Домашнее задание на практику №5.

Следующее задание потребует от Вас знаний некоторых предыдущих тем курса, а именно: I/O, обработка исключений, Stream API.

## Задание 1. Игровой вариант

Реализовать простейший текстовый квест, в котором должны быть следующие функции:

* Герой путешествует по локациям, которые идут друг за другом;
* На каждой локации происходит диалог с персонажем, который даёт игроку загадку. За правильный ответ игрок получает очки, за неправильный - отнимается здоровье; Цель игры - пройти десять локаций и не умереть.
* Все локации имеют у себя в классе счётчик сложности;
* Локации группируются между собой случайным образом, но в рамках своей сложности (то есть, игрок не должен первой локацией попасть на самый сложный уровень); Такую сортировку и фильтрацию можно сделать при помощи методов StreamAPI.
  + Для ясности, условное расположение локаций в массиве в виде чисел сложности заданий:

До сортировки [1, 1, 2, 3, 5, 6, 1, 5, 6, 4, 2, 1, 3]

После: [1, 1, 1, 1, 2, 2, 3, 3, 4, 5, 6, 6] - отсюда выбирается по одной-две локации для каждой сложности и итоговый набор получается из 10 локаций с постепенным увеличением сложности.

* Вывод текста происходит в любую консоль (IDE или Терминал);
* В игре должна присутствовать возможность сохранения текущего состояния - локации, параметров игрока - в файл для последующей загрузки при следующем запуске;
* Желательно, чтобы текст не был однотонным (но в меру! “Вырвиглазное” оформление считается за невыполненную работу);
* NPC обращаются к игроку по имени, введенному в начале игры; Резюмируя, программа должна содержать *примерный* набор классов:

Location - описание локации;

Player - игрок;

NPC - персонаж, который находится на локации; View - всё, что касается вывода в консоль; Input - всё, что игрок вводит;

SaveLoad - сохранение и загрузка;

GameLoop - базовый класс с игровым циклом;

Log - логирование действий в программе; (в случае выполнения доп.задания) Для ясности, типичный игровой цикл выглядит следующим образом:

1. Игрок заходит на локацию и читает её художественное описание;
2. К игроку обращается NPC и предлагает решить загадку;
3. Игрок должен выбрать правильное решение (обычно это просто выбор из 2-4 вариантов, без ввода с клавиатуры);
   1. Если игрок выбрал верный ответ, то получает N-очков (рассчитывается исходя из сложности локации);
   2. Если игрок выбрал неверный ответ, то у него отнимается здоровье с учётом сложности загадки.
4. Если у игрока счётчик жизни не упал до нуля и меньше, то он идёт на следующую локацию (если это не была последняя) и цикл начинается заново.

# Дополнительное задание для всех вариантов:

## Автологирование

В начале работы программы запускается отдельный поток для автоматической записи всех действий в программе с помощью классов FileWriter и BufferedWriter в текстовый файл, пример записи действия в файле:

[“НАЗВАНИЕ\_КЛАССА”][“НАЗВАНИЕ\_МЕТОДА”] Пользователь ввёл следующие данные:

[“НАЗВАНИЕ\_КЛАССА”][“НАЗВАНИЕ\_МЕТОДА”] Произошла ошибка, см. описание: (далее вывод stacktrace из блока catch в обработке исключений)

Очевидно, что логировать *каждое* действие пользователя **не нужно.**

Поток засыпает на 5 секунд и после чего производит перезапись файла новыми данными.

# Общие требования для всех вариантов

## Использование Optional<T> вместо обработки NullPointerException

При обработки функций, которые *потенциально* могут вернуть null (например, поиск конкретного объекта в массиве), получать данные через метод .get(), либо использовать .ifPresentOrElse() если требуется провести вывод/обработку.

Использование try-catch-finally для обработки NPE - **запрещено**. Для ясности пример:

…

Optional<Cat> catOptional;

catOptional = cats.stream().filter(cat -> cat.color.equals("Серый")).findAny(); if(catOptional.isPresent()) {

Cat grayCat = catOptional.get();

}

else {

System.out.println("Нет такого кота");

}

…

# Что должно быть в отчёте

1. Текст **выбранного** задания;
2. Исходный набор данных;
3. Описание решения задания;
4. Листинг всех классов.