Национальный исследовательский университет компьютерных технологий, механики и оптики

Факультет ПИиКТ

Лабораторная работа №5

Работу выполнил: Степанов Михаил

Группа: P3130

Преподаватель: Исаев А.С.

Город: Санкт-Петербург

2021г

ПОСТАНОВКА ЗАДАЧИ

Вариант: 311765

Введите вариант: 311765

Внимание! У разных вариантов разный текст задания!

Реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. В коллекции необходимо хранить объекты класса `Worker`, 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 `csv`
- Чтение данных из файла необходимо реализовать с помощью класса `java.io.BufferedInputStream`
- 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_min {element}` : добавить новый элемент в коллекцию, если его значение меньше, чем у наименьшего элемента этой коллекции
- `remove_lower {element}` : удалить из коллекции все элементы, меньшие, чем заданный
- `history` : вывести последние 14 команд (без их аргументов)
- `filter_contains_name name` : вывести элементы, значение поля name которых содержит заданную подстроку
- `filter_less_than_status status` : вывести элементы, значение поля status которых меньше заданного
- `print_ascending` : вывести элементы коллекции в порядке возрастания

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 Worker {
    private Integer id; //Поле не может быть null, Значение поля должно быть больше 0, Значение этого поля должно быть уникальным, Значение этого поля должно генерироваться автоматичес
ки
    private String name; //Поле не может быть null, Строка не может быть пустой
    private Coordinates coordinates; //Поле не может быть null
    private java.time.LocalDate creationDate; //Поле не может быть null, Значение этого поля должно генерироваться автоматически
    private Double salary; //Поле может быть null, Значение поля должно быть больше 0
    private java.time.LocalDate startDate; //Поле не может быть null
    private Position position; //Поле может быть null
    private Status status; //Поле может быть null
    private Person person; //Поле может быть null
}
public class Coordinates {
    private double x; //Значение поля должно быть больше -623
    private double y;
}
public class Person {
    private Integer height; //Поле не может быть null, Значение поля должно быть больше 0
    private Color eyeColor; //Поле не может быть null
    private Color hairColor; //Поле не может быть null
    private Country nationality; //Поле не может быть null
}
public enum Position {
    DIRECTOR,
    ENGINEER,
    HEAD_OF_DIVISION;
}
```

```
public enum Status {
    FIRED,
    RECOMMENDED_FOR_PROMOTION,
    REGULAR;
}
public enum Color {
    GREEN,
    BLUE,
    ORANGE,
    WHITE;
}
public enum Color {
    YELLOW,
    WHITE,
    BROWN;
}
public enum Country {
    RUSSIA,
    UNITED_KINGDOM,
    GERMANY,
    ITALY;
}
```

Отчёт по работе должен содержать:

1. Текст задания.
2. Диаграмма классов разработанной программы.
3. Исходный код программы.
4. Выводы по работе.

Вопросы к защите лабораторной работы:

1. Коллекции. Сортировка элементов коллекции. Интерфейсы `java.util.Comparable` и `java.util.Comparator`.
2. Категории коллекций - списки, множества. Интерфейс `java.util.Map` и его реализации.
3. Параметризованные типы. Создание параметризуемых классов. Wildcard-параметры.
4. Классы-оболочки. Назначение, область применения, преимущества и недостатки. Автоупаковка и автораспаковка.
5. Потоки ввода-вывода в Java. Байтовые и символьные потоки. "Цепочки" потоков (Stream Chains).
6. Работа с файлами в Java. Класс `java.io.File`.
7. Пакет `java.nio` - назначение, основные классы и интерфейсы.
8. Утилита `javadoc`. Особенности автоматического документирования кода в Java.

Ссылка на код: https://github.com/stmikeal/Lab5_prog/tree/master

ВЫВОД:

Выполняя данную лабораторную работу, я научился работать с коллекциями: сортировать, добавлять и редактировать элементы коллекции. Взаимодействие с коллекцией осуществлялось через интерактивный режим, так я ознакомился с потоком ввода System.in, в котором нужно было предусмотреть всевозможные пользовательские ошибки ввода и обработать их. Трудность составило написание и обработка команды execute_script, которая может рекурсивно запускать и другие скрипты. Сама коллекция автоматически заполняется из входного файла, что реализовано с помощью цепочки потоков FileInputStream и BufferedInputStream. Данные в файлах должны храниться в формате xml – для сериализации/десериализации этих данных понадобилось найти подходящую утилиту и изучить основы работы с ней – таковой оказался парсер JAXB, который был (по какой-то причине) удален из JDK после Java 8. Хотя ошибок во входном файле не предусматривалось, их обработка присутствует. Для возможности сортировки коллекции ознакомился с интерфейсами Comparable и Comparator, которые позволяют самостоятельно выбрать правила сравнения элементов при сортировке. Для автоматического создания и форматированного вывода поля creationDate потребовалось найти необходимые методы классов LocalDateTime и DateTimeFormatter. Так как в коллекции могут храниться только ссылочные типы данных, я узнал о существовании классов-обёрток для примитивов. Для ознакомления стороннего пользователя с проектом значимые части кода были задокументированы в формате Javadoc.