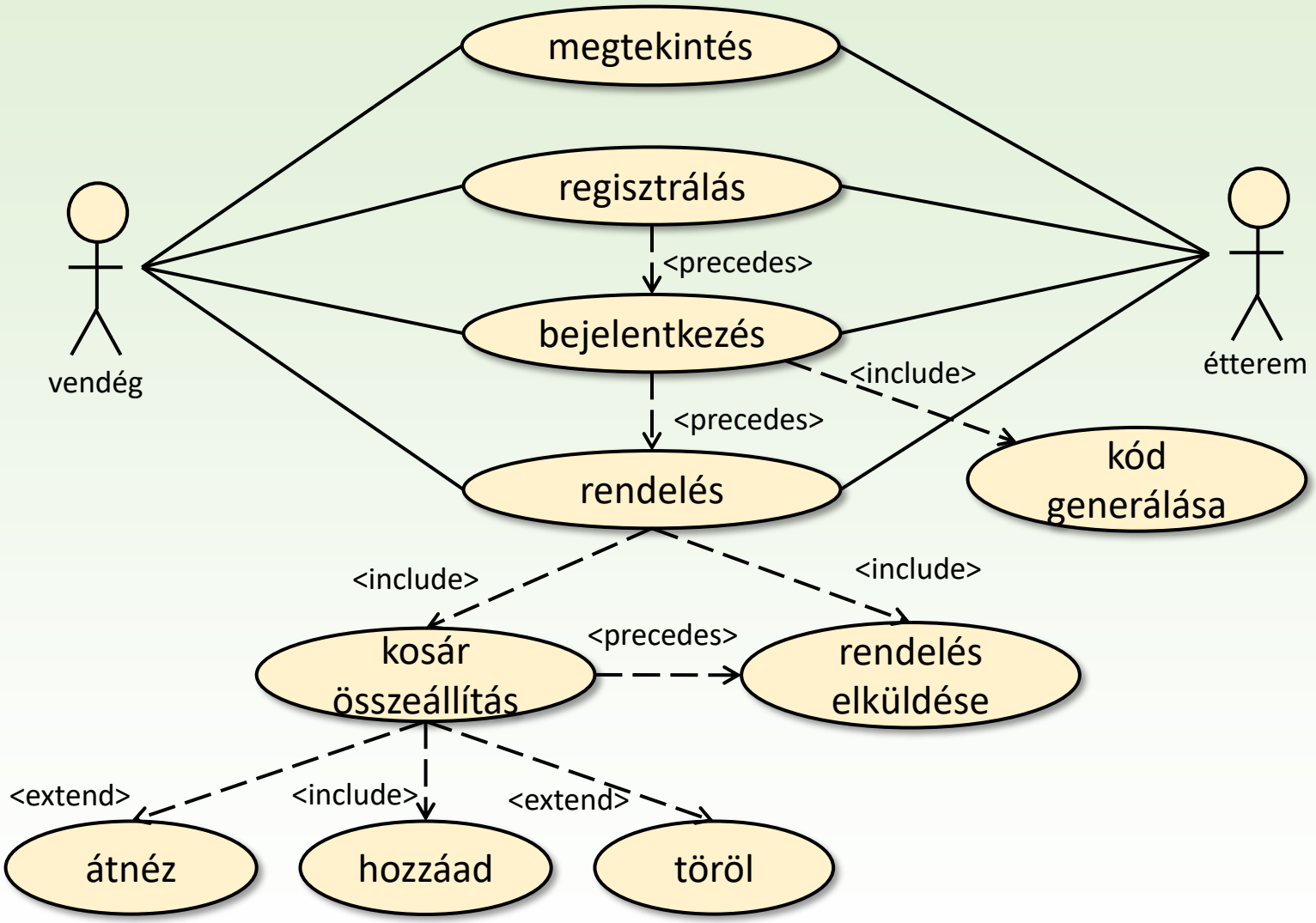
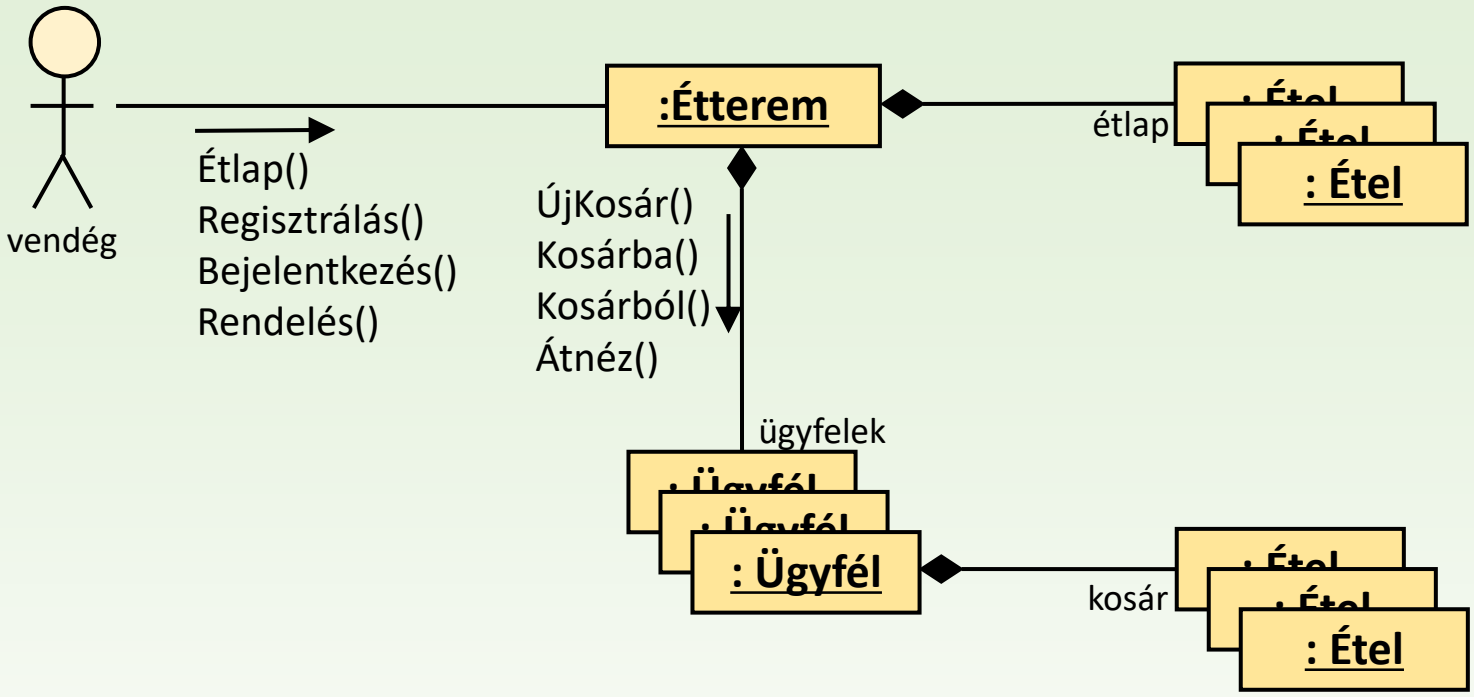
Ételrendelés

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.


megtekintés
regisztrálás
<precedes>
bejelentkezés
<include>
<precedes>
rendelés
kód generálása
vendég
étterem
<include>
<include>
kosár összeállítás
<precedes>
rendelés elküldése
<extend>
<include>
<extend>
átnéz
hozzáad
töröl

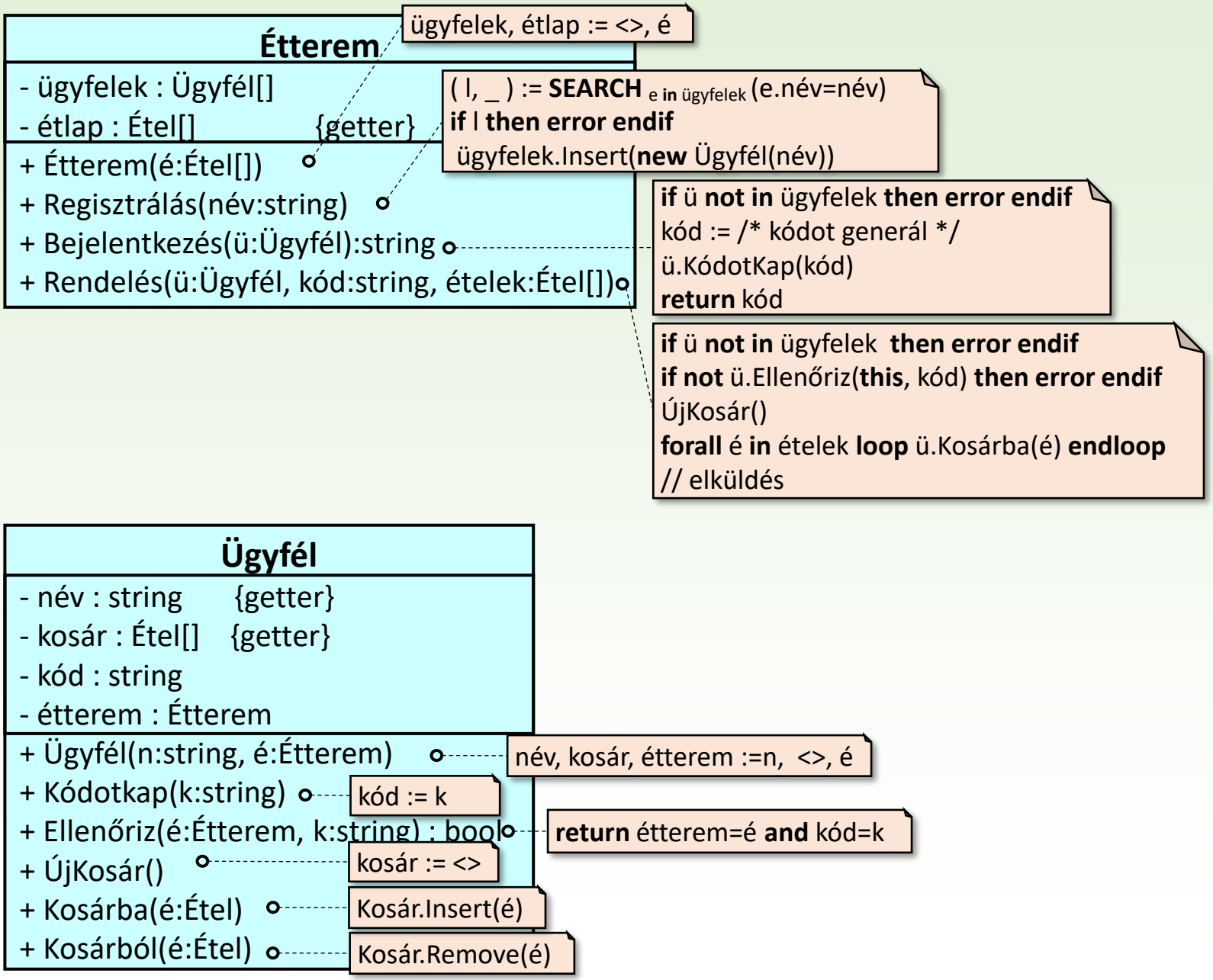


Étterem
- ügyfelek : Ügyfél[] - étlap : Étel[] {getter}
+ Regisztrálás(név:string) + Bejelentkezés(ü:Ügyfél):string + Rendelés(ü:Ügyfél, kód:string, ételek:Étel[])

Étlap()

Ügyfél
- kosár : Étel[] {getter} - kód : string
+ ÚjKosár() + Kosárba(é:Étel) + Kosárból(é:Étel)

Átnéz()

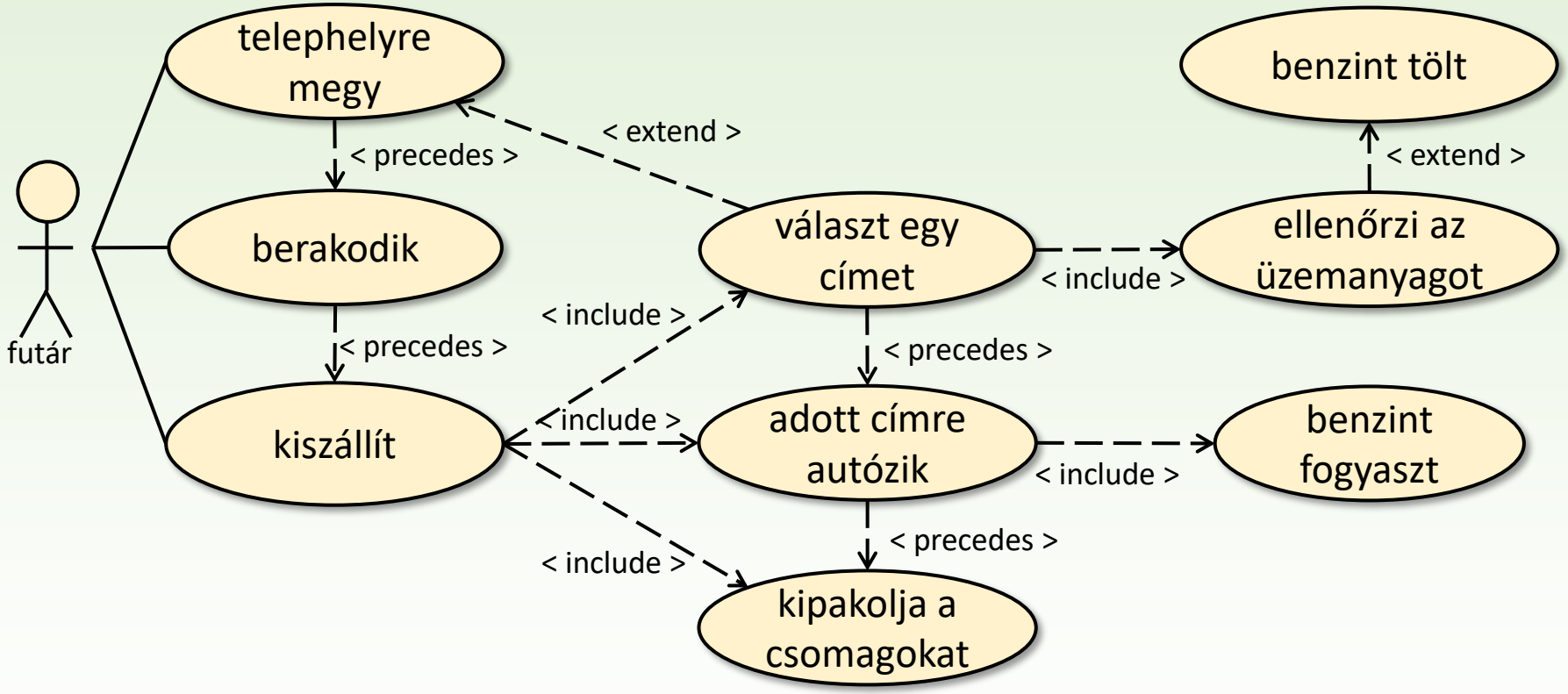

Étterem
- ügyfelek : Ügyfél[]
- étlap : Étel[] {getter}
+ Étterem(é:Étel[])
+ Regisztrálás(név:string)
+ Bejelentkezés(ü:Ügyfél):string
+ Rendelés(ü:Ügyfél, kód:string, ételek:Étel[])
ügyfelek, étlap := <>, é
(l, _) := SEARCH e in ügyfelek (e.név=név)
if l then error endif
ügyfelek.Insert(new Ügyfél(név))
if ü not in ügyfelek then error endif
kód := /* kódot generál */
ü.KódotKap(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)
+ Kódotkap(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)

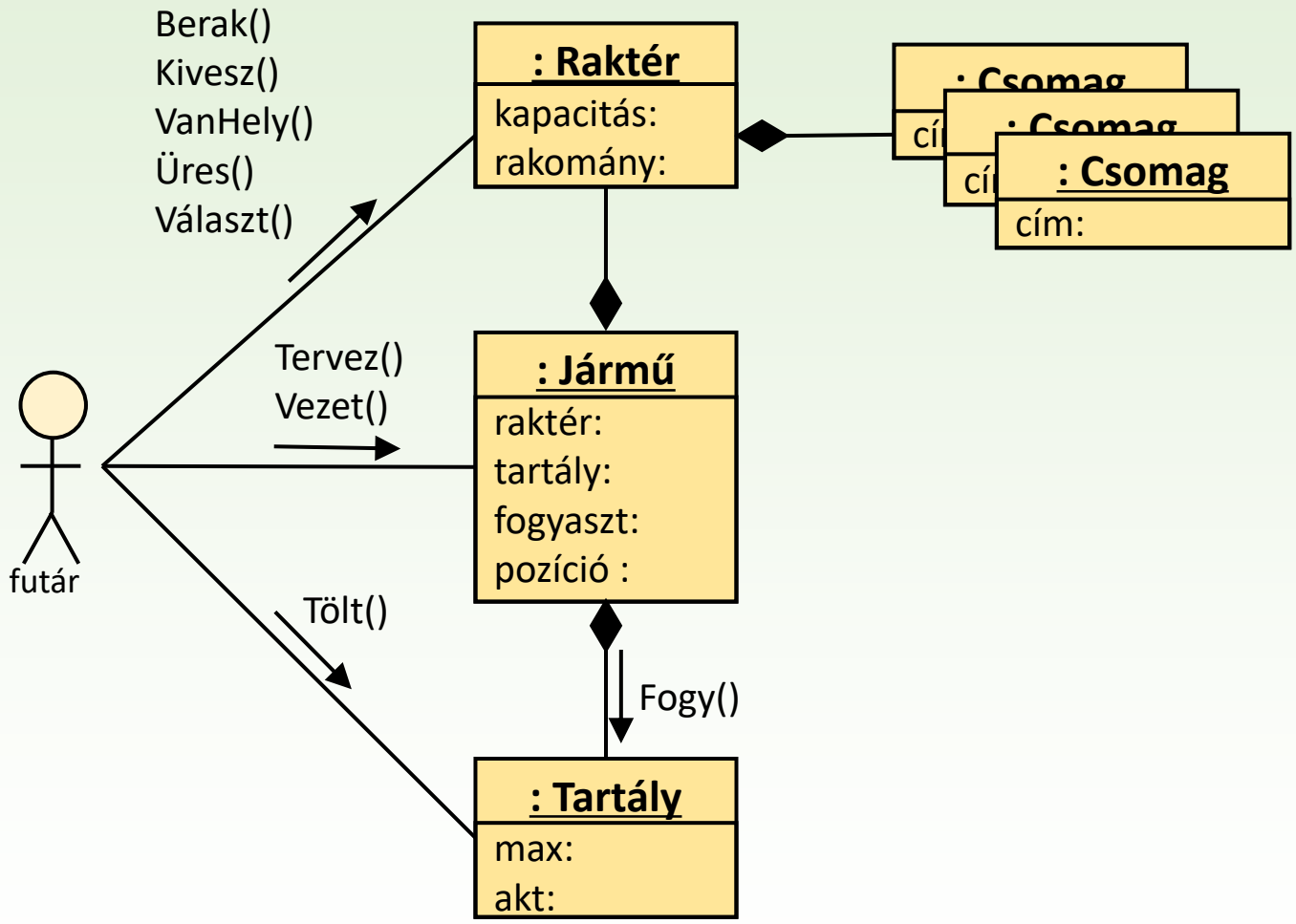
Csomagszállítás

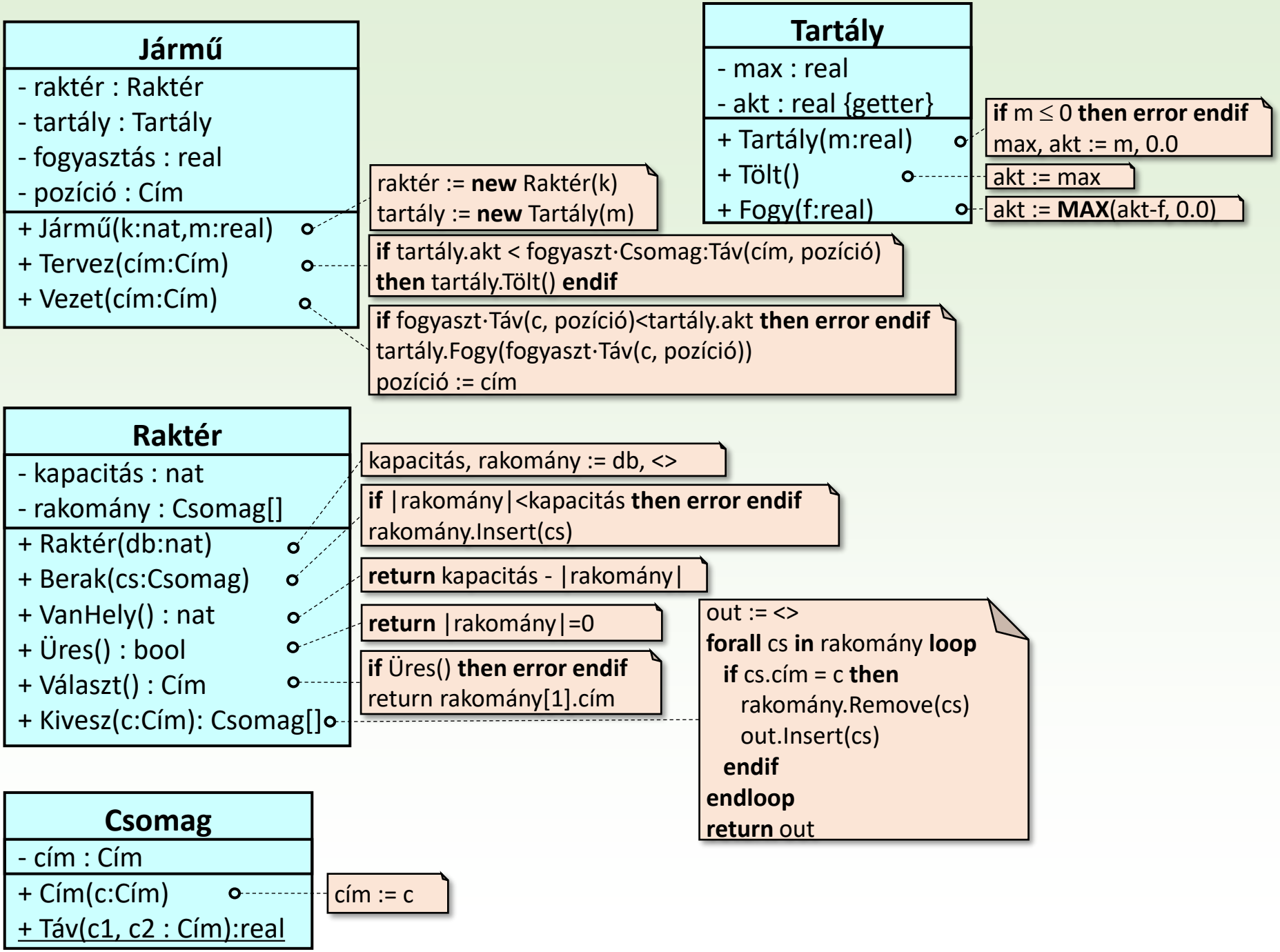
Egy csomag kiszállító futár a telephelyről hordja szét a megrendelt csomagokat különböző benzinkutakhoz telepített PickPack pontokra. (Tehát minden kiszállítási címen tankolni is lehet). A csomagokra rá van írva a kiszállítás címe, így kiszámolható, hogy mekkora távolságot kell autóznia a futárnak az aktuális tartózkodási helyétől a kiválasztott címig (km-ben).

A járműnek van rakodótere és egy benzintartálya. A rakodótér megadott számú csomagot képes tárolni. A benzintartály maximális térfogata is adott. Ismert a jármű fogyasztása (liter/km).

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.


futár
telephelyre megy
< precedes >
berakodik
< precedes >
kiszállít
< extend >
< include >
< include >
< include >
választ egy címet
adott címre autózik
kipakolja a csomagokat
< precedes >
< precedes >
< precedes >
< include >
< include >
benzint tölt
< extend >
ellenőrzi az üzemanyagot
benzint fogyaszt

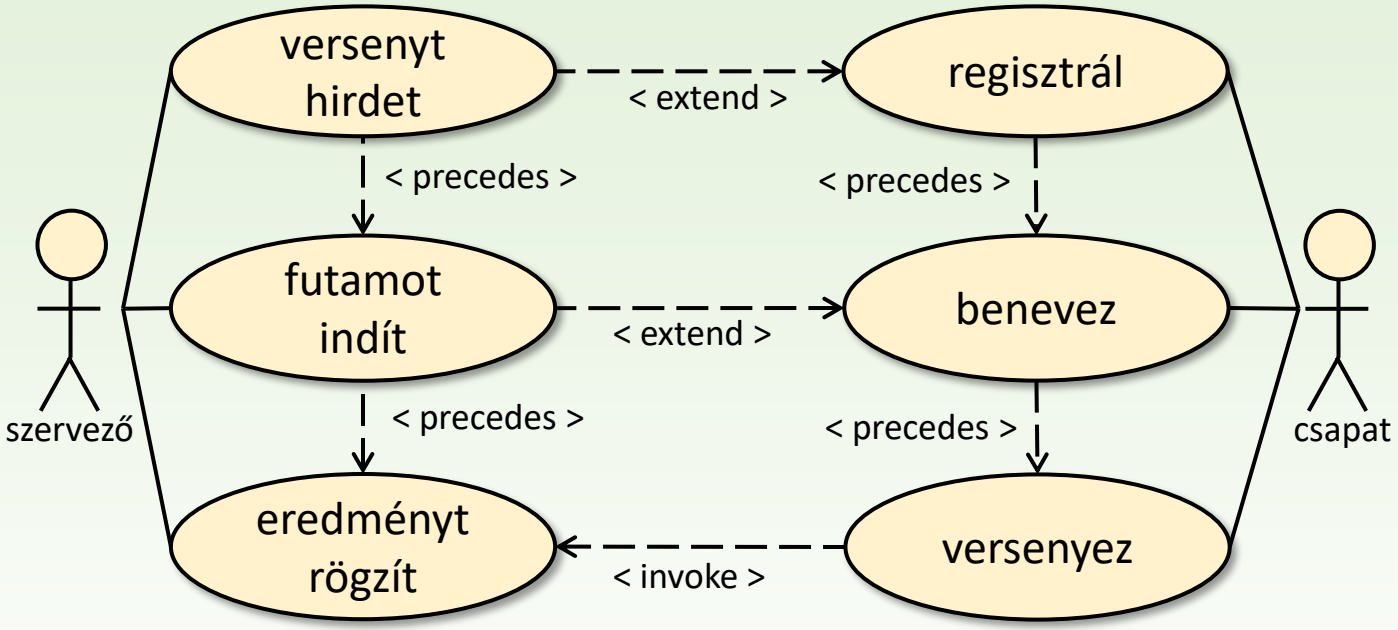

Berak()
Kivesz()
VanHely()
Üres()
Választ()
: Raktér
kapacitás:
rakomány:
: Csomag
cím:
futár
Tervez()
Vezet()
: Jármű
raktér:
tartály:
fogyaszt:
pozíció :
Tölt()
Fogy()
: Tartály
max:
akt:

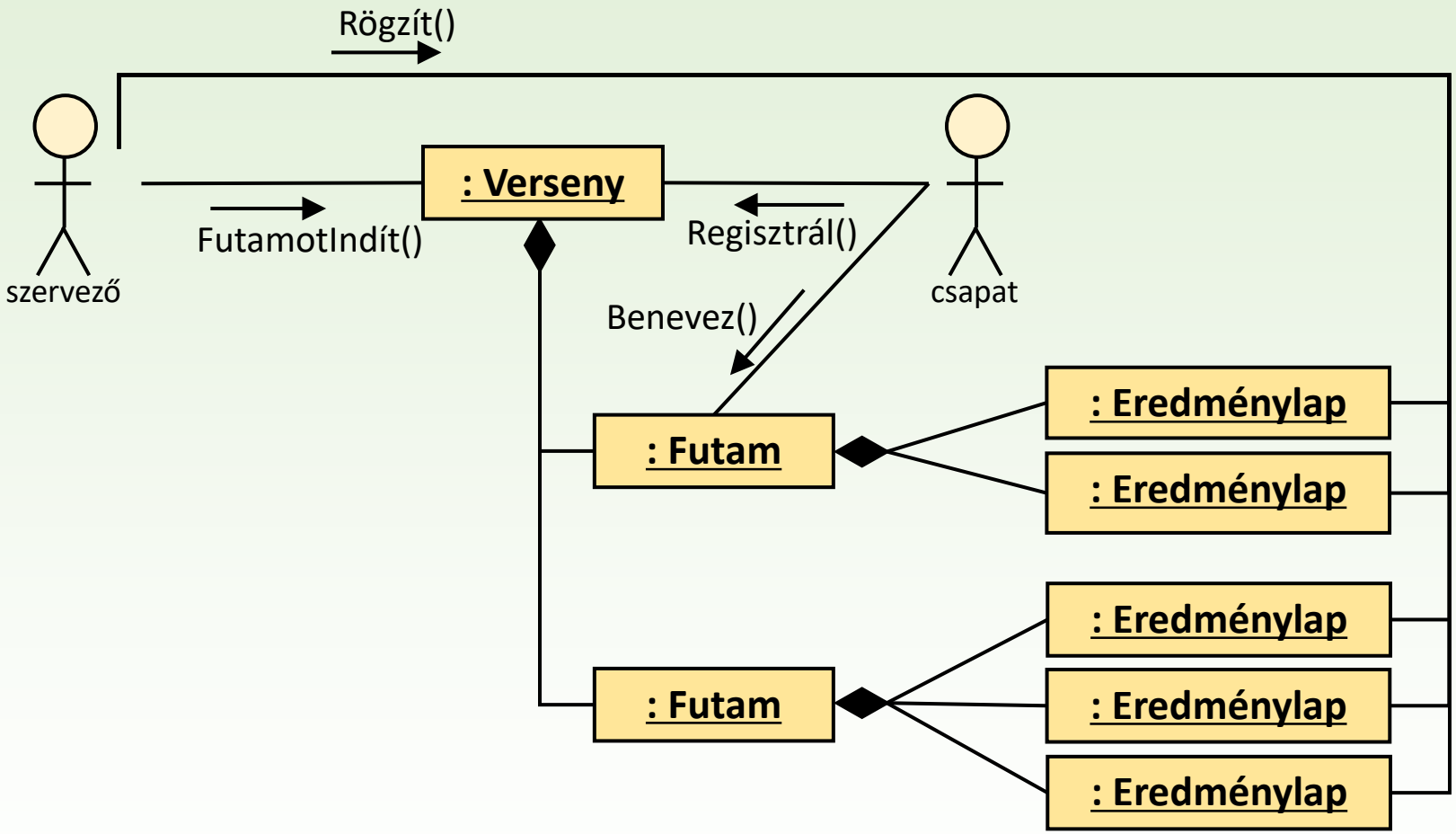

Jármű
- raktér : Raktér
- tartály : Tartály
- fogyasztás : real
- pozíció : Cím
+ Jármű(k:nat,m:real)
+ Tervez(cím:Cím)
+ Vezet(cím:Cím)
raktér := new Raktér(k)
tartály := new Tartály(m)
if tartály.akt < fogyaszt·Csomag:Táv(cím, pozíció)
then tartály.Tölt() endif
if fogyaszt·Táv(c, pozíció)<tartály.akt 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

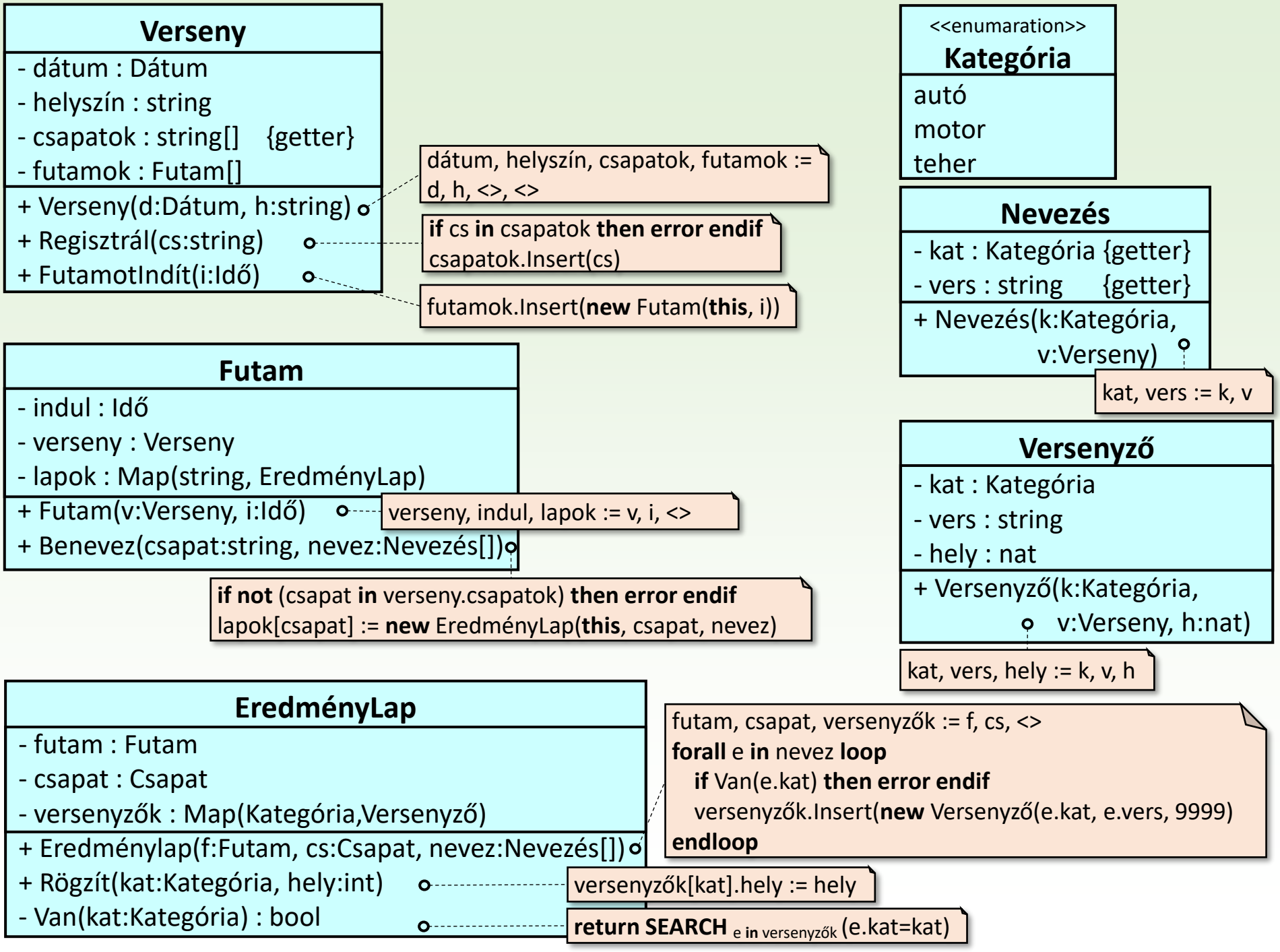
Rally

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.)

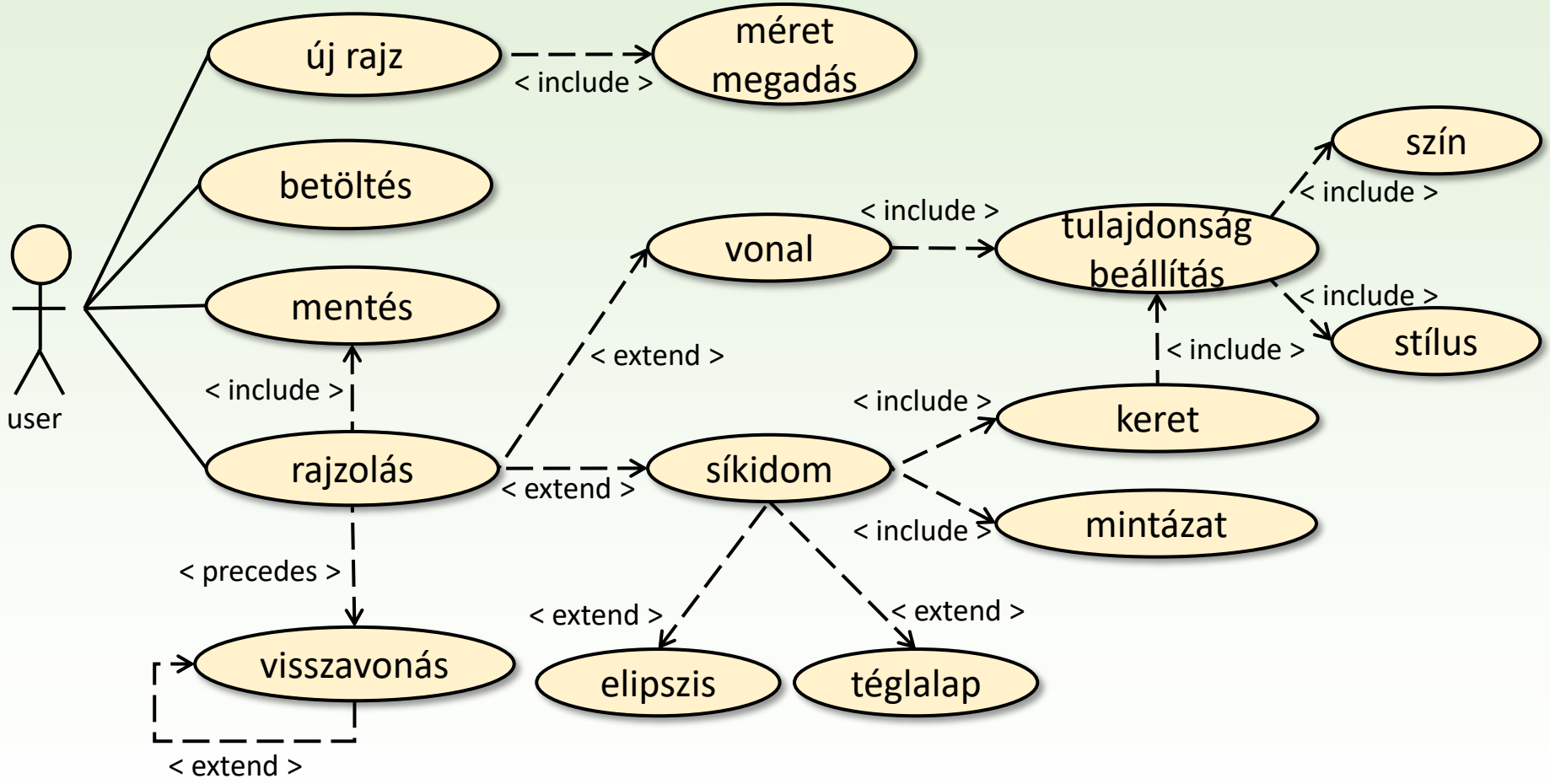

versenyt hirdet
regisztrál
< extend >
< precedes >
< precedes >
futamot indít
benevez
< extend >
szervező
csapat
< precedes >
< precedes >
eredményt rögzít
versenyez
< invoke >

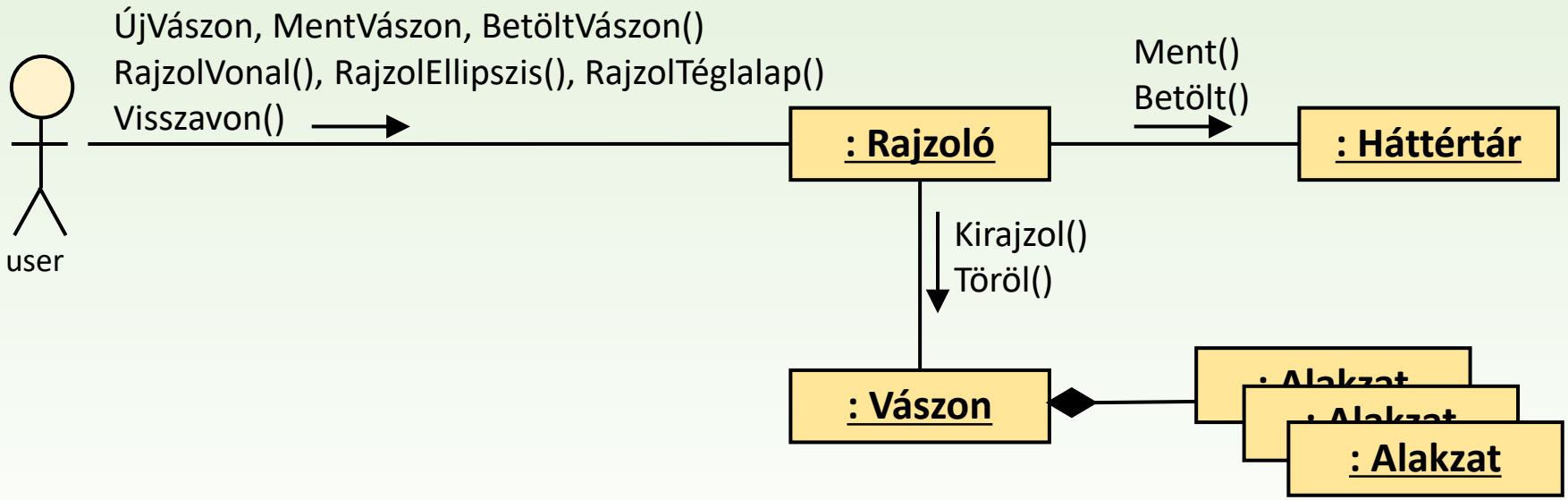

Rögzít()
szervező
: Verseny
FutamotIndít()
Regisztrál()
csapat
Benevez()
: Futam
: Eredménylap
: Eredménylap
: Futam
: Eredménylap
: Eredménylap
: Eredménylap

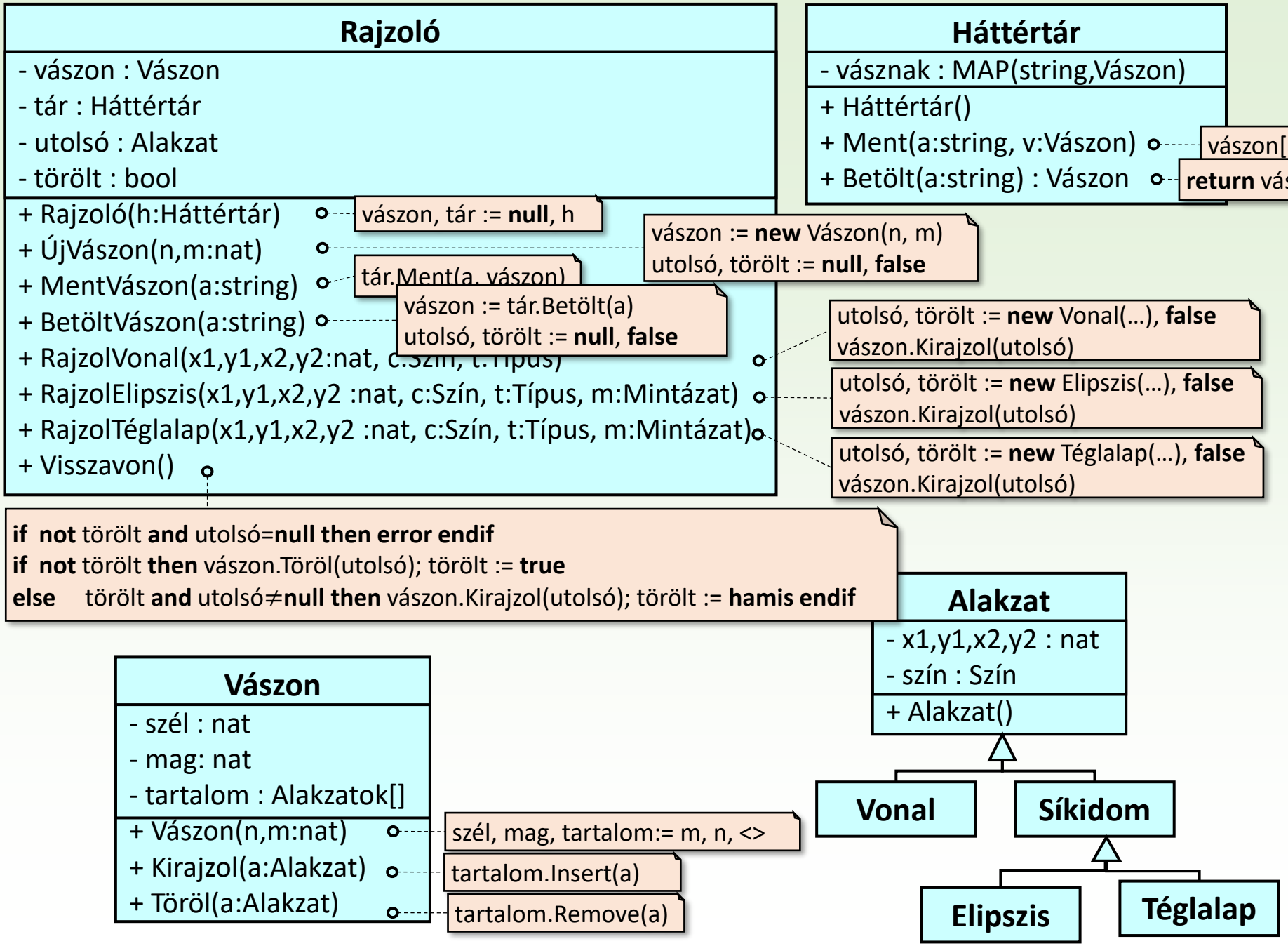

Verseny
- dátum : Dátum
- helyszín : string
- csapatok : string[] {getter}
- futamok : Futam[]
+ Verseny(d:Dátum, h:string)
+ Regisztrál(cs:string)
+ FutamotIndít(i:Idő)
dátum, helyszín, csapatok, futamok := d, h, <>, <>
if cs in csapatok then error endif
csapatok.Insert(cs)
futamok.Insert(new Futam(this, i))
<<enumaration>>
Kategória
autó
motor
teher
Nevezés
- kat : Kategória {getter}
- vers : string {getter}
+ Nevezés(k:Kategória, v:Verseny)
kat, vers := k, v
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)
Versenyző
- kat : Kategória
- vers : string
- hely : nat
+ Versenyző(k:Kategória, v:Verseny, h:nat)
kat, vers, hely := k, v, h
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
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
versenyzők[kat].hely := hely
return SEARCH e in versenyzők (e.kat=kat)

Rajzoló

Egy grafikus rajzolóprogramban lehetőségünk van egy új rajz készítésére, vagy meglévők betöltésére, illetve rajz mentésére. Új rajz létrehozásakor meg kell adnunk a rajzvászon méretét (szélesség, magasságát). Egy rajzra rárajzolhatunk vonalat, téglalapot, valamint ellipszist. A vonalaknak beállíthatjuk a színét és stílusát (szaggatottság, vastagság). Lehetőségünk van az utolsó művelet visszavonására, ha már történt rajzolás, illetve visszavont művelet újbóli alkalmazására. Minden művelet után történik egy automatikus biztonsági mentés, így ha az alkalmazás összeomlik, a program automatikusan az utolsó biztonsági mentést tölti be.


új rajz
méret megadás
< include >
betöltés
mentés
rajzolás
user
< include >
< precedes >
visszavonás
< extend >
< extend >
vonal
< extend >
síkidom
< include >
< include >
tulajdonság beállítás
< include >
szín
< include >
stílus
< include >
keret
< include >
mintázat
< extend >
elipszis
< extend >
téglalap


ÚjVászon, MentVászon, BetöltVászon()
RajzolVonal(), RajzolEllipszis(), RajzolTéglalap()
Visszavon()
user
: Rajzoló
Ment()
Betölt()
: Háttértár
Kirajzol()
Töröl()
: Vászon
: Alakzat
: Alakzat
: Alakzat


Rajzoló
- vászon : Vászon
- tár : Háttértár
- utolsó : Alakzat
- törölt : bool
+ Rajzoló(h:Háttértár)
+ ÚjVászon(n,m:nat)
+ MentVászon(a:string)
+ BetöltVászon(a:string)
+ RajzolVonal(x1,y1,x2,y2:nat, c:Szín, t:Típus)
+ RajzolElipszis(x1,y1,x2,y2 :nat, c:Szín, t:Típus, m:Mintázat)
+ RajzolTéglalap(x1,y1,x2,y2 :nat, c:Szín, t:Típus, m:Mintázat)
+ Visszavon()
vászon, tár := null, h
vászon := new Vászon(n, m)
utolsó, törölt := null, false
tár.Ment(a, vászon)
vászon := tár.Betölt(a)
utolsó, törölt := null, false
utolsó, törölt := new Vonal(…), false
vászon.Kirajzol(utolsó)
utolsó, törölt := new Elipszis(…), false
vászon.Kirajzol(utolsó)
utolsó, törölt := new Téglalap(…), false
vászon.Kirajzol(utolsó)
if not törölt and utolsó=null then error endif
if not törölt then vászon.Töröl(utolsó); törölt := true
else törölt and utolsó≠null then vászon.Kirajzol(utolsó); törölt := hamis endif
Háttértár
- vásznak : MAP(string,Vászon)
+ Háttértár()
+ Ment(a:string, v:Vászon)
+ Betölt(a:string) : Vászon
vászon[a]:=
return vászon
Alakzat
- x1,y1,x2,y2 : nat
- szín : Szín
+ Alakzat()
Vonal
Síkidom
Elipszis
Téglalap
Vászon
- szél : nat
- mag: nat
- tartalom : Alakzatok[]
+ Vászon(n,m:nat)
+ Kirajzol(a:Alakzat)
+ Töröl(a:Alakzat)
szél, mag, tartalom:= m, n, <>
tartalom.Insert(a)
tartalom.Remove(a)