JAVA CORE KICK
TASK ARRAY PART1

October 21, 2025

First Task Array. Part I

Разработать приложение согласно требованиям, приведенным ниже. В приложении должна быть реализована функциональность, определенная заданием.

Requirements

- Разработать entity-класс, например: *«класс массив»*
- Создание объекта должно осуществляться только через Factory Method или Builder.
- Приложение должно быть структурировано **по пакетам**.
- **Оформление кода** должно соответствовать ***Java Code Convention***.
- Для записи логов использовать *Log4J2* или другой логгер.
- Разработать тесты на *JUnit5*.
- Решение задания хранить на *Github, Bitbucket*.
- Методы класса *Objects* использовать **запрещено**.
- Библиотеку *Lombock* использовать **запрещено**.
- ***Изучить appendix 1***

ch 1

- Разработать services, реализующие функциональности:
  - поиск min\max значения массива,
  - замену элементов массива по условию,
  - определение среднего значения элементов массива,
  - определение суммы элементов массива,
  - определение числа положительных\отрицательных элементов массива.

ch 2

- Сортировка массива должна выполняться *тремя различными* алгоритмами.
- **Параметры**, необходимые для создания массивов, получить чтением информации из файла (.txt). ***Часть данных должна быть некорректной***. Файл данных должен находиться в *отдельном* каталоге.
- Для чтения из файла *можно* использовать методы, появившиеся в Java8.
- **Использовать собственные классы исключительных ситуаций**.
- Разработать validation-классы для валидации исходных данных при создании массивов.

  Например:

  Корректная строка в файле для создания массива: *«1, 2, 3» или «1 - 2 - 3» или « » или «3 4 7»*;

  Некорректные данные в строке файла для создания массива: *«1y1 21 32» и т п*.

  Возможный вид файла (могут присутствовать пустые строки):

1; 2; 3
1; 2; x3; 6..5; 77
11; 2

ch 3

Решить задачи, поставленные в *ch 1 и 2*, с помощью StreamAPI (*IntStream, DoubleStream etc.)*.

Appendix 1:

1. **После if всегда следует положительный сценарий, после else - отрицательный**
2. **Если только if, то возможен и отрицательный сценарий**
3. **В if, for, while обязательно использовать { }**
4. **Если генерируется exception, не ловить его сразу же**
5. **В finally не генерировать исключения и не использовать return**
6. **Не генерироваать стандартные исключения. Разрешено только в методах private**
7. **fileWriter.close(); - в блок finally**
8. **Регулярные выражения в константы**
9. **В именах пакетов не использовать большие буквы**
10. **Не класть сторонние файлы в папки рядом или вместе с классами**
11. **Размещать файлы только в папках в корне проекта**
12. **Использовать для файлов только относительные пути. Папка *src* не должна присутствовать в пути к файлу**
13. **Если константа неизменяемая, то имя должно быть в верхнем регистре, если изменяемая, то как правило именуется как обычное поле класса**
14. **Элементы перечисления именуются как неизменяемые константы**
15. **Не увлекаться статическими методами**
16. **Не объявлять get-теры и set-теры абстрактными**
17. ***Не давать классам имена, совпадающие с именами стандартных классов и интерфейсов Java !***
18. **Не разделять объявление переменной и присвоение ей значения в методах, то есть не писать:**
    Integer count;
    count = 0;
    а надо *Integer count = 0;*
19. **Расстояние (в строчках кода) между использованием переменной и ее объявление должно быть минимально!**
20. **В одной строчке - одна точка, то есть надо использовать локальные переменные, не надо:**
    sample.getSomething().getData()
    надо:
    Something something = sample.getSomething();
    Data data = something.getData();
21. **Не писать *if (isValid == true)*, а писать *if (isValid)***
22. **Не писать:**

```
if (someValue == EXPECTED_VALUE) {
    return true;
 } else {
  return false;
 }
```

писать:

return someValue == EXPECTED_VALUE;

23. **Использовать *assertEquals* вместо *assertTrue(some == other)***
24. **Использовать *assertTrue(isValid)* вместо *assertEquals(true, isValid)***
25. **Лучше тестовые объекты размещать в тестах в виде констант, а не создавать их в самом методе**
26. **Не использовать существующий *FactoryMethod* в тестах для создания объектов, объекты в тестах делать через *new***
27. **Тест должен иметь структуру: given, when, then, где given – прекондишны (инициализация данных), when – вызов тестируемого метода (всегда одна строчка), then – постусловие (*assert-метод*)**