Факультет систем управления и робототехники

Программирование

Лабораторная работа №5

Вариант 2585

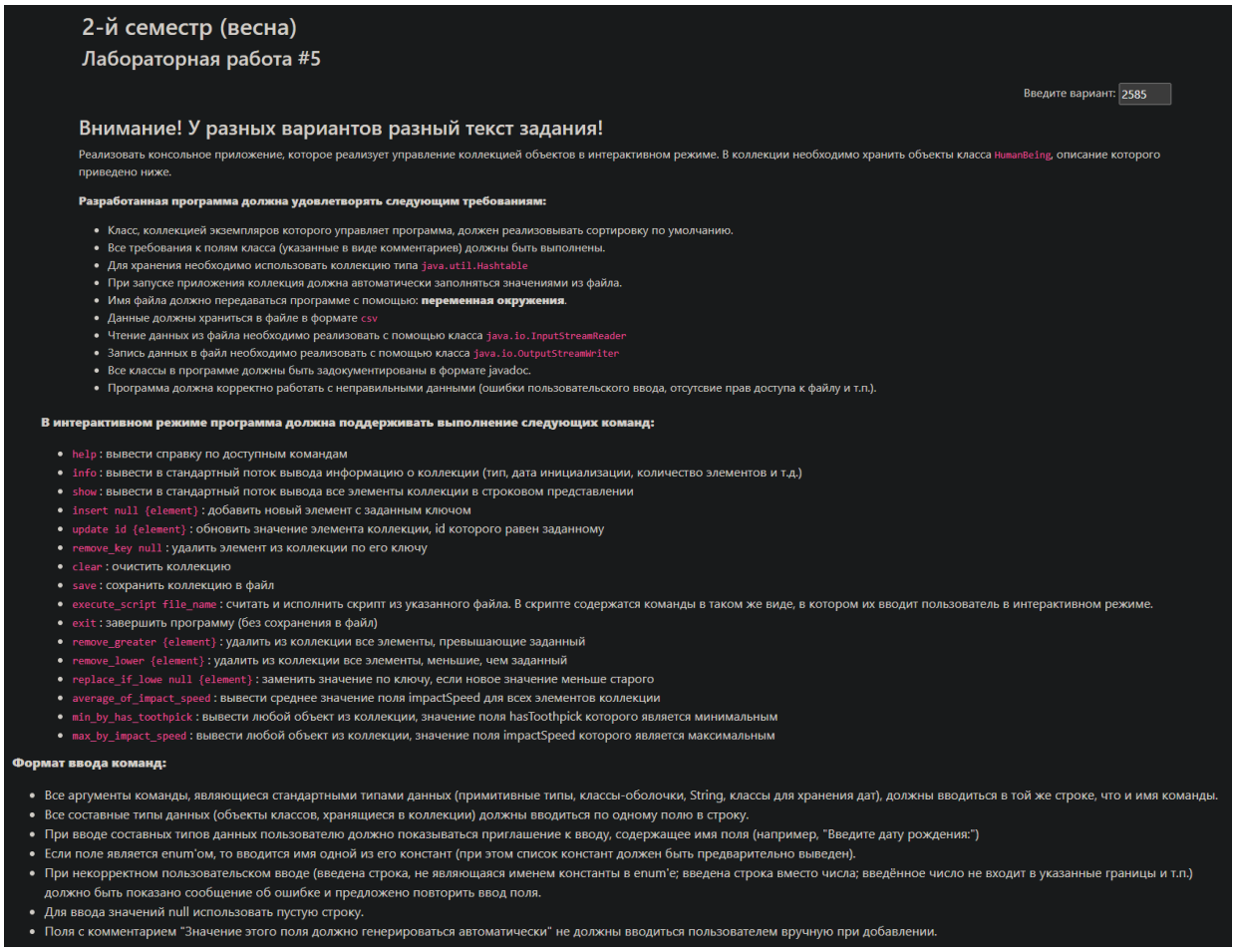
Выполнил: Соловьев Дмитрий

Андреевич, 409609

R3135

Санкт-Петербург, 2024

Условие



2-й семестр (весна)
Лабораторная работа #5

Введите вариант: 2585

Внимание! У разных вариантов разный текст задания!

Реализовать консольное приложение, которое реализует управление коллекцией объектов в интерактивном режиме. В коллекции необходимо хранить объекты класса `HumanBeing`, 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.Hashtable`
- 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.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
- `insert null {element}` : добавить новый элемент с заданным ключом
- `update id {element}` : обновить значение элемента коллекции, id которого равен заданному
- `remove_key null` : 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
- `remove_lower {element}` : удалить из коллекции все элементы, меньшие, чем заданный
- `replace_if_lowe null {element}` : заменить значение по ключу, если новое значение меньше старого
- `average_of_impact_speed` : вывести среднее значение поля impactSpeed для всех элементов коллекции
- `min_by_has_toothpick` : вывести любой объект из коллекции, значение поля hasToothpick которого является минимальным
- `max_by_impact_speed` : вывести любой объект из коллекции, значение поля impactSpeed которого является максимальным

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 HumanBeing {
    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 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.LocalDate creationDate; //Поле не может быть null, Значение этого поля должно генерироваться автоматически
    private Boolean realHero; //Поле не может быть null
    private Boolean hasToothpick; //Поле не может быть null
    private Integer impactSpeed; //Значение поля должно быть больше -355, Поле может быть null
    private WeaponType weaponType; //Поле не может быть null
    private Mood mood; //Поле не может быть null
    private Car car; //Поле не может быть null
}
public class Coordinates {
    private double x;
    private Long y; //Значение поля должно быть больше -507, Поле не может быть null
}
public class Car {
    private String name; //Поле не может быть null
}
public enum WeaponType {
    HAMMER,
    AXE,
    PISTOL,
    MACHINE_GUN;
}
public enum Mood {
    SORROW,
    LONGING,
    CALM,
    FRENZY;
}
```

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.

Диаграмма классов

Исходный код программы

https://github.com/Timo0ooon/lab_5

Вывод

Создание UML диаграммы классов до начала написания кода упрощает процесс и облегчает последующие шаги. Удобно сохранять данные в файл для их сохранения после перезапуска программы. Javadoc очень удобная штука, которая позволяет автоматически генерировать html-документацию, описывающие классы, методы, поля и другие элементы. Это очень сильно облегчает понимание кода.