Лабораторная работа #5

Введите вариант:

1125

Внимание! У разных вариантов разный 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.LinkedHashMap`
- 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.FileReader`
- 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
- `insert key {element}` : добавить новый элемент с заданным ключом
- `update id {element}` : обновить значение элемента коллекции, id которого равен заданному
- `remove_key key` : удалить элемент из коллекции по его ключу
- `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)
- `remove_greater {element}` : удалить из коллекции все элементы, превышающие заданный
- `history` : вывести последние 15 команд (без их аргументов)
- `remove_greater_key key` : удалить из коллекции все элементы, ключ которых превышает заданный
- `remove_any_by_birthday birthday` : удалить из коллекции один элемент, значение поля birthday которого эквивалентно заданному
- `count_less_than_location location` : вывести количество элементов, значение поля location 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 Person {
    private long id; //Значение поля должно быть больше 0, Значение этого поля должно быть уникальным, Значение этого поля должно генерироваться ав
томатически
    private String name; //Поле не может быть null, Строка не может быть пустой
    private Coordinates coordinates; //Поле не может быть null
    private java.util.Date creationDate; //Поле не может быть null, Значение этого поля должно генерироваться автоматически
    private Long height; //Поле может быть null, Значение поля должно быть больше 0
    private java.time.LocalDateTime birthday; //Поле может быть null
    private String passportID; //Поле может быть null
    private Color hairColor; //Поле не может быть null
    private Location location; //Поле не может быть null
}
public class Coordinates {
    private Long x; //Поле не может быть null
    private Double y; //Значение поля должно быть больше -537, Поле не может быть null
}
public class Location {
    private double x;
    private Double y; //Поле не может быть null
    private double z;
    private String name; //Поле не может быть null
}
public enum Color {
    GREEN,
    BLACK,
    YELLOW,
    ORANGE,
    WHITE;
}
```

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.

Лабораторная работа #6

Введите вариант:

Чтобы узнать задание, введите свой номер варианта.

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. Передача данных по сети. Сериализация объектов.
5. Интерфейс `Serializable`. Объектный граф, сериализация и десериализация полей и методов.
6. Многопоточные программы. Концепции.
7. Класс `Thread` и интерфейс `Runnable`.
8. Состояние потока. Синхронизация потока.
9. Пакет `java.util.concurrent`. Интерфейс `Lock` и его реализации.
10. Атомарные операции.
11. Java Stream APi. Создание конвейеров. Промежуточные и терминальные операции.

Лабораторная работа #7