

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ «Санкт-Петербургский национальный исследовательский
университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ
ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №6

по дисциплине

«ПРОГРАММИРОВАНИЕ»

Вариант № 3116100

Выполнил:

Студент группы Р3116

Билошицкий Михаил Владимирович

Преподаватель:

Гаврилов Антон Валерьевич

Санкт-Петербург, 2023

Содержание

Задание.....	3
Исходный код программы	5
UML диаграмма классов	6
Вывод.....	8

Задание

Лабораторная работа #6

Введите вариант: 31161

Внимание! У разных вариантов разный текст задания!

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 имени.
- 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.

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.

Сервер должен работать в **однопоточном** режиме.

Обязанности клиентского приложения:

- Чтение команд из консоли.
- Валидация вводимых данных.
- Сериализация введенной 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**

Отчёт по работе должен содержать:

1. Текст задания.
2. Диаграмма классов разработанной программы (как клиентского, так и серверного приложения).
3. Исходный код программы.
4. Выводы по работе.

Вопросы к защите лабораторной работы:

1. Сетевое взаимодействие – клиент-серверная архитектура, основные протоколы, их сходства и отличия.
2. Протокол TCP. Классы `Socket` и `ServerSocket`.
3. Протокол UDP. Классы `DatagramSocket` и `DatagramPacket`.
4. Отличия блокирующего и неблокирующего ввода-вывода, их преимущества и недостатки. Работа с сетевыми каналами.
5. Классы `SocketChannel` и `DatagramChannel`.
6. Передача данных по сети. Сериализация объектов.
7. Интерфейс `Serializable`. Объектный граф, сериализация и десериализация полей и методов.
8. Java Stream API. Создание конвейеров. Промежуточные и терминальные операции.
9. Шаблоны проектирования: Decorator, Iterator, Factory method, Command, Flyweight, Interpreter, Singleton, Strategy, Adapter, Facade, Proxy.

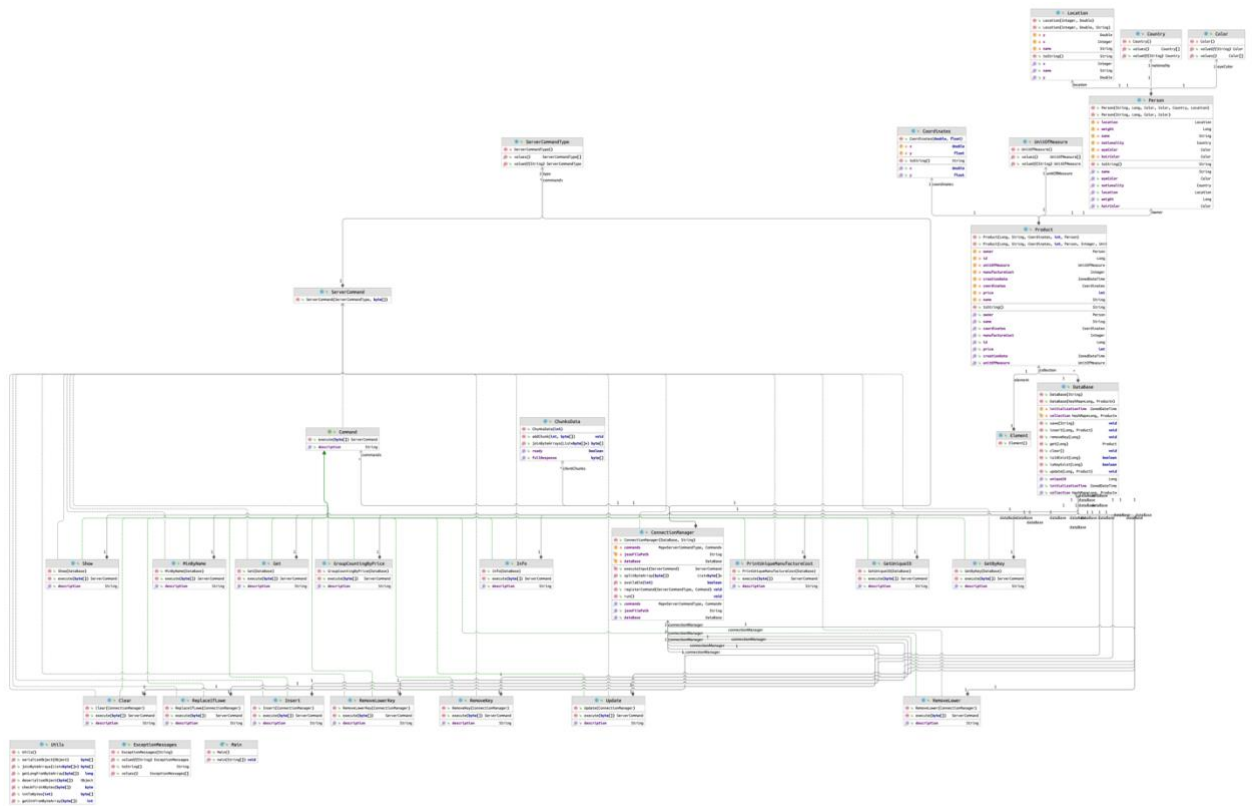
Исходный код программы

Ссылка на GitHub с исходным кодом:

https://github.com/michael-bill/labs_programming_itmo/tree/main/Laba6

UML диаграмма классов

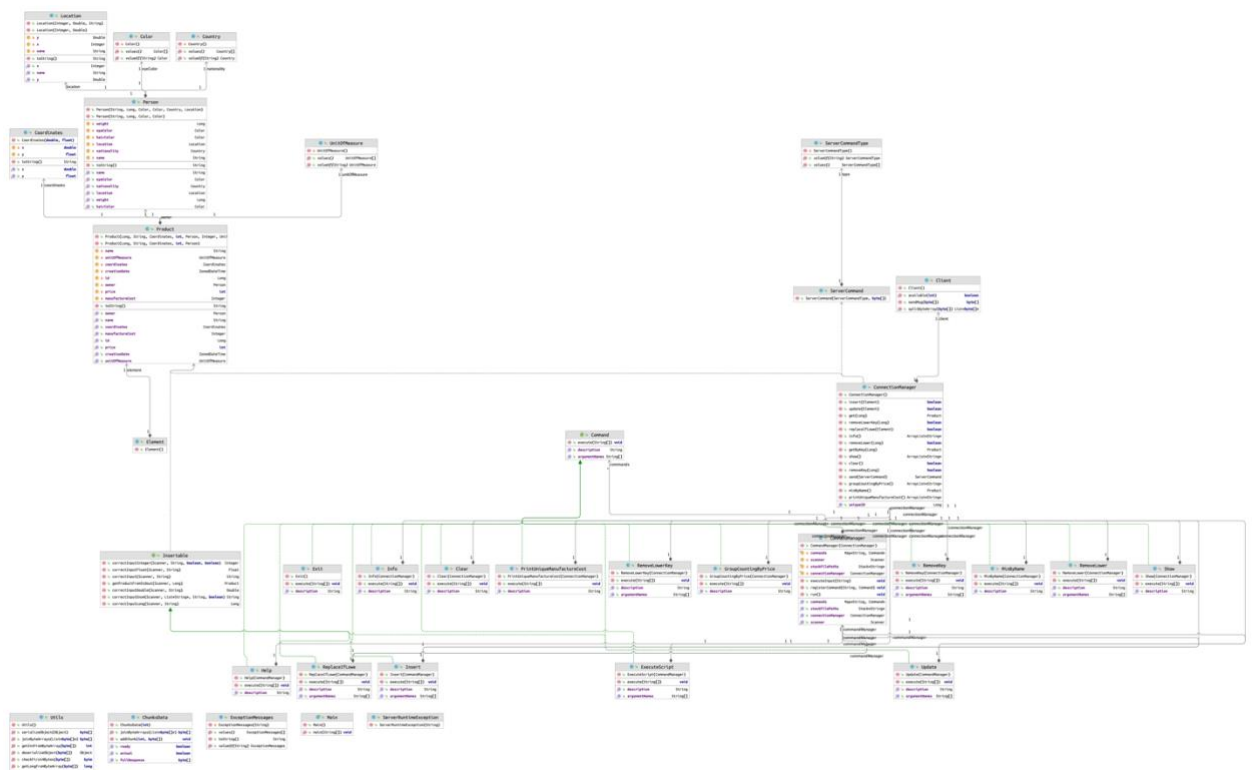
Серверное приложение.



Чтобы ознакомиться с ней поближе, ссылка на .png файл также прикреплена в GitHub репозитории.

https://github.com/michael-bill/labs_programming_itmo/blob/main/Laba6/laba6_server_diagram.png

Клиентское приложение.



Чтобы ознакомиться с ней поближе, ссылка на .png файл также прикреплена в GitHub репозитории.

https://github.com/michael-bill/labs_programming_itmo/blob/main/Laba6/laba6_client_diagram.png

Вывод

В ходе выполнения лабораторной работы я научился писать клиент серверные приложения, научился работать с двумя самыми популярными сетевыми протоколами TCP и UDP, узнал про сериализацию и десериализацию объектов и научился писать приложение в неблокирующем режиме. Полученные знания пригодятся мне в будущем.