МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ордена трудового Красного Знамени федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра Математическая кибернетика и информационные технологии

Отчет по лабораторной работе № 8

Выполнил: студент группы БВТ2402

Косякова Олеся Дмитриевна

Москва, 2025

Цель работы: Освоение практического применения аннотаций и API Stream Java для решения типовых задач обработки информации.

Задание:

Разработать приложение, которое считывает данные из ис ходного источника (например, файл, база данных или сетевой ресурс), применяет к ним различные операции с использованием Stream API и сохраняет результаты в новый источник данных

Ход работы:

Создаем нашу аннотацию, для этого нужно определить аннотатацию target, которая указывает на то, где эта аннотация будет использоваться и Retention, который говорит, сохранить аннотацию до запуска программы

```
@Retention(RetentionPolicy.RUNTIME) /
@Target(ElementType.METHOD) //указыва
public @interface DataProcessor {
}
```

Далее создаем список с нашими обработчиками, исходными данными и обработанными данными. Реализуем метод регистрации объекта, загрузки данных из исходного источника. Получаем путь к файлу и читаем все строки.

```
public class DataManager {

    private final List<Object> processors = new ArrayList<>();
    private List<String> data = new ArrayList<>();
    private List<String> processedData = new ArrayList<>();

    //регистрация объекта
    public void registerDataProcessor(Object processor) {
        processors.add(processor);
    }

    //загружаем данные из исходного источника
    public void loadData(String source) {
        try {
            data = Files.readAllLines(Paths.get(source)); //пас
            System.out.println("Данные загружены: " + data);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

Реализуем метод многопоточной обработки данных. Сначала создаем переменную храняющую в себе ссылку на объект управляющий пулом потоков. Далее список для хранения будущих результатов асинхронных задач наших потоков, результат будет представлен в виде списка строк.

Далее перебираем все обработчики и проходимся по всем методам каждого, и если метод имеет нашу аннотацию, то мы отправляем нашему пулу потоков задачу и добавляем в список future, переопределяем метод call, который будет вызывать наш метод с данными которые мы подали и возвращать результат в виде списка. И возвращаем пустой список даже если произошла ошибка

```
public void processData() {
    ExecutorService executor = Executors.newFixedThreadPool(nThreads: 4); //создаем пе
    List<Future<List<String>>> futures = new ArrayList<>(); //создаем список для хран

    for (Object processor : processors) { //перебираем все обработчики
        for (Method method : processor.getClass().getDeclaredMethods()) { //находим в
            if (method.isAnnotationPresent(annotationClass: DataProcessor.class)) {

                futures.add(executor.submit(() -> { //добавляем в список future задач
                    try {
                        // вызываем метод-обработчик
                        return (List<String>) method.invoke(processor, data);
                        //вызываем метод-обработчик у объекта с определенными данными
                        //возвращем список строк тк мы уверенны в это
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                    return new ArrayList<String>();
                    //возвращает пустой список если произошла ошибка
                }));
            }
        }
    }
```

Далее в переменной обработанных данных превращаем список future в поток и дальше в каждом future дожидаемся результата выполнения задач пулом потока и превращем этот список в поток, а далее собираем поток обратно в список. Далее завершаем работу пула потоков чтобы он больше не ждал задач.

```
processedData = futures.stream() //превращаем список в поток для дальн
        .flatMap(f -> { //flatMap означает что для каждого future выпо
            try { return f.get().stream(); }
            catch (Exception e) { return Stream.<String>empty(); }
        })
        .collect(Collectors.toList()); //собираем все элементы потока

executor.shutdown(); //завершаем работу пул потоков, обозначив что зад
System.out.println("Обработанные данные: " + processedData);
```

```
Данные загружены: [hi, hello, java, yo, stream, api]
Обработанные данные: [hello, java, stream, HI, HELLO, JAVA, YO,
 STREAM, API]
Результат сохранён в lab8/output.txt
PS C:\Users\Олеся\it>
```

Вывод: В ходе лабораторной работы мы освоили аннотации и API Stream для решения типовых задач обработки информации.

Ссылка на репозиторий: https://github.com/swydiz/information_technology