11. táblás gyakorlat

OEP

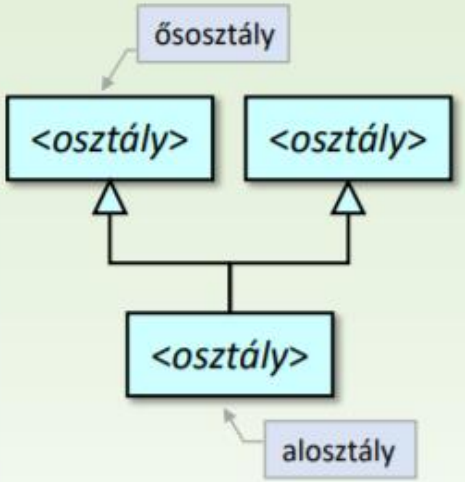
Feladatok

- Könyvkiadó
- Kórház
- Kert
- Fájlrendszer
- Vonat
- Tanulók

6. előadás

Származtatás, öröklődés

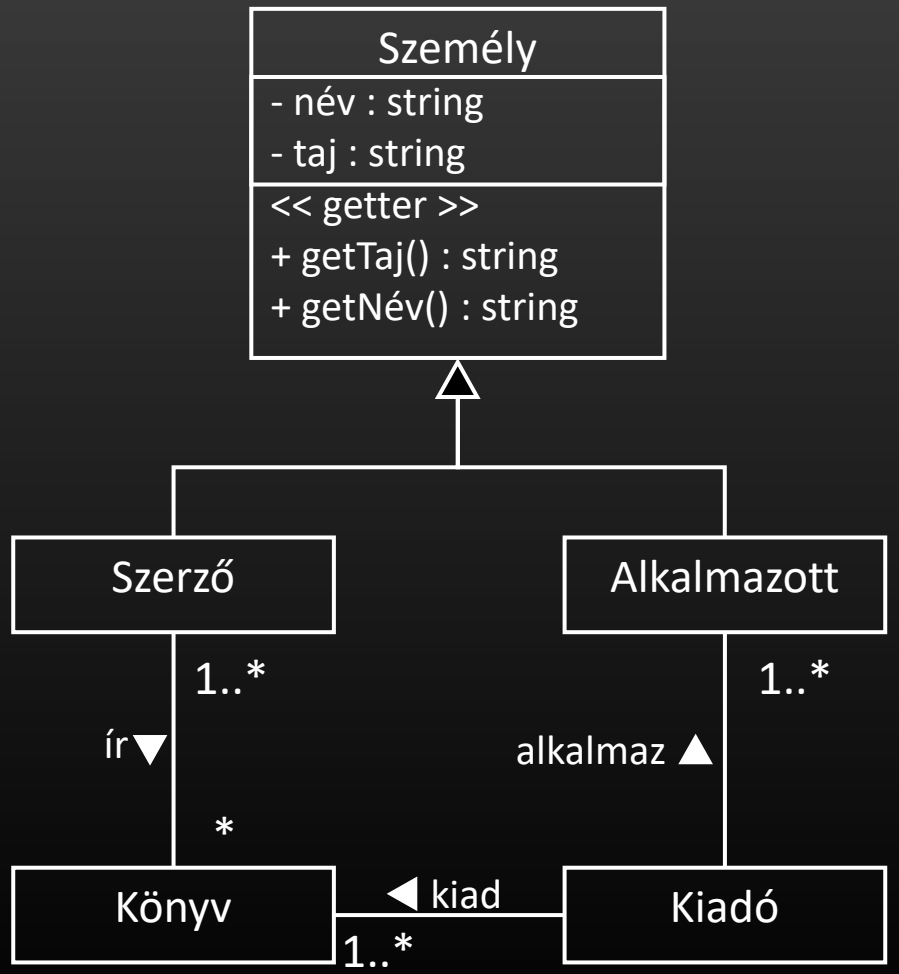
- Ha egy objektum más objektumokra hasonlít, azokkal megegyező adattagjai és metódusai vannak, akkor az osztálya a vele hasonló objektumok osztályainak mintájára írható fel, azaz belőlük származtatható. Más szóval örököli azok tulajdonságait, amelyeket azonban módosíthat is, és ki is egészíthet.



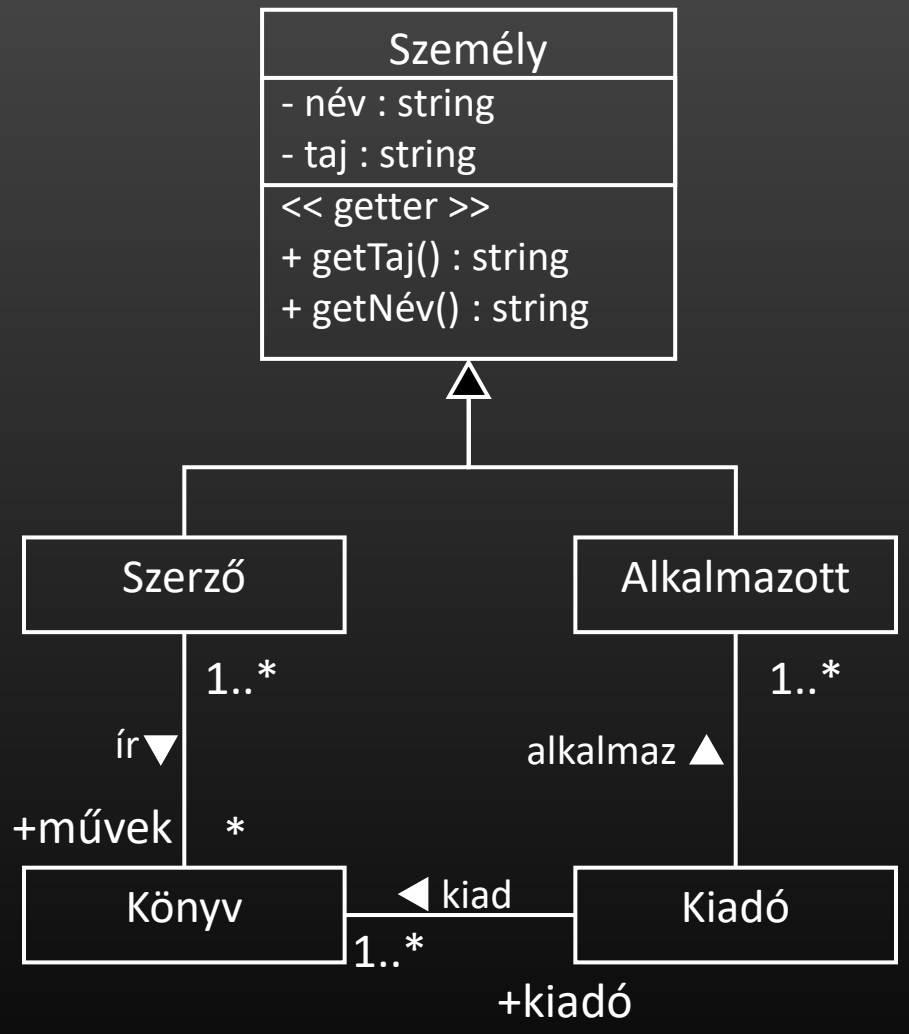
- A modellezés során kétféle okból használunk származtatást:
  - Általánosítás: már meglévő, egymáshoz hasonló osztályoknak a közös tulajdonságait leíró ősosztályát (szuperosztály) hozzuk létre.
  - Specializálás: egy osztályból származtatással hozunk létre egy alosztályt.

5 Az objektum-orientált nyelvek támogatják az öröklést: osztályok már meglévő osztályokból származtathatók.
Ősosztály változójának értékéül adható az alosztályának objektuma.

1. A könyveket legalább egy szerző írja és pontosan egy kiadó adja ki. Egy kiadó legalább egy könyvet kiad. A kiadó legalább egy alkalmazottakat foglalkoztat. Egy alkalmazottat pontosan egy kiadó alkalmaz. Az alkalmazottak és a szerzők is személyek, és nincs olyan szerző, aki kiadói alkalmazott lenne.



Új kapcsolat típus:
Származtatás vagy
öröklődés
(inheritence)



Feladat: Hány könyvet írt egy szerző?

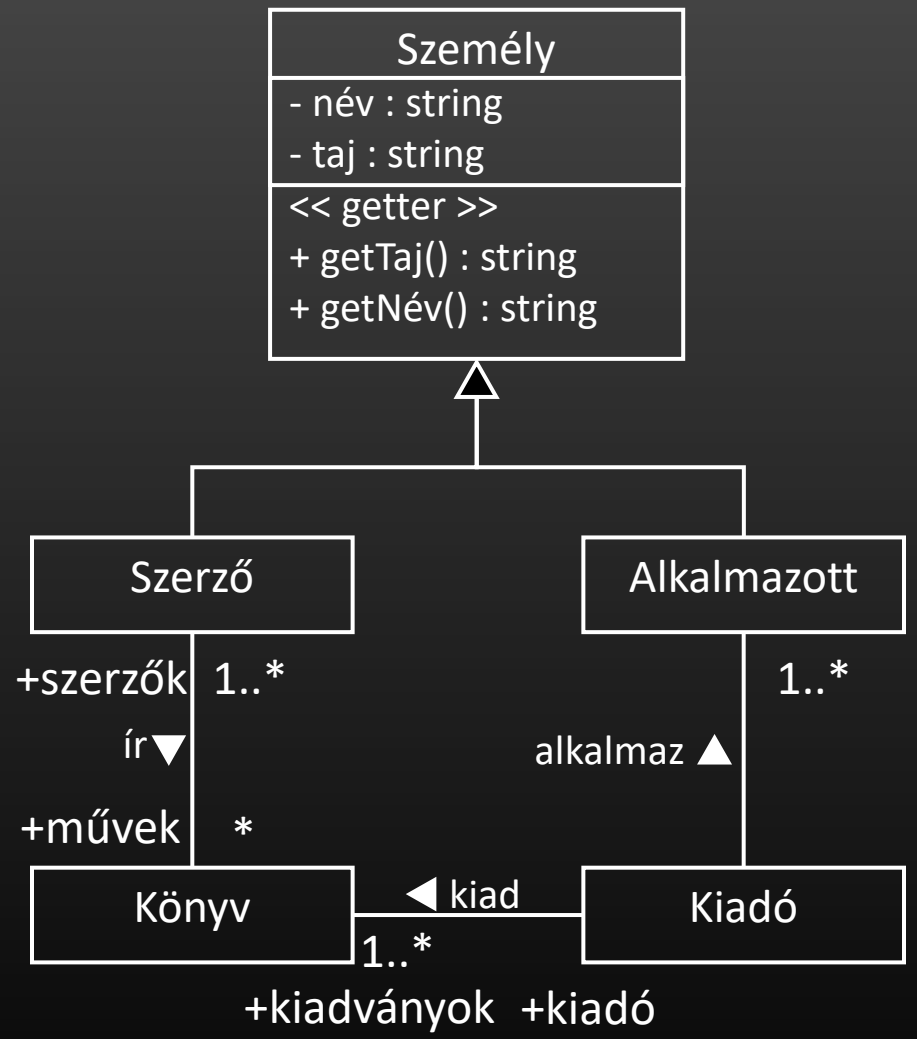
```
c := darabKönyv(szerző:Szerző)
```

```
c :=  | szerző.művek |
```

Feladat: Hány könyvet írt egy szerző egy kiadónál?

```
c := darab(szerző :Szerző, kiadó:kiadó)
```

```
c := 0
forall könyv in szerző.művek loop
    if könyv.kiadó = kiadó then
        c := c + 1
    endif
endloop
```



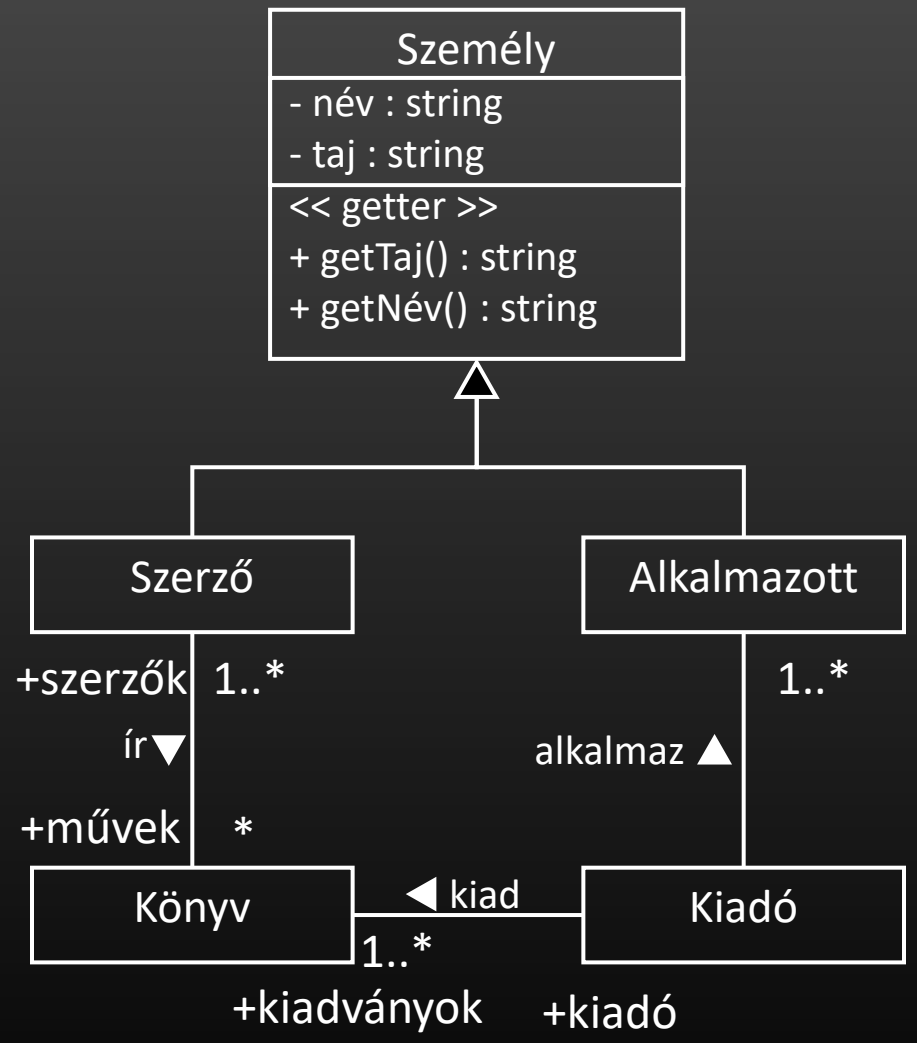
Feladatok: Ki egy kiadó által legtöbbet foglalkoztatott szerző?

maximum kiválasztás (kiadó felől indulva)

```
név := sztár(kiadó:Kiadó)
```

```
elem := kiadó.kiadványok[1].szerzők[1]
max := darab(szerző, kiadó)
forall könyv in kiadó.kiadványok loop
    forall szerző in könyv.szerzők loop
        c := darab(szerző, kiadó)
        if c >max  then
            max, elem := c, szerző
        endif
    endloop
endloop
név := elem.getNév()
```

Túl költséges, javítsunk rajta!



Még egyszer

Feladatok: Ki egy kiadó által legtöbbet foglalkoztatott szerző?

zsákolás

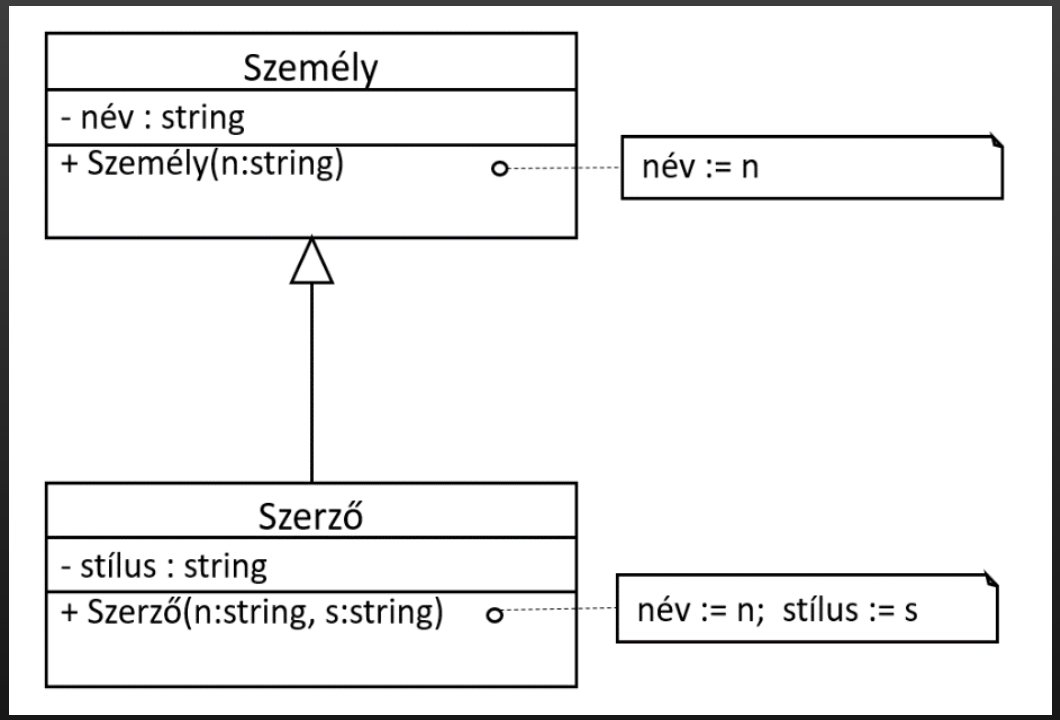
```
név := sztár(kiadó:Kiadó)
zsák : Zsák

zsák.erase()
forall könyv in kiadó.kiadványok loop
    forall szerző in könyv.szerzők loop
        zsák.putIn(szerző)
    endloop
endloop
név := zsák.mostFrequent().getNév()
```

Első kvíz kérdés

- Milyen fordítási hibát okoz a Szerző osztályának konstruktora?

Nem lehet hivatkozni a Személy ősosztály privát név adattagjára.
Személy osztálynak nincs üres konstruktora.



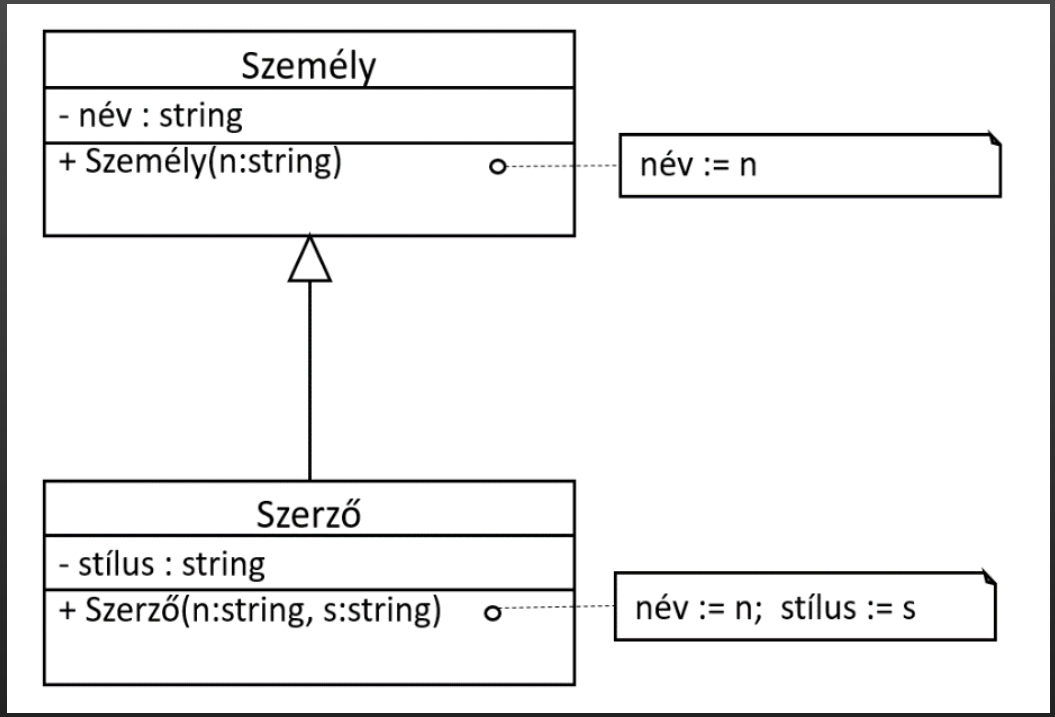
Második kvíz kérdés

- Milyen fordítási hibát okoz a Szerző osztályának konstruktora?

Nem lehet hivatkozni a Személy ősosztály privát név adattagjára.
Személy osztálynak nincs üres konstruktora.

- Milyen módokon lehetne ezt a hibát orvosolni?

Láthatóság: private, protected, public
Vigyázat!
Származtatás is lehet public, protected, private



(1) Legyen a név adattag védett (protected) a Személy osztályban.

(2) Hívjuk meg a Szerző konstruktorából a Személy konstruktorát a név:=n értékadás helyett.

(3) Készítsünk a Személy osztályban egy (publikus vagy legalább védett) setNév(str:string) metódust, amely a név adattagot felülírja, és ezt hívjuk meg az név:=n értékadás helyett.

6. előadás

Származtatás és láthatóság

- Egy alosztályban hivatkozhatunk az ősosztályában definiált publikus és védett tagokra, de nem érjük el az ősosztály privát tagjait, azokhoz csak indirekt módon, az ősosztálytól örökölt metódusokkal férhetünk hozzá.
- A származtatás módja maga is lehet
  - publikus (*public*): ekkor az ősosztály publikus és védett tagjai az ősosztályban definiált láthatóságukkal együtt öröklődnek az alosztályra. (Az UML szerint ez a default, de a C++ nyelvben nem.)
  - védett (*protected*): ekkor az ősosztály publikus és védett tagjai mind védettek lesznek az alosztályban.
  - privát (*private*): ekkor az ősosztály publikus és védett tagjai privátok lesznek az alosztályban.

Bázis osztályok elérése

```
class A{
public:

protected:

private:

};
```

```
class B : public A{
//public
      public marad
//protected
      protected marad

};
```

```
class B : protected A{
//public
    protected lesz
//protected
      protected marad

};
```

```
class B : private A{
//public
      private lesz
//protected
      private lesz

};
```

Vigyázat! Ha elhagyjuk, C++ nyelvben class esetén a „private” lesz az alapértelmezett, struct esetén pedig a ”public”!

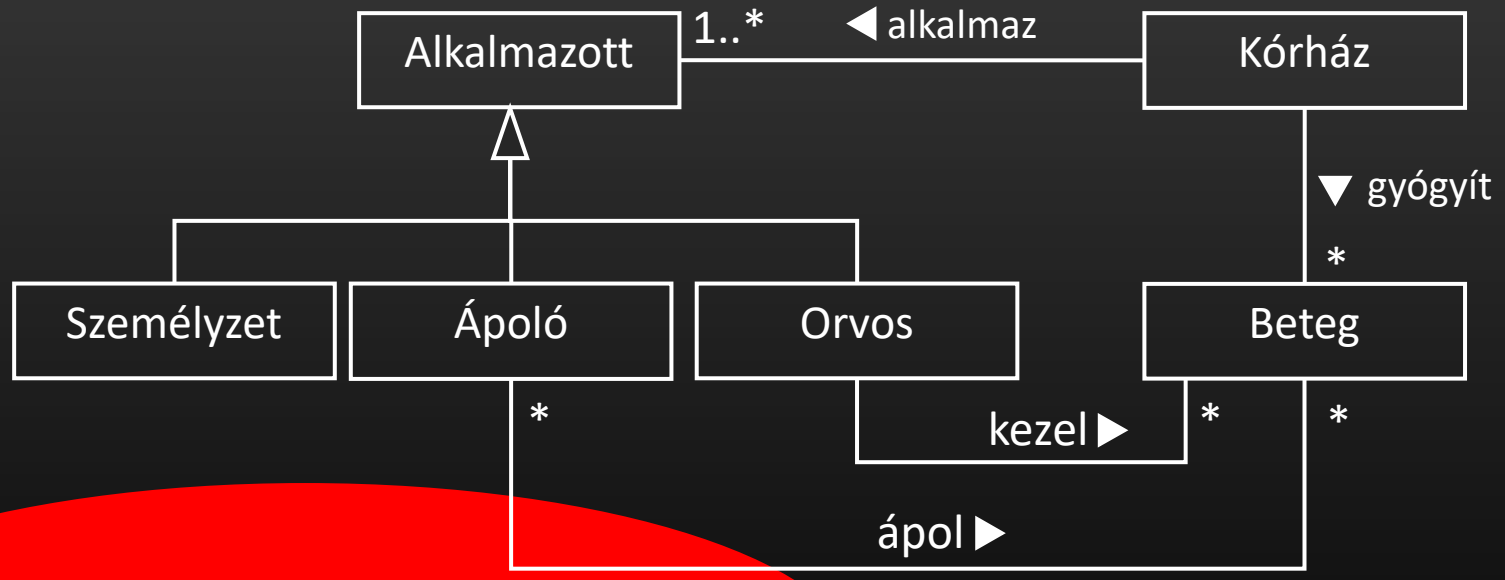
Bázisosztályok elérése útmutató:

- Ha A nyilvános bázisosztály, nyilvános tagjai bárhol használhatók. Ezenkívül védett tagjai B tagfüggvényeiből és barátaiból, valamint B-ből származó osztályok tagfüggvényeiből és barátaiból érhetők el. Bármely függvény végezhet B*-ból A*-ra való konverziót.

- Ha A védett bázisosztály, akkor nyilvános és védett tagjai csak B tagfüggvényeiből és barátaiból, valamint B-ből származó osztályok tagfüggvényeiből és barátaiból érhetők el. B tagfüggvényei és barátai, valamint B-ből származó osztályok tagfüggvényei és barátai végezhetnek B*-ból A*-ra való konverziót.

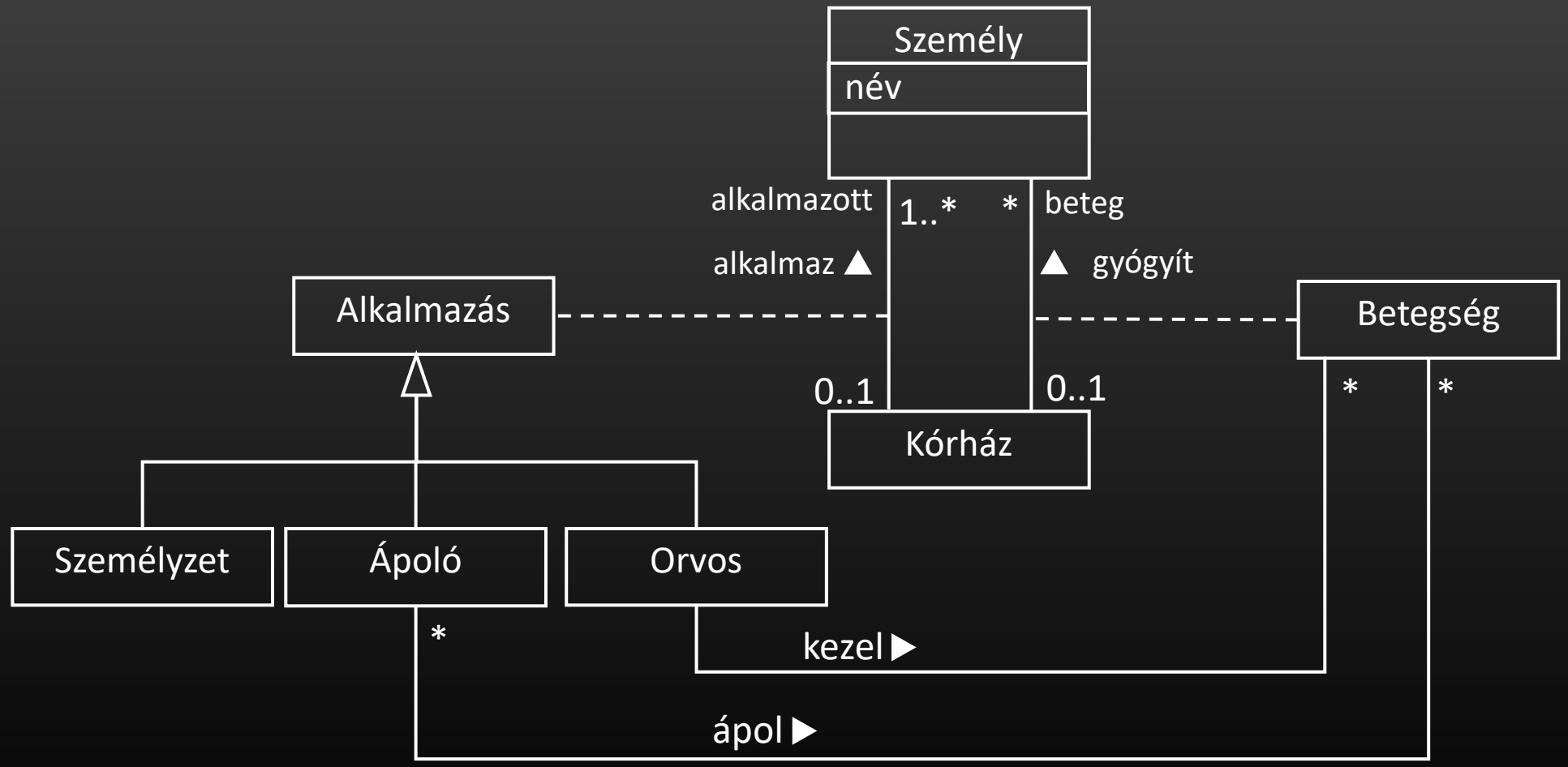
- Ha A privát bázisosztály, akkor nyilvános és védett tagjai csak B tagfüggvényeiből és barátaiból érhetők el. Csak B tagfüggvényei és barátai konvertálhatnak egy B* mutatót A* mutatóvá.

2. A kórházban legalább egy alkalmazott dolgozik, aki lehet orvos, ápoló vagy a személyzethez tartozhat. A kórházban betegek vannak, akiket orvosok kezelnek és ápolók ápolnak. Egy beteget pontosan egy orvos kezel, és tetszőleges számú ápoló ápol. Egy orvos tetszőleges számú beteget kezelhet, egy ápoló tetszőleges számú beteget ápolhat.



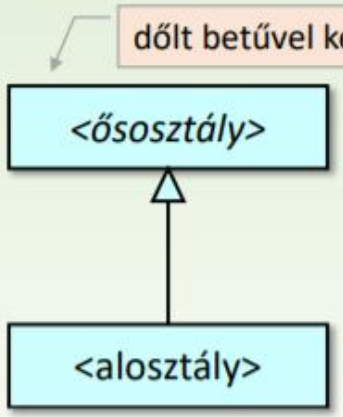
Mi van, ha egy orvos, ápoló, vagy egy tagja a személyzetnek megbetegszik, és a kórházba kerül mint beteg?

Vegyük figyelembe azt is, hogy egy alkalmazott is lehet beteg, és a betegséggel nem szűnik meg az alkalmazotti viszonya! Módosítsuk az előző osztálydiagramot ennek megfelelően!



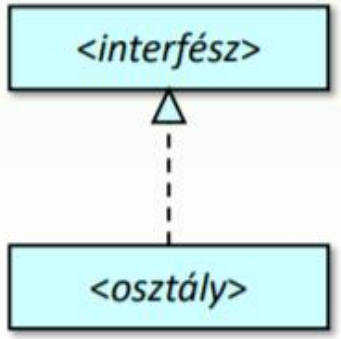
7. előadás

Absztrakt osztály, interfész

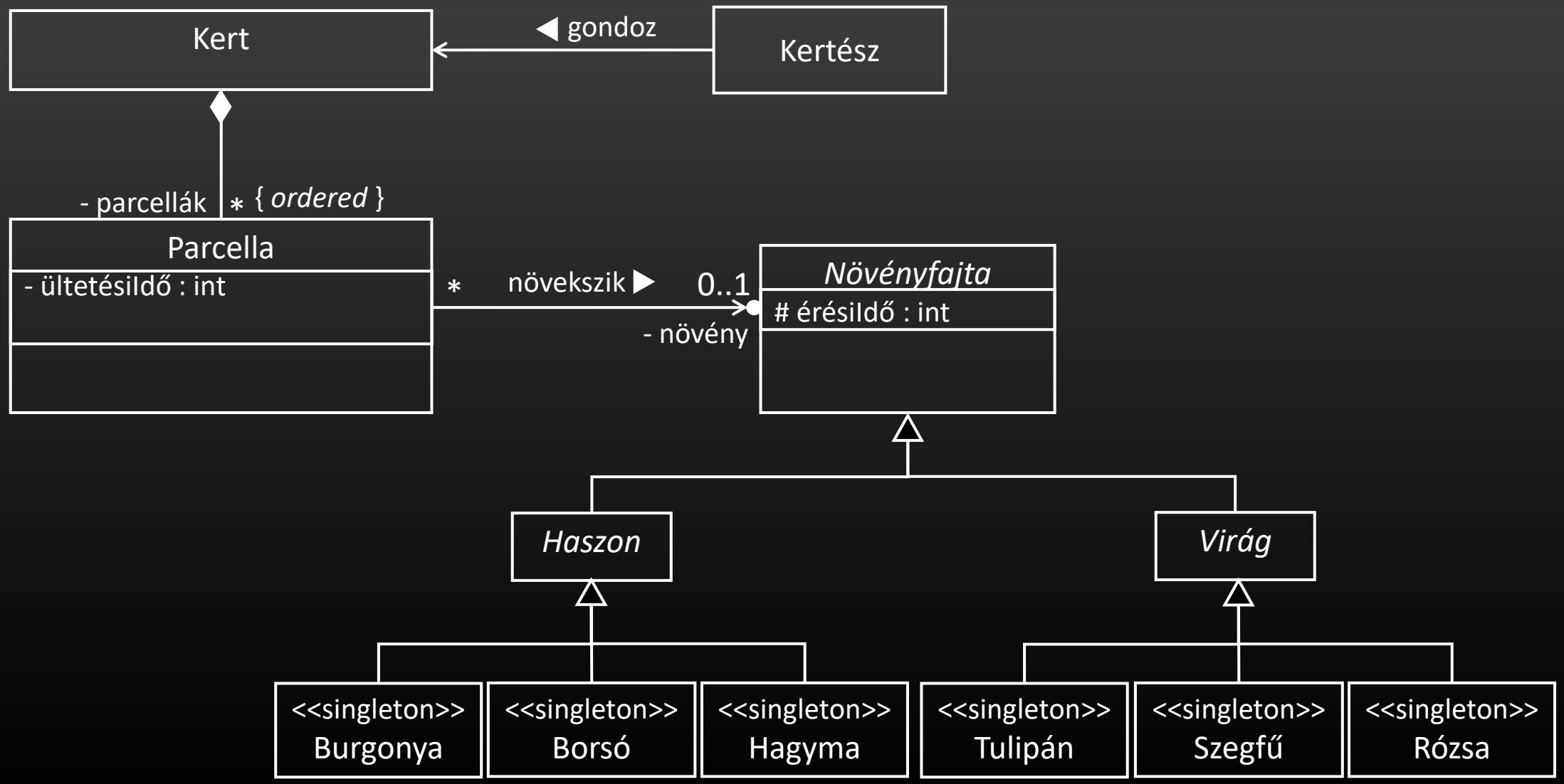


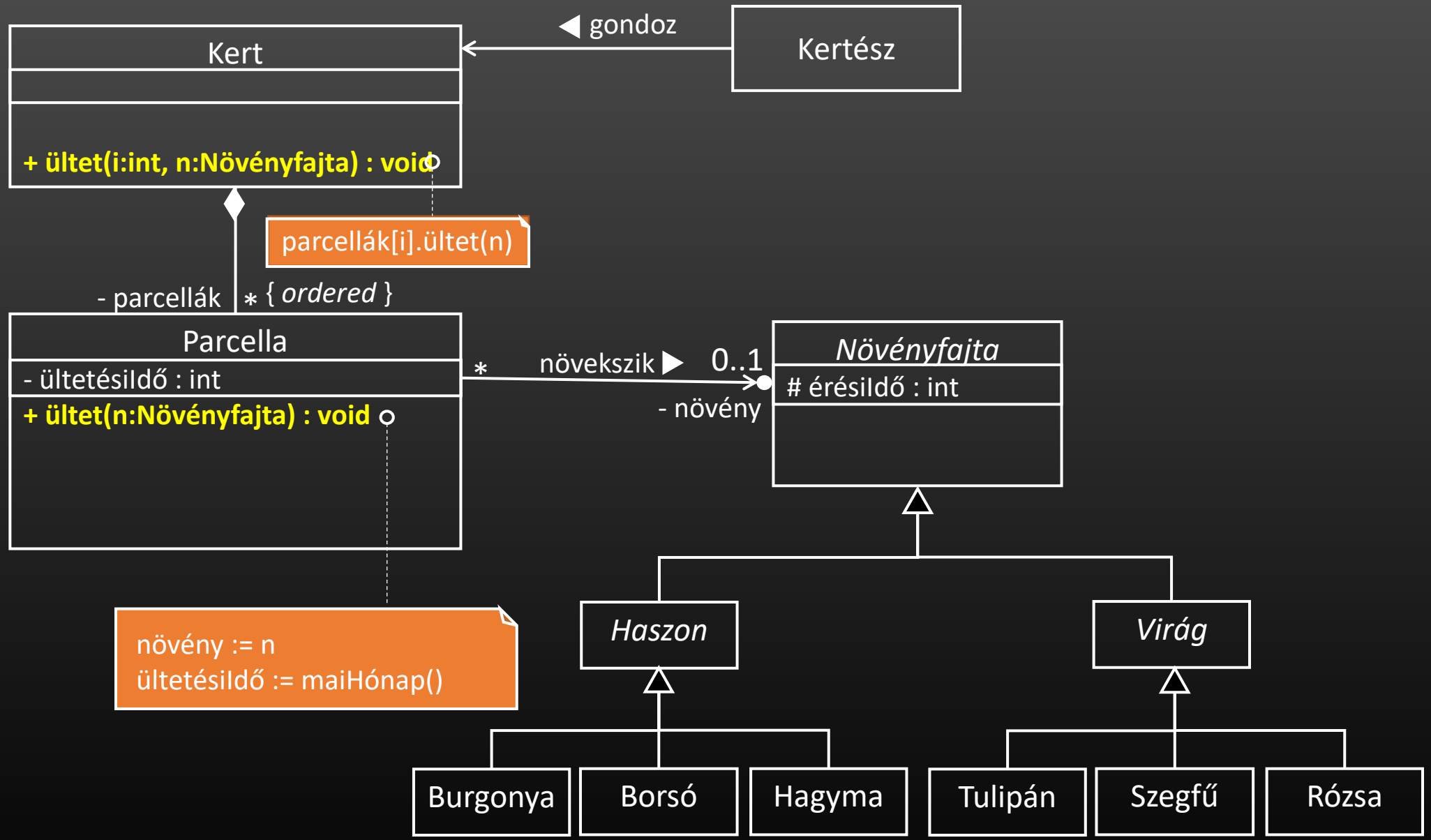
- Absztrakt (*abstract*) osztály az, amelyből nem példányosítunk objektumokat, kizárólag ősosztályként szolgálnak a származtatásokhoz.
- Egy osztály attól lesz absztrakt, hogy
  - konstruktorai nem publikusak,
  - vagy legalább egy metódusa absztrakt (dőlt betűvel kell írni), azaz nincs implementálva, csak származtatás során írjuk majd felül

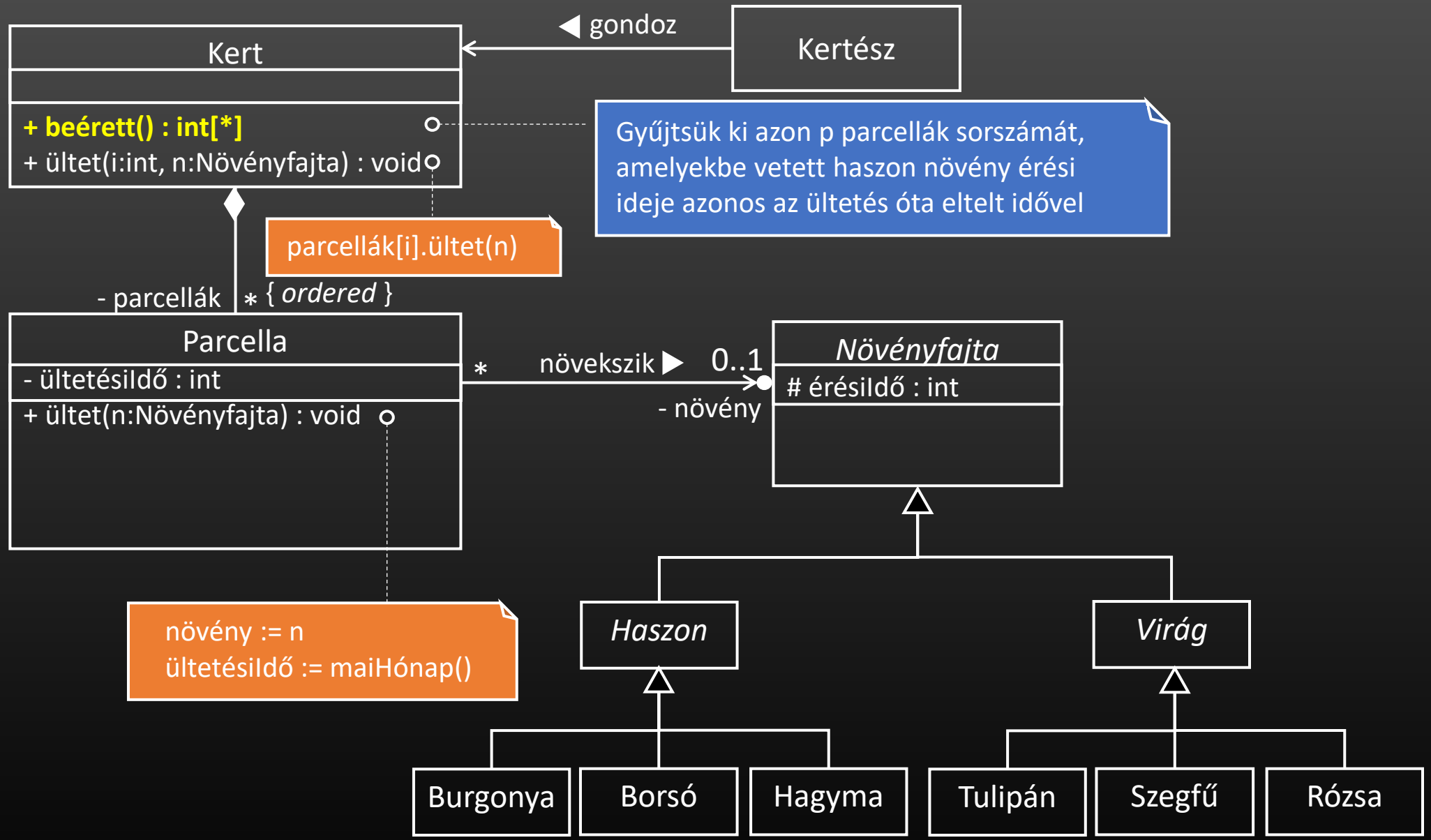
- Interfésznek, azaz tisztán absztrakt (*pure abstract*) osztálynak nevezzük azt az osztályt, amelyiknek egyetlen metódusa sincs implementálva.
- Egy interfészből származtatott osztály, amelyik az interfész minden absztrakt metódusát implementálja, az megvalósítja az interfészt.

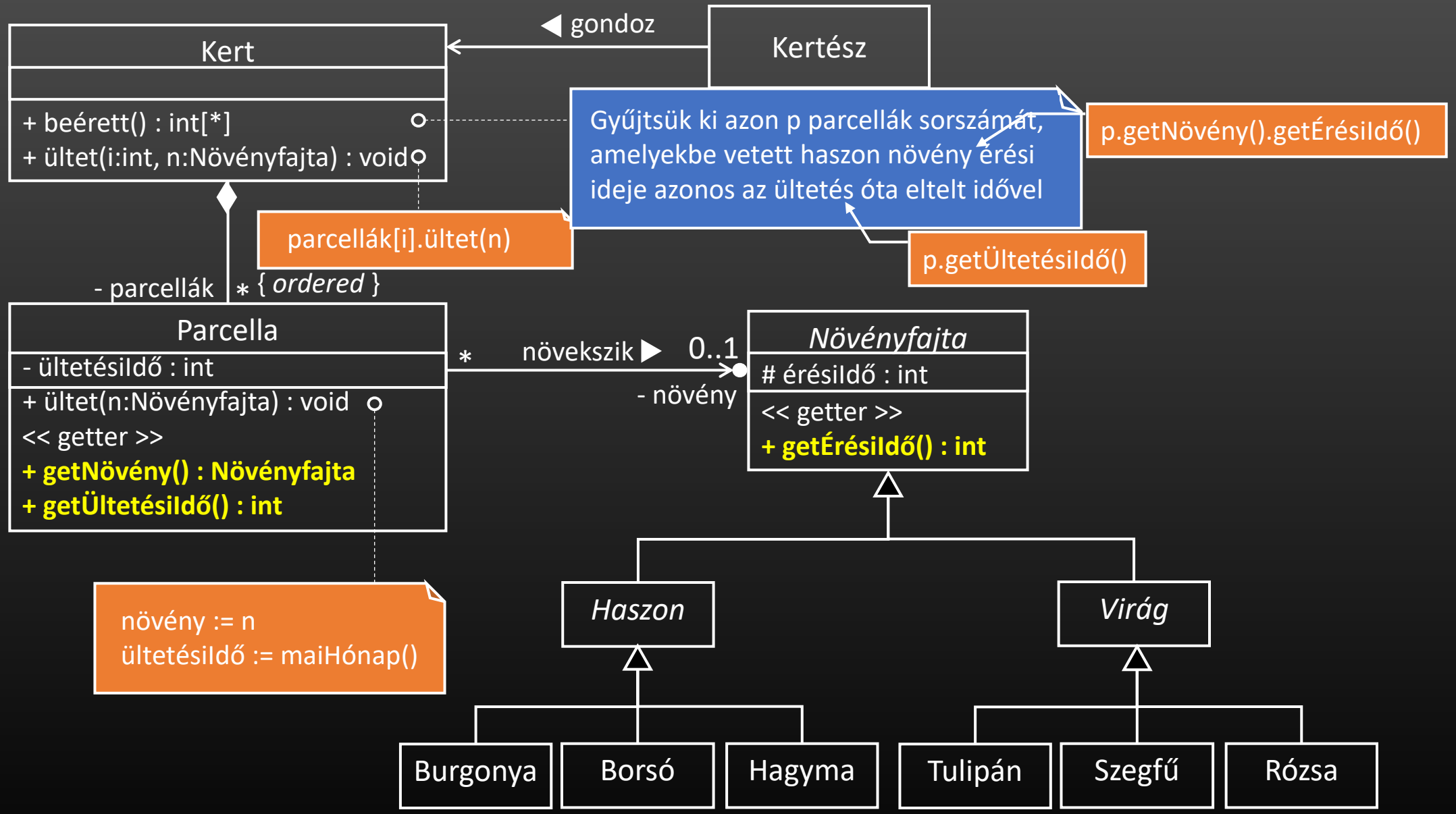


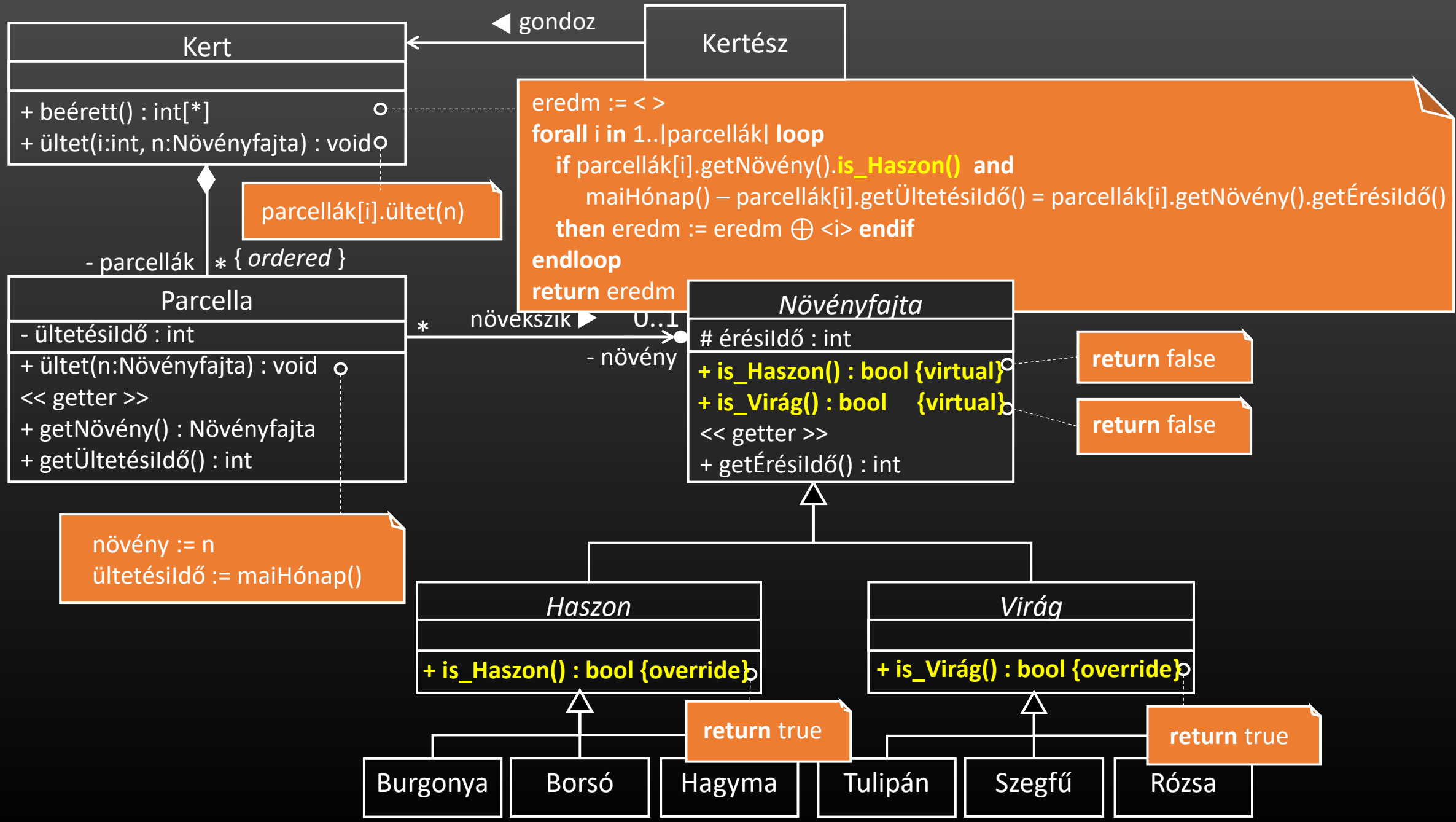
3. Egy kertet egy kertész gondoz. A kert parcellákból áll, minden parcellába egyféle növény ültethető. Az ültetés idejét eltároljuk (hónapban). A növények lehetnek haszonnövények, mint burgonya, borsó, paprika; vagy virágok, mint tulipán, szegfű, rózsa. A növényeknek ismerjük az érési idejét (hónapban). Listázza ki a kertész azokat a parcellákat, ahol az adott hónapban haszonnövények fognak beérni!



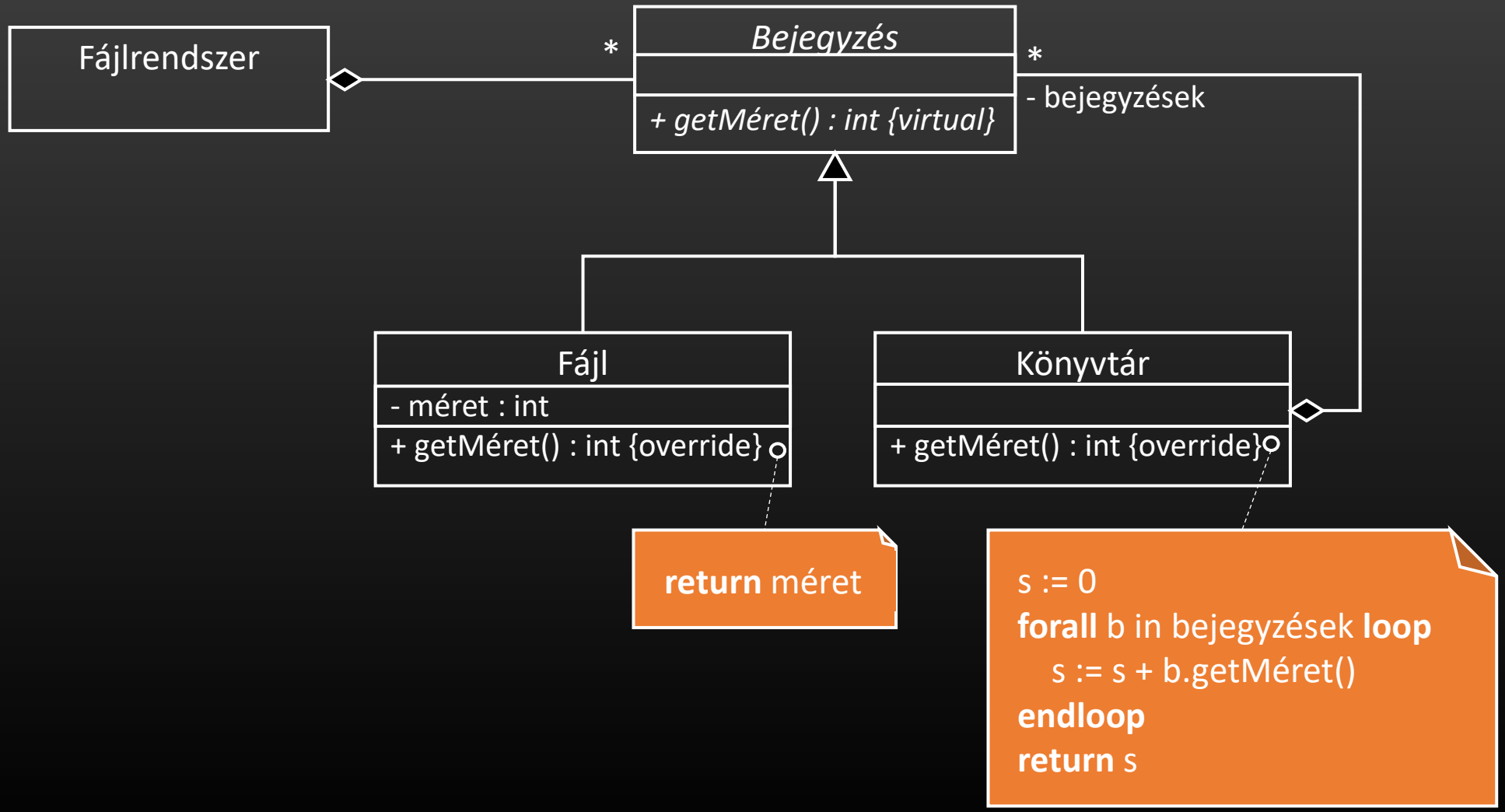

Kert
+ ültet(i:int, n:Növényfajta) : void
gondoz
Kertész
parcellák[i].ültet(n)
- parcellák
* { ordered }
Parcella
- ültetésidő : int
+ ültet(n:Növényfajta) : void
*
növekszik
0..1
- növény
Növényfajta
érésidő : int
növény := n
ültetésidő := maiHónap()
Haszon
Virág
Burgonya
Borsó
Hagyma
Tulipán
Szegfű
Rózsa


Kert
+ beérett() : int[*]
+ ültet(i:int, n:Növényfajta) : void
gondoz
Kertész
Gyűjtsük ki azon p parcellák sorszámát, amelyekbe vetett haszon növény érési ideje azonos az ültetés óta eltelt idővel
parcellák[i].ültet(n)
- parcellák
* { ordered }
Parcella
- ültetésildő : int
+ ültet(n:Növényfajta) : void
*
növekszik
0..1
- növény
Növényfajta
érésildő : int
növény := n
ültetésildő := maiHónap()
Haszon
Virág
Burgonya
Borsó
Hagyma
Tulipán
Szegfű
Rózsa

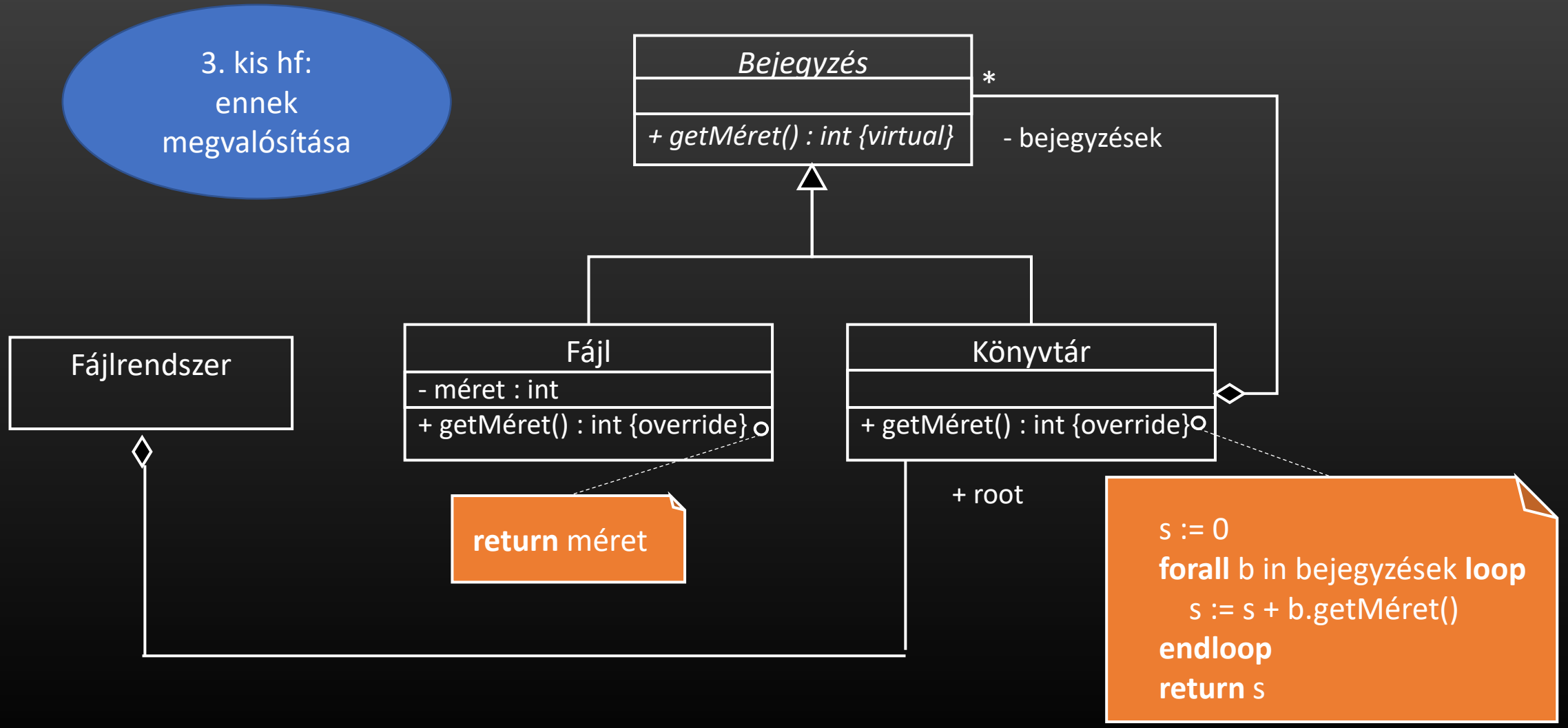

Kert
+ beérett() : int[*]
+ ültet(i:int, n:Növényfajta) : void
gondoz
Kertész
Gyűjtsük ki azon p parcellák sorszámát, amelyekbe vetett haszon növény érési ideje azonos az ültetés óta eltelt idővel
p.getNövény().getÉrésiIdő()
p.getÜltetésiIdő()
parcellák[i].ültet(n)
- parcellák
* { ordered }
Parcella
- ültetésiIdő : int
+ ültet(n:Növényfajta) : void
<< getter >>
+ getNövény() : Növényfajta
+ getÜltetésiIdő() : int
*
növekszik
0..1
- növény
Növényfajta
érésiIdő : int
<< getter >>
+ getÉrésiIdő() : int
növény := n
ültetésiIdő := maiHónap()
Haszon
Virág
Burgonya
Borsó
Hagyma
Tulipán
Szegfű
Rózsa


Kert
+ beérett() : int[*]
+ ültet(i:int, n:Növényfajta) : void
gondoz
Kertész
eredm := < >
forall i in 1..|parcellák| loop
if parcellák[i].getNövény().is_Haszon() and
maiHónap() – parcellák[i].getÜltetésiIdő() = parcellák[i].getNövény().getÉrésiIdő()
then eredm := eredm ⊕ <i> endif
endloop
return eredm
parcellák[i].ültet(n)
- parcellák
* { ordered }
Parcella
- ültetésiIdő : int
+ ültet(n:Növényfajta) : void
<< getter >>
+ getNövény() : Növényfajta
+ getÜltetésiIdő() : int
növekszik
0..1
*
- növény
Növényfajta
érésiIdő : int
+ is_Haszon() : bool {virtual}
+ is_Virág() : bool {virtual}
<< getter >>
+ getÉrésiIdő() : int
return false
return false
növény := n
ültetésiIdő := maiHónap()
Haszon
+ is_Haszon() : bool {override}
Virág
+ is_Virág() : bool {override}
return true
return true
Burgonya
Borsó
Hagyma
Tulipán
Szegfű
Rózsa

4. Egy számítógépes fájlrendszerben a fájlokat könyvtárakba szervezzük. Minden könyvtár tetszőleges számú fájlt vagy könyvtárat tartalmazhat. A fájlrendszerben a fájlok lehetnek közvetlen a fájlrendszerhez kötve (gyökér), vagy valamelyik könyvtárban is elhelyezkedhetnek. Mennyi tárhelyet foglal egy adott könyvtár?



4. Egy számítógépes fájlrendszerben a fájlokat könyvtárakba szervezzük. Minden könyvtár tetszőleges számú fájlt vagy könyvtárat tartalmazhat. A fájlrendszerben a fájlok lehetnek közvetlen a fájlrendszerhez kötve (gyökér), vagy valamelyik könyvtárban is elhelyezkedhetnek. Mennyi tárhelyet foglal egy adott könyvtár?



6. előadás

Dinamikus altípusos polimorfizmus

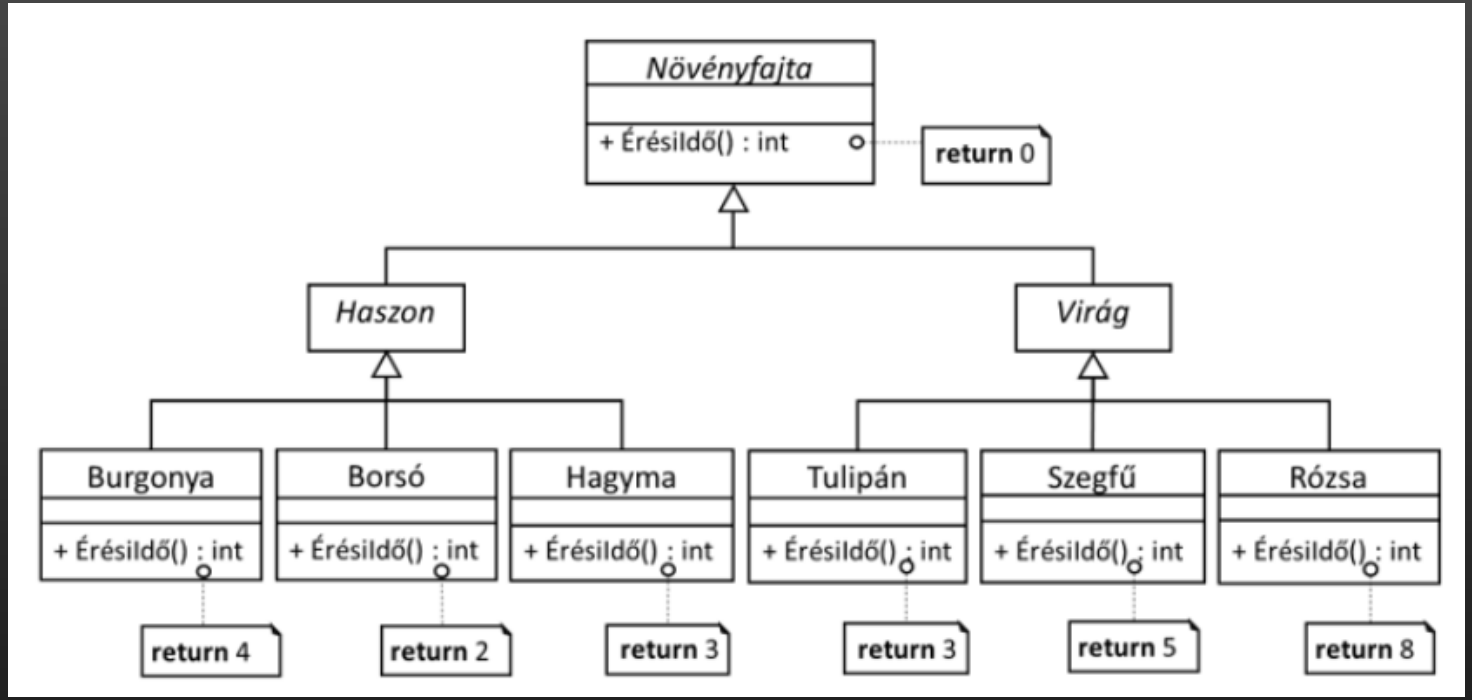
virtual kell legyen!

- Ha egy ősosztály metódusát a leszármazott osztályban felülírjuk (override), akkor ez a metódus több alakkal is rendelkezik (polimorf).
- Mivel egy ősosztály típusú változónak mindig értékül adható az alosztályának egy példánya, ezért csak futási időben derülhet ki, hogy ez a változó az ősosztály egy példányára vagy alosztályának egy példányára hivatkozik-e. (késői vagy futási idejű vagy dinamikus kötés).
- Ha egy referencia- vagy pointer változóra egy polimorf virtuális metódust hívunk meg, akkor e metódusnak azon osztálybeli változata fut majd le, amelyik osztálynak a példányára hivatkozik a referencia- vagy pointer változó hivatkozik (dinamikus altípusos polimorfizmus).

6 Az objektum-orientált nyelvek ismérve a dinamikus altípusos polimorfizmus.

3. Kvíz feladat

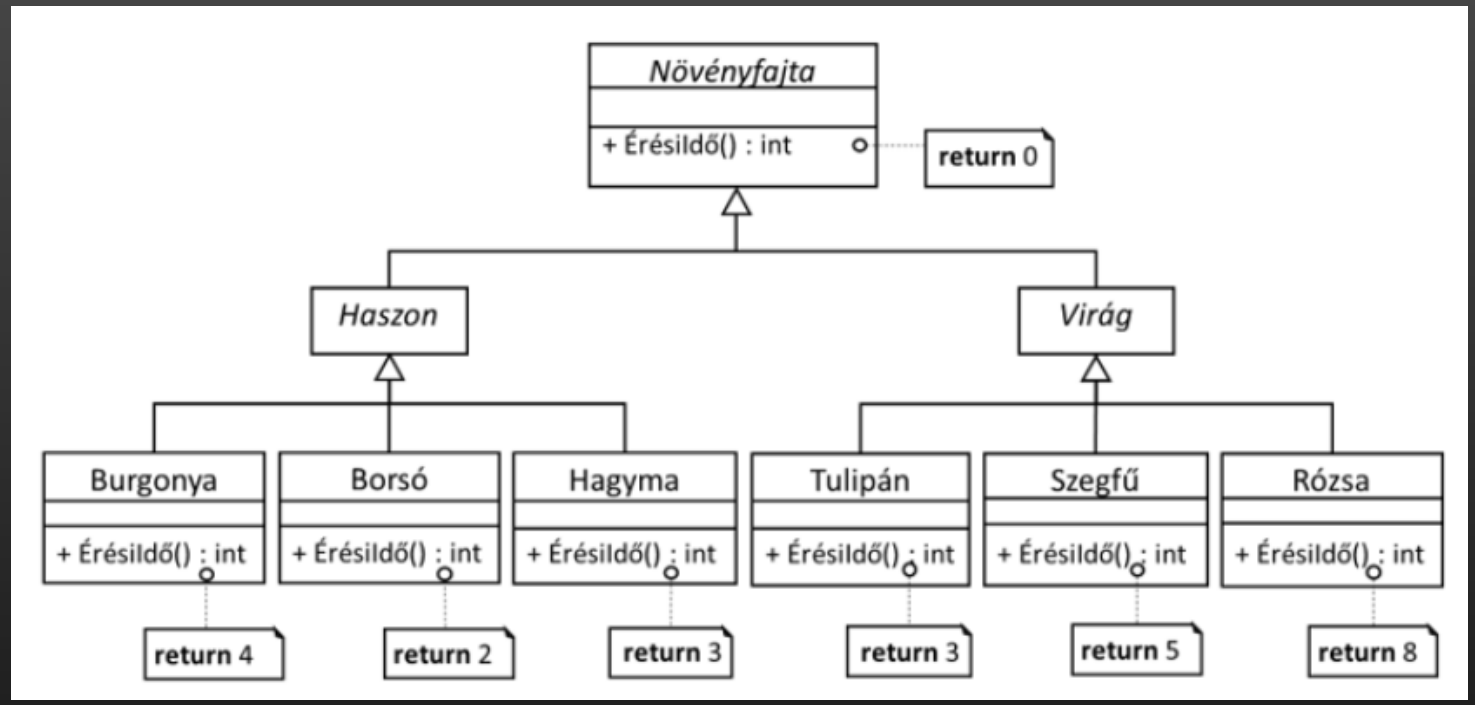
- Tekintsük a mellékelt osztálydiagramot.
- Tekintsük a mellékelt programot.
- Mi lesz az s változó értéke az program lefutásának végén, ha:
  - (1) Érésildő() metódus virtuális? 0
  - (2) Érésildő() metódus nem virtuális? 0



b : Borsó, s1 : Szegfű, s2 : Szegfű, h : Hagyma, r1 : Rózsa, r2 : Rózsa
x: Novenyfajta[]
x := [b, s1, s2, h, r1, r2] -- növényeket tartalmazó sorozat
s := 0; forall e in x loop s := s + e.Érésildő() endloop

3. Kvíz feladat

- Tekintsük ismét a mellékelt osztálydiagramot.
- Tekintsük most a mellékelt másik alakú programot.
- Mi lesz az s változó értéke az program lefutásának végén, ha:
  - (1) Érésildő() metódus virtuális? 31
  - (2) Érésildő() metódus nem virtuális? 0

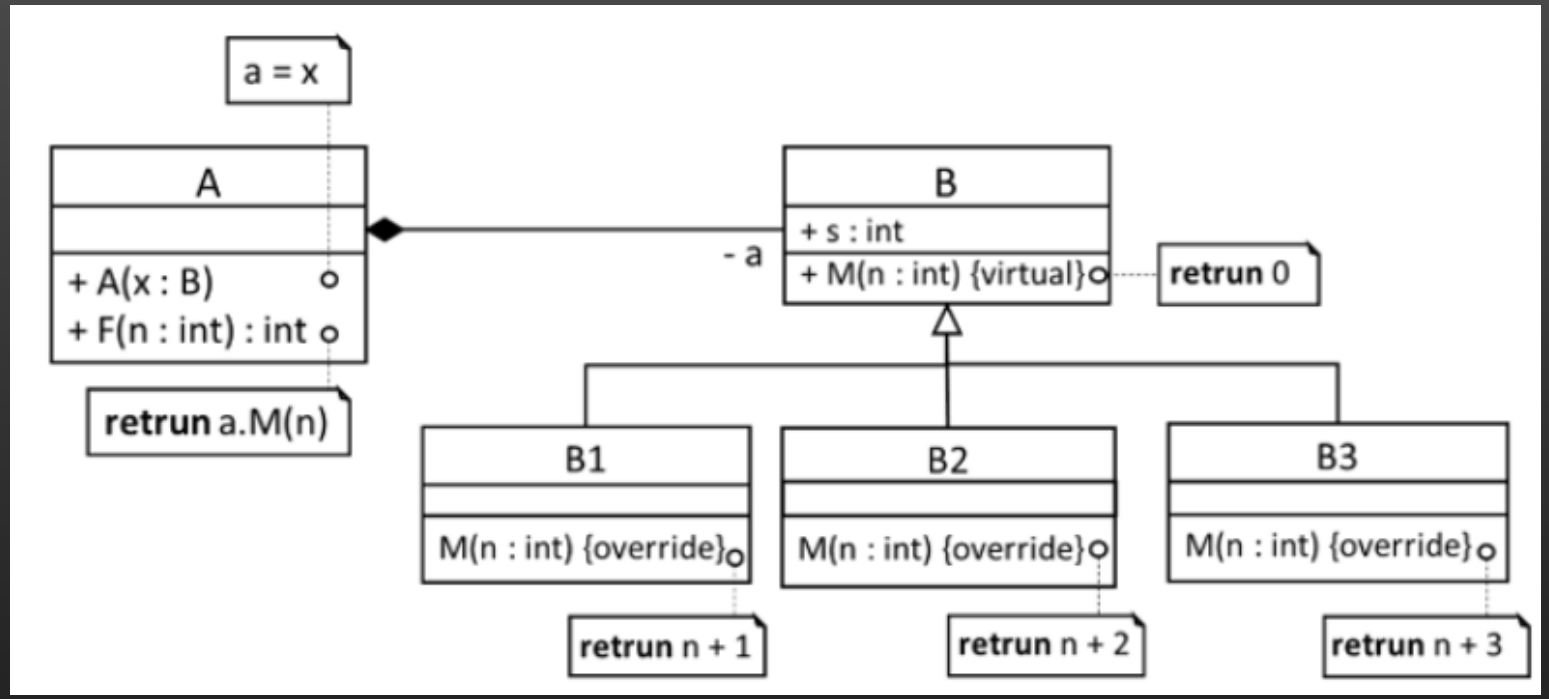


b := new Borsó, s1 := new Szegfű, s2 := new Szegfű, -- referenciák
h := new Hagyma, r1 := new Rózsa, r2 := new Rózsa -- referenciák
x:Novenyfajta*[]
x := [b, s1, s2, h, r1, r2] – növények referenciáit tartalmazó sorozat
s := 0; forall e in x loop s := s + e->Érésildő() endloop

Futási idejű poliformizmus

4. kvíz kérdés

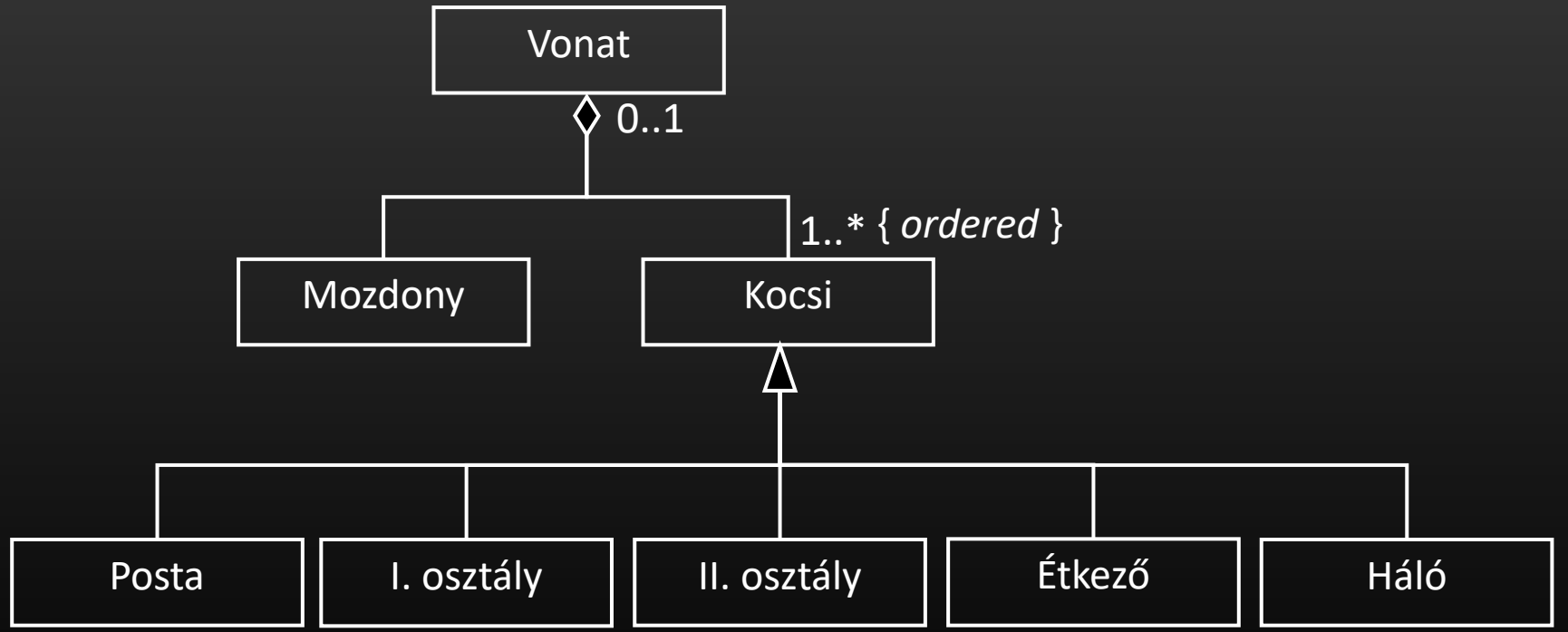
- Tekintsük a mellékelt osztálydiagramot.
- Milyen értéket ad vissza az o.F(13) metódushívás, ha:



(1) o : A(new B1)	14
(2) o: A(new B2)	15
(3) o: A(new B3)	16
(4) o: A(new B)	0

Stratégia tervminta

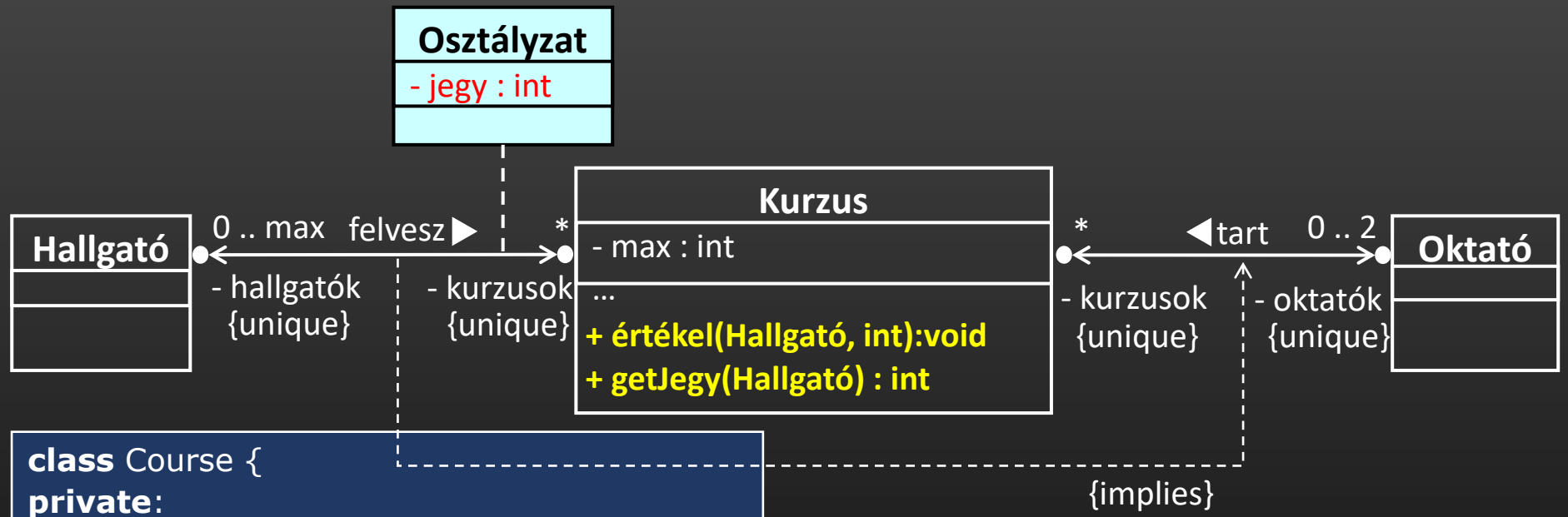
5. Egy vonatszerelvény egy mozdonyból és legalább egy kocsiból áll. A kocsikat a mozdony után adott sorrend szerint kapcsolják össze. A vonatot különböző típusú (eltérő hosszúságú) kocsikból állíthatják össze. A lehetséges típusok: első osztályú, másodosztályú, posta, étkező, háló, tehervagon, platóvagon. Egy szerelvény vagy személyszállító (nincs tehervagon, sem platóvagon), vagy teherszállító (csak tehervagon vagy platóvagon van). Egy mozdony, illetve kocsi egy időben csak egy vonathoz tartozhat. Önálló feladat: milyen hosszú egy adott szerelvény?



Önálló feladat

- Egy vonatszerelvény egy mozdonyból és legalább egy kocsiból áll.
  - A kocsikat a mozdony után adott sorrend szerint kapcsolják össze.
  - A vonatot különböző típusú (eltérő hosszúságú) kocsikból állíthatják össze.
  - A lehetséges típusok: első osztályú, másodosztályú, posta, étkező, háló, tehervagon, platóvagon.
  - Egy szerelvény vagy személyszállító (nincs tehervagon, sem platóvagon), vagy teherszállító (csak tehervagon vagy platóvagon van).
  - Egy mozdony, illetve kocsi egy időben csak egy vonathoz tartozhat. Milyen hosszú egy adott szerelvény?

5. Egészítsük ki az előadáson szerepelt (Kurzusok nyilvántartása) feladat modelljét azzal, hogy jegyeket is lehessen adni egy hallgatónak egy adott kurzusán.




```
class Course {
private:
      int _max;
      std::vector<Teacher*> _teachers;
      std::map<Student*, int> _students;
public:
      Course(int a) { _max = a; }
      bool can_lead(Teacher *pt) ;
      bool has_teacher() const ;
      bool can_sign_up(Student *ps) ;
      void grade(Student* ps, int mark);
      int getMark(Student* ps) const;
};
```

Modellezzük azt is, hogy csak az arra jogosult személyek adhassanak jegyet és olvashassák el azt.