

UML

Gregorics Tibor

`gt@inf.elte.hu`

`http://people.inf.elte.hu/gt/oep`

UML diagramjai

- ❑ Az UML (*Unified Modeling Language*) egy grafikus nyelv, amelyet az objektumelvű szoftverfejlesztési folyamat során az egyes fázisok (az elemzés, a tervezés, a megvalósítás, és a tesztelés) dokumentálására használhatunk.
- ❑ A nyelv különböző diagramokat definiál, amelyek a készülő vagy már elkészült szoftvernek, illetve az azt körül ölelő rendszernek egy-egy részét különféle nézőpontokból képesek bemutatni, és így együttesen egy elosztott dokumentációt alkotnak.
- ❑ A diagramokat aszerint szokás csoportosítani, hogy a rendszer szerkezetét vagy viselkedését mutatják-e meg:
 - **szerkezeti nézet**: pl. objektum-, osztály-, komponens-, kihelyezési-, csomag diagram
 - **viselkedési nézet**: pl. használati eset-, kommunikációs-, szekvencia-, állapotgép-, tevékenység diagram

A kurzuson használt diagramok

- **használati eset diagram** (elemzés)
 - ez az egyetlen nem objektumelvű nézet, amely a feladat fő funkcióit írja le
- **objektum diagram** (elemzés)
 - egy adott időpillanatban létező objektumokat és azok kapcsolatait jellemzi
- **kommunikációs diagram** (elemzés)
 - objektumok közötti üzenetváltásokat (szignál küldés, metódus-hívás) mutatja
- **osztály diagram** (elemzés, tervezés, megvalósítás)
 - az objektumok és a közöttük létrejövő kapcsolatok általános leírására szolgál
- **szekvencia diagram** (elemzés, tesztelés)
 - üzenetváltások szekvenciájának egy lehetséges forgatókönyvét (szkenárióját) mutatja be
- **állapotgép diagram** (tervezés)
 - egy objektumnak a neki küldött üzenetek (pl. metódusainak hívása) hatására bekövetkező állapot-változásait, azaz a működést szemlélteti.

1.rész

Használati eset diagram

Gregorics Tibor

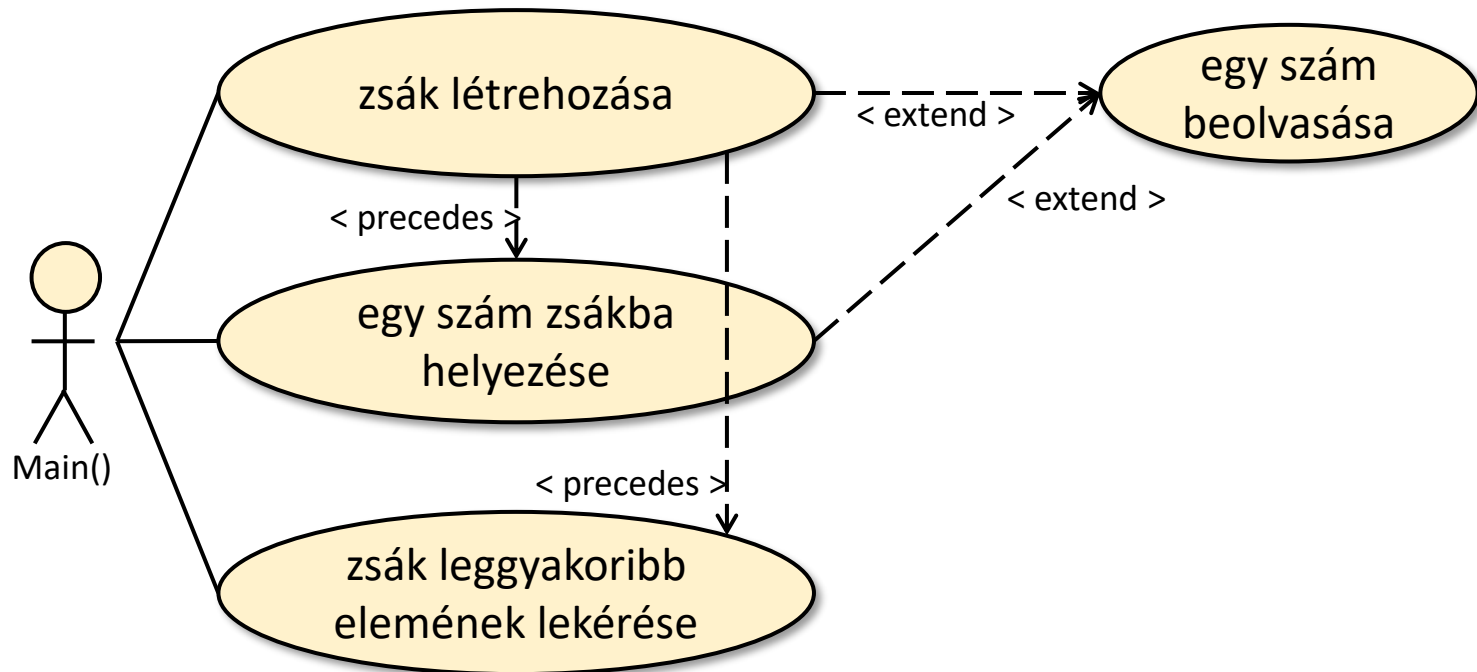
gt@inf.elte.hu

<http://people.inf.elte.hu/gt/oep>

Használati eset diagram

■ Megmutatja, hogy a tervezett rendszer

- milyen funkciókat lát el, milyen szolgáltatásokat biztosít
- kik számára nyújtja a szolgáltatásokat
- milyen követelményeket vár el a környezetétől
- milyen (tartalmazási, rákövetkezési) kapcsolatoknak kell fenn állni az egyes funkciók között



Használati eset diagram elemei

- ❑ **Használati esetek:** azon tevékenységek (funkciók, szolgáltatások), amelyek a rendszert kívülről szemlélő (azaz a felhasználó vagy más külső szereplő) szempontjából azonosíthatóak, és kezdeményezhetőek.
- ❑ **Kapcsolatok** a használati esetek között, amelyek a használati esetek
 - részekre bontását, illetve plusz funkciókkal történő kiegészítését ábrázolják
 - sorrendjét határozzák meg
 - általános használati esetből történő származtatására mutatnak rá
- ❑ **Aktorok:** a funkciókat használó felhasználók vagy más külső szereplők, akik vagy amik a tervezett rendszer működését vezérlik. Jelölhető az, hogy egy aktor az aktorok mely fajtájához tartozik (specializálás), illetve az, hogy egyfajta aktorból hány darab lehet (multiplicitás).

Használati esetek kapcsolatai

□ Használati eset **részei**

- **include**: rámutat egy használati esetnek egy olyan jól elkülöníthető, önállóan is kezdeményezhető részére, amely nélkül a tartalmazó tevékenység már nem lenne teljes (hanem absztrakt)
- **extend**: egy önmagában is teljes (tehát nem absztrakt) használati esetet opcionálisan kiegészítő másik esetre mutat

□ Használati esetek **rákövetkezési sorrendje**

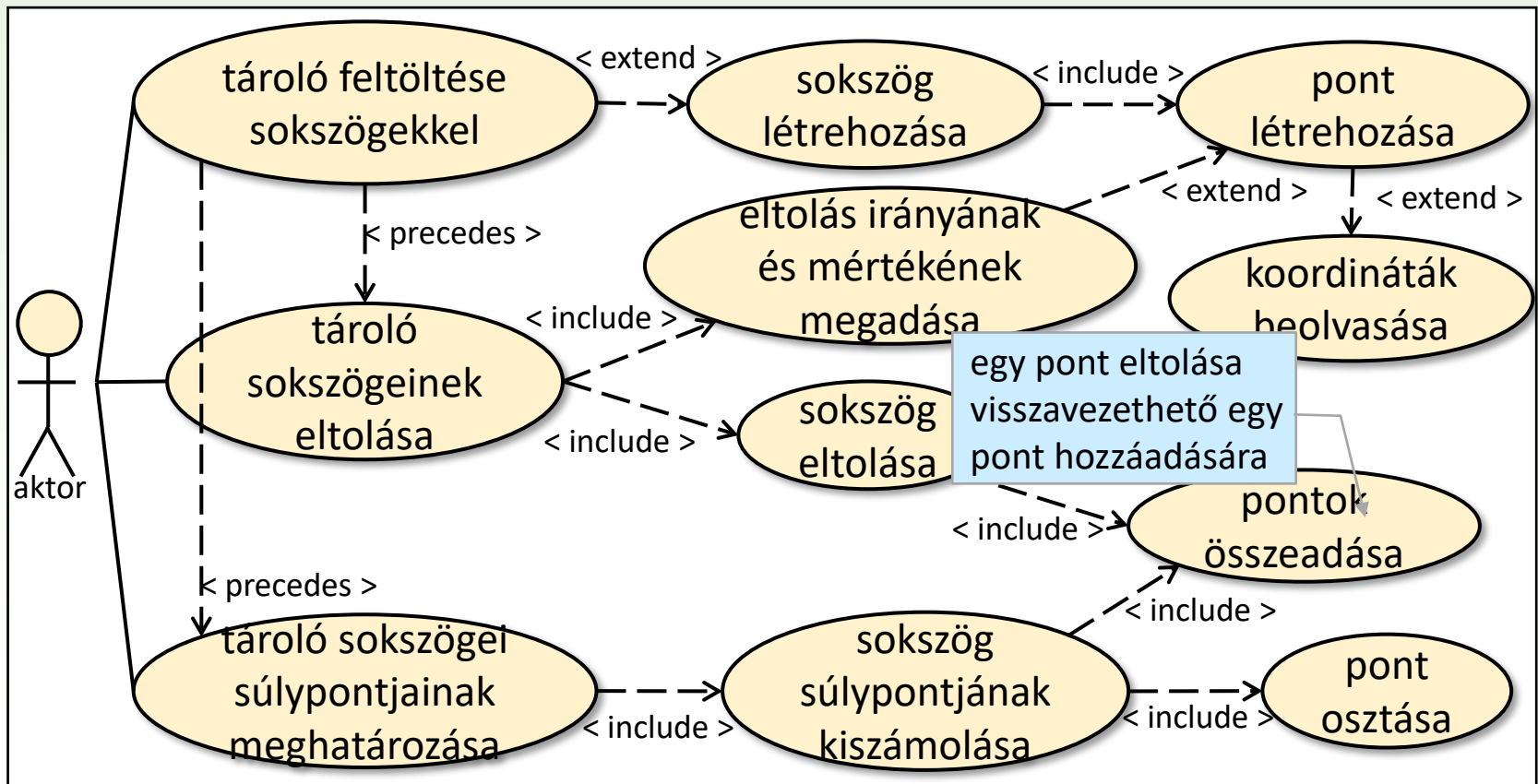
- **precedes**: aktor által közvetlenül kezdeményezhető tevékenységek közötti sorrendet jelöli ki.
- **invokes**: rámutat arra, hogy egy tevékenység végrehajtását mely másik tevékenységnek kell követnie az aktor akaratától függetlenül

□ Használati esetek közötti **származtatás**

- amikor egy általánosan megfogalmazott használati esetnek több eltérő konkrét változata is beazonosítható

Példa

Töltsünk fel egy tárolót különféle sokszögekkel, és mindegyiket toljuk el ugyanazon irányba és mértékkel, majd számoljuk ki az így nyert sokszögek súlypontjait. A csúcspontok és súlypontok koordinátái, sőt az eltolást leíró helyvektor végpontjának koordinátái is legyenek valós számok.



Felhasználói történet (user story)

- ❑ A használati eset diagram önmagában nem ad elégséges képet a megvalósítandó rendszerről, hiszen az egyes tevékenységekről a nevükön kívül nem árul el túl sokat.
- ❑ A felhasználói esetek **felhasználói csoportonként** („AS a ...”) történő táblázatos („user story”) leírásában viszont részletesen ki lehet fejteni a felhasználói tevékenységeket:
 - mi a tevékenység **neve**,
 - milyen **előfeltétel** meglétét feltételezi (GIVEN)
 - milyen **esemény** hatására következik be (WHEN)
 - mi a **hatása** a végrehajtásának, milyen eredményt ad (THEN).

AS a ...		
eset		leírás
tevékenység neve	GIVEN	tevékenység kiváltásakor feltételezett alaphelyzet
	WHEN	tevékenység kiváltása
	THEN	tevékenység hatása

Példa

eset		Leírás
tároló feltöltése sokszögekkel	GIVEN	sokszögeket tárolni képes tároló (pl. sorozat)
	WHEN	egy sokszögnek a tárolóban történő elhelyezésének kérése
	THEN	a tároló sokszögeket tartalmaz
sokszög létrehozása	GIVEN	alkalmazás egy sokszög létrehozására vár
	WHEN	csúcsok megadása (darabszámuk és a csúcsok síkbeli pontjai)
	THEN	létrejön egy sokszög (objektum)
pont létrehozása	GIVEN	síkbeli pont koordinátái
	WHEN	síkbeli pont létrehozása
	THEN	létrejön egy síkbeli pont (objektum)
tároló sokszögeinek eltolása	GIVEN	tároló a sokszögekkel
	WHEN	sokszögek eltolása az eltolás helyvektora végpontjának megadásával
	THEN	A tároló sokszögeit egymás után eltolja a megadott vektorral
eltolás irányának és mértékének megadása	GIVEN	síkbeli pont koordinátái
	WHEN	az eltolás helyvektora létrehozása
	THEN	egy síkbeli pont létrehozása

Példa

eset		leírás
tároló sokszögei súlypontjainak meghatározása	GIVEN	tároló sokszögekkel
	WHEN	súlypontok meghatározásának kérése
	THEN	a tároló sokszögeinek egymás után kiszámolja és kiírja a súlypontját
sokszög súlypontjának kiszámolása	GIVEN	sokszög
	WHEN	súlypont számítás igénye
	THEN	sokszög csúcspontjai alapján létrejön egy síkbeli pont (objektum)
sokszög eltolása	GIVEN	sokszög és az eltolás helyvektorának végpontja
	WHEN	eltolás igénye
	THEN	módosulnak a sokszög csúcspontjai
pontok összeadása	GIVEN	két síkbeli pont
	WHEN	összeadás igénye
	THEN	az egyik pont koordinátái két másik pont koordinátáinak összege
pont osztása	GIVEN	síkbeli pont és egy egész szám
	WHEN	osztás igénye
	THEN	a pont koordinátái az eredeti koordináták adott számmal vett hányadosai

2.rész

Objektum-, kommunikációs diagram

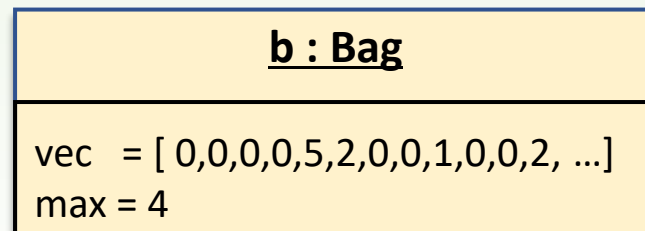
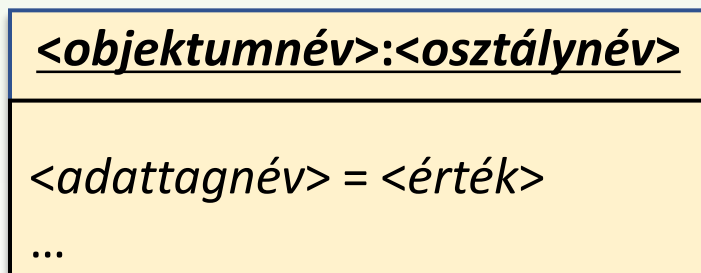
Gregorics Tibor

gt@inf.elte.hu

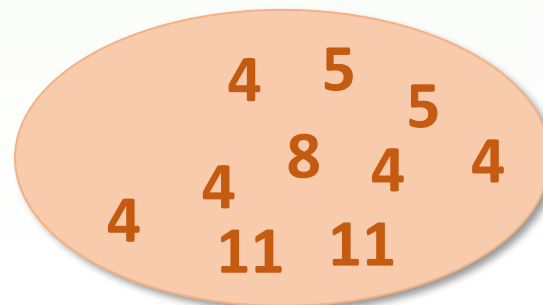
<http://people.inf.elte.hu/gt/oep>

Objektum diagram

- ❑ Az objektum diagram a tervezett rendszer működésének egy adott időpillanatában létező objektumok (a **populáció**) leírására szolgál.
- ❑ Egy objektumot meghatároz
 - az **osztálya**, amely az ugyanolyan adattagokkal és metódusokkal rendelkező objektumokat jellemzi.
 - a **neve** (amit nem kötelező megadni),
 - az **állapota** (amit az adattagjainak értékei jelölnek ki).



Az objektum diagram nem tartalmazza az objektumra meghívható metódusokat.

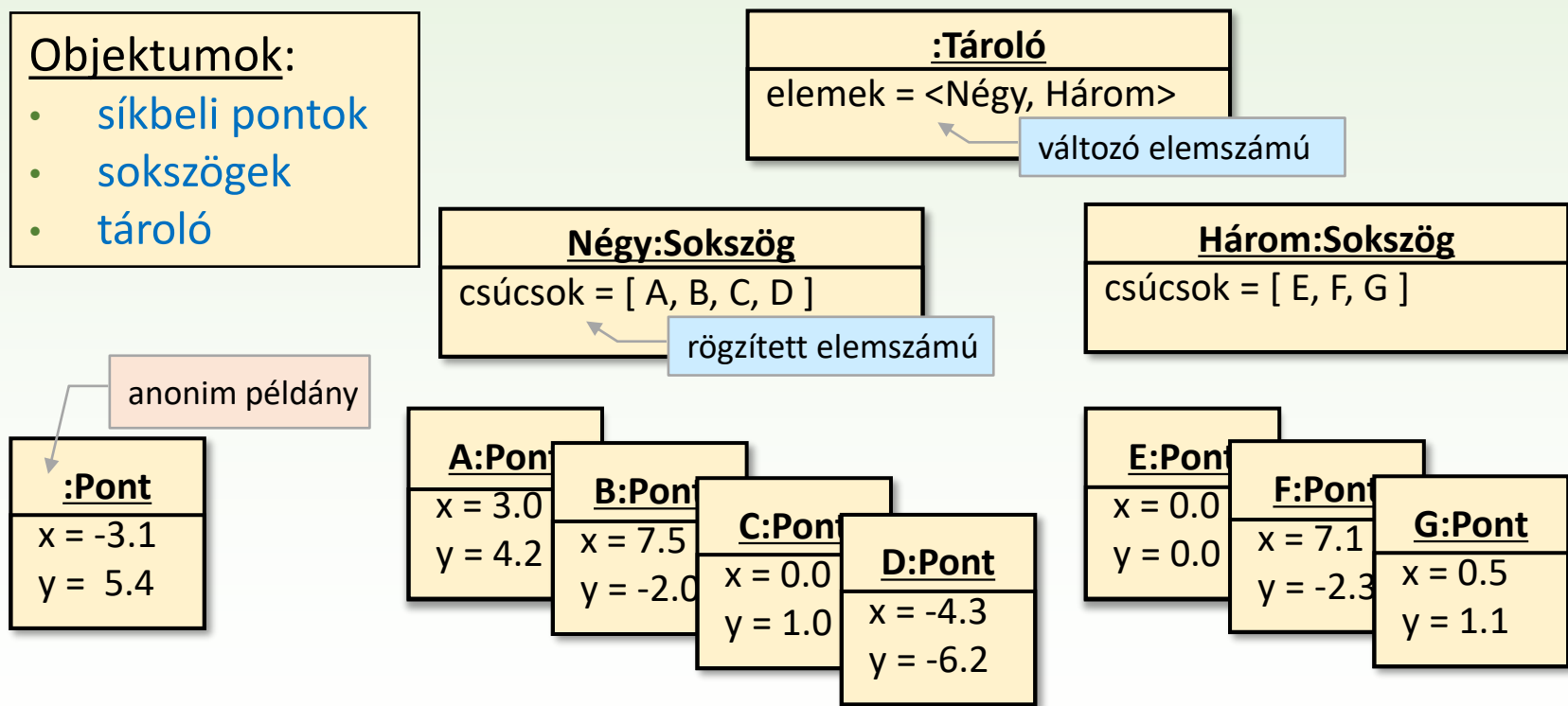


Példa

Töltsünk fel egy tárolót különféle sokszögekkel, és mindegyiket toljuk el ugyanazon irányba és mértékkel, majd számoljuk ki az így nyert sokszögek súlypontjait. A csúcspontok és súlypontok koordinátái, sőt az eltolást leíró helyvektor végpontjának koordinátái is legyenek valós számok.

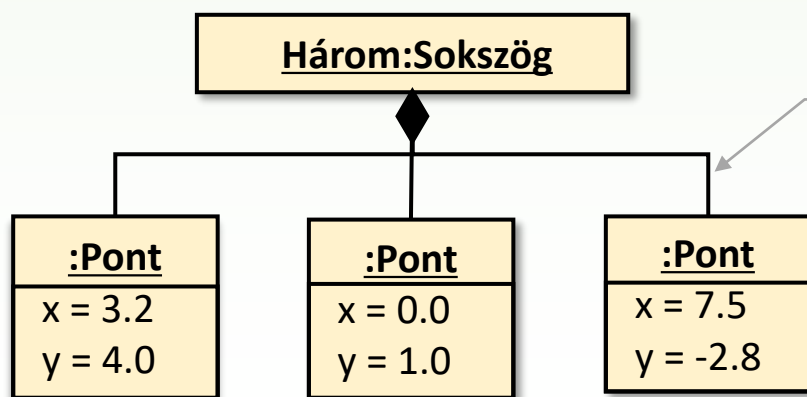
Objektumok:

- síkbeli pontok
- sokszögek
- tároló



Objektumok kapcsolatai

- ❑ Az objektumok között különböző célból jöhetnek létre kapcsolatok:
 - objektum adattagja hivatkozhat másik objektumra (például arra, amely egy részét alkotja),
 - objektum egy metódusa hívhatja egy másik objektum metódusát,
 - objektum küldhet másik objektumnak jelzést (szignált)
- ❑ Egy kapcsolatban az egyik objektumnak ismernie kell a másik objektum hivatkozását: ezt tárolhatja egy adattagjában, vagy egy metódusa megkaphatja paraméterként, esetleg maga példányosítja azt.



Folytonos összekötő vonal jelzi két objektum között fenn álló kapcsolatot. Ennek végén lehet nyíl, amely arra az objektumra mutat, amely a másiktól elérhető (látható); vagy rombusz, ha az így megjelölt objektum tartalmazza a vele kapcsolatban levő objektumokat.

Objektumok felelősségi köre

- ❑ Egy objektum metódusai az objektum adattagjain végeznek olyan műveleteket, amelyek a feladat megoldásának egy részéért felelősek.
- ❑ Amikor egy feladat megoldásához szükséges részfeladatok programjait, mint metódusokat hozzárendeljük az objektumokhoz, akkor kijelöljük az objektumnak a feladat megoldásában játszó felelősségi körét.
- ❑ Ügyelni kell arra, hogy arányosan osszuk szét a felelősségi köröket az objektumok között.

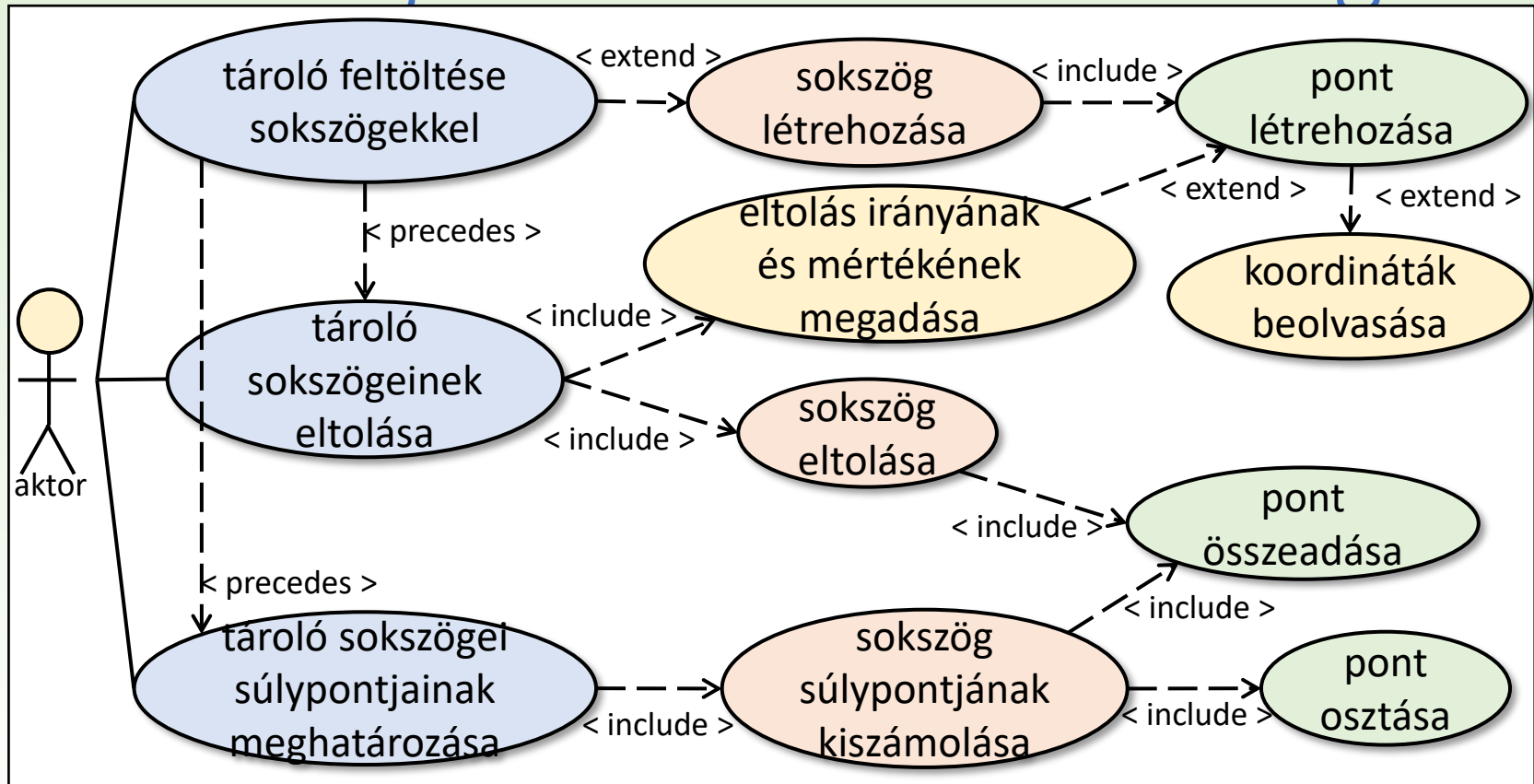
Single responsibility

O
L
I
D

Egy objektum – egy felelősségi kör

- Ne legyenek "mindenható" objektumok, amelyek több felelősségi kör metódusait gyűjtik egybe.
- Ne osztozzon több objektum egy felelősségi körön.

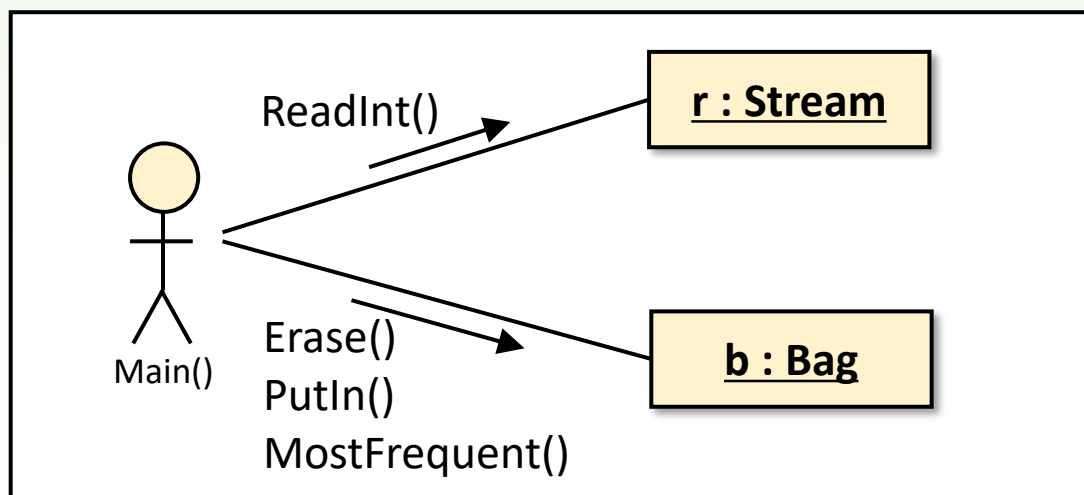
Példa objektumainak felelőssége



- **síkbeli pont:** létrehozás, összeadás, osztás
- **sokszög:** létrehozás, eltolás, súlypont kiszámítás
- **tároló (sorozat):** elem elhelyezése (hozzáfűzése),
elemek kinyerése felsorolásukkal)

Kommunikációs diagram

- ❑ A kommunikációs diagram azt mutatja meg, hogy az egymással kapcsolatba kerülő **objektumok** milyen **üzeneteket** küldenek egymásnak (azaz egy objektum mely **metódusát hívja** meg egy másik objektumnak, vagy milyen **szignált küld** neki).
- ❑ Hasonlít az objektum diagramhoz, de nem tünteti fel az objektumok adattagjait (azok értékét), viszont megmutatja, hogy az objektumok között milyen irányban, milyen tartalmú üzenet valósulhat meg. Ezzel megjeleníti egy-egy objektum felelősségi körét, ezen belül a metódusait.

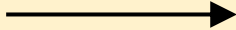
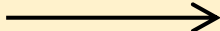


Kommunikációs diagram üzenetei

❑ Üzenet az, amikor

- a küldő objektum meghívja a fogadó objektum egyik **metódusát**,
- a küldő objektum **szignált** küld a fogadó objektumnak.

❑ Az üzenetküldés módja lehet

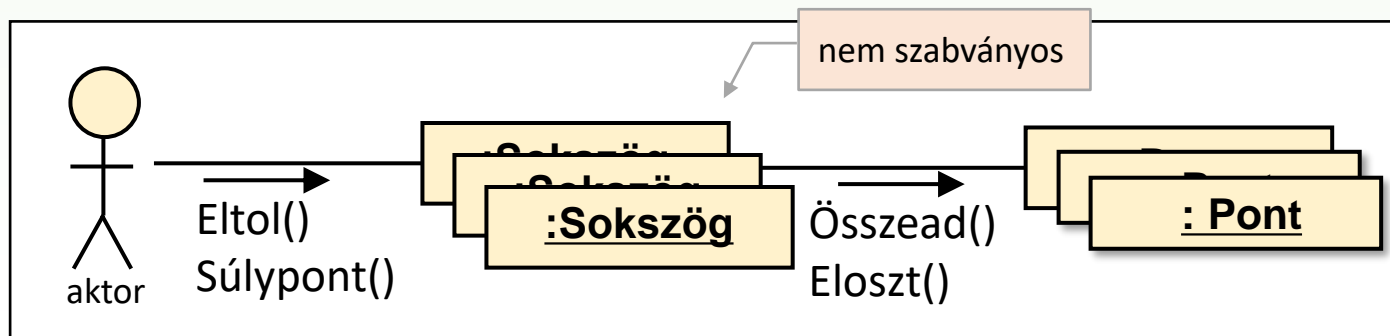
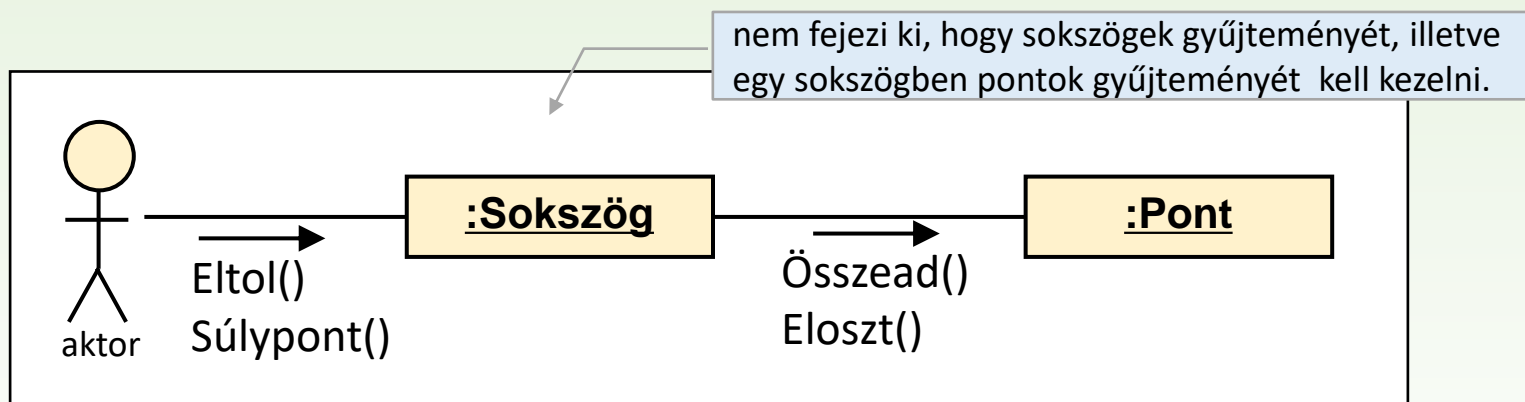
- **szinkron**: a küldő objektum megvárja, amíg a fogadó feldolgozza az üzenetét, és csak ezen feldolgozás esetleges eredményét megkapva folytatja a saját tevékenységét, 
- **aszinkron**: az üzenet elküldése után a küldő és a fogadó objektumok tevékenységei párhuzamosan folynak. 

❑ Egyéb jelölések

- Kijelölhető az üzenetek **sorrendje** (azok sorszámozásával).
- Megadható az üzenetek **előfeltétele** (szögletes zárójelpár között).
- Jelölhető, ha egy üzenet ciklikusan ismétlődik.

Példa

Töltsünk fel egy tárolót különféle sokszögekkel, és mindegyiket toljuk el ugyanazon irányba és mértékkel, majd számoljuk ki az így nyert sokszögek súlypontjait. A csúcspontok és súlypontok koordinátái, sőt az eltolást leíró helyvektor végpontjának koordinátái is legyenek egész számok.



3.rész

Osztály diagram

Gregorics Tibor

gt@inf.elte.hu

<http://people.inf.elte.hu/gt/oep>

Osztály diagram

- ❑ Az osztály diagram a tervezett rendszer működése során létrejövő populációk általános leírására szolgál. Tehát egy osztály diagramnak számos objektum diagramot lehet megfeleltetni, más szavakkal, az osztály diagram az objektum diagramok absztrakciója.
- ❑ Egy osztály az azonos típusú objektumok adattagjait és metódusait írja le.
- ❑ Az objektumok közötti kapcsolatokat az osztály diagram osztályai közötti kapcsolatokkal jellemezhetjük.
- ❑ Az osztály diagram elsősorban egy szoftver tervezését támogató eszköz, de már a szoftver elemzésénél is használható, valamint a megvalósítás leírására is alkalmas.

Osztály leírás elemei

- ❑ Egy osztálynak a leírása a belőle példányosítható objektumokat jellemzi. Tartalmazza az
 - **osztálynevet**, ami az objektumok típusának neve is egyben
 - **adattagok** (tulajdonság, attribútum, mező) felsorolását a nevükkel, típusukkal, és egyéb jellemzőikkel
 - **metódusok** (művelet, tagfüggvény) felsorolását a nevükkel, paraméterlistájukkal, visszatérési típusukkal, és egyéb jellemzőikkel
- ❑ Az adattagok és metódusok egyenként beállítható láthatósága azt jelzi, hogy egy tagra hivatkozhatunk-e az osztály-leíráson kívül vagy sem.
 - kívülről is látható, azaz publikus (*public +*)
 - külvilág elől rejtett: privát (*private -*) vagy védett (*protected #*)

<osztálynév>	
<+ - #> <adattagnév> : <típus>	
...	
<+ - #> <metódusnév>(<paraméterek>) : <típus>	
...	

Típus és Osztály

- Az objektumelvű tervezés során osztályként adjuk meg az egyedi, vagy ún. felhasználói típusokat (*custom type*).

Zsák típus:

típus-specifikáció

zsákok

$b := \emptyset$
 $b := b \cup [e]$
 $e := \text{leggyakoribb}(b)$
 $b: \text{Zsák}, e: \mathbb{N}$

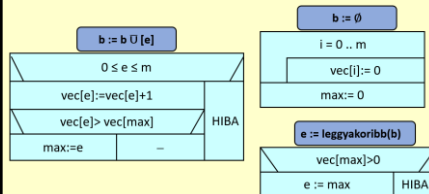
típus-műveletek

típus-értékek

reprezentáció

$\text{vec} : \mathbb{N}^{0..m}$
 $\text{max} : \mathbb{N}$
 $\{ \text{max} \in [0..m] \wedge$
 $\text{vec}[\text{max}] =$
 $\max_{i=0..m} \text{vec}[i] \}$

típus-megvalósítás



Zsák osztály:

$\text{max} \in [0..m] \wedge$
 $\text{vec}[\text{max}] = \max_{i=0..m} \text{vec}[i]$

Bag

$- \text{vec} : \text{int}[0..m]$
 $- \text{max} : \text{int}$
 $+ \text{Erase}() : \text{void}$
 $+ \text{PutIn}(e:\text{int}) : \text{void}$
 $+ \text{MostFrequent}() : \text{int}$

for i=0..m loop vec[i] := 0 endloop
max := 0

if vec[max] ≤ 0 then error endif
return max

if not (0 ≤ e ≤ m) then error endif
++vec[i]
if vec[e] > vec[max] then max := e endif

Példa

Töltsünk fel egy tárolót különféle sokszögekkel, és mindegyiket toljuk el ugyanazon irányba és mértékkel, majd számoljuk ki az így nyert sokszögek súlypontjait. A csúcspontok és súlypontok koordinátái, sőt az eltolást leíró helyvektor végpontjának koordinátái is legyenek egész számok.

<u>Négy:Sokszög</u>
csúcsok = [A, B, C, D]

<u>Három:Sokszög</u>
csúcsok = [E, F, G]

<u>Sokszög</u>
csúcsok : Pont[]
Eltol()
Súlypont()

<u>A:Pont</u>	<u>B:Pont</u>	<u>C:Pont</u>	<u>D:Pont</u>
x = 3.0 y = 4.2	x = 7.5 y = -2.0	x = 0.0 y = 1.0	x = -4.3 y = -6.2

<u>E:Pont</u>	<u>F:Pont</u>	<u>G:Pont</u>
x = 0.0 y = 0.0	x = 7.1 y = -2.3	x = 0.5 y = 1.1

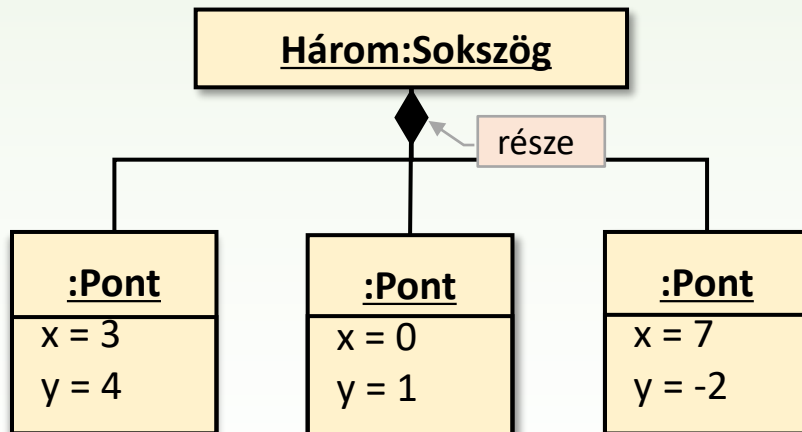
<u>Pont</u>
x : real y : real
Összead() Eloszt()

a láthatóság még nincs eldöntve

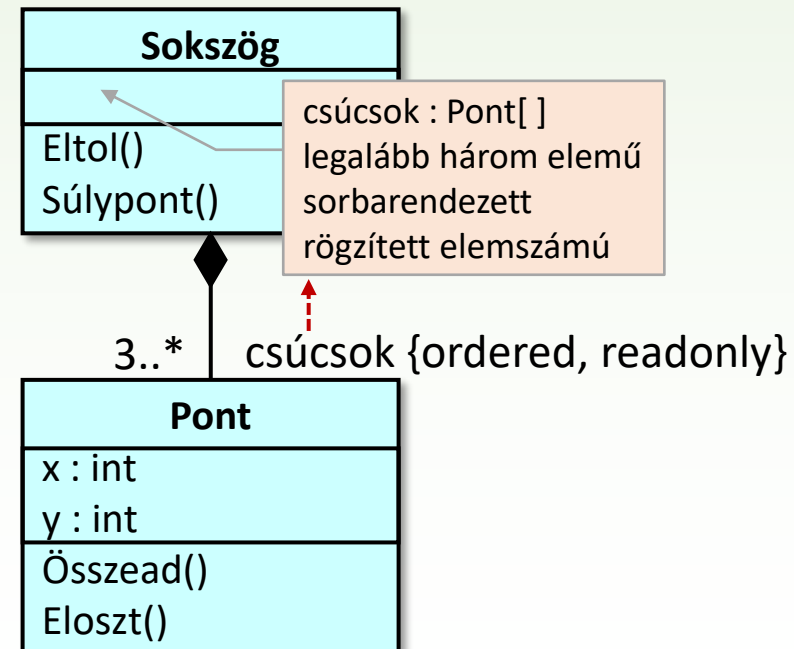
Osztály diagramban ábrázolt objektum-kapcsolatok

- Az objektumok közötti kapcsolatok osztályszintű leírását egy későbbi előadásban részletesen tárgyaljuk.

Objektumok közötti kapcsolatok
ábrázolása objektum diagramban:



Objektumok közötti kapcsolatok
ábrázolása osztály diagramban:



Osztály diagram részletezettsége

- Az osztály diagram **a modellezés során fokozatosan alakul ki**. Az elemzés fázisában még számos részlete hiányozhat, de a tervezés fázisa sem tartalmaz minden, a megvalósításhoz szükséges információt.
- Hiányozhatnak belőle az attribútumok és/vagy a metódusok.
 - Elemzéskor elmaradhat még az adattagok típusa, a metódusok paraméterezése és a visszatérési típusa, valamint a láthatósági jelölések feltüntetése.
 - Az osztályok közötti kapcsolatok jellemzői is (ezeket később tanuljuk) csak fokozatosan kerülnek fel a diagramra.

<osztálynév>

<osztálynév>

<metódusnév>()

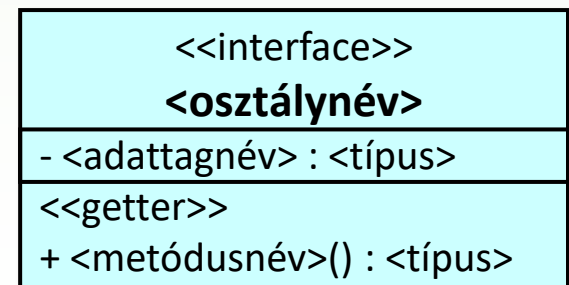
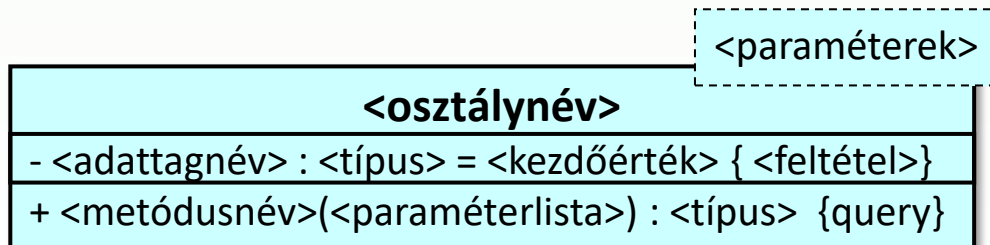
<osztálynév>

- <adattagnév> : <típusnév>

+ <metódusnév>(<paraméterek>) : <típusnév>

Osztály diagram kiegészítései

- ❑ Az osztályok a példányaik jellegzetes **viselkedése** alapján különféle kategóriákba (sztereotípiákba) sorolhatók, amelyet <<...>> jelzés között írhatunk le (pl. <<interface>> , <<enumeration>>, <<singleton>>).
- ❑ Egy osztályt általánosítva is felírhatunk (generikus-, sablon-osztály), ha **paramétereket** (pl. típust helyettesítve) vezetünk be a leírásában.
- ❑ Az adattagokhoz (a '=' jel után) **kezdőértéket** rendelhetünk, amiket a konstruktor állít majd be, illetve {...} jelzésben **megszorításokat** (típus invariáns) is adhatunk rájuk.
- ❑ A metódusok sajátos tulajdonságai (pl. query, override, virtual) is megadható {...} jelzés között. A <<getter>> az adattagok értékét **lekérdező műveletek** csoportját, a <<setter>> az adattagok értékét **felülíró műveletek** csoportját vezeti be.



4.rész

Szekvencia diagram

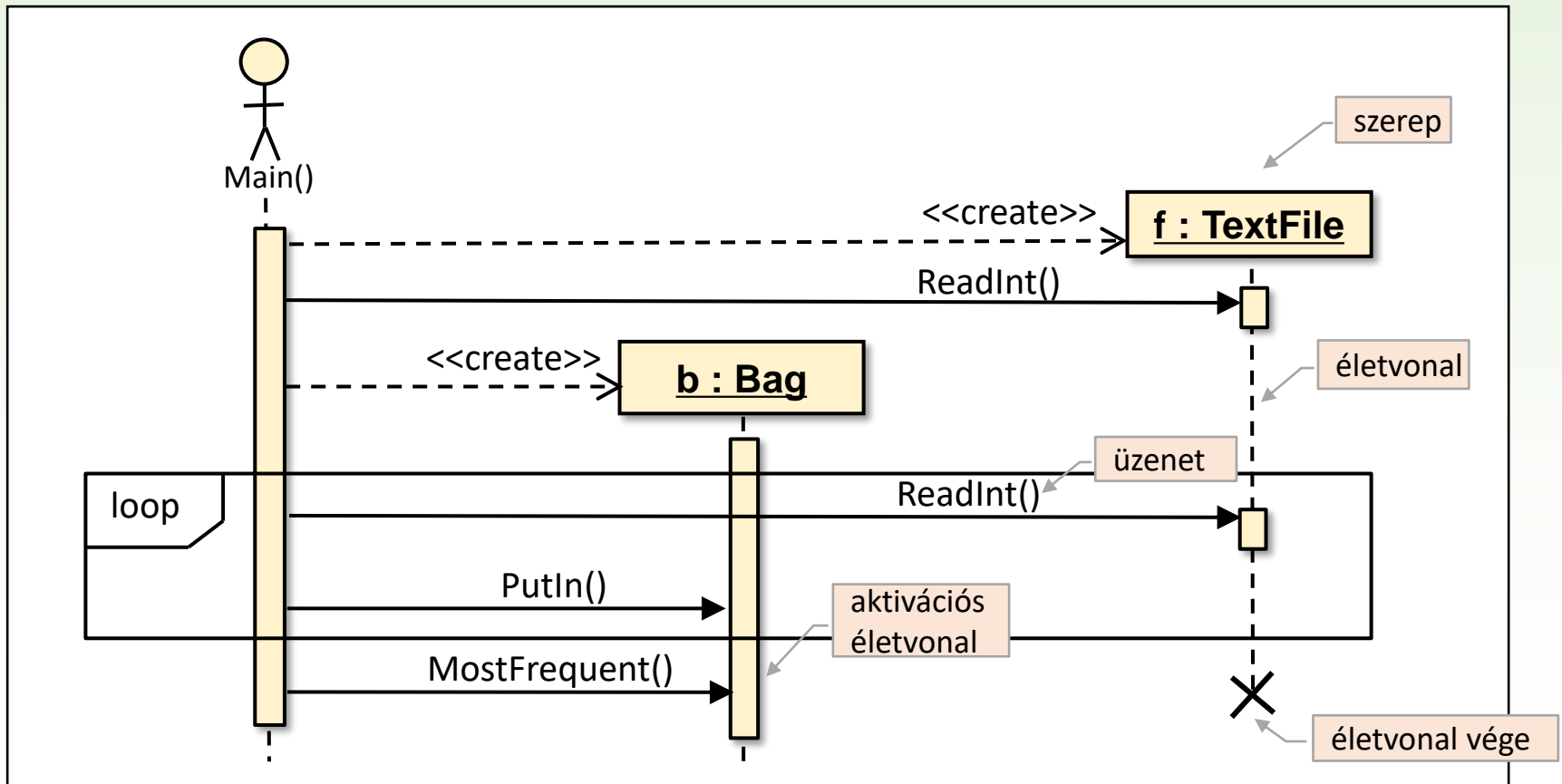
Gregorics Tibor

gt@inf.elte.hu

<http://people.inf.elte.hu/gt/oep>

Szekvencia diagram

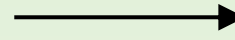
- Az objektumok közötti **üzenetváltások időbeli sorrendjét** mutatja meg.
- Ez a diagram a rendszer vagy a rendszer egy részének egy lehetséges működését írja le, ezért több is szokott készülni belőle. Nemcsak az elemzés, hanem a tesztelés dokumentálásához is használják.



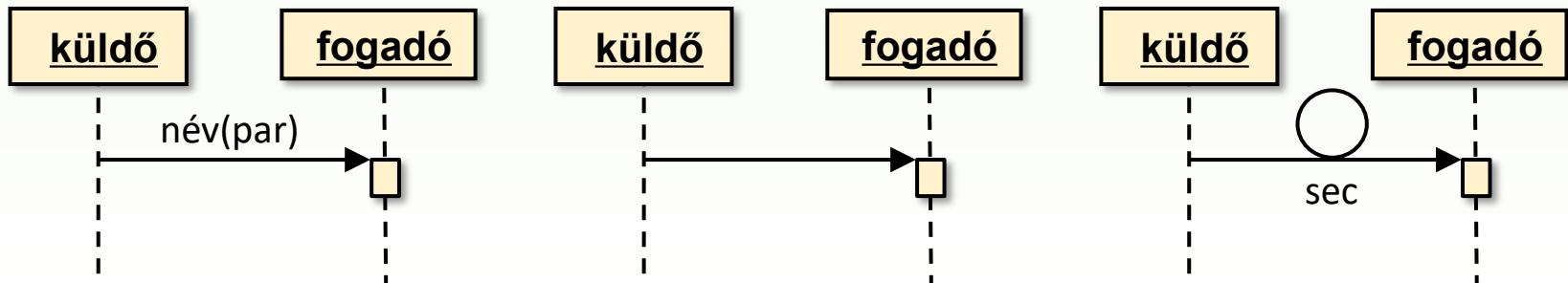
Szekvencia diagram elemei

- ❑ **Osztályszerp** : egy vagy akár több osztály egy vagy több objektumát testesíti meg, sokszor csak egyetlen objektumot.
- ❑ **Osztályszerp életvonal** (szerepből kiinduló függőleges szaggatott vonal) : egy osztályszerp létezését ábrázoló idővonal.
- ❑ **Osztályszerp aktivációs életvonal** (egy hosszú elnyújtott téglalap az életvonalon) : egy osztályszerpnek azon időszaka, amelyben az osztályszerpet megtestesítő objektumok más objektumok vezérlése alatt állnak, azaz egy vagy több metódusuk végrehajtás alatt áll.
- ❑ **Üzenet** (általában vízszintes nyíl az életvonalak között) : az objektumok közötti információ átadás formája. Megmutatja, hogy egy objektum egy üzenet elküldésével mikor hoz működésbe (akció) egy másik objektumot. (pl. meghívja annak egy metódusát.)
 - Az egymás alatt jelzett üzenetek azok időbeli sorrendjét mutatja.
 - Feltüntethető két üzenet között eltelt idő is.

Szinkron üzenetek



- ❑ Szinkron üzenetről akkor beszélünk, amikor a küldő objektum átadja a vezérlést a fogadó objektumnak, és a saját tevékenységét mindaddig blokkolja, amíg a fogadó objektum ezt nem oldja fel. Ez lehet
 - a fogadó objektum egy **metódusának hívása**. Ilyenkor az üzenet címkéje a meghívott metódus neve az esetleges paraméterekkel.
 - egy **szinkronizációs üzenet**, amely során a küldő objektum várakozik a fogadó objektum visszajelzésére.
 - **időhöz kötött várakozó üzenet**, amikor a küldő objektum megjelölt ideig várakozik a fogadó objektum visszajelzésére.



Aszinkron üzenetek \longrightarrow régebben: \longrightarrow

- ❑ Aszinkron üzenetről akkor beszélünk, amikor a küldő szerep az üzenet elküldése után nem szakítja meg a saját tevékenységét, nem várakozik feltétel nélkül a fogadó objektum visszajelzésére. Ez lehet
 - aszinkron módon küldött **szignál** (ami kiegészülhet paraméterekkel) vagy aszinkron **metódus-hívás**, amikor a szignál feldolgozása, illetve a metódus futtatása külön szálon történik.
 - **randevú üzenet**, amely során a küldő objektum megjelölt ideig várakozik arra, hogy a fogadó objektum fogadja az üzenetét.

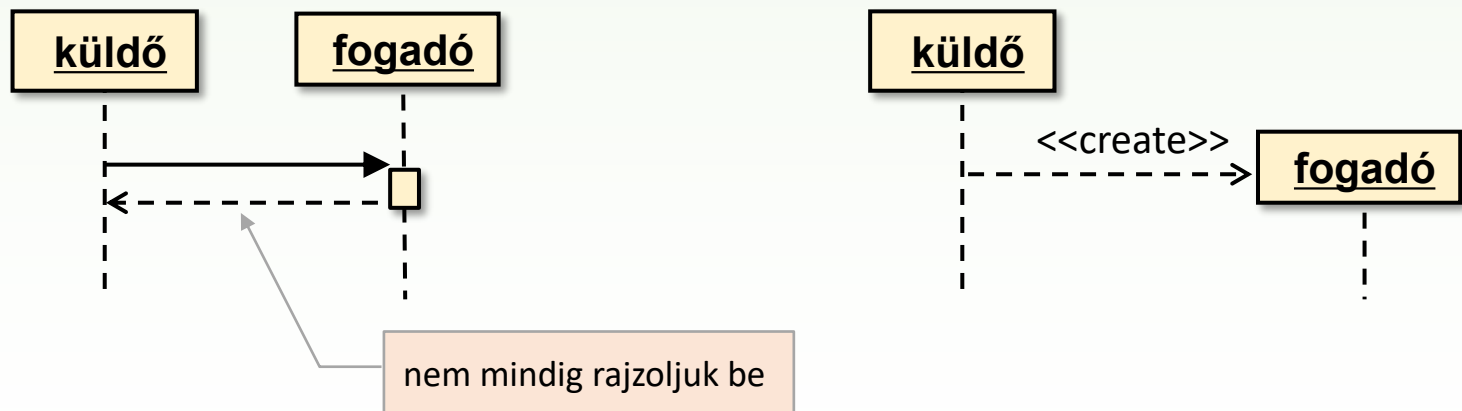


Speciális üzenetek

----->

□ Ezek azok az üzenetek, amelyeket gyakran fel sem tüntetünk a könnyebb áttekinthetőség érdekében:

- **visszatérési üzenet**, amelyet a fogadó objektum küld el az őt aktiváló objektumnak egy korábban attól kapott üzenetre válaszként. Ha ez a korábbi üzenet szinkron üzenet volt, akkor a visszatérési üzenet szünteti meg a küldő objektum blokkolását.
- **példányosító (create) üzenet**: létrehoz egy új osztályszeretet

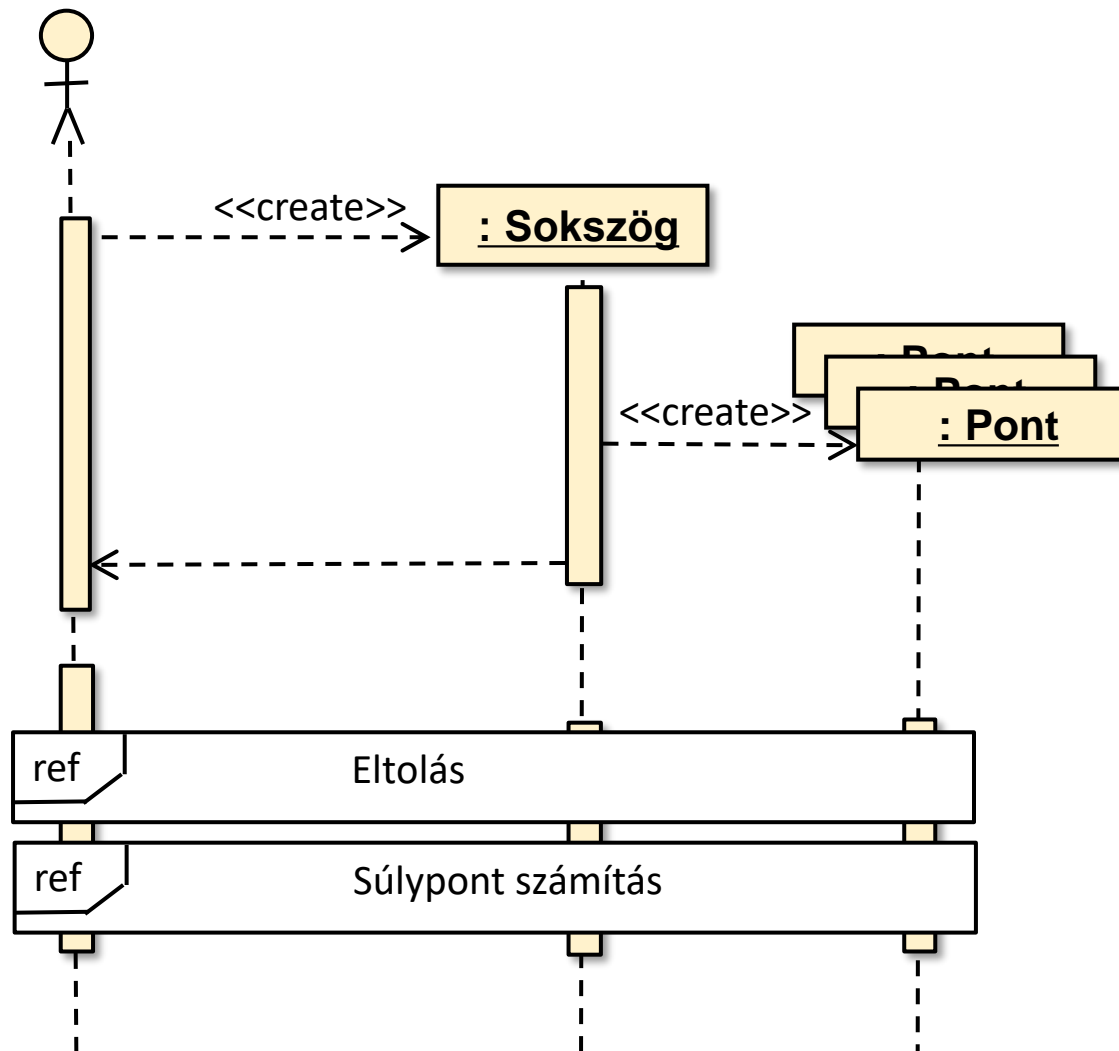


Szekvencia diagramok hierarchiája

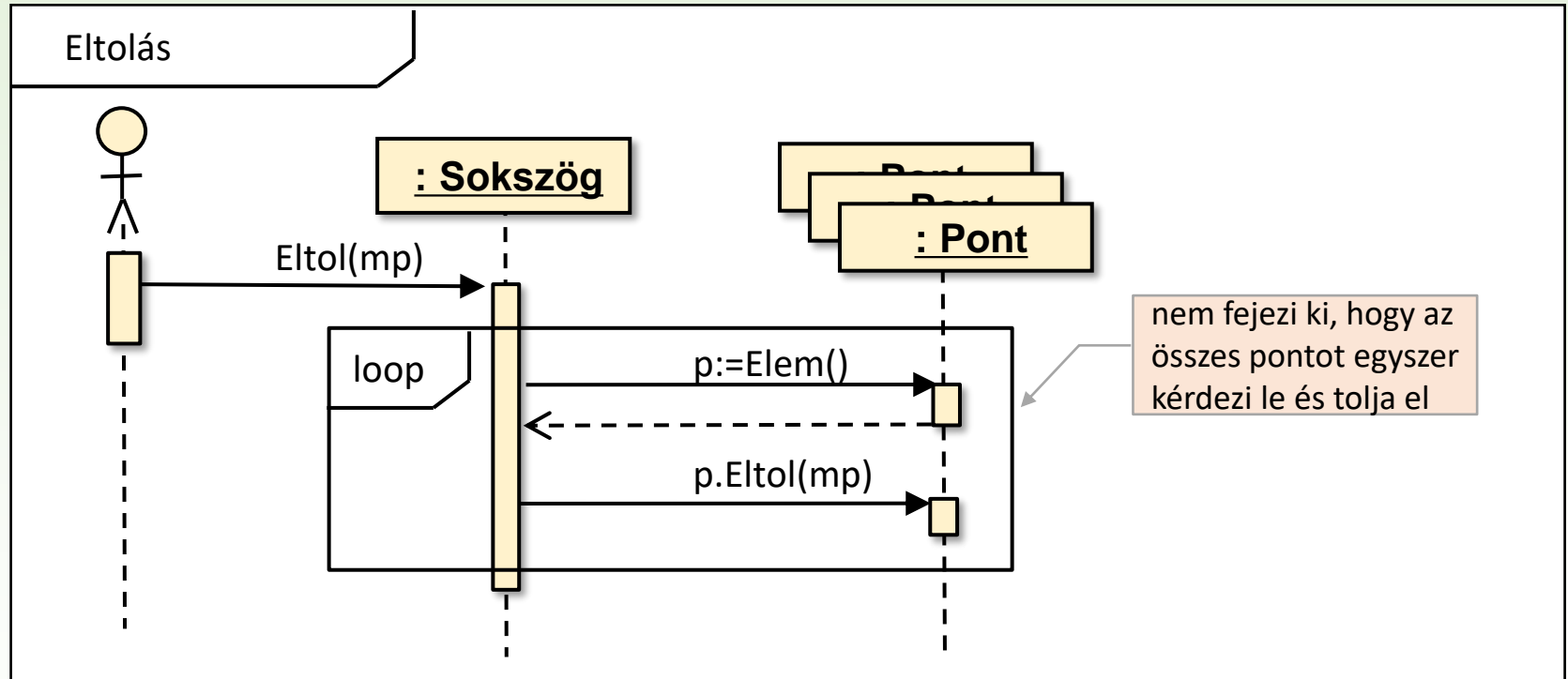
Lehetőség van arra, hogy egy összetett szekvencia diagram bizonyos részeit az olvashatóság érdekében ideiglenesen elrejtjük, és ezeket majd külön részletezzük, finomítsuk.

- ❑ Egy szekvencia diagram egyetlen osztályszerepe elfedheti például a szerepet alkotó objektumok, objektum-csoportok közötti üzenetváltásokat.
- ❑ Kiemelhető a diagram egy adott időintervallumának egy része is (ref).
- ❑ Kijelölhetünk olyan részleteket (fragmentek), amelyek
 - ciklikusan ismétlődhetnek (loop)
 - adott feltétel esetén következhetnek csak be (alt, opt)
 - párhuzamosan futnak (par)
 - ...

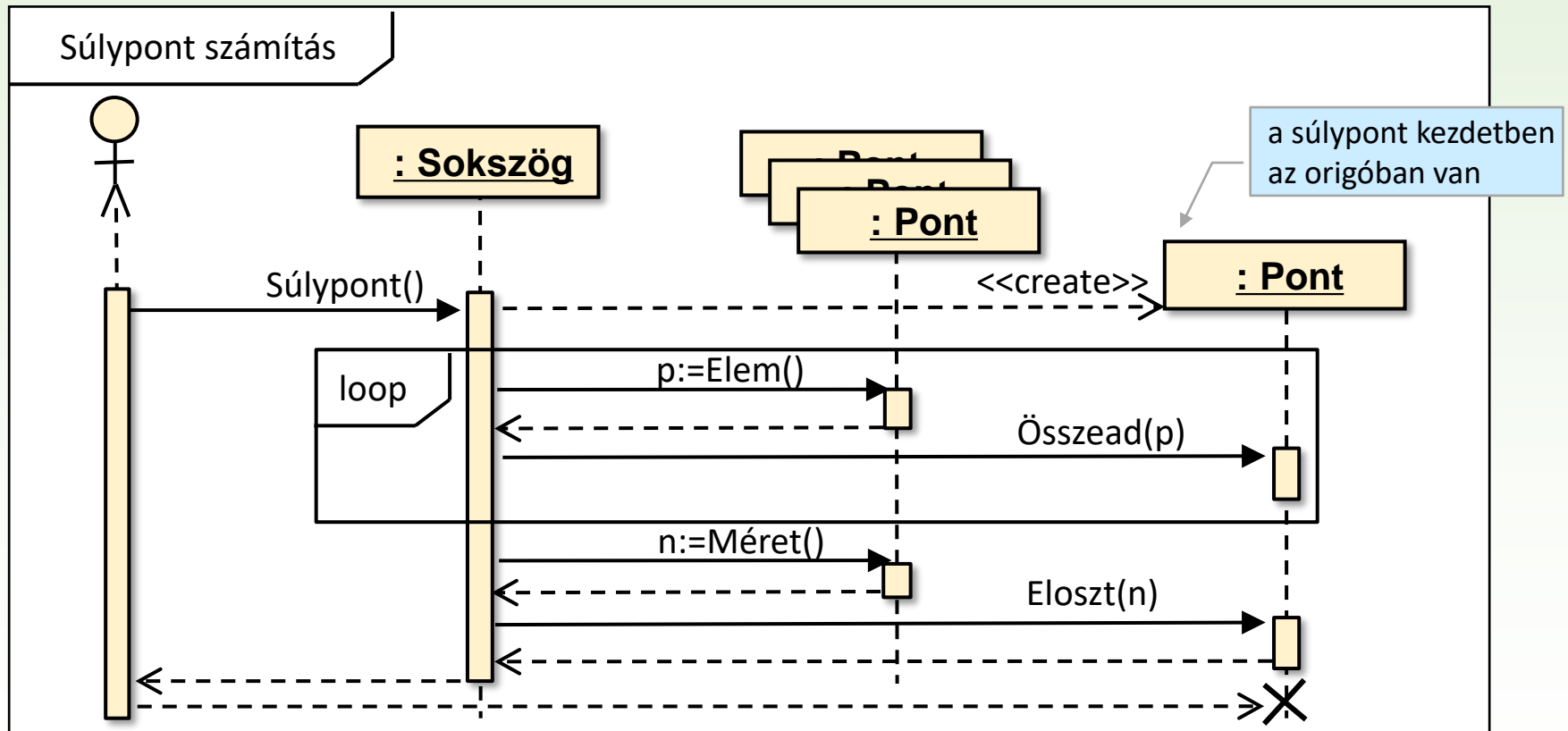
Példa



Példa



Példa



5.rész

Állapotgép diagram

Gregorics Tibor

gt@inf.elte.hu

<http://people.inf.elte.hu/gt/oep>

Állapotgép diagram

- ❑ Egy objektumnak különböző állapotai lehetnek. Állapotnak tekinthető
 - az objektum adatai által felvett értékek együttese (**fizikai állapot**)
 - közös tulajdonságú fizikai állapotok halmaza (**logikai állapot**).
- ❑ Az objektum állapotai közül egyszerre csak egy lehet **aktív**, amely valamilyen esemény (pl. metódushívás vagy szignál) hatására változhat meg. Az a változás az **állapot-átmenet**.
- ❑ Az objektum (logikai) állapotainak változásait egy **irányított gráffal** ábrázolhatjuk, amelyben
 - a csúcsok jelölik az objektum állapotait,
 - az irányított élek mutatják az állapotok közötti átmeneteket.
- ❑ Amikor egy átmenet az objektumnak küldött üzenet (metódus hívás vagy szignál küldés) hatására következik be, akkor az átmenetet jelző nyílra ráírjuk az üzenet nevét (kiegészítve bemenő paramétereivel és a tevékenységével).
- ❑ Ezt a diagramot később ennél sokkal részletesen tárgyaljuk majd.

Példa: egy nyomtató állapotgépe

- ❑ A nyomtató kétféle állapotban lehet: szabad vagy foglalt.
- ❑ Amikor egy kliens használja a nyomtatót, akkor az **foglalt**, amikor senki nem használja, akkor **szabad** állapotban van.
- ❑ Ezek az állapotok ciklikusan változnak.

