**Progtech beadandó**

## Készítette: Szalóki Dávid, Seres Péter , Kerepesi Gergő

# Projektről röviden:

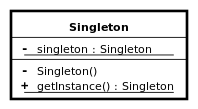
Szeretnénk készíteni egy Vatera alkalmazást, ahol lesz egy nagy Raktár. Amiből a Felhasználók tudnak válogatni a benne lévő Termekből. Az alkalmazásunk ezt hivatott megvalósítani.

Tervben : felhasználónak bevásárló kosár -> a rendeléshez

# Csomagok:

Terv: C# singleton + Diszítő tervezési minták

Singleton használata a raktár létrehozásához, csak egy raktár létezik, amiből lehet kivenni és pakolni. Új raktár létrehozását pedig megtiltjuk a Singleton tervezési minta alkalmazásával.



# Osztályok:

**Interfészek**: A különböző szinteken található interfészeket csak azon osztály implementálják, amiknek szükségük van a bennük lévő metódusokra.

**Absztrakt osztályok**: A legutolsó gyermek osztály kivételével, minden szülő absztrakt. Ennek célja, hogy a közös tulajdonságok összegyűjtve legyenek, így megspórolásra kerülnek a fölösleges ismétlések. Ennek lényege, hogy az azonos tulajdonsággal rendelkező dolgok kódjai csak egyszer legyenek megírva, viszont, ha a tárgyak különböznek (például a leírásaik), akkor azt a legalsó szinten lehessen kifejteni.

Az osztályokhoz kapcsolódóan felhasználtuk az **OOP alapelveket**

**-Egységbezárás (Encapulation) :** Az adattagokatés a hozzájuk kapcsolódó műveleteket egy egységbe zártuk (class)

**-Öröklődés (Inheritance) :** A gyermek osztály az ős minden mezőjét és metódusát megörökli

**-Többalakúság (Polymorphism) :** A tervezési minták szinte mindegyike ezen alapszik , maga a többalakúság az öröklődés következménye. Mivel a gyermek örökli az ős felületét igy megkapja azok típusait is, igy az objektum több alakban is használható

# Felhasznált tervezési minták:

**Singleton:** Olyan osztályok, amelyeket csak egyszer van szükségünk példányosítani mert csak a metódusokra van belőlük szükség. Nem hozható belőlük létre új példány PL: különböző storagek vagy factoryk. Kiegészítés : többszálú alkalmazás esetén le kell védeni a péndányosítást.

Felhasznált viselkedési minta:

**Megfigyelő-Observer:** Az observer egy olyan viselkedési minta, amely az egy-több kapcsol létrehozására alkalmas objektumok között. Ha az egyik objektumot megváltoztatjuk, az observer értesítést ad a tőle függő objektumoknak, és automatikusan frissíti őket.

Egy olyan minta, amely az objektum (Subject) fenntartottainak listáját (Observer) tárolja, és egy metódus meghívásával automatikusan értesíti őket, ha egy állapot megváltozik.

Célja egy Modell-Nézet-Vezérlő, vagy röviden MVC.

Részei:

* **Subject:** Interfészt biztosít a Megfigyelő objektumok csatolására és leválasztására.
* **Observer:** Frissítő interfész az értesítendő objektum/ok számára (update művelet).
* **ConcreteSubject:** A ConcreteSubject érdekes állapotokat tárol az Observer számára, és saját állapotváltozásairól értesíti őket.

Példa:

