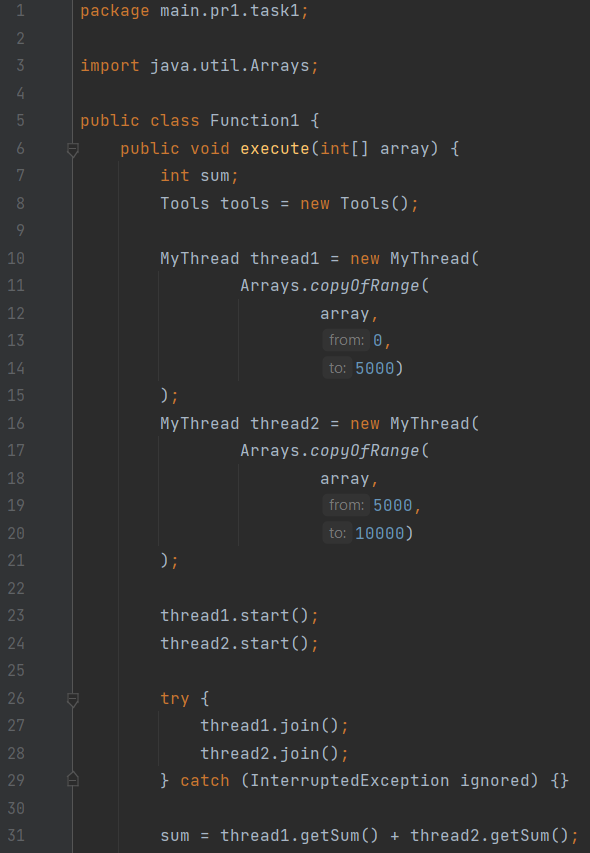


Рисунок 1.3 – Скриншот кода класса для реализации метода многопоточности (Часть 1)

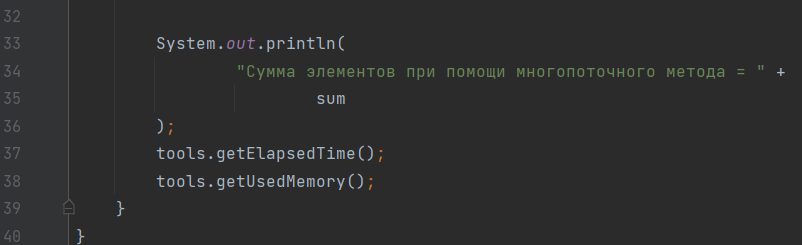


Рисунок 1.4 – Скриншот кода класса для реализации метода многопоточности (Часть 2)

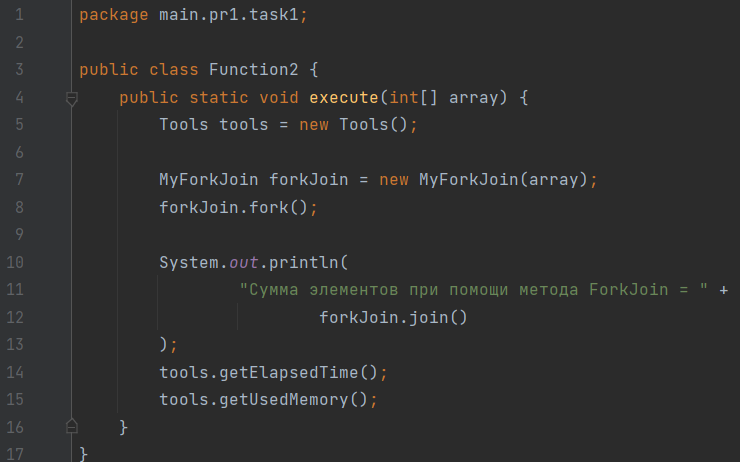


Рисунок 1.5 – Скриншот кода класса для реализации метода ForkJoin



Рисунок 1.6 – Скриншот кода класса наследника Thread

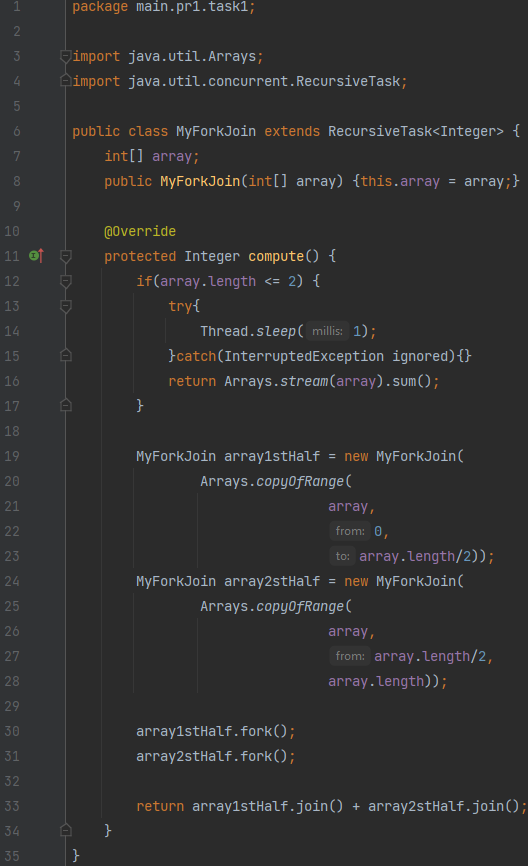


Рисунок 1.7 – Скриншот кода класса наследника для реализации метода ForkJoin



Рисунок 1.8 – Скриншот кода класса вспомогательных методов

После запуска первой программы и ожидания около одной минуты для всех методов выводятся результаты вычислений, что можно увидеть на следующем рисунке (Рисунок 1.9):

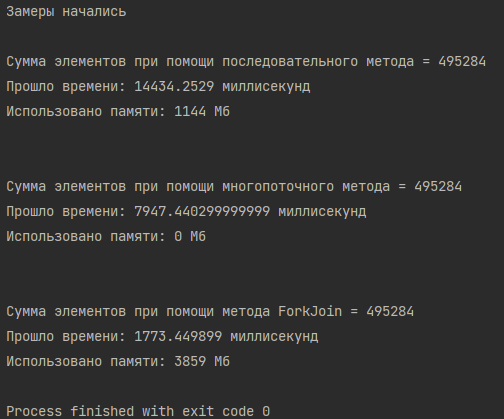


Рисунок 1.9 – Скриншот результата выполнения первой задачи

Затем производился запуск второй программы, в которой вводятся последовательно числа до получения результата вычислений, поскольку вычисления производятся с желаемой задержкой (1-5 секунд), результат выполнения задания представлен на рисунке 1.10.

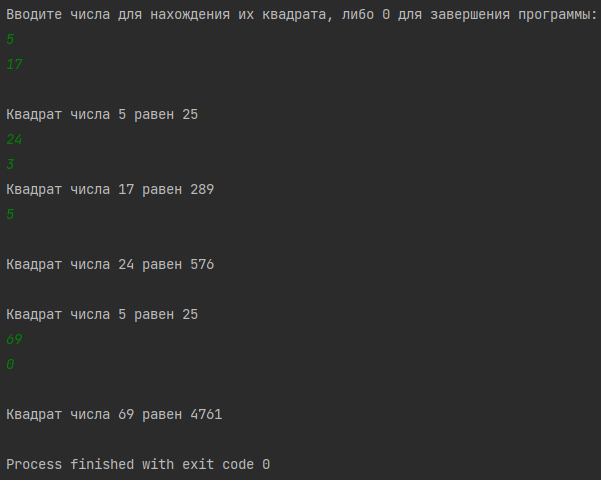


Рисунок 1.10 – Скриншот результата выполнения второй задачи

Наконец, запуск третьей программы производился с целью генерации и дальнейшей обработки ранее сгенерированных файлов в разных потоках, поскольку записей с новых строк больше, чем умещается на скриншоте, результат выполнения третьей задачи представлен на рисунках 1.11-1.12.

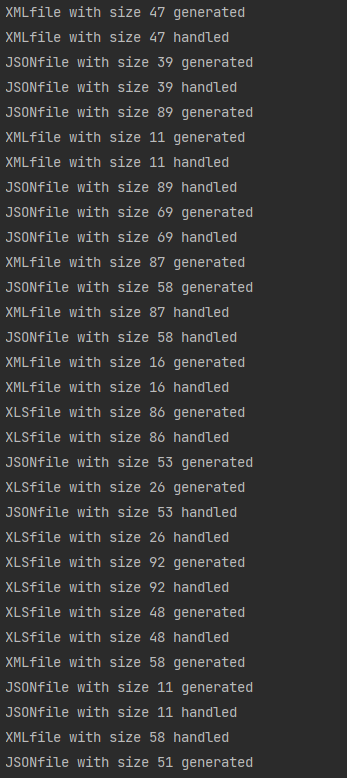


Рисунок 1.11 – Скриншот результата выполнения третьей задачи (Часть 1)

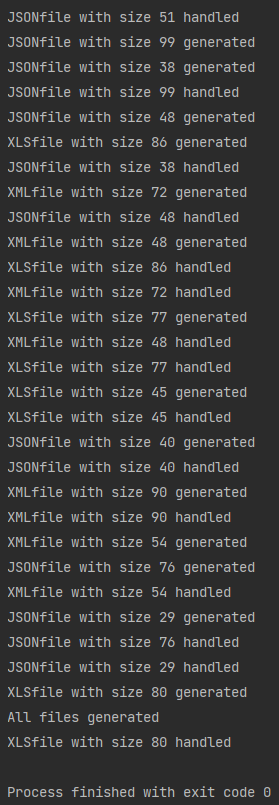


Рисунок 1.12 – Скриншот результата выполнения третьей задачи (Часть 2)

**Практическая работа №2: Java NIO**

**Ход работы:** для решения второй практической работы были написаны класс Main (Рисунок 2.1), четыре класса для функций (Рисунки 2.2-2.7), классы для реализации различных методов копирования во втором задании и вспомогательные интерфейс Example и Timer для подсчета времени (Рисунки 2.8-2.13).

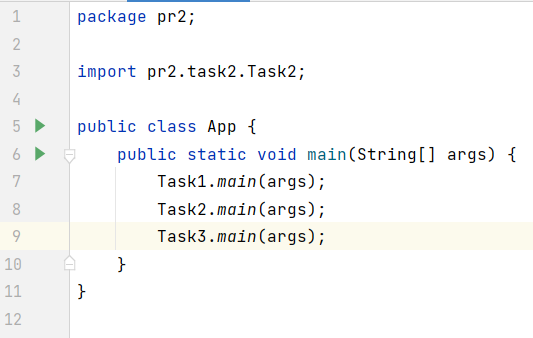


Рисунок 2.1 – Скриншот кода класса Main



Рисунок 2.2 – Скриншот кода класса для решения первого задания



Рисунок 2.3 – Скриншот кода класса для решения второго задания

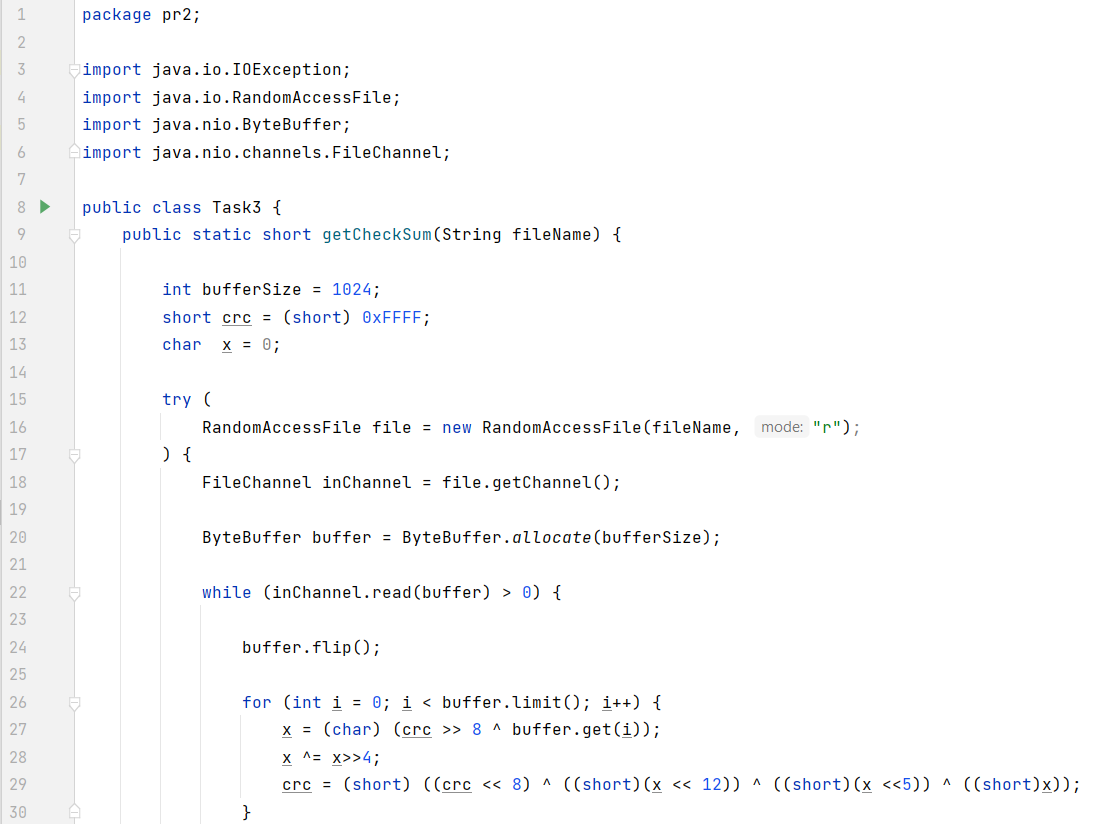


Рисунок 2.4 – Скриншот кода класса для реализации третьей задачи (Часть 1)

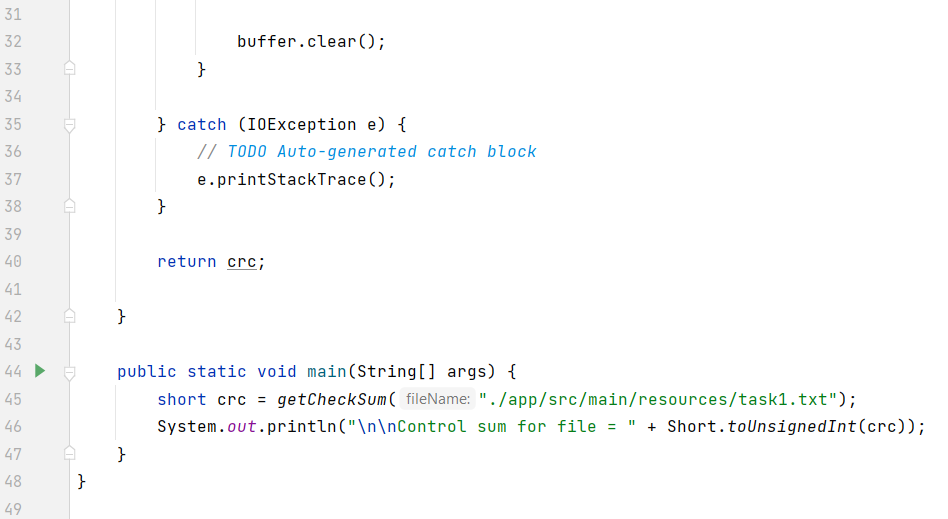


Рисунок 2.5 – Скриншот кода класса для реализации третьей задачи (Часть 2)



Рисунок 2.6 – Скриншот кода класса реализации четвертой задачи (Часть 1)

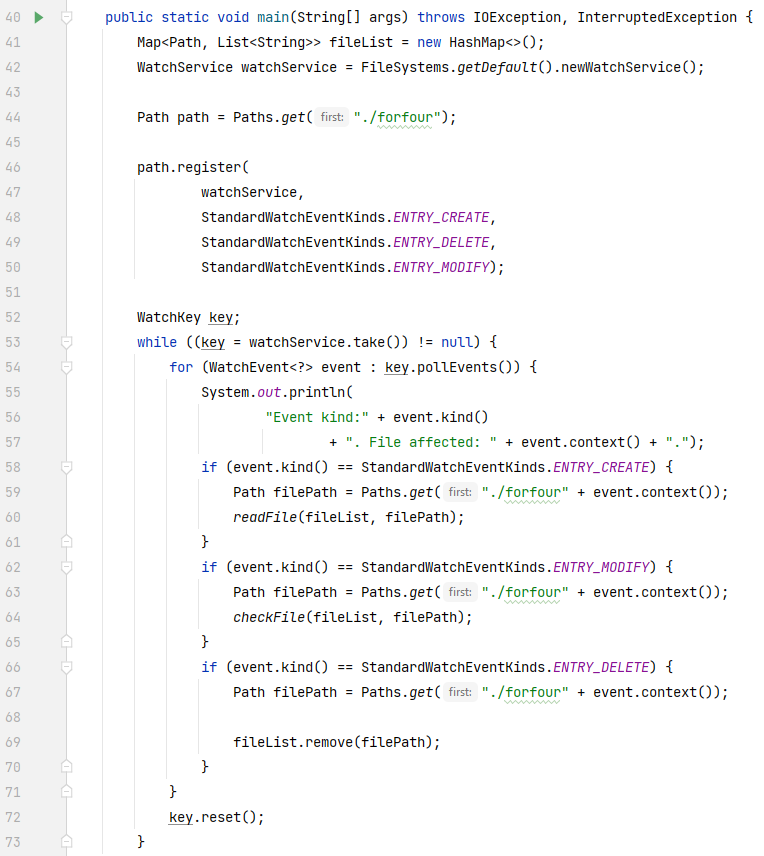


Рисунок 2.7– Скриншот кода класса реализации четвертой задачи (Часть 2)



Рисунок 2.8 – Скриншот кода класса реализации метода Apache Commons IO

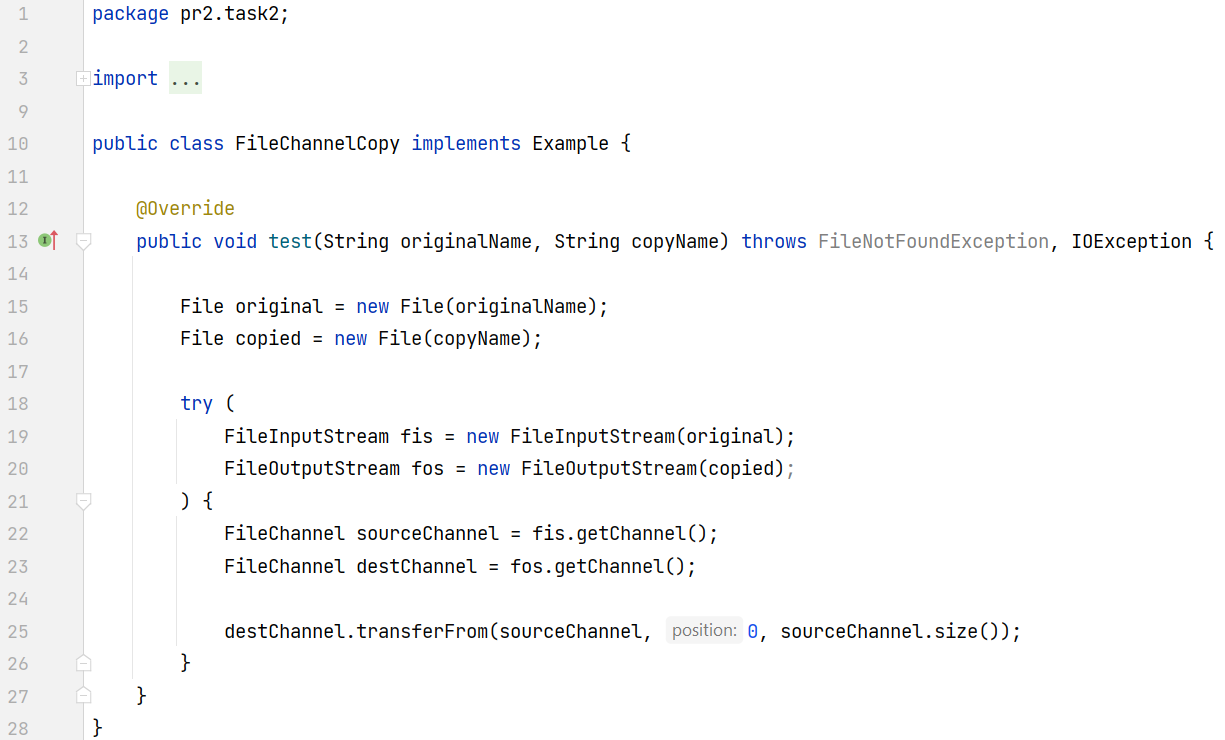


Рисунок 2.9 – Скриншот кода класса реализации метода FileChannel



Рисунок 2.10 – Скриншот кода класса реализации метода File



Рисунок 2.11 – Скриншот кода класса реализации метода FileInputStream/FileOutputStream

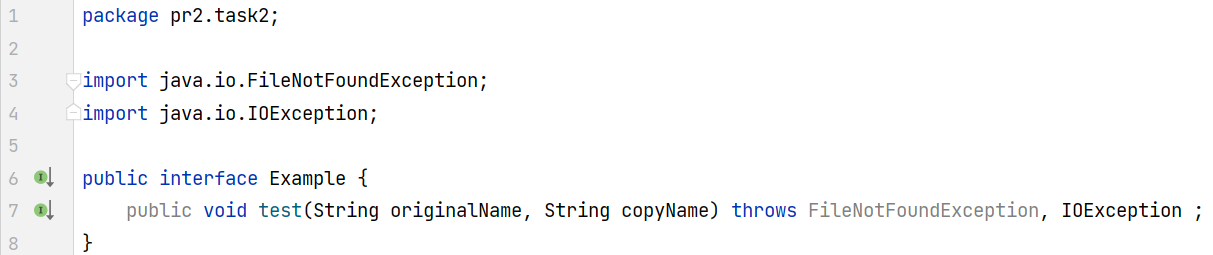


Рисунок 2.12 – Скриншот кода интерфейса Example

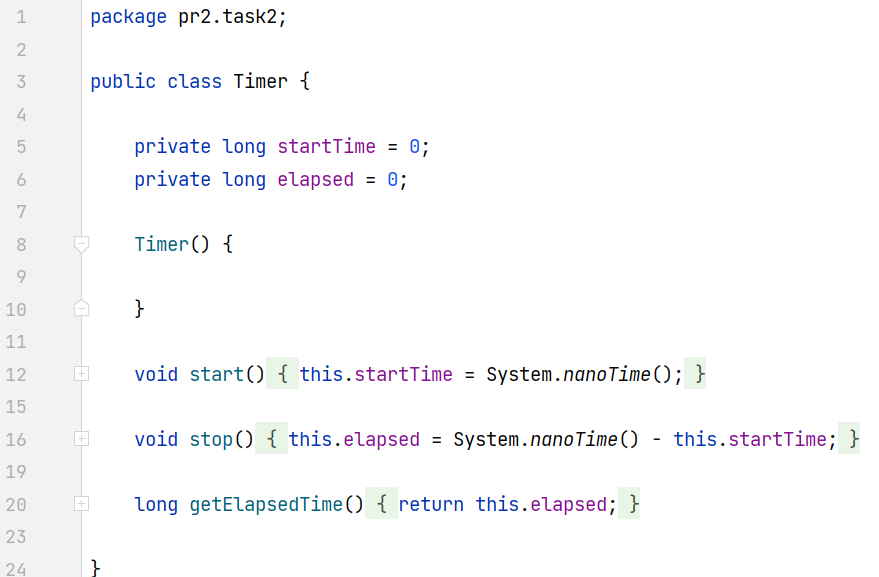


Рисунок 2.13 – Скриншот кода вспомогательного класса Timer

После запуска метода main класса App.java происходит выполнение первого, второго и третьего заданий, что изображено на рисунках 2.14-2.15.

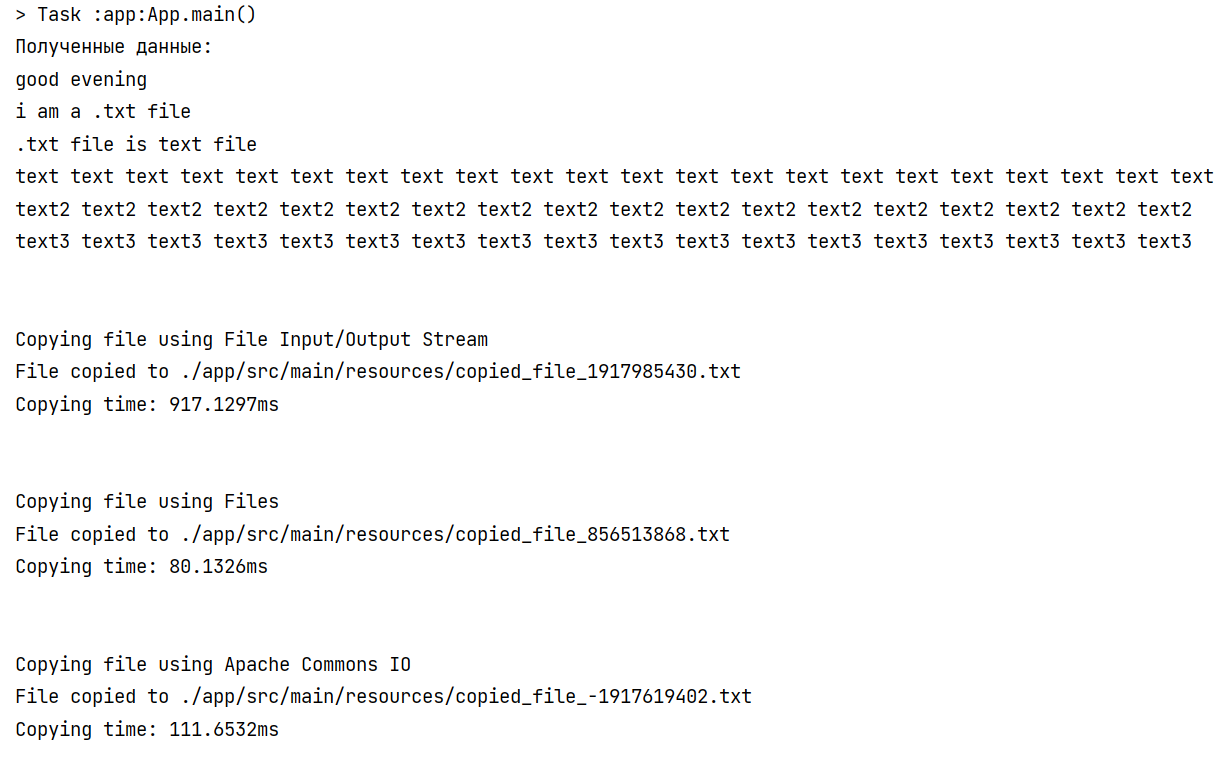


Рисунок 2.14 – Скриншот успешного запуска первого, второго и третьего заданий (Часть 1)

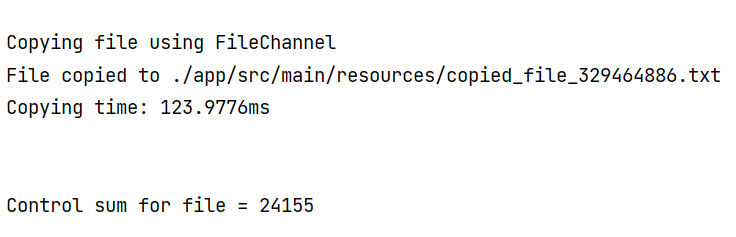


Рисунок 2.15 – Скриншот успешного запуска первого, второго и третьего заданий (Часть 2)

Наконец, после запуска четвертой задачи был создан текстовый документ в соответствующей папке, переименован, изменен и затем удален – все это отражено в консоли, что можно видеть на рисунке 2.16.

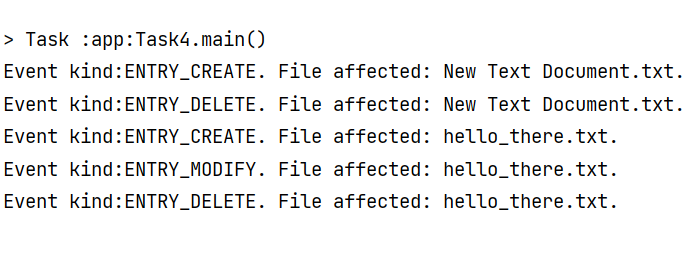


Рисунок 2.16 – Скриншот успешного запуска и работы четвертого задания

**Вывод:** в ходе выполнения практических работ по дисциплине «Разработка клиент-серверных приложений» были получены полезные теоретические, а главное практические навыки создания приложений, использования многопоточности разными методами (наследованием классов Thread и ForkJoin), создания и обработки файлов, создания и обработки файлов, использования пакета java.nio, а также реализаций копирования методами FileInputStream/FileOutputStream, FileChannel, Apache Commons IO, Files class.