Федеральное государственное
автономное учебное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Мегафакультет компьютеных технологий и управления
Факультет программной инженерии и компьютерной техники

Отчёт
по лабораторной работе №6
по дисциплине «Программирование»

Вариант 31209

Группа: P3118

Студенты: Иванова Александра Юрьевна,
Кожухин Иван Алексеевич

Преподаватель: Кулинич Ярослав Вадимович

Санкт-Петербург
2023

Содержание

1 Задание

Разделить программу из лабораторной работы №5 на клиентский и серверный модули. Серверный модуль должен осуществлять выполнение команд по управлению коллекцией. Клиентский модуль должен в интерактивном режиме считывать команды, передавать их для выполнения на сервер и выводить результаты выполнения.

Необходимо выполнить следующие требования:

- Операции обработки объектов коллекции должны быть реализованы с помощью Stream API с использованием лямбда-выражений.
- Объекты между клиентом и сервером должны передаваться в сериализованном виде.
- Объекты в коллекции, передаваемой клиенту, должны быть отсортированы по местоположению
- Клиент должен корректно обрабатывать временную недоступность сервера.
- Обмен данными между клиентом и сервером должен осуществляться по протоколу UDP
- Для обмена данными на сервере необходимо использовать **датаграммы**
- Для обмена данными на клиенте необходимо использовать **сетевой канал**
- Сетевые каналы должны использоваться в неблокирующем режиме.

Рис. 1: Текст задания варианта, ч. 1

Обязанности серверного приложения:

- Работа с файлом, хранящим коллекцию.
- Управление коллекцией объектов.
- Назначение автоматически генерируемых полей объектов в коллекции.
- Ожидание подключений и запросов от клиента.
- Обработка полученных запросов (команд).
- Сохранение коллекции в файл при завершении работы приложения.
- Сохранение коллекции в файл при исполнении специальной команды, доступной только серверу (клиент такую команду отправить не может).

Серверное приложение должно состоять из следующих модулей (реализованных в виде одного или нескольких классов):

- Модуль приёма подключений.
- Модуль чтения запроса.
- Модуль обработки полученных команд.
- Модуль отправки ответов клиенту.

Рис. 2: Текст задания варианта, ч. 2

Сервер должен работать в **однопоточном** режиме.

Обязанности клиентского приложения:

- Чтение команд из консоли.
- Валидация вводимых данных.
- Сериализация введённой команды и её аргументов.
- Отправка полученной команды и её аргументов на сервер.
- Обработка ответа от сервера (вывод результата исполнения команды в консоль).
- Команду `save` из клиентского приложения необходимо убрать.
- Команда `exit` завершает работу клиентского приложения.

Важно! Команды и их аргументы должны представлять из себя объекты классов. Недопустим обмен "простыми" строками. Так, для команды add или её аналога необходимо сформировать объект, содержащий тип команды и объект, который должен храниться в вашей коллекции.

Дополнительное задание:
Реализовать логирование различных этапов работы сервера (начало работы, получение нового подключения, получение нового запроса, отправка ответа и т.п.) с помощью **Logback**

Рис. 3: Текст задания варианта, ч. 3

2 Выполнение работы

Ссылка на репозиторий на GitHub

Ссылка диаграмму классов программы
(перейдя по ссылке, следует нажать «Открыть в приложении»)

3 Вывод

В ходе выполнения лабораторной работы были изучены: сетевое взаимодействие (клиент-серверная архитектура, основные протоколы, их сходства и отличия); протокол TCP, классы Socket и ServerSocket; протокол UDP, классы DatagramSocket и DatagramPacket; отличия блокирующего и неблокирующего ввода-вывода, их преимущества и недостатки, работа с сетевыми каналами; классы SocketChannel и DatagramChannel, передача данных по сети, сериализация объектов; интерфейс Serializable, объектный граф, сериализация и десериализация полей и методов; Java Stream API, создание конвейеров, промежуточные и терминальные операции; шаблоны проектирования: Decorator, Iterator, Factory method, Command, Flyweight, Interpreter, Singleton, Strategy, Adapter, Facade, Proxy.