Домашнее задание на практику №4.

**Задание 1.**

Задание состоит из двух частей.

1. Описать структуру данных класса Character, взяв за основу описание характеристик персонажей в Genshin Impact (или любой другой RPG, но не менее 5 базовых характеристик):
   * Максимальное кол-во HP;
   * Сила атаки;
   * Защита;
   * Мастерство стихий;
   * Выносливость.
   * Тип Архонта (для геншина) или специализация (для других RPG);

Далее взять одну произвольную характеристику из списка второстепенных:

* + Шанс крит.попадания;
  + Крит.урон;
  + Бонус лечения;
  + Бонус получаемого лечения;
  + Восст. энергии;
  + Снижение времени отката;
  + Прочность щита.

После чего создать динамический массив из объектов данного класса, и заполнить не менее 15-20 элементами (имя, характерстики можно задавать рандомом или брать из офф.источников).

1. С имеющейся коллекцией объектов, при помощи методов Stream API произвести следующие действия:
2. Вывести персонажа с максимальным количеством HP среди всех;
3. Вывести персонажа с минимальным количеством HP среди всех;
4. Сортировать персонажей по силе атаки, выбрать только тех, у кого она меньше 100. Вывести на экран;
5. Тоже самое, что в (c), но убрать из выборки всех, у кого меньше 100;
6. Произвести выборку персонажей по одному типу архонта/специализации и для всех изменить произвольную характеристику умножив её значение на 2. Выборку собрать в коллекцию List и вывести на экран;
7. Найти персонажа со значением характеристики силы атаки в 1337, вывод функции .get() произвести в объект класса Optional<Character>. Если такой объект существует, то вывести на экран его и добавить перед его именем слово l33t. Если такого объекта нет, вывести на экран “No l33t”);
8. Посчитать для каждой специализации количество персонажей. Вывести на экран в виде таблички.

**Альтернативное задание 1.**

Альтернативные и более приземленные условия задания для тех, кто по какой-то причине не хочет игровой проект.

Одновременно два задания делать НЕ НАДО.

Описать структуру данных класса Worker (работник) некой компании со следующими свойствами:

* + Имя (отдельное поле)
  + Фамилия (отдельное поле)
  + Возраст
  + Пол
  + Имя кота (если есть)
  + Название отдела
  + Должность
  + Зарплата
  + Премия (если есть)
  + Длительность работы в компании (в годах)

После чего создать динамический массив из объектов данного класса, и заполнить не менее 15-20 элементами (все свойства можно задавать рандомом).

1. С имеющейся коллекцией объектов, при помощи методов Stream API произвести следующие действия:
   1. Вывести работника с максимальной зп среди всех;
   2. Вывести работника с минимальной зп среди всех;
   3. Сортировать работников по возрасту, выбрать только тех, у кого он меньше 50. Вывести на экран тех, у кого есть кот;
   4. Тоже самое, что в (c), но убрать из выборки всех, у кого меньше 50;
   5. Произвести выборку работников по одному отделу и для всех изменить премию умножив её значение (если она больше 0) на 2. Выборку собрать в коллекцию List и вывести на экран;
   6. Найти работника со значением суммы зарплаты + премии больше или равно 100к, вывод функции .get() произвести в объект класса Optional<Worker>. Если такой объект существует, то вывести на экран его и добавить после суммы зарплаты+премии текст “наносек”. Если такого объекта нет, вывести на экран “Компания нищебродов”);
   7. Посчитать для каждого отдела количество работников. Вывести на экран в виде таблички.

# Что должно быть в отчёте

1. Текст **выбранного** задания;
2. Исходный набор данных;
3. Для каждого пункта должно быть:
   1. Что надо получить через StreamAPI;
   2. Код вызова;
   3. Полученный вывод.
4. Листинг кода по каждому классу в конце отчёта.