

3. heti feladatok

1 Készítsen egy tetszőleges osztályt, amely megvalósítja az **Comparable** interfészt

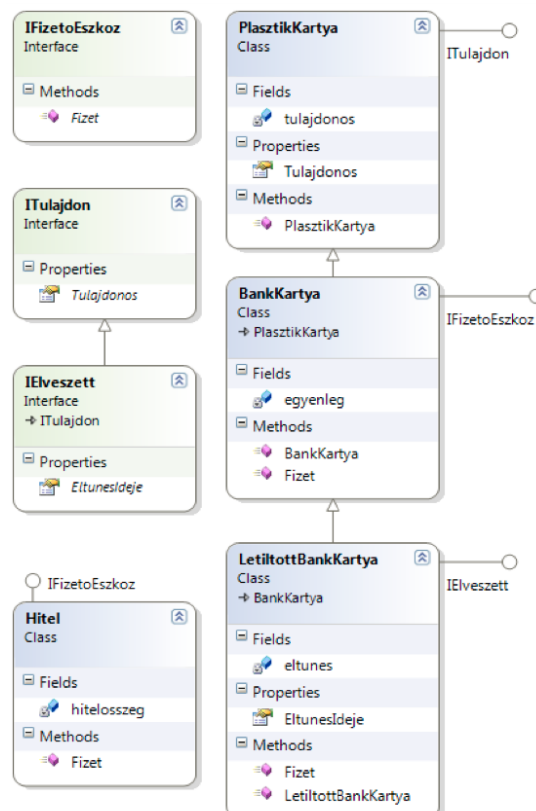
- A .NET osztálykönyvtár tartalmaz egy **Comparable** nevű interfészt, amelyet megvalósítva egy objektum össze tudja hasonlítani önmagát egy másikkal.
- Az interfész egyetlen metódust határoz meg: `int CompareTo(object obj)`
Ennek a metódusnak a lehetséges visszatérési értékei:
 - kisebb mint 0, ha a példány megelőzi a paraméterként átadott objektumot sorbarendezésnél
 - 0, ha a példány és a paraméterként átadott objektum azonosnak tekinthető sorbarendezésnél
 - nagyobb mint 0, ha a példány a paraméterként átadott objektum után áll sorbarendezésnél

Próbálja ki az így megvalósított objektumot. A fent megvalósított osztály példányaiból létrehozott tömböt adjon át a beépített rendező algoritmusnak.

- A .NET osztálykönyvtár rendelkezik egy `Array.Sort(Array)` statikus metódussal.
- Ez a metódus rendezi a paraméterként átadott **Comparable** interfészt megvalósító objektumokat.

2

- Valósítsa meg az ábrán látható interfész és osztályhierarchiát.
- Készítsen főprogramot, ami egy közös tömbben tárol **IFizetoEszkoz** interfészt megvalósító objektumokat.
- Legyen egy statikus `Fizetes(IFizetoEszkoz[])` metódus a **Program** osztályban, ami igazat ad vissza, ha a bemeneti tömb bármelyik eleme végre tudta hajtani a `Fizet` metódusát.
- Legyen egy ellenőrzött fizetést végrehajtó metódus, ami a hasonlóan működik az előző metódushoz, de a fizető nevét is ellenőrzi.



- Írjon egy **IRealEstate** interfészt, ami egy paraméter nélküli `int TotalValue()` metódust tartalmaz. A metódus az ingatlan értékét adja majd vissza.
- Írjon egy **IRent** interfészt, ami három tagot ír elő:
 - `int GetCost(int months)` metódus megadja, hogy mennyibe kerül az ingatlan adott hónapnyi bérlete egy fő számára.
 - `bool IsBooked` getter visszatadja, hogy éppen foglalt-e az ingatlan.
 - `bool Book(int months)` metódus adott hónapra lefoglalja az ingatlan, ha még nincs lefoglalva. Visszatérési értéke a foglalás sikerességéről ad információt.
- Írjon egy **Flat** absztrakt osztályt, ami implementálja az **IRealEstate** interfészt
 - Az osztály a következő adattagokkal rendelkezik, amik elérhetők a leszármazott osztályokból is:
 - * `area` a lakás alapterülete
 - * `roomsCount` a szobák száma
 - * `inhabitantsCount` a lakók száma
 - * `unitPrice` a lakás négyzetméterenkénti ára
 - Paraméteres konstruktoron keresztül lehessen kezdeti értéket adni az adattagoknak
 - Készítsen egy `bool MoveIn(int newInhabitants)` absztrakt metódust, ami embereket költöztet a lakásba. A visszatérési érték attól függ, hogy sikeres volt-e a beköltözés.
 - Valósítsa meg az interfész metódusát, ami a terület és a négyzetméter ár alapján visszaadja a lakás teljes értékét.
 - A lakók számát lehessen publikus geteren keresztül lekérni.
 - Írja felül a `string ToString()` metódust, ami visszaadja az egyes adattagok értékeit.
- Írjon egy **Lodgings** osztályt, ami a **Flat**-ből származik és megvalósítja az **IRent** interfészt.
 - Az örökölteken kívül egy foglalt hónapok száma (`bookedMonths`) adattaggal is rendelkezik.
 - Készítsen paraméteres konstruktort, ami az adattagok beállítását teszi lehetővé. A foglalt hónapok száma és a lakók száma kezdetben legyen 0.
 - Implementálja az interfész metódusait és tulajdonságát:
 - * Az albérlet egy főre jutó havi költsége a lakás értékének 240-ed része osztva a lakók számával.
 - * Az albérlet akkor nincs lefoglalva, ha a foglalt hónapok száma 0.
 - * A lakást akkor lehet lefoglalni, ha még nem lett korábban lefoglalva.
 - Implementálja az ösosztály absztrakt metódusát az alábbiak figyelembevételével:
 - * Csak akkor lehet beköltözni egy lakásba, ha az már ki lett bérelve, azaz le lett foglalva.
 - * Az albérletben egy szobában maximum 8 fő lakhat.
 - * Egy főre minimum 2 m^2 területnek kell jutnia.
 - * Ha minden feltétel teljesül, akkor sikeres a beköltözés. Ne feledkezzen el a lakók számának növeléséről.
 - Írja felül a `string ToString()` metódust, hogy a már foglalt hónapok száma is szerepeljen benne. A metóduson belül használja az ösosztály `ToString` metódusát is.
- Írjon egy **FamilyApartment** osztályt, ami a **Flat**-ből származik.
 - Az örökölteken kívül egy `childrenCount` adattaggal is rendelkezik, ami megadja, hogy az összes

lakóból mennyi gyerek.

- Készítsen paraméteres konstruktort az adattagok kezdeti értékadásához. A lakók és a gyerekek száma kezdetben legyen 0.
 - Készítsen egy paraméter nélküli `bool ChildIsBorn()` metódust. A metódus ellenőrizze le, hogy van e legalább két felnőtt lakója az apartmannak, és ha igen, úgy növelje a lakók és a gyerekek számát is 1-gyel. Térjen vissza logikai értékkel attól függően, hogy megszületett e a gyermek.
 - Implementálja az `IRealEstate` absztrakt metódusát az alábbiak szerint:
 - * Az apartman egy szobájában maximum 2 fő lakhat.
 - * Egy főre minimum 10 m²-nek kell jutnia.
 - * A gyerekek csak fél főnek számítanak és egy gyereknek 5 m² terület is elegendő.
 - * Az újonnan beköltözők mind felnőttek.
 - Írja felül a `string ToString()` metódust, hogy a gyerekek száma is szerepeljen benne.
- Írjon egy **Garage** osztályt, ami implementálja mindkét fenti interfészt.
 - Az osztály a következő adattagokkal rendelkezik:
 - * `area` a garázs alapterülete
 - * `unitPrice` a garázs négyzetméterenkénti ára
 - * `isHeated` fűtött e a garázs
 - * `months` hány hónapra van lefoglalva
 - * `isOccupied` áll-e benne autó
 - Implementálja az **IRent** interfész metódusát.
 - Implementálja az **IRent** interfészt tagjait.
 - * A garázs havi költsége a garázs értékének 120-ad része, de ha fűtött, akkor ezt még 1,5-del meg kell szorozni.
 - * A garázs akkor foglalt, ha a hónapok száma nagyobb mint 0, vagy ha áll bent autó.
 - * Ha még nem volt lefoglalva a garázs, akkor lehetőség van foglalni.
 - Készítsen egy paraméter nélküli `UpdateOccupied()` metódust, ami egy autó ki-, illetve beállítását reprezentálja.
 - Írja felül a `string ToString()` metódust, az adattagok megjelenítése érdekében.
 - Írjon egy **ApartmentHouse** osztályt.
 - Az osztálynak legyen egy publikusan lekérdezhető és privát módon beállítható `auto-property`-je, ami képes tárolni **Flat** és **Garage** objektumokat.
 - Tárolja még el, hogy aktuálisan hány lakás és hány garázs van a házban, illetve a maximális lakás- és garázs számot. Két utóbbit az osztály konstruktorán keresztül lehet beállítani. Kezdetben legyen üres a ház.
 - Írjon metódust lakások és garázsok felvételére. A metódus legyen képes új lakások és új garázsok felvételére is, attól függően, hogy majd milyen objektummal hívjuk meg a metódust. Ha nem sikerül felvenni az új lakást vagy garázst, akkor `false` értékkel térjen vissza a metódus.
 - Írjon egy `int InhabitantsCount` publikus gettert, ami visszaadja házban lakók számát.
 - Írjon egy `int TotalValue()` metódust, ami visszaadja a házban lévő és használatban lévő lakások és garázsok összértékét. Lakások esetén azok vannak használatban, amikben legalább 1 lakó lakik, míg garázs esetén az, ami le van foglalva.
 - Legyen egy statikus `ApartmentHouse LoadFromFile(string fileName)` metódusa, ami egy fájlt

feldolgozva képes példányosítani egy új társasházat és abba felvenni új albérleteket, családi apartmanokat és garázsokat. A fájl felépítése az alábbi mintát kövesse:

Alberlet 50.2 5 30000000

CsaladiApartman 62.8 2 40000000

Garazs 10.3 5000000 futott

- A főprogramból tesztelje az elkészített osztályokat és azok publikus metódusait és tulajdonságait.