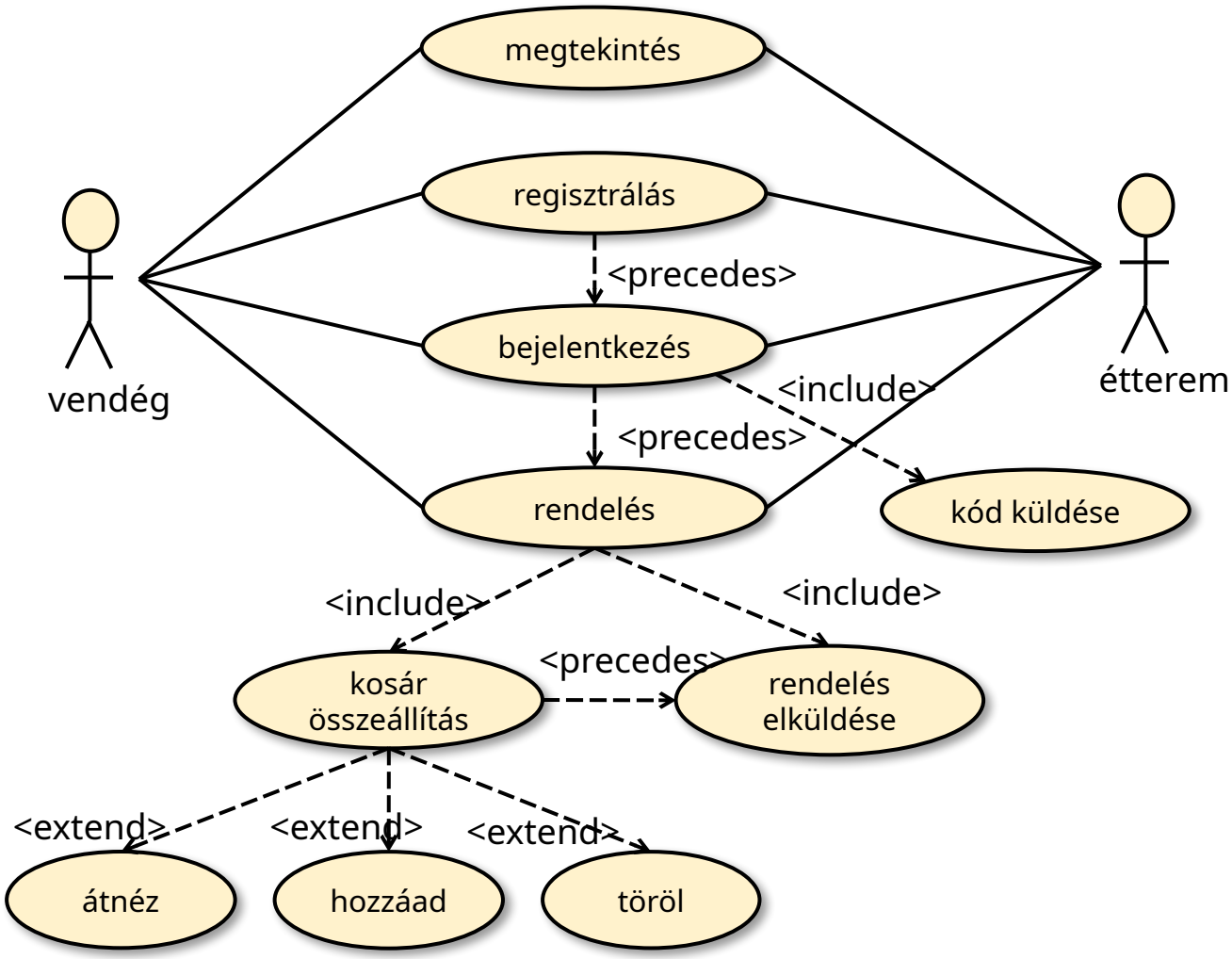
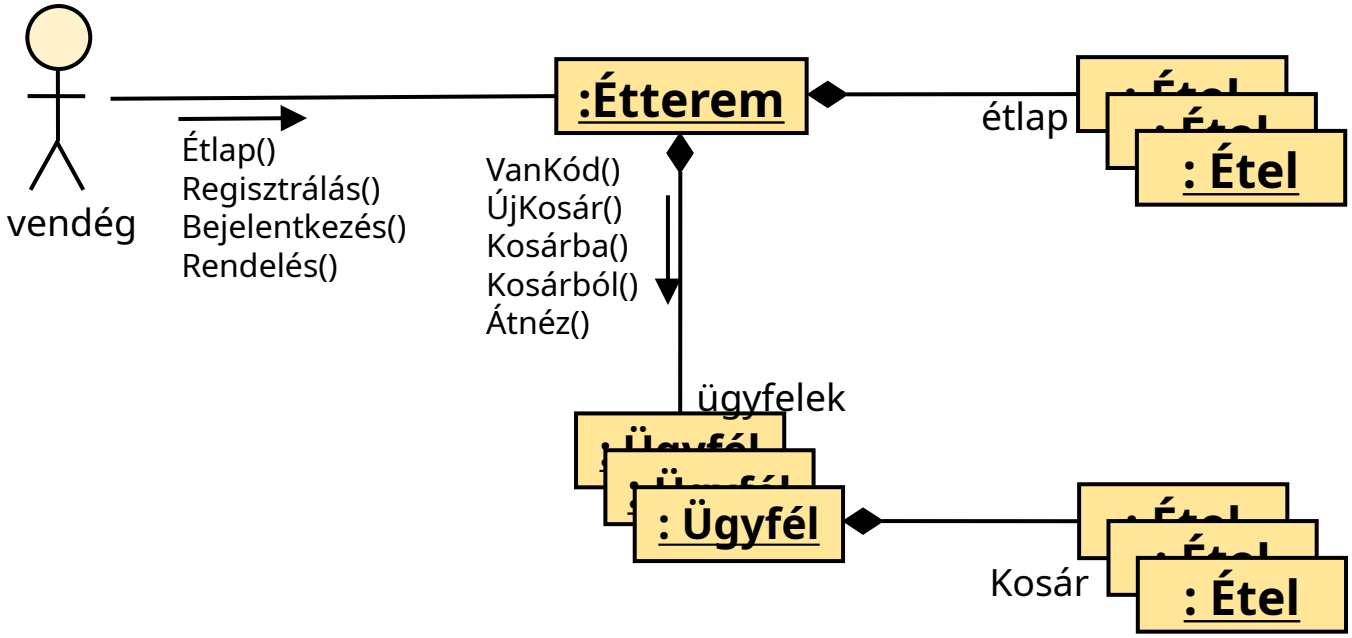
Egy ételrendelő weblapra érkezve a felhasználó megtekintheti az étlapot, de rendelni csak regisztrációt, illetve bejelentkezést követően tud. A már regisztrált ügyfél a bejelentkezéskor kap egy kódot, amellyel a rendelését tudja intézni: kiválaszthatja az étlapról a megrendelni kívánt ételeket (ugyanazt az ételt többször is), amelyek bekerülnek a „kosarába”. Rendelés közben bármikor átnézheti a kosarát, ahonnan tud törölni tételeket, és újabbakat tud hozzáadni. Végül elküldheti a rendelést.





Étterem

- ügyfelek : Ügyfél[]
- étlap : Étel[] {getter}

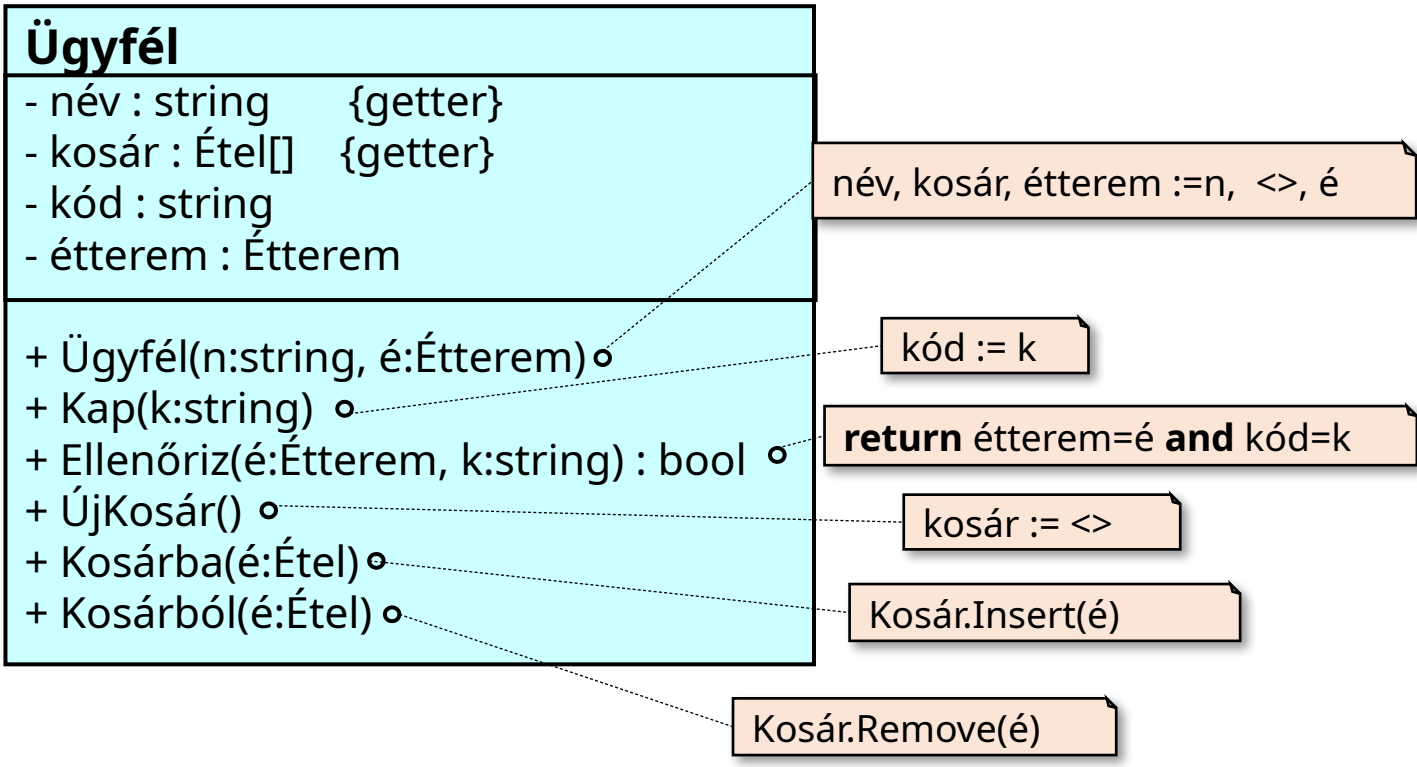
+ Étterem(é:Étel[])
+ Regisztrálás(név:string)
+ Bejelentkezés(ü:Ügyfél):string
+ Rendelés(ü:Ügyfél, kód:string, ételek:Étel[])

ügyfelek, étlap := <>, é

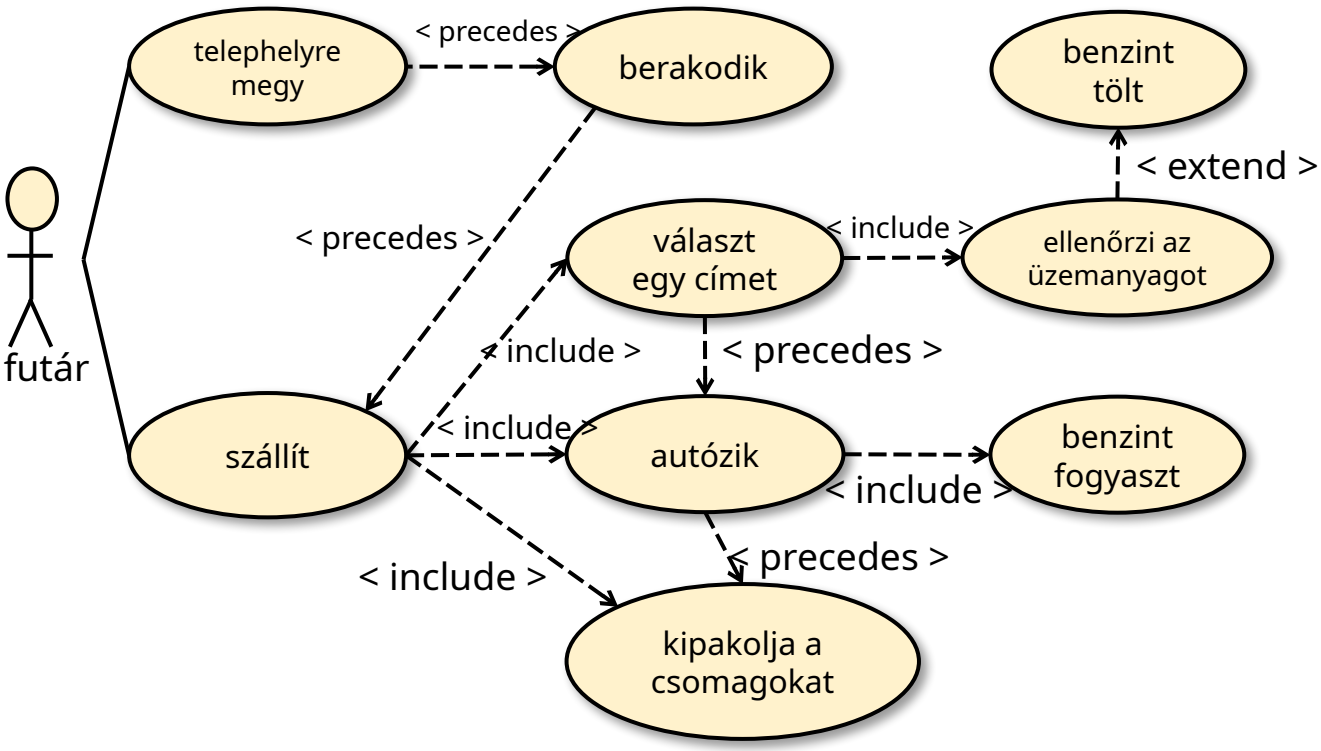
if ü **not in** ügyfelek
ügyfelek.Insert(**new** Ügyfél(név))
endif

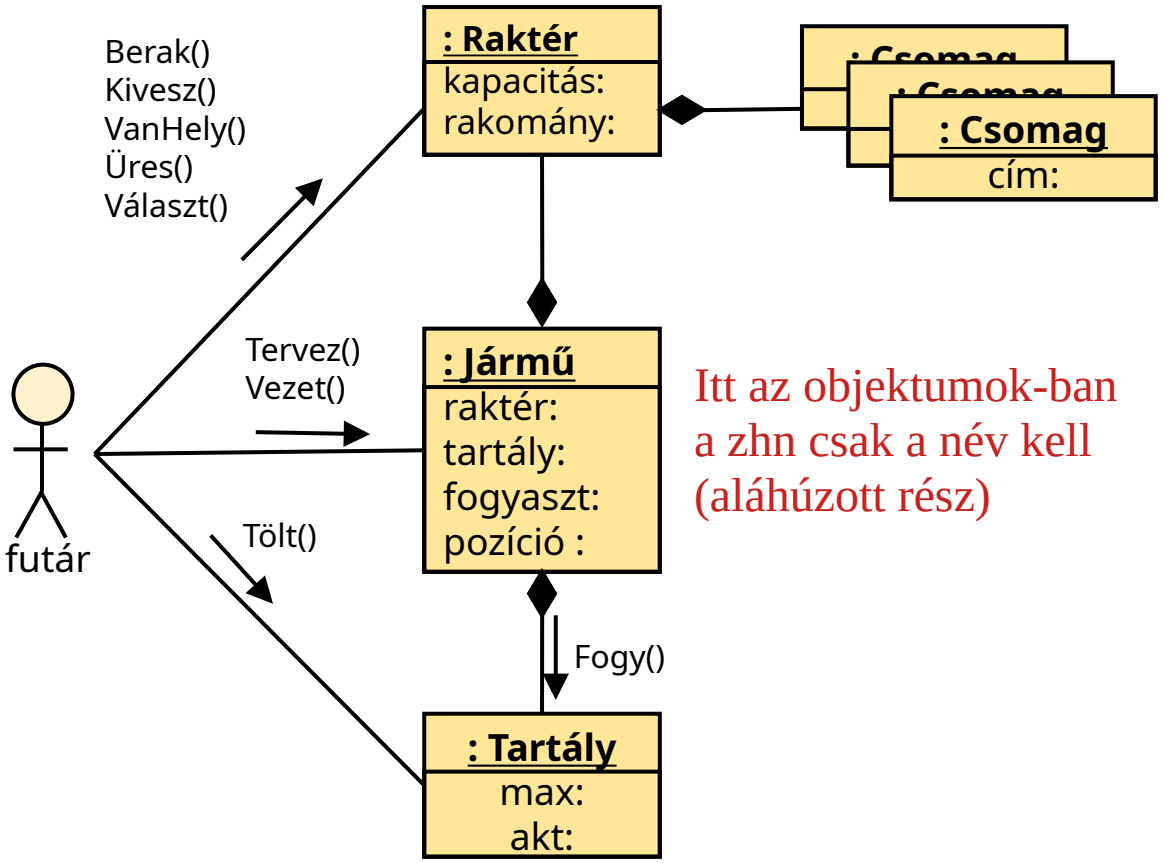
if ü **not in** ügyfelek **then error endif**
kód := /* kódot generál */
ü.Kap(kód)
return kód

if ü **not in** ügyfelek **then error endif**
if not ü.Ellenőriz(**this**, kód) **then error endif**
ÚjKosár()
forall é **in** ételek **loop** ü.Kosárba(é) **endloop**
// elküldés

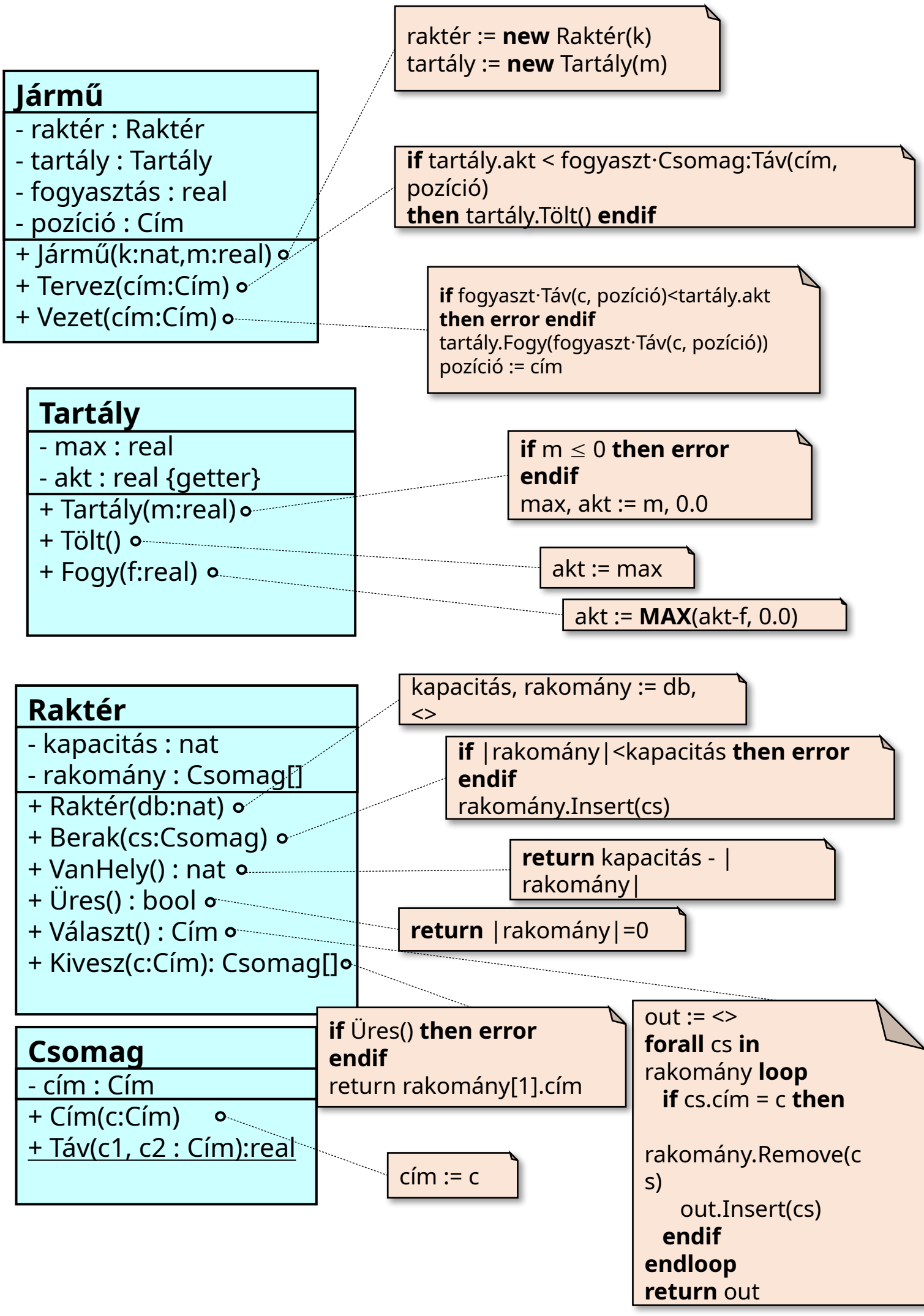

Ügyfél
- név : string {getter}
- kosár : Étel[] {getter}
- kód : string
- étterem : Étterem
+ Ügyfél(n:string, é:Étterem)
+ Kap(k:string)
+ Ellenőriz(é:Étterem, k:string) : bool
+ ÚjKosár()
+ Kosárba(é:Étel)
+ Kosárból(é:Étel)
név, kosár, étterem :=n, <>, é
kód := k
return étterem=é and kód=k
kosár := <>
Kosár.Insert(é)
Kosár.Remove(é)

Egy csomag kiszállító futár a telephelyről hordja szét a megrendelt csomagokat különböző benzinkutakhoz telepített PickPack pontokra. (Tehát minden kiszállítási címen tankolni is lehet). A csomagokra rá van írva a kiszállítás címe, így kiszámolható, hogy mekkora távolságot kell autóznia a futárnak az aktuális tartózkodási helyétől a kiválasztott címig (km-ben).
A járműnek van rakodótere és egy benzintartálya. A rakodótér megadott számú csomagot képes tárolni. A tartályba a maximális benzinszint figyelembe vételével tankolhatunk. Ismert a jármű fogyasztása (liter/km mértékegységben).
A futár a telephelyen bepakol annyi csomagot a járműve rakterébe, amennyit csak tud, majd a következők szerint jár el: kiválasztja az egyik csomagjának a címét (ha üres a rakodótér, akkor a telephelyének címét); ellenőrzi a benzinszintet, hogy elegendő üzemanyaga van a következő címre autózáshoz (ha nem, akkor tankol); elautózik a kiválasztott címre (emiatt csökken az autóban a benzinszint); majd kipakolja az adott címre küldött csomagokat.

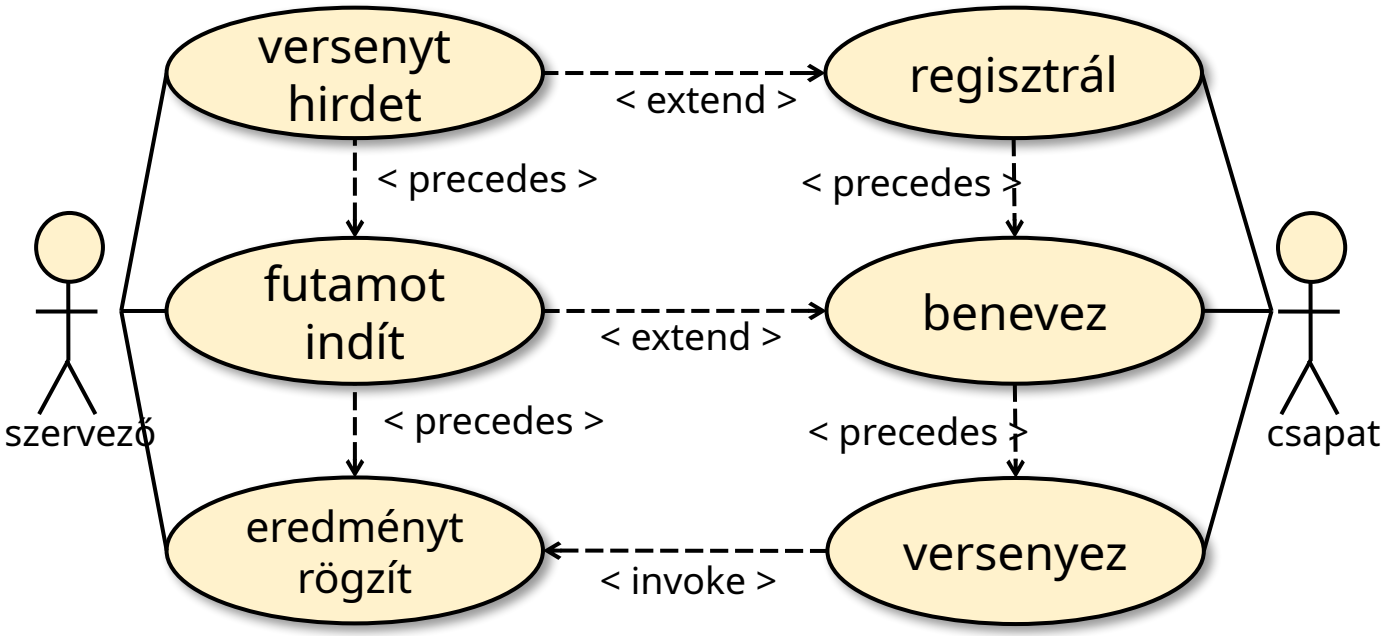


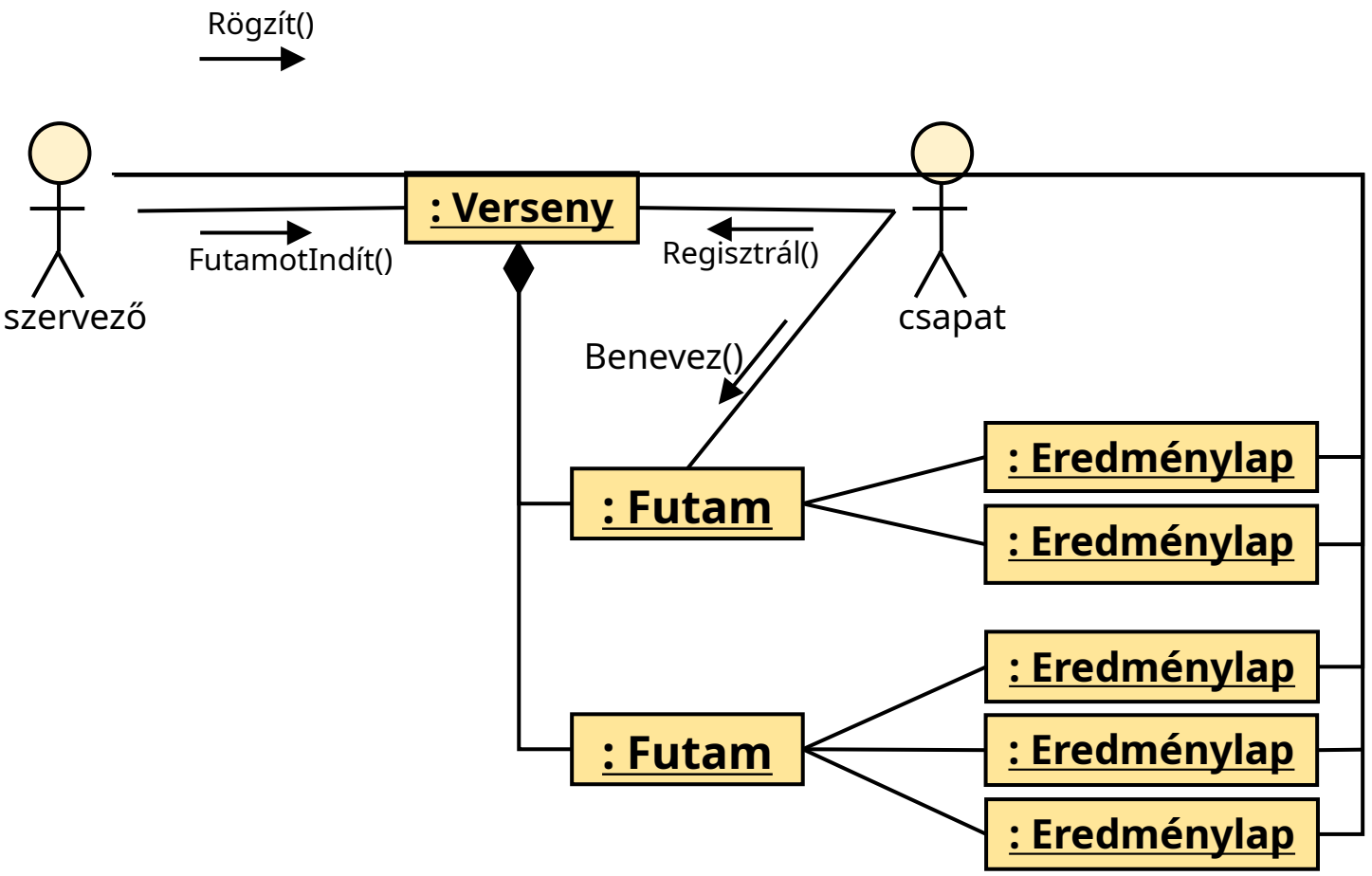


Itt az objektumok-ban a zhn csak a név kell (aláhúzott rész)


Jármű
- raktér : Raktér
- tartály : Tartály
- fogyasztás : real
- pozíció : Cím
+ Jármű(k:nat,m:real)
+ Tervez(cím:Cím)
+ Vezet(cím:Cím)
raktér := new Raktér(k)
tartály := new Tartály(m)
if tartály.akt < fogyaszt·Csomag:Táv(cím, pozíció)
then tartály.Tölt() endif
if fogyaszt·Táv(c, pozíció)<tartály.akt
then error endif
tartály.Fogy(fogyaszt·Táv(c, pozíció))
pozíció := cím
Tartály
- max : real
- akt : real {getter}
+ Tartály(m:real)
+ Tölt()
+ Fogy(f:real)
if m ≤ 0 then error
endif
max, akt := m, 0.0
akt := max
akt := MAX(akt-f, 0.0)
Raktér
- kapacitás : nat
- rakomány : Csomag[]
+ Raktér(db:nat)
+ Berak(cs:Csomag)
+ VanHely() : nat
+ Üres() : bool
+ Választ() : Cím
+ Kivesz(c:Cím): Csomag[]
kapacitás, rakomány := db, <>
if |rakomány|<kapacitás then error
endif
rakomány.Insert(cs)
return kapacitás - |rakomány|
return |rakomány|=0
if Üres() then error
endif
return rakomány[1].cím
out := <>
forall cs in rakomány loop
if cs.cím = c then
rakomány.Remove(cs)
out.Insert(cs)
endif
endloop
return out
Csomag
- cím : Cím
+ Cím(c:Cím)
+ Táv(c1, c2 : Cím):real
cím := c

Egy országos rally autóversenyre történő regisztrációkor a csapatok egyedi azonosítót kapnak, amellyel az egyes futamokra benevezhetnek. A futamokon több kategóriában versenyezhetnek a regisztrált csapatok. Egy csapat egy futamra kategóriánként egy-egy versenyzőt nevezhet be.
A futamok megadott időben indulnak. A futamra benevezett csapatokról egy ún. eredménylap készül, amely tartalmazza a csapat által elindított versenyzők adatait (versenyző neve, kategóriája), valamint az elért helyezést, amelyet a versenyzés befejezésekor rögzítenek. Egy csapatnak egy futamon elért eredménye az egyes kategóriákban elért helyezéseitől függ. (Ennek kiszámolásával most nem foglalkozunk.)





Verseny

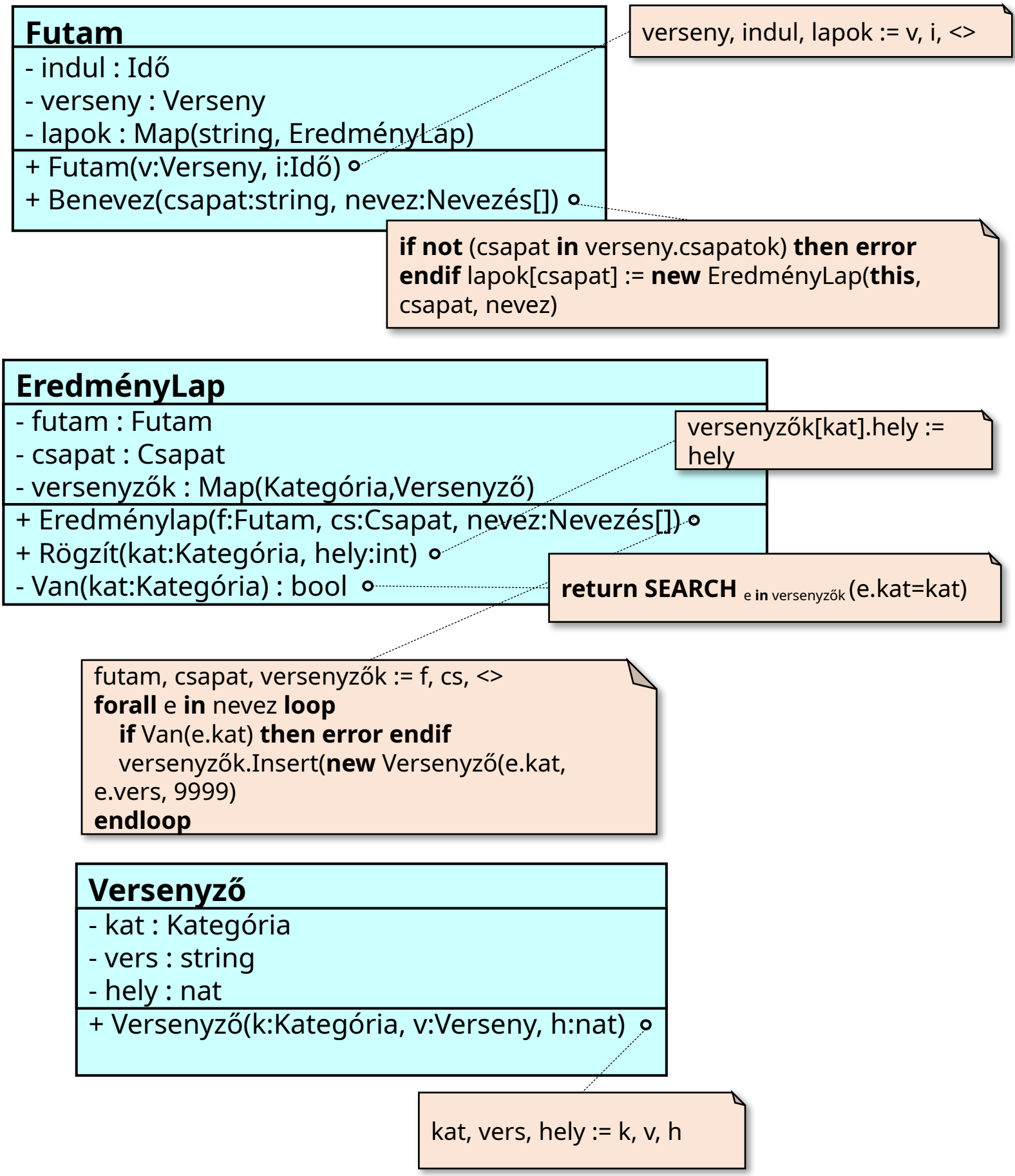
- dátum : Dátum
- helyszín : string
- csapatok : string[] {getter}
- futamok : Futam[]

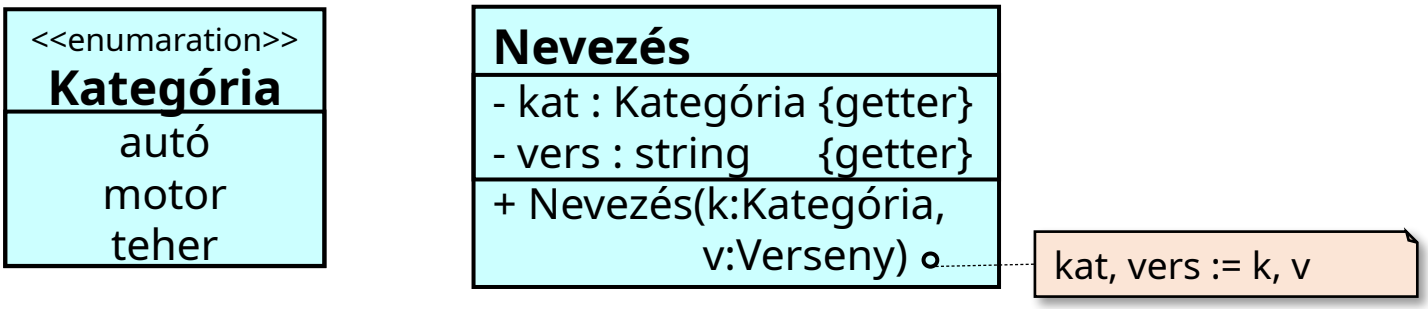
+ Verseny(d:Dátum, h:string)
+ Regisztrál(cs:string)
+ FutamotIndít(i:Idő)

dátum, helyszín, csapatok,
futamok :=
d, h, <>, <>

if cs **in** csapatok **then error**
endif csapatok.Insert(cs)

futamok.Insert(**new**
Futam(**this**, i))


Futam
- indul : Idő
- verseny : Verseny
- lapok : Map(string, EredményLap)
+ Futam(v:Verseny, i:Idő)
+ Benevez(csapat:string, nevez:Nevezés[])
verseny, indul, lapok := v, i, <>
if not (csapat in verseny.csapatok) then error endif lapok[csapat] := new EredményLap(this, csapat, nevez)
EredményLap
- futam : Futam
- csapat : Csapat
- versenyzők : Map(Kategória,Versenyző)
+ Eredménylap(f:Futam, cs:Csapat, nevez:Nevezés[])
+ Rögzít(kat:Kategória, hely:int)
- Van(kat:Kategória) : bool
versenyzők[kat].hely := hely
return SEARCH e in versenyzők (e.kat=kat)
futam, csapat, versenyzők := f, cs, <>
forall e in nevez loop
if Van(e.kat) then error endif
versenyzők.Insert(new Versenyző(e.kat, e.vers, 9999)
endloop
Versenyző
- kat : Kategória
- vers : string
- hely : nat
+ Versenyző(k:Kategória, v:Verseny, h:nat)
kat, vers, hely := k, v, h


<<enumaration>>
Kategória
autó
motor
teher
Nevezés
- kat : Kategória {getter}
- vers : string {getter}
+ Nevezés(k:Kategória,
v:Verseny)
kat, vers := k, v