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-camp.json**, **in-team-1.txt**, **in-team-2.txt**) teszteléskor a megfelelő helyre kell másolni.
- Feltehetjük, hogy a bemenő/teszt adatok helyesek, érvényesek (kivéve, ha a feladat mást nem állít).
- 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 kalandorok menedzselése a cél, akik meg akarnak küzdeni egy sárkánnyal. Minden kalandornak van egy szöveges azonosítója és zsoldja (price, egész), és három karakterosztály valamelyikébe tartozik: kardforgató, mesterlövész, vagy mágus. A kardforgatónak van ereje (strength, egész). A mesterlövésznek van ügyessége (agility, egész), és tároljuk, hogy elit-e (logikai). A mágusnak van varázsereje (magic, egész). Mindegyik numerikus tulajdonság pozitív egész. Legyen egy megfelelő osztályszerkezet, ami tárolja a kalandorokat.

A **Camp** osztály tárolja a kalandorokat, tetszőleges számban. A *load* metódusa legyen képes beolvasni egyetlen JSON fájlból minden kalandort, a sorrendjük lényeges. A **"Type"** kulcs mondja meg, hogy ki melyik karakterosztályba tartozik.

Innentől kezdve (az előbb említett beolvasást leszámítva) a teljes pontszámhoz a **Camp** és **Team** osztályok nem hivatkozhatnak a konkrét gyerekosztályokra, csak az ősosztályra. Vagyis, a gyerekosztályok szerinti esetszétválasztásokat ajánlott virtuális metódusokkal megoldani.

A **Camp** osztály a *print* metódussal tudja kalandorok adatait kiírni. Ha nem kap paramétert, akkor az összes kalandort írja ki. Legyen paraméterben megadható egy azonosító, ekkor csak azt az egy kalandort írja ki. Legyen megadható egyetlen **vector<string>** is, ilyenkor a felsorolt azonosítójú kalandorokat kell kiírni.

Lehessen a kalandorok támadóerejét tesztelni. A **Camp** osztály *getAttackPower* metódusa egy azonosítót kap. Szimuláljon egy támadást és adja vissza, hogy a kalandor mennyi sérülést okozott (egész). A támadások sikerességét és hatásosságát kockadobással döntjük el. A **Dice** osztály statikus *roll* metódusa visszaad egy 1 és 6 közötti értéket, ez 1 db kockadobást reprezentál. Lásd a *main* függvény elejét a példa használathoz. Fontos, hogy pontosan annyiszor hívjuk meg a *roll* metódust, ahányszor arra szükség van, mert így garantált a minta kimenettel azonos végeredmény.

Legyen egy virtuális metódus a karaktereket reprezentáló ősosztályban, ami megvalósítja a háromféle karakterosztály támadását az alábbiak szerint.

- A kardforgató annyi kockával dob, amennyi az ereje, és a dobások nyers összege adja a támadóerejét. (Példa: 3 a kardforgató ereje, a három dobás 1, 6, 4, tehát 11 a támadóerő.)

- A mesterlövész dob egy kockával – ha elit, akkor 2 kockával, és ha dobott 5-öst vagy 6-ost, akkor a lövés talált, ilyenkor a támadóerő a mesterlövész ügyessége. Ha a lövés nem talált, a támadóerő 0. (Példa: 7 az elit mesterlövész ügyessége, a két dobás 1 és 5, ami talált, tehát 7 a támadóerő. Ha a két dobás 4 és 3, akkor nincs találat, 0 a támadóerő. Ha a két dobás 6 és 6, akkor is 7 a támadóerő.)
- A mágus addig dob egyesével, amíg nem dob 1-est. Minden 1-esnél nagyobb dobása annyi támadóerőt ér, amennyi a mágus varázsereje. (Példa: 4 a mágus varázsereje, és a dobásai 2, 6, 4, 3, 2, 5, 2, 3, 1. Itt összesen 9 dobással lett vége a dobásoknak, tehát a támadóerő 32. Ha már a legelső dobás 1-es, a támadóerő 0.)

Ha nem sikerül a fenti módszert implementálni, akkor a *getAttackPower* adjon vissza valami számértéket, amivel lehet a feladatsort folytatni.

Legyen egy **Team** osztály, ami egy, a kalandorok közül kikerülő csapatot tárol. A **Camp** osztály *readTeam* metódusa egy szövegfájl nevét kapja, amiben a csapat tagjainak azonosítói vannak felsorolva. Az azonosítók nem tartalmaznak szóközt. A sorrendjük számít. A *readTeam* létrehozza és visszaadja a csapatot. (Tipp: célszerű a csapattagok mutatóit tárolni.) Legyen a **Team** osztályban is *print* metódus, ami kiírja a csapat tagjainak adatait, eredeti sorrendben. Ugyanaz a kalandor több csapatban is tag lehet.

Legyen a **Team** osztályban egy *calculateStats* metódus, ami megbecsüli a csapat tagjainak a várható támadóerejét, és ez alapján egy ár/érték arányt számol. Ehhez minden csapattag támadóerejét ki kell próbálni *elég sokszor*, és az eredményekből egy átlagot számolni, ez a várható támadóerő.[1] Kis eltérés megengedett a minta kimenethez képest. Az ár/érték arány a várható támadóerőnek és a csapattag zsoldjának a hányadosa. A *calculateStats* írja ki csapattagonként az eredményeket a standard kimenetre, valamint exportálja is ezeket JSON formátumban (tagonként csak az azonosító, a várható támadóerő és az ár/érték arány kell). A JSON fájl nevét a *calculateStats* paraméterben várja.

Pontozás

1. A kalandorok osztályai, megfelelő kapcsolatok, adatszerkezet **(8 pont)**
2. **Camp** osztály, *load* metódus **(6 pont)**
3. *print* metódus, háromféle paraméterezés **(8 pont)**
4. Virtuális metódus, a karakterosztályok támadási módjaihoz **(4 pont)**
5. *getAttackPower* metódus **(4 pont)**
6. *readTeam* metódus, **Team** osztály feltöltése, *print* **(4 pont)**
7. *calculateStats*, ami kiír standard outputra és JSON fájlba **(6 pont)**

Összesen: **40 pont**.

[1] Ez a Monte Carlo módszer. Lehetséges próbálgatás helyett képlettel pontos értéket számítani mindhárom támadási módra. Ha valaki így csinálja és helyesek a képletek, az is OK.