Feladat – NagyZH – FONTOS INFÓK

- **Alkalmazni kell a megoldás során a tanult objektum-orientáltsági elveket (ősosztály-gyerekosztályok, virtuális metódusok, egységbe zárás stb).**
- A bemeneti fájlokat (**in-arena.json**, **in-actions.txt**) teszteléskor a megfelelő helyre kell másolni.
- Feltehetjük, hogy a bemenő/teszt adatok helyesek, érvényesek (kivéve, ha a feladat mást mond).
- A megoldás teljes forráskódját egyetlen ZIP fájlba csomagolva kell feltölteni.

A feladat leírása

A *main*-ben és ebben a leírásban szereplő osztályok és metódusok igény szerint **átnevezhetők**, az itteni nevek csak példák.

A feladatban játékos karakterek (szereplők) menedzselése a cél, akik egy arénában harcolnak egymással. Mindenkinek van egy szöveges azonosítója, aktuális életereje (HP) és maximális életereje. Kezdetben az életerő mindig megegyezik a maximális életerővel. Minden szereplő három karakterosztály valamelyikébe tartozik: katona, mágus vagy pap. A katonának van ereje (power), és páncélja (armor). A mágusnak és a papnak van varázsereje (magic). Feltehetjük, hogy az említett tulajdonságok mindegyike pozitív egész (de az aktuális életerő leeshet 0-ra, lásd később).

Legyen egy megfelelő osztályszerkezet, amiben tárolni tudjuk a különböző szereplőket. Az **Arena** osztály *load* metódusa legyen képes beolvasni egyetlen JSON fájlból mindenkit (lásd az **in-arena.json** mintafájlt). A **"CharacterType"** kulcs mondja meg, hogy ki melyik karakterosztályba tartozik.

Innentől kezdve (az előbb említett beolvasást leszámítva) a teljes pontszámhoz az **Arena** osztály nem hivatkozhat a konkrét gyerekosztályokra, csak az ősosztályra. Vagyis, a gyerekosztályok szerinti esetszétválasztásokat ajánlott virtuális metódusokkal megoldani.

Legyen az **Arena** osztályban egy *printOne* metódus azonosító alapján keressen meg egy szereplőt és írja ki minden adatát. A *printAll* metódus írja ki az összes szereplő minden adatát.

Tudjanak a szereplők akciókat végrehajtani egymáson. Legyen egy virtuális metódus a karaktereket reprezentáló ősosztályban, ami visszaadja az adott karakter páncélját. Legyen egy másik virtuális metódus, ami egy végrehajtott akciót reprezentál, és a támadás célpontja egy másik karakter, amit a metódus paraméterben kap meg. A végrehajtott akció hatása az akciót végrehajtó és a célpont karaktertől függ, az alábbiak szerint.

- A katona megtámadja a célpontját. A célpont életereje annyival csökken, amennyi a katona ereje. Ha viszont van a célpontnak páncélja, akkor az semlegesít az okozott sérülésből annyit, amennyi a páncél értéke. Viszont a célpont mindenképp sérül legalább 1-et.
- A mágus is megtámadja a célpontját, akinek az életereje annyival csökken, amennyi a mágus varázsereje. A védekező páncéljának ilyenkor nincs hatása.
- A pap gyógyítja a célpontját, akinek az életereje annyival nő, amennyi a pap varázsereje.

- Soha nem csökkenhet egy szereplő életereje 0 alá, és nem nőhet a maximális életereje fölé. Ha ez történne, akkor az életerő nulla, illetve maximális lesz.
- Ha egy szereplő életereje 0-ra csökken, akkor elesik. Innentől kezdve minden általa vagy rajta végrehajtott akció teljesen hatástalan.

Legyen az **Arena** osztályban legyen egy *performAction* metódus, ami megkapja az akció végrehajtójának és a célpontnak az azonosítóját, és elvégez 1 db akciót. Nem kell kiírni semmit.

(Megjegyzés: a *main*-ben lévő tesztkód a *printOne* metódus segítségével megjeleníti sorban néhány akció hatását, mindig akció után a célpont kerül kiírásra, aminek elsősorban az életereje az érdekes.)

Lehessen ilyen akciók sorozatát a programba beolvasni és tárolni egy **ActionList** osztályban. Ez konstruktorban egy szöveges fájl nevét várja, innen beolvassa és eltárolja a végrehajtó-célpont párokat. Az azonosítókról feltehetjük, hogy nem tartalmaznak whitespace-t. Legyen egy *printPairs* metódusa, ami kiírja ezeket a beolvasott azonosító-párokat. Végül, az **Arena** osztályban legyen egy *performList* metódus, ami a paraméterben kapott **ActionList**-ben tárolt akciókat sorban végrehajtja.

(Megjegyzés: az **ActionList**-re vonatkozó feladatok tökéletesen megoldhatók az akcióktól és a *performAction*-ös feladattól függetlenül is. Például maximális pontszám járhat akkor is, ha a *performAction* metódus üres, de jól van meghívva a *performList* metódusban. Az **in-actions.txt** fájlban egyébként pontosan ugyanaz az akció-sorozat van, amit a *main* korábban a másik feladat tesztelése végett maga is végrehajt).

Pontozás

1. A karakterek osztályai, megfelelő adatszerkezet **(8 pont)**
2. **Arena**, beolvasás JSON fájlból **(6 pont)**
3. **Arena**, *printOne* és *printAll* metódusok **(6 pont)**
4. Virtuális metódus: páncél számítása **(4 pont)**
5. Virtuális metódus: a különféle akciók kódolása **(4 pont)**
6. *performAction* metódus, a különféle szabályok kezelése **(4 pont)**
7. **ActionList** osztály, adatszerkezet, beolvasás szövegfájlból **(4 pont)**
8. *printPairs* metódus **(2 pont)**
9. *performList* metódus **(2 pont)**

Összesen: **40 pont**.