# Feladat – Kiszh 2 – FONTOS INFÓK

- A feladat során alkalmazd a megtanult objektum-orientáltsági elveket!
- A megadott példakódon ne módosíts, hacsak a feladat nem kéri! Ez alól kivétel a Program.cs elején lévő kommentek (#define). A megoldásnak ezen fájlokkal kell mennie, hiszen az ellenőrzés során a Moodle biztosítja őket.
- Figyelj a kiírás megfelelő formátumára, szóközökre, új sorokra! Az automata javító csak a tényleges kimenetet látja, nem tudja mit akartál.
- Minden pont értékeléséhez szükséges, hogy az adott ponthoz tartozó #define szerepeljen a Program.cs fájl elején (#define PART1 az 1. feladathoz, #define PART4 a 4. feladathoz, stb.).
- Csak olyan kódot tölts fel moodle-be, ami nálad fordul. Ami nálad nem fordul, a Moodle-ben sem fog.
- A fájlokat nem tömörítve kell feltölteni, hanem önmagukban (drag-and-drop-pal egyszerre be lehet húzni az összeset).
- Figyelj rá, hogy a fájlnevek pontosan azok legyenek, amiket a feladat kér!
- A Moodle által használt fordító nem feltétlenül támogatja az újabb nyelvi fejlesztéseket. Amit az órákon tanultunk és használtunk, az mind működik, de az újabb fejlesztések nem biztos.

# Feladat – Kiszh 2

- A feladat elkezdéséhez a mellékelt projekt tartalmaz kódokat. A megadott **Program.cs** fájl a tesztelésben segít. Módosítani csak az elején lévő #define-okon lehet, amik a teszteket aktiválják.
- A feladatban egy stratégiai játék néhány leegyszerűsített funkcióját valósítjuk meg.
- Az alapkódban adott egy **Position3D** osztály, ami egy térbeli pozíciót ír le (x, y, z) adatok. Van konstruktora, illetve egy távolság-számító metódusa. Az osztályt módosítani nem lehet!
- Adott továbbá egy absztrakt **GameObject** osztály, ami a játékban kezelt objektumok összefogó ős osztálya. Tárol egy pozíciót, maximális életet és aktuális életet, amiket a konstruktor beállít. Két virtuális metódusa van: *TakeDamage*, ami csökkenti az életet egy adott mennyiséggel (0 alá nem engedi), és *Heal*, ami növeli az életet egy adott mennyiséggel (a max fölé nem engedi). Az osztályt módosítani nem lehet!
- A feladatban egy interfészt, valamint több osztályt kell készíteni, mindegyiket csak kevés funkcióval.
- A feladatok között vannak összefüggések.

## A feladatok:

1. Készíts egy **IAttacker** interfészt a támadásra képes objektumok funkcióihoz. Egy metódusa legyen: *Attack*, amely egy tetszőleges játékbéli objektumot vár (a támadás célpontja). **(1 pont)**
2. Származtass a **GameObject** osztályból egy **Unit** osztály, ami egy egységet jelöl, és implementálja az **IAttacker** interfészt. Extra adatként tárolja az egység sebzését, a konstruktor pedig várja a pozíciót, max életet és a sebzést. **(1 pont)**
    a. Az *Attack* metódus sebezze meg a célpontot az egység sebzésével. **(1 pont)**
3. Származtass a **GameObject** osztályból egy **Building** osztályt, ami egy épületet jelöl. Extra adatként tárolja az épület maximális és aktuális páncélját, a konstruktor várja a pozíciót, max életet és max páncélt. Az aktuális páncél kezdetben mindig a max érték, ahogy az ősosztály esetében az életnél is van. Az aktuális páncél a *CurrentArmor* property-n keresztül legyen lekérdezhető, a max páncél pedig a *MaxArmor* property-n keresztül. **(2 pont)**
    a. Legyenek az épületnek *AddArmor* és *DamageArmor* metódusai, melyek paraméterben egy egész számot kapnak, és az épület páncélját ennyivel növelik, illetve csökkenti, figyelve arra, hogy az mindig 0 és a max érték között maradjon. **(1 pont)**
4. Írd felül a **Building** osztályban a *Heal* és *TakeDamage* metódusokat úgy, hogy sebzés először a páncélt csökkentse, majd, ha az elfogyott, akkor a maradék mennyiséggel az életet, a gyógyulás pedig először az életet töltse, majd, ha az tele van, akkor a maradék mennyiséggel a páncélt. Az élet és páncél értékéről a függvény hívásakor csak annyi feltételezhető, hogy valahol 0 és a max érték között vannak. **(2 pont)**
5. Származtass a **Building** osztályból egy **DefenseTower** osztályt, ami implementálja az **IAttacker** interfészt. Extra adatként a torony támadó erejét tárolja, a konstruktor ezt is kapja meg, a többi adat után. Az *Attak* metódus sebezze meg a célpontot a támadóerő értékével, de, amennyiben a célpont 50 egységnél közelebb van, akkor a támadóérték kétszeresével sebezzen. **(2 pont)**