**Домашнее задание на практику №2.**

## Задание 1.

Теперь вы - подкованный разработчик и вас не испугать словами инкапсуляция или полиморфизм. Время проверить свои навыки на прочность. Задание состоит в следующем:

На основе выполненной работы в предыдущей практике и следующих требований, сформировать структуру игрового проекта:

1. Класс GameObject является абстрактным и имеет два поля: идентификатор объекта (int) и имя объекта(String);
2. Интерфейс Damager определяет поведение *предметов*, наносящих урон и содержит сигнатуру метода damage(), возвращающего значение int;
3. Класс Creature расширяет GameObject и является родителем для всех *существ*;
4. Класс Item расширяет GameObject и является базовым для любых предметов, *которыми может владеть игрок*;
5. Класс Equipment расширяет базовый класс предметов и является родителем для любой *экипировки* (то, что герой носит на себе в том числе *оружие*);
6. Класс EquipmentCell - класс-дженерик, который обрабатывает любые типы T расширяющие класс экипировки; по смыслу представляет собой ячейку, в которую размещается элемент экипировки.
7. Класс Inventory - класс-джереник, который обрабатывает типы T, расширяющие класс предметов. Класс описывает стандартный инвентарь игрока и содержит список объектов типа Item.
8. Интерфейс Lootable описывает поведение существ, *с которых после убийства падают предметы* и содержит сигнатуру метода dropLoot(), возвращающего список (List) объектов типа Item;
9. Интерфейс Attacker описывает поведение сущности, *способной атаковать* и содержит сигнатуру метода attack(Damageable target), которая возвращает значение int;
10. Интерфейс Damageable описывает сущности или объекты, которым *можно нанести урон* и содержит сигнатуру метода getHit(Damager, damager);
11. Созданные классы в практике 1 соответствующим образом переписать под новую структуру с небольшими изменениями: Класс Character переименовать в Player, а Enemy - в Mob.

В данном задании необходимо подготовить только структуру классов

## Задание 2.

Вы хорошо потрудились и разработали хорошую (хорошую ведь?) иерархию классов. Теперь структуру классов надо заполнить основными свойствами и поведением, соответственно назначению классов, соблюдая логику наследования. Например, у всех существ есть здоровье, но не каждое существо умеет атаковать и имеет экипировку\оружее.

Поэтому, задача будет считаться выполненной, если общие и частные свойства\поведения объектов будут правильно распределены по иерархии классов.

Также не забыть о реализации соответствующих интерфейсов, например, для существа реализация интерфейса Damageable подразумевает уменьшение его здоровья в методе getHit(Damager damager). Объект, который реализует Damager - должен возвращать значение наносимого урона через метод getDamage().

health -= damager.getDamage();

## Задание 3.

Далее для функционирования базовой игровой логики необходимо создать класс GameLogic, содержащий статический метод void fight(Attacker attacker, Damagable target) для обмена уроном между двумя существами.

После чего создать класс GameLoop в котором будет реализован игровой цикл с несколькими объектами типа оружия, существ и героя.

Количество мобов, их свойства, сила оружия, здоровье и т.д. заполняется случайным образом (как в практике 1).

Герой и моб получают своё оружие и далее идёт следующий цикл:

1. Герой наносит урон мобу
2. Если моб жив, то он наносит урон герою
   1. Если моб погиб, то в инвентарь герою падает лут с моба (то есть вызов метода getLoot().
3. Повторять пока герой не победит всех мобов или не погибнет сам.

## Что требуется в отчёте

1. Текст заданий;
2. Листинг кода по каждому классу в конце отчёта
3. Для задания 1 - рисунок иерархии классов из IDE (см.рекомендации по выполнению);
4. Для задания 2:
   1. Тот же рисунок иерархии с готовыми полями и методами;
   2. Результат выполнения игрового цикла в виде скриншота окна консоли.