Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторные работы по Программированию №5, №6, №7

Вариант №367786

Работу выполнил:
Агаев Х. Р.
Группа:
P3234

Санкт-Петербург,
2024

СОДЕРЖАНИЕ

Стр.

1 Текст задания

Реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. В коллекции необходимо хранить объекты класса Vehicle, описание которого приведено ниже.

Разработанная программа должна удовлетворять следующим требованиям:

- Класс, коллекцией экземпляров которого управляет программа, должен реализовывать сортировку по умолчанию.
- Все требования к полям класса (указанные в виде комментариев) должны быть выполнены.
- Программа должна корректно работать с неправильными данными (ошибки пользовательского ввода, отсутсвие прав доступа к файлу и т.п.).

В интерактивном режиме программа должна поддерживать выполнение следующих команд:

- help : вывести справку по доступным командам
- info : вывести в стандартный поток вывода информацию о коллекции (тип, дата инициализации, количество элементов и т.д.)
- show : вывести в стандартный поток вывода все элементы коллекции в строковом представлении
- add element : добавить новый элемент в коллекцию
- update id element : обновить значение элемента коллекции, id которого равен заданному
- remove_by_id id : удалить элемент из коллекции по его id
- clear : очистить коллекцию
- execute_script file_name : считать и исполнить скрипт из указанного файла. В скрипте содержатся команды в таком же виде, в котором их вводит пользователь в интерактивном режиме.
- exit : завершить программу
- remove_at index : удалить элемент, находящийся в заданной позиции коллекции (index)
- add_if_min element : добавить новый элемент в коллекцию, если его значение меньше, чем у наименьшего элемента этой коллекции

- sort : отсортировать коллекцию в естественном порядке
- sum_of_number_of_wheels : вывести сумму значений поля numberOfWheels для всех элементов коллекции
- max_by_name : вывести любой объект из коллекции, значение поля name которого является максимальным
- filter_contains_name name : вывести элементы, значение поля name которых содержит заданную подстроку

Формат ввода команд:

- Все аргументы команды, являющиеся стандартными типами данных (примитивные типы, классы-оболочки, String, классы для хранения дат), должны вводиться в той же строке, что и имя команды.
- Все составные типы данных (объекты классов, хранящиеся в коллекции) должны вводиться по одному полю в строку.
- При вводе составных типов данных пользователю должно показываться приглашение к вводу, содержащее имя поля (например, "Введите дату рождения:").
- Если поле является enum'ом, то вводится имя одной из его констант (при этом список констант должен быть предварительно выведен).
- При некорректном пользовательском вводе (введена строка, не являющаяся именем константы в enum'е; введена строка вместо числа; введённое число не входит в указанные границы и т.п.) должно быть показано сообщение об ошибке и предложено повторить ввод поля.
- Для ввода значений null использовать пустую строку.
- Поля с комментарием "Значение этого поля должно генерироваться автоматически"не должны вводиться пользователем вручную при добавлении.

Описание хранимых в коллекции классов:

Листинг 1.1: Классы приложения

```
public class Vehicle {
    private int id; //Значение поля должно быть больше
      0, Значение этого поля должно быть уникальным, З
```

```
        начение этого поля должно генерироваться автомат
        ически
    private String name; //Поле не может быть null, Стр
        ока не может быть пустой
    private Coordinates coordinates; //Поле не может бы
        ть null
    private java.time.ZonedDateTime creationDate;
        //Поле не может быть null, Значение этого поля д
        олжно генерироваться автоматически
    private Integer enginePower; //Поле может быть
        null, Значение поля должно быть больше 0
    private Long numberOfWheels; //Поле не может быть
        null, Значение поля должно быть больше 0
    private int capacity; //Значение поля должно быть б
        ольше 0
    private VehicleType type; //Поле может быть null
}
public class Coordinates {
    private double x; //Значение поля должно быть больш
        е -832
    private int y; //Максимальное значение поля: 802
}
public enum VehicleType {
    PLANE,
    BOAT,
    SPACESHIP;
}
```

Программа делится 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 умолчанию.
- 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:

- Управление коллекцией объектов.
- Назначение автоматически генерируемых полей объектов в коллекции.
- Ожидание подключений и запросов от клиента.
- Обработка полученных запросов (команд).

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.

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).
- 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.

Доработать программу следующим образом:

- Организовать хранение коллекции в реляционной СУБД (PostgresQL).
- Для генерации поля id использовать средства базы данных (sequence).
- Обновлять состояние коллекции в памяти только при успешном добавлении объекта в БД.
- Все команды получения данных должны работать с коллекцией в памяти, а не в БД.
- Организовать возможность регистрации и авторизации пользователей. У пользователя есть возможность указать пароль.
- Пароли при хранении хэшировать алгоритмом MD2.
- Запретить выполнение команд не авторизованным пользователям.
- При хранении объектов сохранять информацию о пользователе, который создал этот объект.
- Пользователи должны иметь возможность просмотра всех объектов коллекции, но модифицировать могут только принадлежащие им.
- Для идентификации пользователя отправлять логин и пароль с каждым запросом.

Необходимо реализовать многопоточную обработку запросов.

- Для многопоточного чтения запросов использовать ForkJoinPool.
- Для многопотчной обработки полученного запроса использовать Fixed thread pool.
- Для многопоточной отправки ответа использовать ForkJoinPool.
- Для синхронизации доступа к коллекции использовать потокобезопасные аналоги коллекции из java.util.concurrent.

2 Диаграмма классов разработанной программы

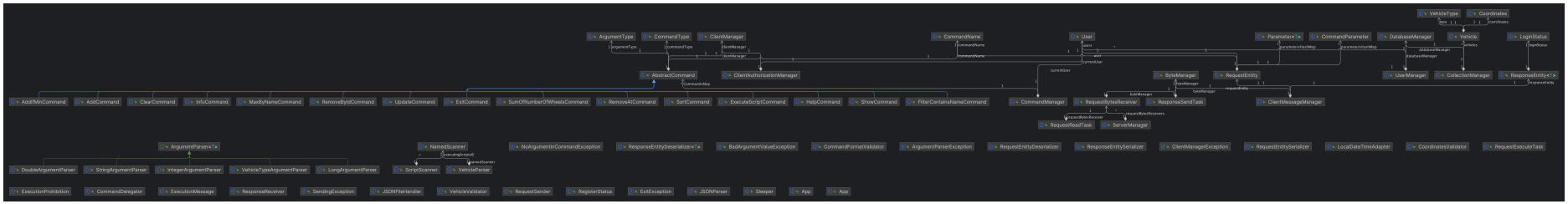

Рисунок 2.1 — Диаграмма классов разработанной программы

Также диаграмма доступна по ссылке: `https://github.com/HamzaAgaev/ICS_ITMO/blob/main/2nd_semester/programming/lab_7/uml_diagram.png`.

3 Исходный код программы

Исходный код программы находится в Github: `https://github.com/HamzaAgaev/ICS_ITMO/tree/main/2nd_semester/programming/lab_7`.

ЗАКЛЮЧЕНИЕ

Разработанная система представляет собой многомодульное приложение для управления коллекцией объектов класса Vehicle в интерактивном режиме, включающее клиентский и серверный модули. Основываясь на требованиях, были реализованы функции добавления, обновления, удаления и сортировки объектов, а также обработка некорректного пользовательского ввода и взаимодействие с базой данных PostgreSQL для хранения данных коллекции.

Программа обеспечивает многопоточную обработку запросов, используя различные пулы потоков, что способствует повышению производительности и эффективности обработки данных. Кроме того, реализована система регистрации и авторизации пользователей, с возможностью просмотра и модификации данных согласно правам доступа.