Órai Feladat

Határidő:

150 perc

Játékhősök Képviselete és Küzdelem

Cél:

A feladat célja egy játékrendszer létrehozása, amelyben különböző hőstípusokat (pl. Harcos, Mágus, Íjász) modellezünk. Az osztályok a következőket tartalmazzák:

- Az egyes hősök támadási módjai (pl. harcos karddal támad, mágus varázslatokat használ).
- A hősök életerő (HP) és szintje (Level).
- Sebzés osztály, amely képes kiszámítani a hősök sebzését szintjük és a támadás típusának figyelembevételével.
- Az adatokat JSON fájlban tároljuk, és különböző karaktereket adhatunk hozzá, tárolhatunk és betölthetünk.
- A hősök közötti küzdelmek lefolytatása, ahol a támadások és a sebzések kezelésre kerülnek.

Részletes Feladatok:

1. **Sebzés Osztály (Damage):**
    - A sebzés egy osztályban van kezelve, amely képes a támadás sebzését kiszámolni. A sebzés tartalmaz egy alapsebzés tartományt (pl. 10-20) és egy szintet, amely növeli a támadás erejét. A sebzés minden támadáskor változó, és véletlenszerűen számítódik ki a tartományon belül.

    - A sebzés osztály támogatja a __str__ és __repr__ metódusokat, hogy az objektumok könnyen kiírhatók és olvashatók legyenek.

    - Emellett az osztály lehetővé teszi a sebzések összeadását, ha két különböző sebzés összegzésére van szükség.

2. **Hősök (Hero osztály):**
    - Az **Hero** osztály egy absztrakt osztály, amely az alapvető jellemzőket és metódusokat tartalmazza a hősökhöz: név, HP, szint és sebzés. Az osztály absztrakt támadás metódust tartalmaz, amelyet az alosztályoknak kell implementálniuk.

    - A hősök rendelkeznek egy hp (életerő) property-vel, amely nem csökkenhet nulla alá. Ha a hős életpontjai 0-ra csökkennek, az életpontok nem csökkenhetnek tovább.

    - A __str__ és __repr__ metódusok segítségével az objektumok emberi olvasható módon és technikai szinten is megjeleníthetők.

3. **Alosztályok: Harcos (Warrior), Mágus (Mage), Íjász (Archer):**
    - A **Harcos (Warrior)**, **Mágus (Mage)** és **Íjász (Archer)** osztályok a **Hero** osztályból öröklődnek, és mindegyik saját támadási mechanizmusokat tartalmaz:
        - A **Harcos** karddal támad, a támadás sebzése a Damage osztály által generált értékkel számol.
        - A **Mágus** varázslatot használ, amely szintén a Damage osztály segítségével van meghatározva.
        - Az **Íjász** íjjal támad, és az ő támadásának is egyedi sebzése van.

4. **Karakterek Adatainak Kezelése JSON fájlban:**
    - Az **CharacterDataManager** osztály kezeli a karakterek adatainak olvasását és írását JSON fájlokba.
    - Az osztály képes karaktereket hozzáadni a fájlhoz (név, típus, HP, szint) és a fájlból történő adatbetöltést is végrehajtani. A karakterek típusát (pl. Warrior, Mage) és adatait JSON formátumban tároljuk.
    - Az adatok JSON fájlba való írása és olvasása a write_data és read_data metódusok segítségével történik.

5. **Küzdelem Osztály (Battle):**
    - A **Battle** osztály két hőst vesz fel (pl. egy Harcost és egy Mágust), és folytat egy kört, ahol a hősök egymásra támadnak, amíg valamelyik hős életereje el nem fogy.
    - A küzdelem folyamán a támadások számítása a megfelelő támadási mechanizmusokkal történik, és a hősök sebződnek.
    - A küzdelem addig folytatódik, amíg az egyik hős el nem esik (HP = 0).

6. **Fő Program (main):**
    - A fő program létrehozza a karakterek adatainak kezelésére szolgáló **CharacterDataManager** objektumot, amely a karaktereket hozzáadja és betölti a JSON fájlból.
    - A karakterek típusától függően (pl. Warrior, Mage) új hősök jönnek létre, és a **Battle** osztály segítségével küzdelmet indítanak közöttük.
    - A küzdelem eredménye kiírásra kerül a képernyőre, jelezve, hogy melyik hős győzött.

Követelmények:

- Az osztályok és metódusok használata során ügyelni kell a **SOLID** elvekre, a **KISS** elvre (Keep It Simple, Stupid) és a **DRY** (Don't Repeat Yourself) elvre.
- Az adatokat JSON fájlban kell tárolni és kezelni.

- Az osztályok és metódusok legyenek jól dokumentáltak, és használjanak megfelelő Python szintaktikai elemeket, mint például **@property** a getter és setter metódusokhoz, **__str__** és **__repr__** metódusok az objektumok szép megjelenítéséhez, valamint **__add__** a sebzés összeadásához.

Tesztelés:

- A kódot futtatva figyeljük meg, hogy a karakterek megfelelően támadnak, sebződnek és a küzdelem végén az egyik hős győz.
- Az adatokat a characters.json fájlban tároljuk, és az új karakterek megfelelően hozzáadódnak a fájlhoz.

Példa kimenet:

```
Thor attacks with a sword, dealing 20 damage!
Merlin has 65 HP left.
Merlin casts a spell, dealing 29 damage!
Thor has 91 HP left.
Thor attacks with a sword, dealing 24 damage!
Merlin has 41 HP left.
Merlin casts a spell, dealing 21 damage!
Thor has 70 HP left.
Thor attacks with a sword, dealing 25 damage!
Merlin has 16 HP left.
Merlin casts a spell, dealing 25 damage!
Thor has 45 HP left.
Thor attacks with a sword, dealing 27 damage!
Merlin has 0 HP left.
Thor wins!

Process finished with exit code 0
```