EXTERNAL JAVA SE
FIRST TASK ARRAY

www.training.by

March 22, 2021

First Task Array

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

Requirements

- Разработать entity-класс, например: «*класс Массив*»
- **Entity-класс не следует наполнять методами**, выполняющими функциональные действия (методами бизнес-логики, такими как вычисление, поиск и т.д.).
- Все **классы** приложения должны быть структурированы **по пакетам (package)**.
- **Оформление кода** должно соответствовать ***Java Code Convention***.
- Для записи логов использовать *Log4J2*
- Разработать тесты на *TestNG*.
- Решение задания хранить на *Github*.
- Методы класса *Objects* использовать **запрещено**.
- ***Изучить appendix 1***

part 1

- Создать класс *Массив*.
- Разработать service-классы 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.

part 2

- Сортировка массива *тремя различными* алгоритмами.
- **Параметры**, необходимые для создания массивов, получить чтением информации из файла (.txt). ***Часть данных должна быть некорректной***. Если встретилась некорректная строка, приложение должно переходить к следующей строке. Чтение может быть прекращено по достижении первой корректной строки. 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»*;
  Некорректная строка в файле для создания массива: *«1z1 21 32»*. Присутствует недопустимый символ «z», следовательно всю строку следует считать некорректной.

part 3

Решить задачи, поставленные в *part 1 и 2*, с помощью возможностей *IntStream*.

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 генерировавать 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 перерменной 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 использование 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*)

Ver.	Description of Change	Author	Date	Approved	
				Name	Effective Date
<1.0>		Игорь Блинов	09-10-2018		
<2.0>		Игорь Блинов	02-07-2019		
<3.0>		Игорь Блинов	17-01-2021		