МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики»

Факультет информационных технологий и программирования

Кафедра информационных систем

Лабораторная работа № 3

Выполнил студент группы М3201:

Кулаженко Дмитрий Алексеевич

САНКТ-ПЕТЕРБУРГ

2019

**Задание:**

Определить класс **BattleUnitsStack**. Этот класс должен описывать всю необходимую информацию для участия стека в сражении (следует учесть возможность воскрешения, изменения параметров и т.п.). Экземпляр должен формироваться на основе “походного” аналога. Реализовать все необходимые методы для управления параметрами.

Определить класс **BattleArmy**. Этот класс должен описывать всю необходимую информацию о “боевой” армии (содержит “боевые” стеки). Следует учесть возможность добавления стеков (например, магия “Призвать элементалей” - добавляет стек элементалей). Максимальное количество стеков в боевой армии - девять. Реализовать все необходимые методы для управления армией.

Определить класс **Battle**. Это основной класс движка, все взаимодействие должно происходить через него. Экземпляр создается на основании двух боевых армий (при желании можно сделать боевку не для двух, а произвольного количества армий). Класс должен предоставлять информацию о текущем ходе (какой юнит ходит, какие действия доступны) и возможность совершить действие. Также необходимо получать информацию о статусе сражения: завершено или нет, кто победил.

Пока реализацию данного класса можно полностью не делать (то, что связано с информацией о ходе и совершении действия), достаточно будет продемонстрировать интерфейс с “заглушками”. Эти заглушки заменятся на нормальную реализацию в следующей лабе.

Весь функционал необходимо продемонстрировать - создание “боевой” армии/стека, управление параметрами, создание экземпляра сражения, проведение сражения. Можно как с помощью тестов, так и с помощью демо примера.

**Ход рассуждений:**

1. BattleUnitStack должен содержать основную логику атаки / применения каста / защиты и тд. Также следует сделать какой-либо интерфейс, чтобы можно было управлять юнитов не использую напрямую методы, а через какой-нибудь отдельный метод: так получились методы make\_move и enum Actions, который хранится в виде аттрибута: Dict[Actions, bool (доступно действите для стека в текущий момент или нет)]
2. BattleArmy должен предоставлять интерфейс для управление стеками, а также порядок ходов юнитов 🡪 получился класс BattleQueue, который управляет порядком ходов стеков.
3. Battle для управление боем, но так как интерфейса как такого еще нет, то без должного представления организовывать большую архитектуру смысла не очень много (придется переписывать).
4. Написал unit-тесты.

**Код: https://github.com/dmkulazhenko/meopasl/**