Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Программирование

Лабораторная работа №5

Работу выполнил:

Бавыкин Роман

Группа: Р3110

Вариант 311680

Санкт-Петербург

2020 г.

1. Текст задания.

Реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. В коллекции необходимо хранить объекты класса Person, 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.
* Для хранения необходимо использовать коллекцию типа java.util.TreeSet
* При запуске приложения коллекция должна автоматически заполняться значениями из файла.
* Имя файла должно передаваться программе с помощью: **переменная окружения**.
* Данные должны храниться в файле в формате xml
* Чтение данных из файла необходимо реализовать с помощью класса java.io.InputStreamReader
* Запись данных в файл необходимо реализовать с помощью класса java.io.OutputStreamWriter
* Все классы в программе должны быть задокументированы в формате javadoc.
* 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
* save : сохранить коллекцию в файл
* 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 (без сохранения в файл)
* add\_if\_max {element} : добавить новый элемент в коллекцию, если его значение превышает значение наибольшего элемента этой коллекции
* add\_if\_min {element} : добавить новый элемент в коллекцию, если его значение меньше, чем у наименьшего элемента этой коллекции
* history : вывести последние 9 команд (без их аргументов)
* max\_by\_eye\_color : вывести любой объект из коллекции, значение поля eyeColor которого является максимальным
* filter\_contains\_name name : вывести элементы, значение поля name которых содержит заданную подстроку
* print\_field\_descending\_eye\_color : вывести значения поля eyeColor всех элементов в порядке убывания\

**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:**
* public class Person {
* private int id; //Значение поля должно быть больше 0, Значение этого поля должно быть уникальным, Значение этого поля должно генерироваться автоматически
* private String name; //Поле не может быть null, Строка не может быть пустой
* private Coordinates coordinates; //Поле не может быть null
* private java.time.LocalDateTime creationDate; //Поле не может быть null, Значение этого поля должно генерироваться автоматически
* private long height; //Значение поля должно быть больше 0
* private java.time.LocalDateTime birthday; //Поле не может быть null
* private Color eyeColor; //Поле не может быть null
* private Color hairColor; //Поле не может быть null
* private Location location; //Поле не может быть null
* }
* public class Coordinates {
* private Double x; //Максимальное значение поля: 882, Поле не может быть null
* private long y; //Значение поля должно быть больше -266
* }
* public class Location {
* private float x;
* private long y;
* private String name; //Длина строки не должна быть больше 480, Поле может быть null
* }
* public enum Color {
* BLACK,
* ORANGE,
* WHITE;
* }
* public enum Color {
* GREEN,
* BLACK,
* BLUE,
* YELLOW,
* WHITE;
* }

2, 3. Исходный код и диаграмма классов разработанной программы:

[robqqq Lab5 (github.com)](https://github.com/robqqq/ITMO/tree/master/Second%20semestre/Programming/Lab5)

4. Выводы по работе.

Во время выполнения лабораторной работы научился работать с коллекциями в Java, применял не только коллекцию TreeSet, реализация работы с которой была необходима по заданию, но и такие коллекции, как HashMap – для хранения соответствий между объектом команды и строкой ее вызова, HashSet – для хранения всех id элементов и хранения списка запущенных скриптов, чтобы исключить случаи конфликта id элементов и предотвращения вызова рекурсии с помощью команды execute\_script, Stack – для хранения истории команд. Научился работать с паттернами программирования, на примере паттерна команда, строитель. Понял, что паттерны – это очень круто, их использование облегчает написание кода. Также получил много практики в обработке исключений. Много времени было потрачено на поиск багов в программе. Скорее всего выявлено и исключено было не больше 10% багов. Рейтинг вылезших исключений при изменении чего-либо возглавил NullPointerException, который я поймал не менее 20 раз. В какой-то момент команда exit завершала программу из-за того, что вылазил NullPointerException.