2024.12.09.

Órai Feladat

Határidő:

65 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
```

Gyakorló feladatok

1. JSON: Random személyek

Generálj 10 személyt, ahol a nevek tartalmaznak véletlenszerű kezdőbetűt és számot, az életkorok pedig egy megadott tartományból származnak. Mentsd el JSON fájlba.

Az előző JSON fájlból olvasd be az adatokat, és szűrd ki azokat, akik 30 év alattiak, majd mentsd új fájlba.

Egészítsd ki a JSON adatokat véletlenszerű email-címek generálásával. Az email legyen a névből és egy véletlen domainből generálva.

---

2. CSV: Városok adatai

Adj hozzá egy "kontinens" oszlopot, ahol a kontinens véletlenszerűen van kiválasztva az "Európa", "Ázsia", "Amerika" opciók közül.

Készíts egy új CSV fájlt, amelyben csak azok a városok szerepelnek, ahol a lakosság meghaladja az 500,000 főt.

Írj egy programot, amely számolja és kiírja az átlagos városméretet (terület) az eredeti CSV fájl alapján.

---

3. TXT: Véletlenszerű dátumok

Generálj véletlenszerű időpontokat (óra és perc) a következő 24 órán belül, és írd őket egy TXT fájlba.

Olvass be egy TXT fájlt dátumokkal, és ellenőrizd, hogy vannak-e duplikált dátumok. Az egyedi dátumokat írd egy új fájlba.

Generálj dátumokat és hozzájuk tartozó szöveges megjegyzéseket (pl. eseményneveket) ugyanabba a TXT fájlba.

---

4. JSON: Eseménynaptár

Egészítsd ki az eseménynaptár JSON-t véletlenszerű eseménytípusokkal (pl. "meeting", "party", "deadline"), és mentsd új fájlba.

Olvasd be az eseménynaptárat, és számold meg, hány esemény van hétvégén (szombaton vagy vasárnap).

Generálj egy új JSON fájlt, amely az eseményekből külön "munka" és "szabadidő" kategóriákat készít, a dátumok alapján csoportosítva.

---

5. CSV: Átlaghőmérséklet

Írd ki, melyik hónapban volt a legnagyobb hőmérsékletingadozás (maximum - minimum).

Generálj egy új CSV-t, amely minden hónaphoz véletlenszerű időjárási eseményeket ad (pl. "eső", "hó", "száraz").

Számítsd ki, hogy az első 6 hónap vagy az utolsó 6 hónap volt melegebb, és az eredményt írd ki egy új CSV fájlba.

---

6. TXT: Naplóbejegyzések

Egészítsd ki a naplóbejegyzéseket időbélyegekkel (pl. "2024-11-25 14:32"), és írd őket egy új TXT fájlba.

Írj egy programot, amely megszámolja, hányszor szerepel egy bizonyos szó vagy kifejezés a naplófájlban, és az eredményt írd ki egy új fájlba.

Hozz létre egy új fájlt, amely csak azokat a sorokat tartalmazza, amelyek dátuma egy adott intervallumba esik.

---

7. JSON: Véletlenszerű időpontok

Generálj véletlenszerű időpontokat (óra és perc) a következő 3 napon belül, minden napra 3 eseményt, és mentsd JSON fájlba.

Olvasd be a JSON fájlt, és válaszd ki azokat az időpontokat, amelyek este 6 után vannak, majd mentsd új fájlba.

Készíts egy statisztikát, amely számolja, melyik napra esik a legtöbb esemény, és az eredményt írd ki JSON-ba.

---

8. CSV: Véletlenszerű dátumok az év elejéből

Egészítsd ki a CSV fájlt az év minden hónapjának átlagos hőmérsékletével, véletlenszerű értékeket használva.

Írj egy programot, amely összeszámolja, melyik napból van a legtöbb előfordulás a fájlban (pl. hétfőből vagy keddből).

Hozz létre egy új CSV fájlt, amely a hét napjaira csoportosítja a dátumokat, és minden naphoz hozzáad egy véletlenszerű esemény nevét.

---

9. TXT: Véletlenszerű nevek és születési dátumok

Olvass be neveket egy TXT fájlból, és rendelj hozzájuk véletlenszerű születési dátumokat, majd írd vissza őket egy új fájlba.

Készíts egy programot, amely a neveket és születési dátumokat átrendezi név szerinti ábécésorrendbe.

Egészítsd ki a TXT fájlt azzal, hogy minden személyhez hozzárendelsz egy véletlenszerű lakóhelyet egy előre definiált városlistából.

---

10. JSON: Termékek és átlagár

Számold ki, hány termék ára magasabb az átlagos árnál, és az eredményt mentsd egy új JSON fájlba.

Írd ki egy új JSON fájlba azokat a termékeket, amelyek ára egy bizonyos tartományban van (pl. 100 és 500 között).

Egészítsd ki a JSON fájlt minden terméknél egy "kedvezményes ár" mezővel, amely az eredeti ár véletlenszerű csökkentésével jön létre (pl. 5-20%).