**Класс Runner**

пункт 1.

Реализуйте в 2 этапа.

1. Создание листа экземпляров JsonObject:

Type type = new TypeToken<List<JsonObject>>(){}.getType();

List<JsonObject> jsonObjects = new Gson().fromJson(new FileReader(args[0]), type);

2. Маппинг листа jsonObjects в лист экземпляров Trial по аналогии с задачей start2:

List<Trial> trials = jsonObjects.stream()

.map(...)

.filter(Optional::isPresent)

.map(Optional::get)

.collect(Collectors.toList());

Реализация аргумента метода map() рассмотрена ниже.

пункт 2.

тривиально через method reference.

пункт 3.

аналогично 2.

пункт 4.

используйте метод sort() у колекции.

аргумент метода - компаратор, инициализированный лямбдой.

пункт 5.

intermediate operation - метод map у стрима, в который передается та же лямбда, как в пункте 4.

копи-паст - антипаттерн.

очевидное решение - реализовать в суперклассе дополнительный **финальный** метод и использовать method reference.

более изящное решение - создать ссылку на лямбду и использовать ее в пунктах 4 и 5 (и далее еще в 7).

пункт 6.

выполнение данного пункта через 3+ оператора не представляет трудностей.

минимально возможное количество и при этом изящно - 2 оператора.

даункасты - антипаттерн, поэтому предварительно надо в суперклассе определить метод:

1. clearMarks() и переопределить его в подклассе ExtraTrial,
2. copy() и переопределить его в каждом подклассе.

в первом операторе после извлечения стрима из коллекции надо применить операции filter(), map(), дважды peek() и, наконец, collect() для создания новой коллекции.

второй оператор нужен для подтверждения факта, что все испытания в новой коллекции провалены.

пункт 7.

оператор 1 - преобразование коллекции в целочисленный массив через операцию mapToInt().

оператор 2 - преобразование массива в строковую коллекцию и ее дальнейшее обработка через функционал класса Collectors.

**Реализация аргумента метода map() для пункта 1**

Аргументом метода map() может быть ссылка на статический метод фабричного класса, аргументом которого является экземпляр JsonObject:

public static Optional<Trial> getTrial(JsonObject jsonObject)

В теле метода получите из jsonObject необходимые для создания экземпляров Trial и его подклассов аргументы.

Например,

String className = jsonObject.get("class").getAsString();

JsonObject args = jsonObject.get("args").getAsJsonObject();

Преобразуйте className в экземпляр Type методом Class.forName(). Далее создайте экземпляр Trial или его подкласса:

Trial trial = GSON.fromJson(args, trialType);

где GSON - это константный экземпляр класса Gson.

Если jsonObject и сгенерированный по нему trial валидные, то возвратите Optional.of(trial), иначе Optional.empty().

**Классы сущностей**

Реализуйте аналогично классам сущностей задачи 2.