Szoftverfejlesztési technológiák 04-06

/óravázlat/

Programfejlesztés (objektumelvű) modellalkotással

Soroljuk fel a modell néhány fontos jellemzőjét!

- A modell a valóság egy leegyszerűsített megvalósítása.
- A modell alapján a problémát jobban meg tudjuk érteni.
- A modell alapján a probléma megoldásainak tulajdonságait vizsgálni tudjuk.
- A modell alapján megalapozott, dokumentált döntést tudunk hozni a konkrét megvalósításról.

Ha összehasonlítjuk a programozást és a modellalkotást, arra az eredményre jutunk, hogy a programozás során a problémát a számítástechnikai térben (megoldástérben) oldjuk meg, a modellalkotás során pedig valamilyen absztrakt térben tesszük ugyanezt.

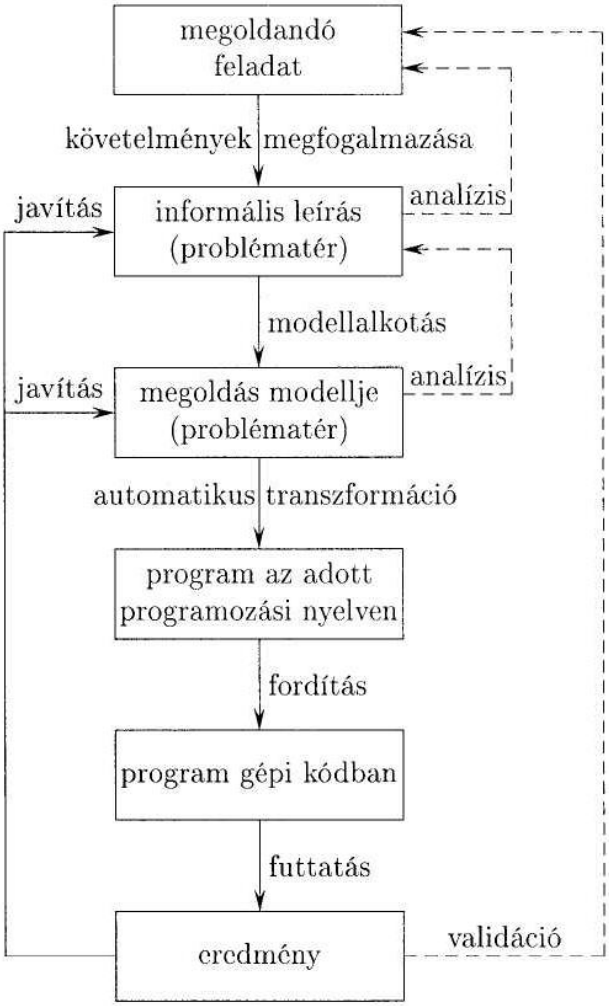
Hagyományosan nagy rendszerek fejlesztésekor a tervezés során létrehozunk egy absztrakt modellteret, amelyben leírjuk a tervet, azaz a problémateret leképezzük egy absztrakt térre. Az implementáció során az absztrakt teret képezzük le a megoldástérre.
A terek közötti transzformációk bonyolítják a folyamatot, és hibákat okozhatnak.

→Probléma
→Követelmények megfogalmazása
→Modellek
→Specifikációk
→Tervek
→…
→Magasszintű forrásnyelvi program
→…
→Végleges gépi kódú program

Az *objektumelvű modellalkotás* alapja és lényege, hogy a probléma megoldása a valós világ objektumainak a terében, azaz *közvetlenül a problématérben* történik. Ekkor a tervezés során elvileg nem kell teret váltanunk. Ez rendkívüli módon leegyszerűsíti a modellezés folyamatát, és csökkenti az elkövethető hibák számát. Léteznek előre definiált, szabványos jelölés rendszerek (pl.: UML), amelyekkel a megoldás objektumelvű modelljét leírhatjuk. Ez a szabványos leírás veszi át a hagyományos programfejlesztési módokban szereplő formális specifikáció, illetve rendszerterv, stb. szerepét.

Mi a modellalkotás során a továbbiakban az ún. **UML** (**Unified Modeling Language**) nyelvet fogjuk használni. Ma már rendelkezésre állnak olyan eszközök, amelyek ebből a leírásból automatikusan előállítják a program kódját, illetve annak vázát (osztályok, műveletek deklarációi) egy adott nyelven. *Ennek megfelelően a kész kódot nem is kell változtatni, a*

javítások egy szinttel feljebb tolódnak. Továbbá optimális esetben nincs szükség a programkód és a megoldás modelljének összevetésére sem, így az ennek megfelelő elemzés (analízis) is elmaradhat:



A modellalkotás néhány alapelvét (ökölszabályát) soroljuk fel a következőkben:

- A modellt tudni kell jól megválasztani. (Ezt a gyakorlat, tapasztalat alakítja ki.)
- Egy modell nem modell. (Több lehetőséget kell megvizsgálni, mielőtt kiválasztanánk a használandót.)
- A modellnek az elvárásoknak megfelelő pontossággal kell tükröznie a valóságot.

Hova lehet még fejlődni programfejlesztési technológiák tekintetében?

A jövőre nézve azt mondhatjuk, hogy a XXI. századi szoftvertechnológiát várhatóan a következők jellemzik majd:

- nyílt,
- szabványokra épülő,

- objektumelvű,
- komponens alapú,
- teljes életciklust lefedő,
- elosztott rendszerű,
- távmunkában felhasználható, azt támogató,
- minőségbiztosítást támogató.

A komponens alapú rendszerekben a komponens fogalmát a következőképpen határozzuk meg: egy önálló részfeladat megoldására szolgáló fizikailag létező része a rendszernek, amely más egységgel helyettesíthető, amennyiben az új komponens megvalósítja a csatlakoztatáshoz előírt felületet.
Az elosztott rendszerű alkalmazások alapjában kliens-szerver rendszerek, illetve a WEB alapú alkalmazások. A kliens-szerver rendszerekben a kliens a felhasználói kommunikációt látja el, míg a szerver a megosztott erőforrásokat használja, és ezekkel nyújt szolgáltatást. A WEB alapú alkalmazásokban a kliens és a szerver közötti kapcsolat több szerveren keresztül jöhet létre.

Unified Modeling Language (UML)

Az objektumelvű modellalkotás során az úgynevezett UML (Unified Modeling Language) nyelvet fogjuk használni, ezért vázlatosan áttekintjük ennek főbb jellemzőit és történetét.

Az UML egy grafikus nyelv, amelyben lehetőségünk van a probléma

- specifikációjára,
- megoldására,
- a megoldás dokumentálására.

Az UML-hez léteznek az alábbi programozási nyelvekhez transzformációs eszközök:

- JAVA,
- C++,
- C#,
- Smalltalk,
- Visual Basic,
- 4 GLs,
- …

Az UML az 1990-es években alakult ki, és 2000-ben az ISO szabványa lett.
Az UML fokozatos fejlődésen ment keresztül:

Szabvány	Dátum
UML 0.8	1995.
UML 1.0	1997. jan.
UML 1.1	1997. szept.
UML 1.2	1998.
UML 1.3	1999.

UML 1.4 – ?
UML 1.5 – ?
UML 2.0 – 2005.
UML 2.1.1 – 2007.
UML 2.1.2 – 2007.
UML 2.2 – 2009. február
UML 2.3 – 2010. május
UML 2.4 (béta) – 2011. március
UML 2.4.1 – 2011. augusztus
UML 2.5 – 2015. június

Folyamatosan fejlődik, nem kiforrott.

Soroljuk fel az UML néhány jellemzőjét:

1. Az UML-ben létrehozott modell grafikus szemléltetést nyújt.

2. A nyelv lehetővé teszi modellek megalkotását különböző nézetekből, nézőpontokból.

3. Alkalmas a probléma megoldásának pontos leírására.

4. Alkalmas a probléma megoldásának dokumentálására, ezen belül:

   • a projektterv dokumentálására,
   • a szoftverkészítés fázisainak (követelmények, tervezés, ...) dokumentálására,
   • a prototípus dokumentálására.

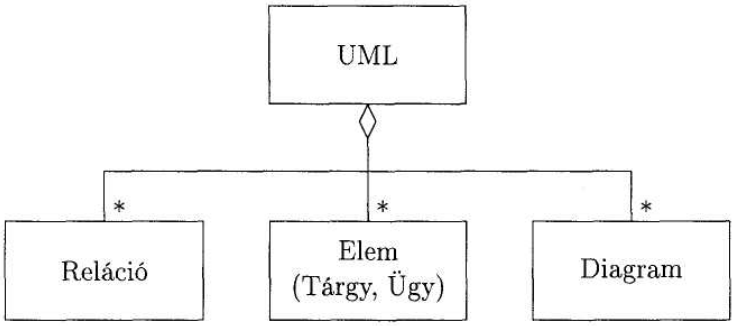
5. Az UML konstrukciós eszköz.

6. Nemcsak szoftvermodellek leírására alkalmas, de munkafolyamatok, szervezetek, különböző üzleti tevékenységek, stb. leírására is.

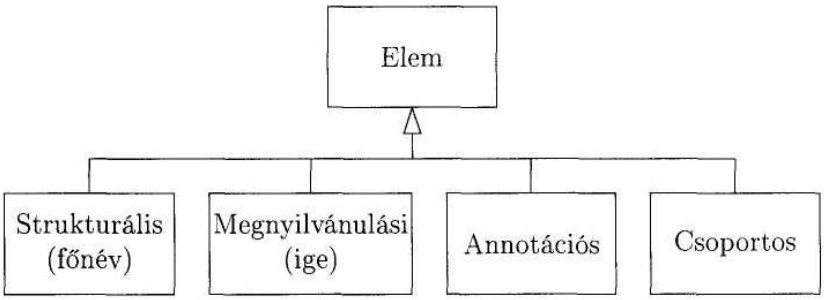
Az UML építőkövei

A következőkben áttekintjük az UML alkotóelemeit. A leíráshoz felhasználjuk kiegészítésként az UML jelölésrendszerét is, ezzel később ismerkedünk meg részletesen.

Egy UML leírás *relációkból, elemekből* (tárgy, ügy) és *diagramokból* áll:

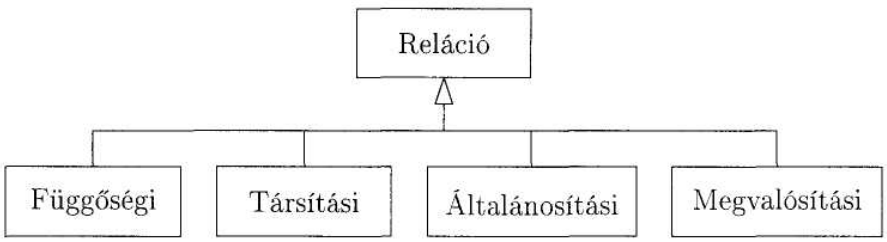


Az elemek lehetnek:

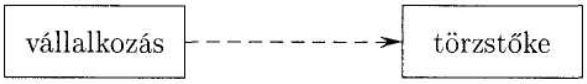


- Strukturális elemek:
  - Osztály
  - Objektum
  - Használati eset
- Megnyilvánulási elemek:
  - Művelet végzése
  - Interakció (üzenetküldés)
  - Állapotautomata (különböző állapotok közötti átmenetek)
- Annotációs elemek:
  - Kiegészítés
  - Megjegyzés
  - Megszorítás
- Csoportos elemek:
  - Alrendszer
  - Package (csomag)
  - ...

A relációkat 4 csoportba oszthatjuk:



- Függőségi reláció: szemantikai összefüggés. Például:



- Társítási reláció: strukturális, szerkezeti összefüggés. Például:

- Általánosítási reláció: általános és speciális kapcsolata. Például:

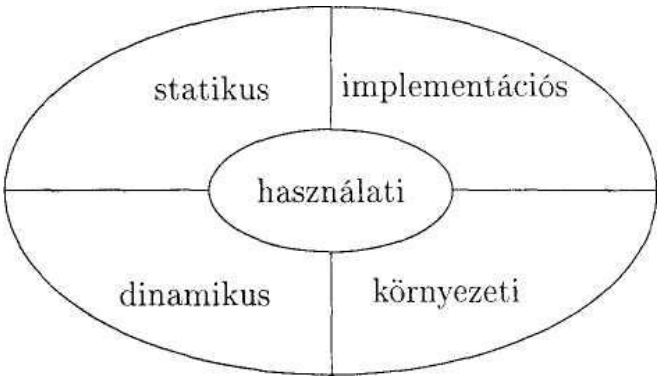
- Megvalósítási reláció: szemantikai kapcsolat a fogalom és annak megvalósítója között. Például:

A diagramok az elemeket és a köztük fennálló relációkat szabványos jelöléssel leíró ábrák. Lásd később!

Nézetrendszer (nézőpontrendszer, szempontrendszer)

Megvizsgáljuk, hogy milyen szempontok, nézőpontok szerint lehet egy megoldandó problémát megközelíteni, szemlélni, hogyan kapcsolódik mindez az UML jelölésrendszeréhez; és megadjuk néhány alapvető fogalom definícióját.

A következőkben megadjuk a probléma megoldásának nézetrendszerét néhány mondatban:



• *Használati szempont:* A használati szempontból történő vizsgálat arra keresi a választ, hogy kiknek nyújt ez a rendszer szolgáltatást. Ezek lehetnek személyek, de lehetnek más rendszerek, programok is. A rendszer viselkedését, funkcionalitását írja le a szereplők és a feladatok megjelölésével, a felhasználó szemszögéből nézve. Azaz alapvetően az egyes

felhasználási eseteket vizsgálja.

• *Szerkezeti, strukturális, statikus szempont:* Itt arra vagyunk kíváncsiak, hogy a rendszer milyen egységekből (osztályokból és objektumokból) épül fel, mi ezeknek az egységeknek a feladata, milyen kapcsolatban vannak egymással a megoldás elérésének az érdekében.

• *Dinamikus szempont:* Ennek során arra keressük a választ, hogy a rendszer egyes részegységei hogyan viselkednek a megoldás során. Az egységek milyen állapotokat vesznek fel, milyen események hatására változik az állapotuk, milyen a közöttük lévő együttműködés mechanizmusa, időben hogyan játszódnak le közöttük az üzenetek, stb. Ez a szempont az, amely a legtöbb kérdésre keresi a választ.

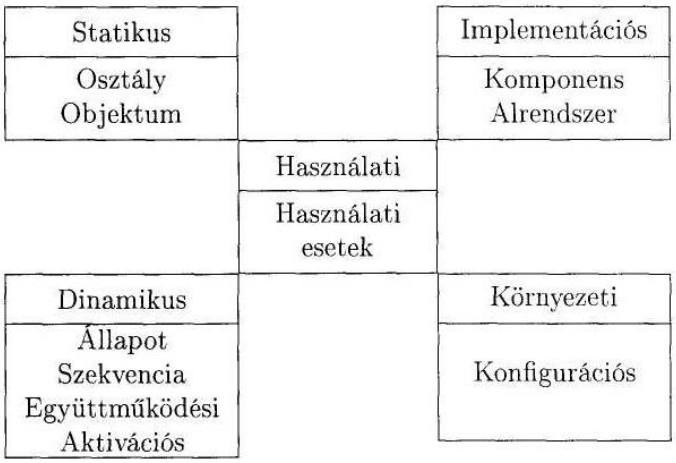
• *Implementációs szempont:* Ekkor a megoldás megvalósításának szoftver kérdéseire keressük a választ: milyen szoftverkomponensek vesznek részt a megoldásban, és azok között milyen kapcsolatok állnak fenn.

• *Környezeti szempont:* Ilyenkor a megoldás hardver és szoftver konfigurációjára vagyunk kíváncsiak. Azt vizsgáljuk, hogy a rendszer milyen hardver és szoftver erőforrást igényel a megoldás során.

A felsorolt nézetrendszer sokrétű, és az elemzés során a megoldás csak egy-egy szempontból szemléltethető. Ezért alakult ki sokféle szemléltető eszköz az idők folyamán. Ezek nagyrészt diagram formájában láttak napvilágot.
Az UML-ben megtestesülő filozófia az, hogy a lényeges, a gyakorlatban bevált diagramokat szabványos formában, egy egységes nyelvezetben foglalja össze. Ha ez a nyelv olyan, hogy ahhoz bizonyos szinten egyértelmű szintaxis és szemantika rendelhető, akkor szoftvereszközökkel egy adott objektumelvű programozási nyelvre áttranszformálható.

Az UML diagramjai

Most megadjuk, hogy a később bevezetésre kerülő UML diagramok mely szempont, nézőpont szerinti nézetrendszerhez kapcsolódnak:



• Statikus szempont szerint:
- *Osztálydiagram (Class):* a rendszer objektumelvű szerkezetének leírása.

- *Objektumdiagram (Object):* az osztálydiagram egy példányát mutatja be.

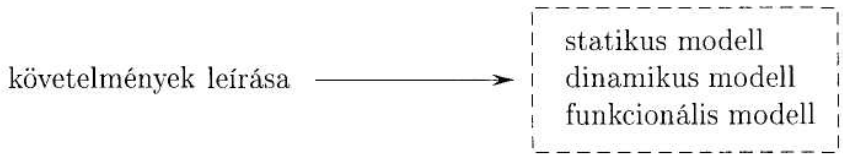
• Dinamikus szempont szerint:
  - *Állapotdiagram (Statechart):* azt mutatja meg, hogy a rendszer milyen állapotokon keresztül, milyen állapotátmenetekkel oldja meg a feladatot.
  - *Szekvenciadiagram (Sequence):* az objektumok közötti üzenetváltások időbeli menetét szemlélteti.
  - *Együttműködési diagram (Collaboration):* az objektumoknak a probléma megoldásában való együttműködését mutatja be.
  - *Aktivációs diagram (Activity):* a tevékenységek és az objektumok egymásra gyakorolt hatását fejezi ki (vezérlések, rendszerfunkciók).

• Implementációs szempont szerint:
  - *Komponensdiagram (Component):* a komponensekből felépülő szoftverrendszert mutatja be.
  - *Alrendszerdiagram:* az alrendszerek kapcsolatát írja le.

• Környezeti szempont szerint:
  - *Konfigurációs diagram (Deployment):* a szoftverrendszer környezetének, a hardver-szoftver konfigurációnak a szemléltetésére szolgál.

• Használati szempont szerint:
  - *Használati esetek diagramja (Use case):* a rendszernek és felhasználóinak kapcsolatát adja meg a felhasználási esetek bemutatásán keresztül.

Az objektumelvű modellezés során a *követelmények* leírásából, specifikációjából a megoldás modelljét három részből, három *részmodellből* állítjuk elő a *problématérben*:



A statikus modell a rendszer szerkezetét adja meg (statikus szempont), a dinamikus és funkcionális modell a rendszer viselkedését határozza meg (dinamikus szempont). Ezek közül elsőként a *statikus modellel* foglalkozunk. A statikus modell az osztálydiagramokból és a hozzájuk tartozó osztályleírásokból, valamint az objektumdiagramokból és az ezekhez tartozó objektumleírásokból áll. Azaz:

statikus modell = osztálydiagram + osztályleírások,
objektumdiagram + objektum leírások.

A továbbiakban a statikus modell elemeivel foglalkozunk.

Az osztály leírását alkotó 4 összetevőt:

• *A paraméter rész leírása* az osztály, illetve objektumainak felépítésében szereplő jellemzők leírását tartalmazza. Ezeket a jellemzőket nevezzük *attribútumoknak,* és ezek két csoportra oszthatók:

- *Objektum attribútumok* azok, amelyek értékei egy-egy objektumra nézve specifikusak. Például az objektum állapotát meghatározó értékek.
- *Osztály attribútumok,* amelyeknek értékei az osztály minden objektumát jellemzik. Például az osztály objektumainak maximális mérete.

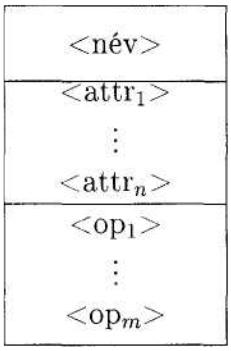
• *A szolgáltatások leírása* az osztály által nyújtott szolgáltatások formájának és jelentésének leírását tartalmazza. Ez a leírás három részre tagolódik:

- Az egyes objektumokra vonatkozó szolgáltatások, operációk *(objektum operációk)* leírása.
- Az osztályra, mint egészre vonatkozó operációk *(osztály operációk)* leírása. Például az osztály egy példányának létrehozása, az attribútumok listájának kiíratása.
- A szolgáltatások igénybevételére vonatkozó feltételek *(korlátozások)* leírása.

• *Az import rész leírása* az osztály szolgáltatásainak megvalósításához szükséges, más osztályoktól igénybe vett szolgáltatások leírását tartalmazza.

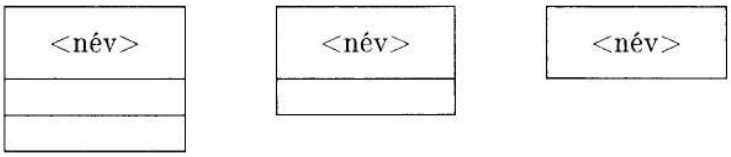
• *A megvalósítás leírása* az osztály szerkezeti tulajdonságainak ábrázolását és viselkedésbeli tulajdonságainak konkrét implementációját írja le.

A diagramokban az osztály ábrázolásánál, jelölésénél a nevét, az attribútumok neveit és a műveletek absztrakt formáját tüntetjük fel. Absztrakt osztály esetén a nevet dőlt betűkkel adjuk meg:



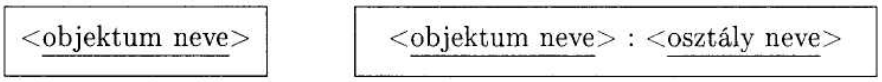
A láthatóság jelölése:

Gyakran azonban a modellalkotás során egyszerűbb formákat használunk a terv áttekinthető ábrázolásának érdekében. Ekkor az adott absztrakciós szinten nem érdekes részeket (műveletek, attribútumok) elhagyjuk:

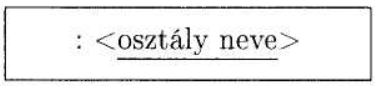


Tekintsük konkrét példaként a kerékpárok osztályát, ahol ismerjük az egyes kerékpárok színét, típusát és azonosítóját; a lehetséges műveletek pedig a kölcsönzés és a javítás. A háromféle jelölés erre a konkrét példára alkalmazva:

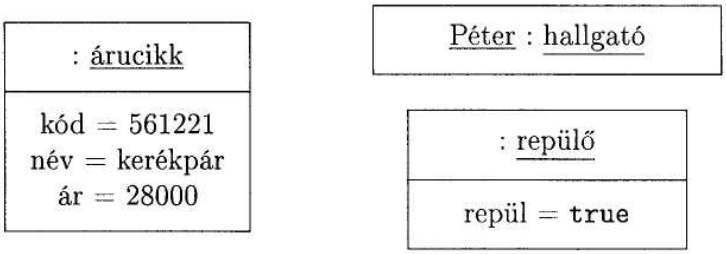
Az objektumok jelölése:



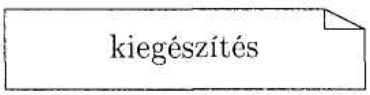
Az objektum neve sokszor nem elegendő az objektum azonosításához – több osztálynak lehet ugyanolyan nevű példánya –, vagy kifejezőbbé akarjuk tenni a diagramot az osztály explicit feltüntetésével. Ezekben az esetekben az ábra jobb oldalán látható jelölést használhatjuk.
Ha egy osztály tetszőleges objektumát szeretnénk jelölni, akkor:



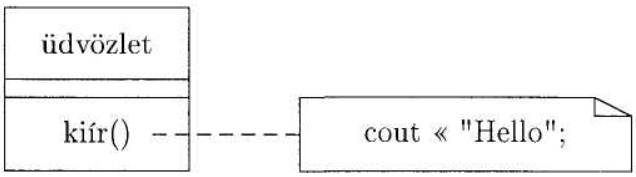
Ezen a konkrét példán látható, hogy az azonosító mellett megjelenhetnek az attribútumok értékei is:



A műveletek argumentumainak meghatározása, végrehajtásának tisztázása az implementáció kérdéskörébe tartozik. Ezt támogatja az UML-ben az *annotáció.* Az annotáció az UML nyelv szemantikai kiterjesztését teszi lehetővé. Jelölése:

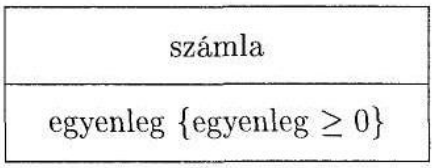


Egy lehetséges példa az operáció kiegészítése:



Itt a művelet jelentését procedurálisan, C++ nyelven adjuk meg az annotációban.
Ez valójában a C++-ban történő implementációhoz egy ajánlásként is értelmezhető.

Az attribútumok lehetséges értékeire tehetünk *megszorításokat.* Ekkor az attribútum mellé, kapcsos zárójelek közé írhatjuk a megszorítást kifejező feltételt. Például bizonyos számlák esetén a számla egyenlege nem lehet negatív:



Az UML diagramokban értelemszerűen máshol is elhelyezhetünk megszorításokat. A megszorítást kifejező feltételt minden esetben kapcsos zárójelek között kell megadni.

Osztálydiagram

A statikus modellben szereplő osztálydiagram fogalma:
Az osztálydiagram a problématérben a megoldás szerkezetét leíró, összefüggő gráf, amelynek

- csomópontjaihoz az osztályokat,
- éleihez pedig az osztályok közötti relációkat rendeljük.

(A gráf összefüggő, hiszen az osztályoknak kapcsolódniuk kell egymáshoz, különben független rendszert alkotnának. Két osztályt ugyanakkor több él is összeköthet, mert több reláció is lehet közöttük, illetve egy relációhoz több él is tartozhat.)

A relációk osztályok, illetve azok objektumai közötti kapcsolatot fejeznek ki. Az osztályok között a következő relációk állhatnak fenn:

- öröklődés,
- asszociáció,
- aggregáció,
- kompozíció.

Az öröklődés osztályok közötti kapcsolat, a másik három a résztvevő osztályok objektumait kapcsolja össze.

Objektumdiagram

A statikus modellben szereplő objektumdiagram fogalma:
Az objektumdiagram egyszeresen összefüggő gráf, amelynek

- csomópontjaihoz az objektumokat,
- éleihez pedig az objektumok közötti összekapcsolásokat rendeljük.

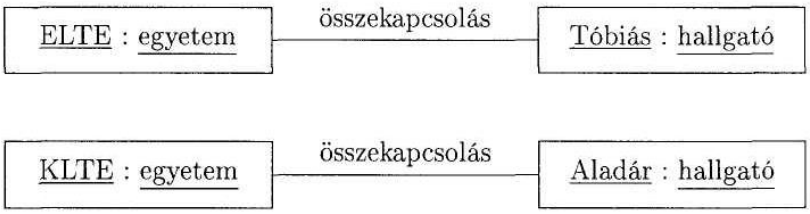
A rendszerhez egy osztálydiagram tartozik, ugyanakkor egy osztálydiagramhoz több objektumdiagram tartozhat. Mindegyik objektumdiagramnak összhangban kell lennie az osztálydiagrammal. A rendszer működése során dinamikusan jönnek létre, változnak és szűnnek meg objektumok, ezért az idő függvényében változhat az objektumdiagram. *Az osztálydiagram a rendszer egész idejére jellemző, az objektumdiagram egy pillanathoz köthető.* Az objektumdiagramban az osztályok helyébe azok példányai, az objektumok kerülnek. Az összekapcsolások az osztálydiagramban szereplő relációk példányai. Az objektumdiagram összekapcsolásai multiplicitás nélküliek, mert az osztálydiagramban szereplő relációk multiplicitásának megfelelő számú objektum jelenik meg az adott helyen. Az öröklődési reláció nem jelenik meg az objektumdiagramban, hiszen ebben a diagramban konkrét objektumok szerepelnek.

Osztályok közötti kapcsolatok

(Asszociáció, aggregáció, kompozíció és öröklődés.)

Társítási reláció, asszociáció

Ez a legáltalánosabb, „leglazább” reláció két osztály között. Az asszociáció két osztály közötti absztrakt reláció, amely kétirányú társítást fejez ki. A reláció absztrakt volta azt jelenti, hogy a reláció konkretizálása osztályok objektumainak összekapcsolásában valósul meg.
Konkrét esetben összekapcsolásról beszélünk (azaz az objektumdiagramban), míg absztrakt esetben társításról (vagyis az osztálydiagramban).
Konkrét, objektumok közötti összekapcsolás:



Osztályok közötti absztrakt társítás:

Fontos megállapítások az asszociációról:

1. Az asszociáció két vagy több osztály objektumainak valamilyen relációval történő *összekapcsolása.*

2. Az asszociáció lehet *reflexív,* azaz azonos osztályon belüli objektumok összekapcsolását is megengedi.

3. Az asszociációnak lehet neve, *azonosítója.*

4. Az összekapcsoláshoz *irány* is tartozhat, amely az aktív objektumtól a passzív objektum felé mutat, azaz a reláció értelmezésének irányát adja meg. Jele: ►.

5. Az asszociáció részleteinek leírása a hozzá *társult osztályban* kaphat helyet.

6. Az összekapcsolt objektumoknak lehet *multiplicitása* is:

- pontosan i, jele: i;
- i és j közötti, jele: $i..j$;
- 0 vagy több, azaz valamennyi, jele: $*$;
- legalább i, jele: $i..*$.

7. Az asszociációban résztvevő objektumnak lehet *szerepe* is:

• névvel azonosított szerep,

• kiemelt szerep,

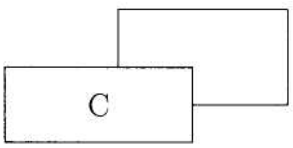
• sorrendiségi szerep.

8. Az asszociációhoz *minősítő* társulhat, amelynek értékei az osztály objektumait (a társítás szempontjából) diszjunkt partíciókhoz rendelik.

9. Az asszociáció esetén megadhatjuk a *navigálhatóságot.* Előfordulhat ugyanis, hogy a társított osztályok objektumai nem ismerik egymást kölcsönösen, csak az egyik osztály objektumai érhetik el a másik osztályba tartozó objektumokat. Ezt fejezhetjük ki ezzel a tulajdonsággal. Ha nem tüntetjük fel, akkor kölcsönös elérhetőséget tételezünk fel.

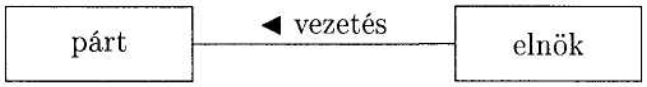
Az asszociáció jelölése:

Ezt a jelölést kiegészíthetik egyéb jellemzők, például multiplicitás.
A reflexív asszociáció jelölése:



Tekintsük például a pártokat, amelyeket egy osztályba szervezünk, és a pártok elnökeit, akiket egy másik osztályba szervezünk. A két osztály objektumait a *„vezetés”* reláció kapcsolja

össze. Feltesszük, hogy minden pártnak pontosan egy elnöke van, és egyvalaki csak egy pártnak az elnöke lehet. Az ennek megfelelő osztálydiagram:



A multiplicitás jelölése példákon keresztül:

A példában a *személy* és a *telefonszám* osztályokat kapcsoljuk össze asszociációval.

Minden személynek van telefonszáma, és pontosan egy van:

Mindenkinek van telefonszáma, de lehet olyan szám, amelyhez nem tartozik személy (pl.: cégé), és egy számhoz legfeljebb egy személy tartozik:

Mindenkinek van telefonszáma, és egy telefonszám több személynek is (akár 0) lehet a közös telefonszáma:

Minden telefonszám legalább *i* db személy közös tulajdona:

Minden telefonszám pontosan *i* db személy közös tulajdona:

Minden telefonszámhoz *i* és *j* közötti számú személy tartozik:

Minden telefonszám tartozik valakihez, és vagy van valakinek telefonszáma vagy nincs:

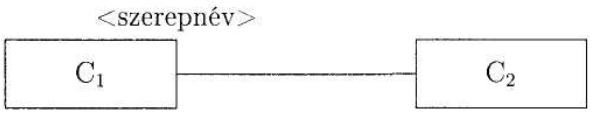
Minden telefonszám tartozik valakihez, és mindenkinek legalább 2 telefonszáma van:

Az objektum szerepének jelölése

Az asszociációval összekapcsolt osztályok objektumai különféle szerepeket tölthetnek be. A szereppel kapcsolatban az alábbi fogalmak jelölését kell bemutatni:

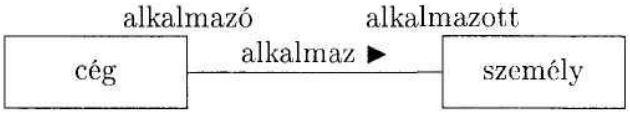
- a szerep megnevezése,
- kiemelt szerep,
- sorrendiségi szerep,
- több szerep megnevezése.

A szerep megnevezése:



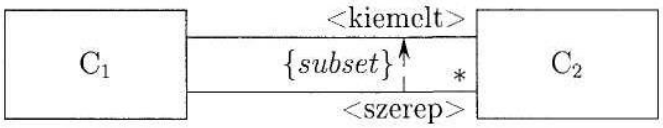
(A C_1 és C_2 osztályok közötti asszociációban C_1 objektumai a <szerepnév> szerepet töltik be.)

Tekintsük példaként a cégek és a személyek kapcsolatát. A két osztály a *cég* és a *személy,* amelyek asszociációban állnak egymással aszerint, hogy mely cég kit alkalmaz. Az *alkalmaz* relációban a cégek az *alkalmazó,* a személyek pedig az *alkalmazott* szerepet töltik be. A multiplicitástól tekintsünk most el. Ekkor a megfelelő diagram:



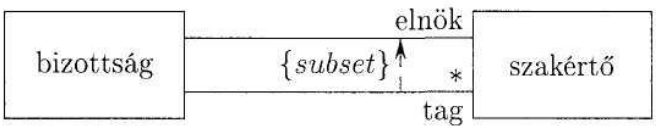
Kiemelt szerep

Ebben az esetben az osztály egy vagy több objektuma a relációban, a többitől eltérő módon, más, kiemelt szerepet is betölt. Ennek jelölése:



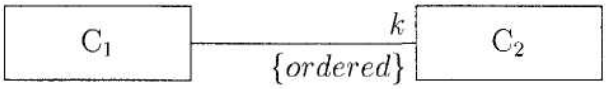
(A C_1 és C_2 osztályok között fennálló asszociációban C_2 objektumai a <szerep> szerepet töltik be, de van egy objektum, amely a <kiemelt> szerepet is betölti.)

Konkrét példaként tekintsük a szakértői bizottságokat. Ekkor a *bizottság*ok osztálya kapcsolatba hozható a *szakértő*k osztályával. A szakértők szerepe ebben a relációban a tagság, de van minden bizottságnak egy olyan tagja, aki egyben elnök is. Ezt szemlélteti az alábbi:

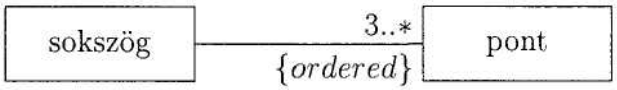


Sorrendiségi szerep

Ebben az esetben megadjuk, hogy hány objektum veszt részt a relációban, és az *ordered* alapszó feltüntetésével jelezzük, hogy ezek kötött sorrendben vesznek abban részt:

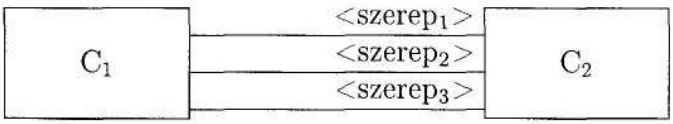


Vegyük példaként a *sokszög*eket és a *pont*okat. A két osztály objektumai közötti kapcsolat kifejezhető, ha a sokszög csúcsait valamilyen sorrendben (óramutató járásával ellenkező irányban) adjuk meg. Egy sokszöghöz legalább három csúcspont tartozik. Az ennek megfelelő osztálydiagram:

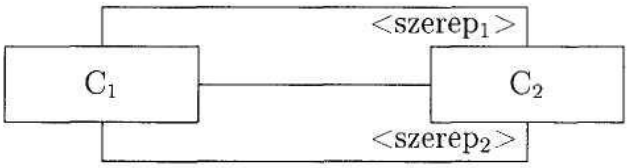


Több szerep megjelölése

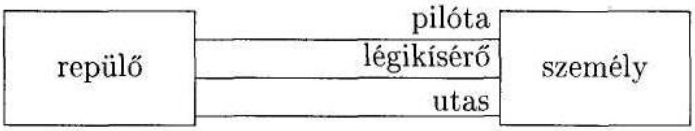
Ebben az esetben a két osztály közötti asszociációt több vonallal szemléltetjük, és az egyes vonalakra ráírjuk a megfelelő szerepeket:



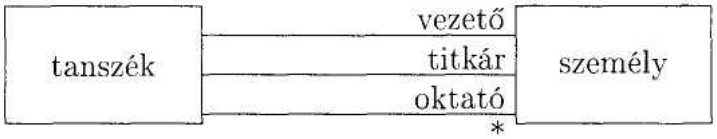
Ha az áttekinthetőség úgy kívánja, másik elrendezést is használhatunk:



Példánkban a C_1 osztálynak feleljen meg a *repülő*k osztálya, C_2-nek pedig a *személy*eké. A személyek többféleképpen is kapcsolatba hozhatók egy repülővel: lehet köztük pilóta, légikísérő vagy utas:



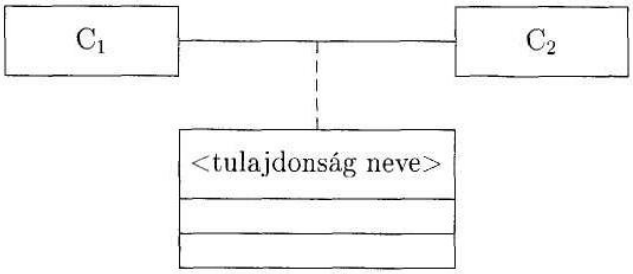
Egy másik példában vegyük az egyetemi tanszékeket. Egy *tanszék*nek egy vezetője és egy titkára van, továbbá több oktató dolgozik ott, de ezek mindegyike *személy*:



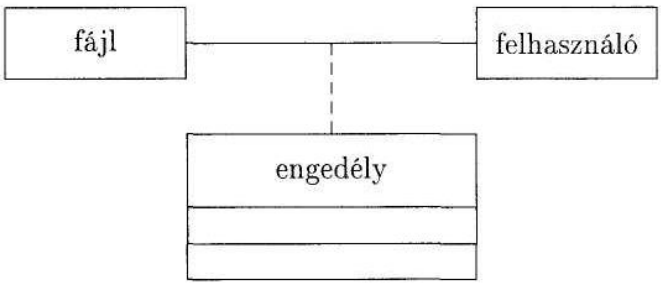
Az asszociációhoz kapcsolódó további jelölések

A reláció tulajdonságainak megadása

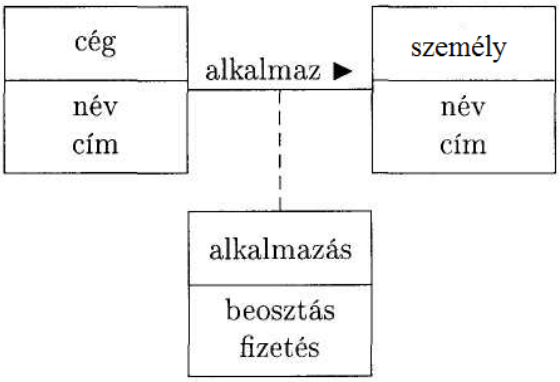
A reláció tulajdonságait, a relációra vonatkozó korlátozásokat rendszerint nem tudjuk elhelyezni a reláció mellett, noha rendkívül fontos lenne a feltüntetésük, jelölésük. Ezeket ilyenkor egy külön osztályban adjuk meg:



Példaként nézzük a *fájl*okat és a *felhasználó*kat. A két osztály objektumai közötti kapcsolat fontos jellemzője az engedély, amely a felhasználók hozzáférését szabályozza a fájlokhoz:

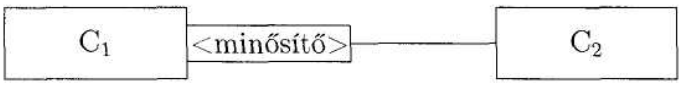


E külön osztály bizonyos attribútumait, amelyek a relációhoz köthetők, a reláció tulajdonságaként tüntethetjük fel egy osztály keretében. Erre példa a cégeknél dolgozó alkalmazottak alkalmazásukhoz köthető tulajdonságainak, adatainak (pl.: beosztás, fizetés) külön osztályban történő megadása:

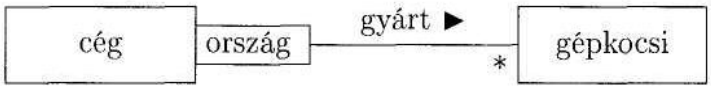


A reláció minősítése

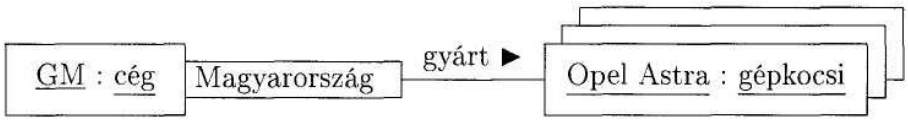
A minősítő konkrét értékei a relációban részt vevő konkrét objektumok egy példányát, vagy részhalmazát azonosítják. Az alkalmazott jelölés:



Példaként a multinacionális *cég*eket említjük meg, ahol a minősítő azt az országot azonosítja a relációban, amelyben a szóban forgó cégek működnek. Ilyen példa a *gépkocsi*k és multinacionális gyártóik közötti reláció:

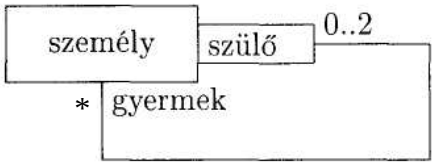


Egy konkrét esetnek megfelelő objektumdiagram:



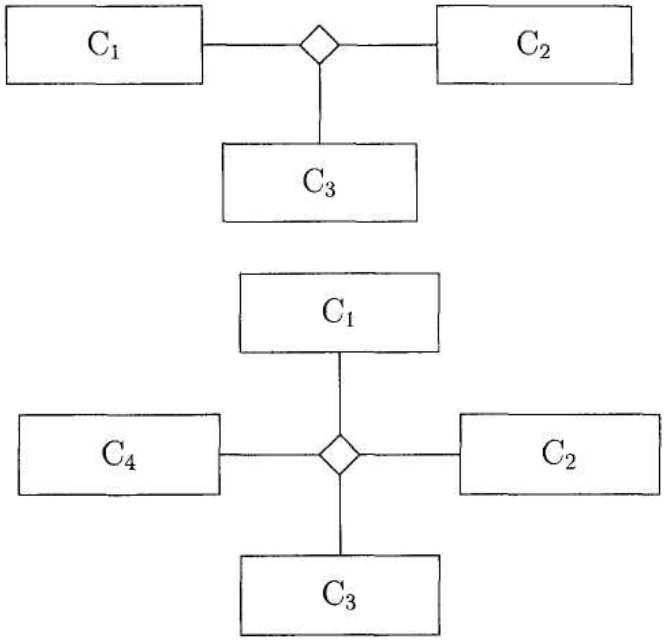
Reflexivitás

Korábban azt mondtuk, hogy az asszociáció lehet reflexív. Ennek szemléltetésére tekintsük a *személy*ek osztályát. (Élő személyekkel foglalkozunk.) Minden olyan személynek, aki szülő van vagy volt legalább 1 gyermeke, és minden személynek vannak szülei, akik nem feltétlen élnek:

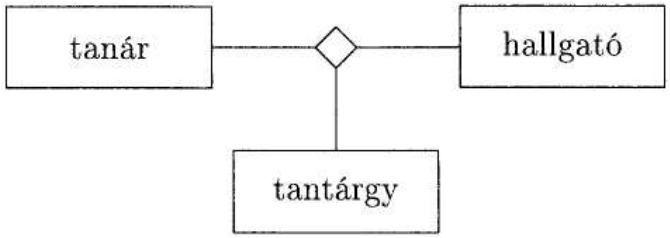


Több osztály között fennálló asszociáció

Az asszociáció nem feltétlen két osztályt kapcsol össze. Előfordulhat, hogy több osztályt hozunk kapcsolatba. Ekkor ezt a vonalak metszéspontjában elhelyezkedő rombusszal szemléltetjük:



Egy lehetséges példa három osztály közötti asszociációra a *tanár*ok, *hallgató*k és *tantárgy*ak között fennálló kapcsolat:

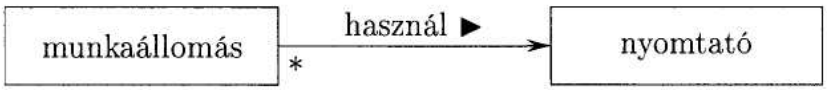


Navigálhatóság

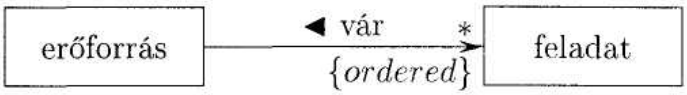
Említettük, hogy a társított osztályok objektumai nem feltétlen ismerik egymást kölcsönösen. Ha az ismeret csak egyirányú, akkor lehetőségünk van ennek kifejezésére, amivel az implementáció számára is útmutatást adunk:

(Ebben az esetben csak C_1 objektumai ismerik C_2 objektumait.)

Tegyük fel, hogy egy rendszerben több *munkaállomás*ról lehet egy *nyomtató*t használni. Ekkor a nyomtatónak nem feltétlen kell ismernie a munkaállomásokat, elég a munkaállomásoknak elérniük a nyomtatót ahhoz, hogy ki tudják nyomtatni a kívánt dokumentumokat. Ezt szemlélteti az alábbi ábra:

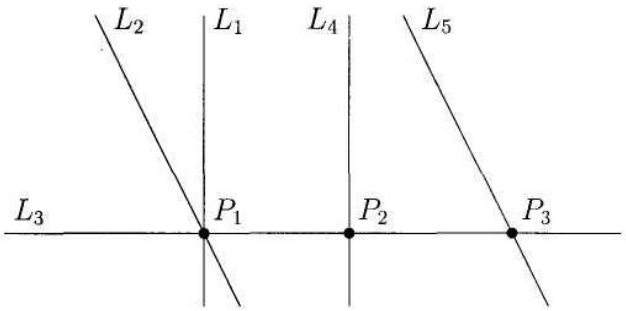


A navigálhatóság iránya nem feltétlen esik egybe a reláció irányával. Például egy *erőforrás*nál kiszolgálásra várakozhatnak *feladat*ok. Ha a reláció neve „vár”, akkor az irány a feladat felől mutat az erőforrás felé. Ugyanakkor elegendő az erőforrásnak ismernie a feladatokat, hiszen így mindig végre tudja hajtani a következőt:

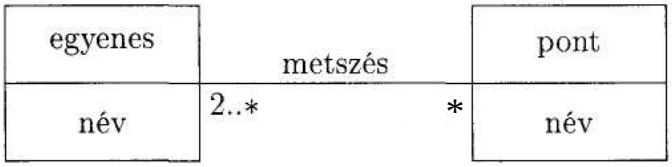


Példák asszociációra

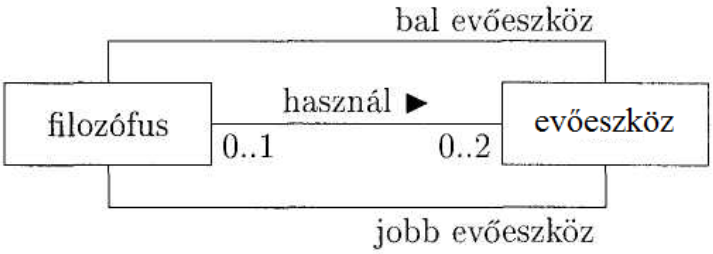
Az első példa síkbeli *egyenes*ek és metszés*pont*jaik kapcsolatát kifejező asszociáció. A síkbeli egyenesek lehetnek párhuzamosak, azaz nem metszik egymást; illetve metszik egymást: ekkor legalább két egyenesnek van ebben szerepe, és ez esetben van egy közös metszéspontjuk is.



Ezt felhasználva megadhatjuk az *egyenes*ek és a *pont*ok osztályát összekapcsoló reláció diagramját:



A második példa az étkező filozófusok problémája. Ez szavakban megfogalmazva a következő: *n* filozófus ül egy kör alakú asztal körül, és van *n* evőeszköz az asztalon, bármely két szomszédos filozófus között pontosan egy. Minden filozófus egy bizonyos ideig gondolkodik, ezután megpróbálja felvenni a tőle balra és jobbra eső evőeszközt, és ha ez sikerül, eszik, majd leteszi az evőeszközöket. A köztük fennálló reláció: a filozófus *használja* az evőeszközt. Vizsgáljuk meg a multiplicitás kérdését! Ezt megtehetjük, ha megvizsgáljuk a *filozófus* használatában levő *evőeszköz*ök számát a tevékenységei során:



Tehát a filozófus objektumok közül 0 vagy 1, az evőeszköz objektumok közül 0, 1 vagy 2 vesz részt a kapcsolatban.

Aggregáció

Az asszociáció az osztály objektumainak egymáshoz rendelését fejezi ki. Az egymáshoz rendelés különböző erősségű kapcsolódást jelenthet. Az asszociációt általában egymástól független osztályok társításának a kifejezésére használjuk. Az aggregáció ennél erősebb kapcsolat, amely olyan jellegű kapcsolatokat fejez ki, mint:

- egész és annak részei (valami valamiből áll),
- felépítmény és annak komponensei.

Fontos megállapítások az aggregációról:

1. Az aggregáció egy *speciális asszociáció.*

2. Az aggregációs reláció azt fejezi ki, hogy az egyik osztály objektumai részét képezik egy másik osztály objektumainak.

3. Az aggregáció *tranzitív:* ha **A** osztály aggregálja a **B** osztályt (azaz a **B** osztály objektumai részét képezik az **A** osztály objektumainak), **B** pedig aggregálja a **C** osztályt, akkor **A** aggregálja a **C** osztályt is.

5. Az aggregáció lehet *reflexív*.

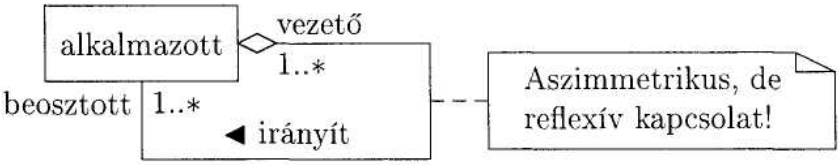
6. Ha **A** aggregálja **B**-t és *f* az **A** osztály egy szolgáltatása, akkor *f* a **B** osztálynak is egy szolgáltatása lesz.

7. Ha **A** aggregálja **B**-t és *attr* az **A** osztály egy attribútuma, akkor *attr* a **B** osztálynak is egy attribútuma lesz.

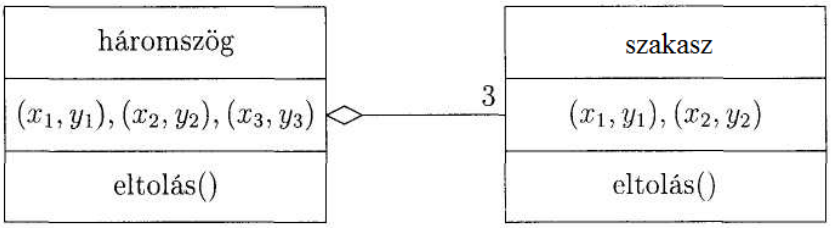
8. Ha **A** és **B** osztályok között aggregációs kapcsolat áll fenn, akkor az **A** osztály objektumai és a **B** osztály objektumai *egymástól függetlenül is létezhetnek.*

Az aggregáció jelölése (**A** aggregálja **B**-t):

Példa reflexív aggregációra:

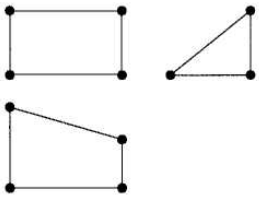


Példa közös attribútumra és szolgáltatásra aggregáció esetén:

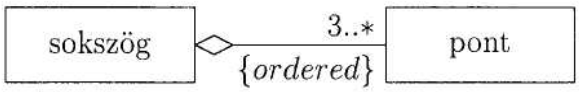


Aggregációra példaként tekintsük az írott szövegeket. Ezek bekezdésekből állnak, a bekezdések pedig mondatokból. Így három osztályunk lesz: a *szöveg*, a *bekezdés* és a *mondat*, amelyekből a fentiek alapján az osztálydiagram:

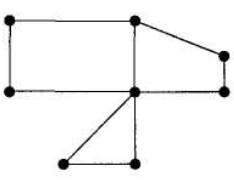
A következő példában vizsgáljuk meg a síkbeli *sokszög*ek és a sík *pont*jainak viszonyát. A sokszögek csúcsai pontok – méghozzá megfelelő sorrendben, ahogy az asszociáció tárgyalásakor már láttuk –, így a két osztály között aggregációs kapcsolat áll fenn. Ha feltesszük, hogy egy síkbeli pont csak egy sokszöghöz tartozhat, és csak ilyen pontokat veszünk figyelembe,



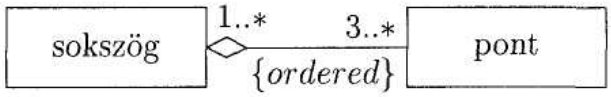
akkor az osztálydiagram:



Hagyjuk el az előbbi feltevést, azaz most már engedjük meg, hogy egy pont akárhány sokszöghöz tartozzon. Továbbra is csak olyan pontot vegyünk figyelembe, amelyik legalább egy sokszög csúcspontja:



Ekkor az osztálydiagram:



Kompozíció

Az aggregációs társítások között a legerősebb kapcsolat a kompozíciós kapcsolat.

Fontos megállapítások a kompozícióról:

1. A kompozíció egy *speciális aggregáció.*

2. A kompozíciós kapcsolat azt fejezi ki, hogy az egyik osztály objektumai a másik osztály objektumait *fizikailag tartalmazzák.*

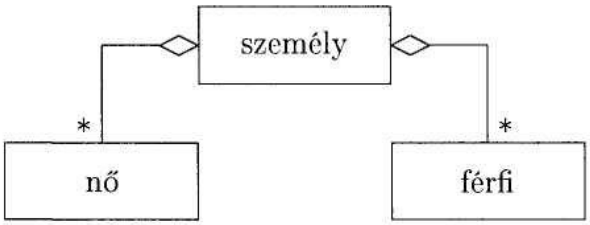
3. Egy komponens objektum legfeljebb csak egy kompozíciós objektumhoz tartozhat.

4. A kompozíciónak tetszőleges számú komponense lehet.

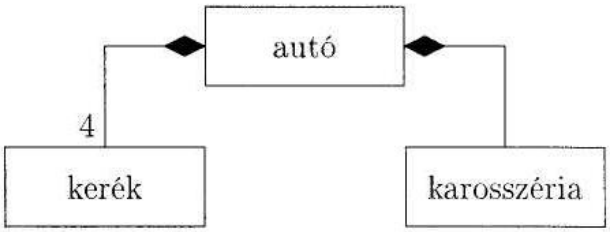
5. A kompozíciós objektum és annak komponensei azonos életciklusban léteznek, azaz egyszerre jönnek létre, és egyszerre szűnnek meg.

A kompozíció jelölése:

Vizsgáljuk meg egy-egy példán keresztül az aggregáció és a kompozíció közötti különbséget. Elsőként tekintsük a *személy*ek, illetve a *nő*k és *férfi*ak osztályát. Azt mondhatjuk, hogy aggregációs kapcsolat áll fenn a személyek és nők, illetve a személyek és férfiak között, hiszen minden nő és férfi egyben személy is. Ugyanakkor ebben az esetben fizikai tartalmazás nem áll fenn, tehát kompozícióról nem lehet szó:

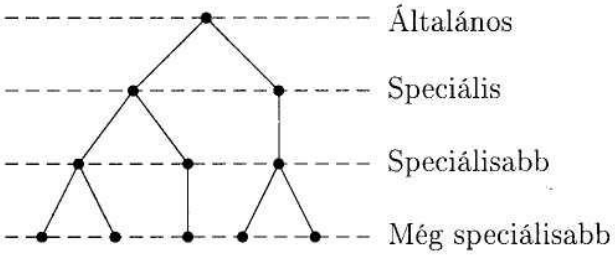


Másodikként foglalkozzunk az *autó*k, *kerek*ek és *karosszériá*k osztályával.
Tudjuk, hogy egy autónak négy kereke és egy karosszériája van, továbbá ez fizikai tartalmazást is jelent. Az autó ezek nélkül nem is létezhet, így a kompozíció minden feltétele megvan. Ebben az esetben tehát a kompozíció a megfelelő reláció:



Általánosítás és specializáció (öröklődés)

Az általánosítás a modellalkotásban az általános tulajdonságokkal rendelkező dolog (superclass, ősosztály) és a kevésbé általános, speciálisabb dolog (subclass, származtatott osztály) között fennálló reláció. Először létrehozzuk az általános tulajdonságokkal felruházott osztályt, majd annak a tulajdonságait átvéve származtatjuk a speciálisabb tulajdonságokkal rendelkező osztályt. Ezt az eljárást tovább folytatva az osztályokat egy hierarchikus szerkezetbe rendezzük. Többszörös öröklődést is megengedve az osztályokat egy fa struktúrába rendezhetjük:



Fontos megállapítások az öröklődésről:

1. A reláció egy általános osztály és egy speciális osztály közötti kapcsolatot fejez ki.

2. A reláció azt fejezi ki, hogy a speciális osztály az általános osztályból származtatással jön létre.

3. A származtatásnál a származtatott speciális osztály:

   • átveszi az általános osztály tulajdonságait (név, attribútum, operáció, asszociáció);

   • az átvett jellemzőkhöz, attribútumokhoz, operációkhoz stb. bővítésként új jellemzőket, operációkat stb. vezethet be;

   • az átvett jellemzőket újrafogalmazhatja.

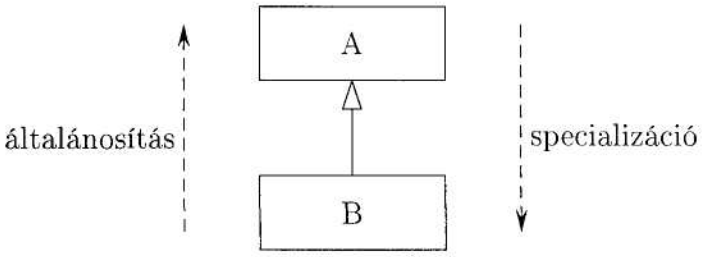
4. A származtatás nem szimmetrikus.

5. A származtatás nem lehet reflexív.

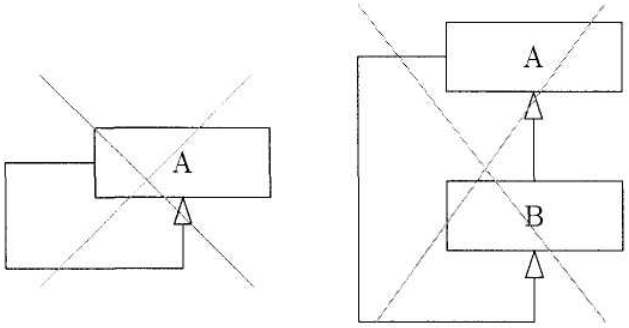
6. A specializáció lehet többszörös, ekkor egy általánosításból több származtatott jön létre.

7. Az általánosítás is lehet többszörös, amikor a származtatás több általánosításból történik.

Az általánosítás és a specializáció (öröklődés) jelölése:

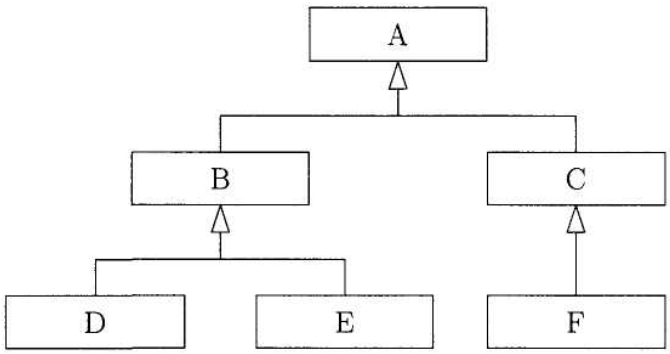


Az általánosítás, illetve a specializáció nem lehet sem reflexív, sem szimmetrikus:

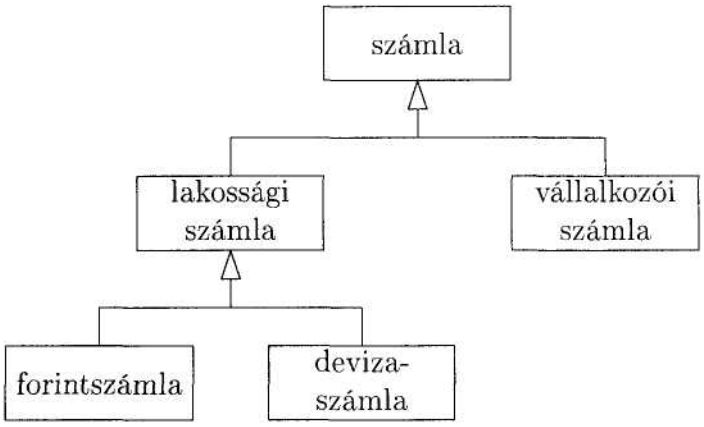


Többszörös specializáció

Ebben az esetben az absztrakciós szintek hierarchikus szerkezetét hozzuk létre, az osztályokat fa struktúrába szervezve:

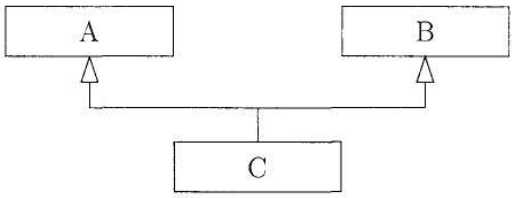


Példaként tekintsük a bankszámlákat, röviden *számlá*kat. Ezeket tovább lehet bontani *lakossági* és *vállalkozói számlá*kra. A lakossági számlákon belül megkülönböztethetünk *forint-* és *devizaszámlá*kat:

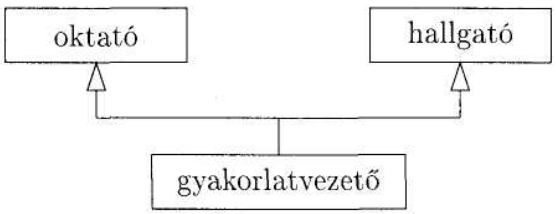


Többszörös általánosítás

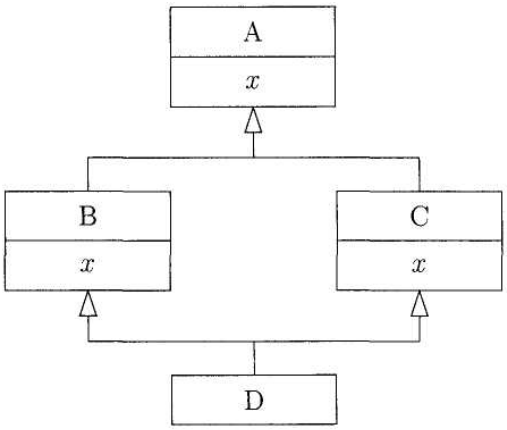
Ebben az esetben egy osztály több osztály tulajdonságaival rendelkezik, ennek megfelelően a diagramokban ez „fordított fa”-ként jelenik meg:



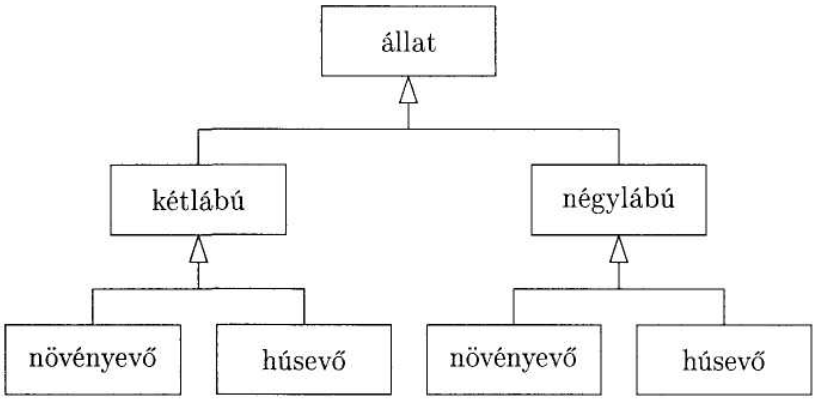
Számos egyetemen az oktatók is és a hallgatók is vezetnek gyakorlatokat. A *gyakorlatvezető* tehát rendelkezik mind az *oktató*k, mind a *hallgató*k bizonyos tulajdonságaival:



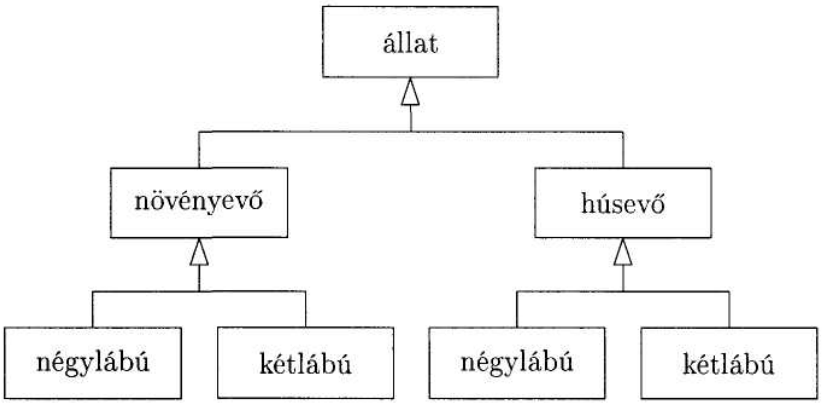
A többszörös specializáció és általánosítás együttes használata kerülendő, mert az átvett információk keveredését eredményezheti. Ez az úgynevezett *kovariancia problémája,* azaz a különböző utakon származtatott tulajdonságok kezelésének problémája. A probléma szemléltetésére tekintsük a következő osztálydiagramot:



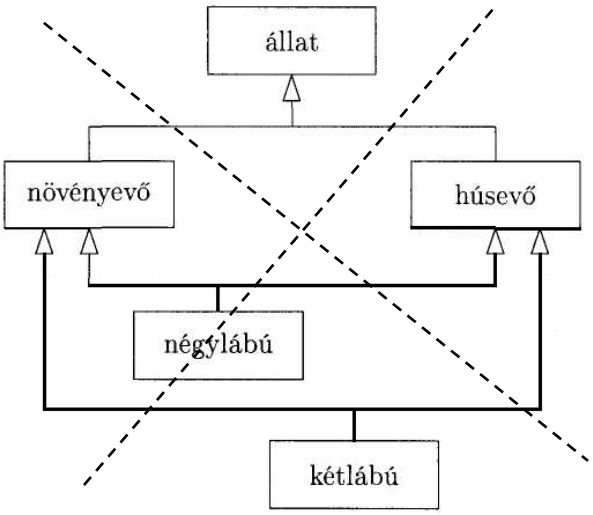
A **D** osztály egy *d* objektuma rendelkezik az *x* attribútummal. Ugyanakkor *d.x* jelentheti az **A**, a **B** vagy a **C** osztályhoz tartozó attribútumot, ebben az esetben ez nem egyértelmű.
A probléma megoldására jó példa az állatok osztályozása:



vagy



de pl. nem

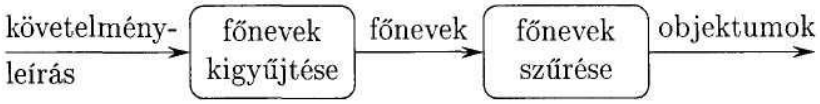


Osztálydiagramok készítése

Eddig megismertük az osztályok és a köztük lévő relációk jelöléseit. Felmerül a kérdés: ezek után hogyan lehet egy problémához tartozó osztálydiagramot megalkotni a követelmények elemzése alapján. Ezzel fogunk most foglalkozni feladatok megoldásán keresztül.

Az osztálydiagram megalkotásának lépései:

1. A követelmények elemzése alapján meghatározzuk azokat a főneveket, amelyek potenciálisan objektumjelöltek lehetnek.

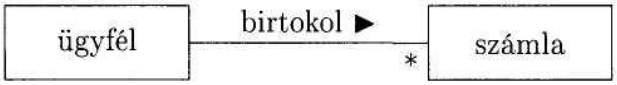


2. Objektumok leírása, tulajdonság jellemzőik meghatározása. Ez tartalmazza a tulajdonságok, műveletek és relációk felderítését.
Például a banki számlák esetén:

- Tulajdonságok: tulajdonos azonosítója, számla típusa (folyószámla, lekötött számla), ...
- Műveletek: felújítás, lekötés, ...
- Relációk: a számlát bankban tárolják, egy ügyfélnek több számlája is lehet, ...

3. Objektumok osztályba sorolása hasonló tulajdonságaik alapján.

4. Az osztályok közötti relációk meghatározása. Például a számla és az ügyfél között asszociációs kapcsolat áll fenn, amelynek neve „birtokol”.



5. Kezdeti osztálydiagram megszerkesztése.

6. Az objektumok attribútumainak, a relációk tulajdonságainak, szerepeknek, multiplicitásoknak, irányoknak, navigálhatóságoknak stb. a meghatározása.

7. Általánosítás és specializáció (öröklődés) segítségével az osztálydiagram hierarchikus szerkezetének kialakítása, áttekinthetőségének növelése, egyszerűsítése.

8. Osztályleírások elkészítése.

Példa:
Készítsük el az irányított gráf osztálydiagramját a következő leírás alapján. Az irányított gráf csomópontokból és élekből áll. Az élek csomópontból csomópontba mutatnak, egy csomópontba több él is mutathat be, és egy csomópontból több él is mutathat ki. A bemenő él egy csomópontba mutat be, a kimenő él pedig egy csomópontból mutat ki. Egy pont is lehet gráf. A gráf síkbeli alakzat, ahol az élet két végpontjának koordinátáival (első honnan, második hova mutat), a csomópontot annak síkbeli koordinátáival adjuk meg.

Megoldás:
A leírás első mondata alapján három osztályt határozhatunk meg:

- irányított gráf,
- él,
- csomópont.

Ugyancsak az első mondatból derül ki az irányított gráf társításának jellege az élekhez, illetve a csomópontokhoz. Az utolsó mondat megadja a csomópont és az él attribútumait. A köztes mondatok megadják a csomópontok és az élek közti relációt, a hozzá tartozó szerepekkel, illetve tisztázzák a multiplicitások kérdését. Összefoglalva:

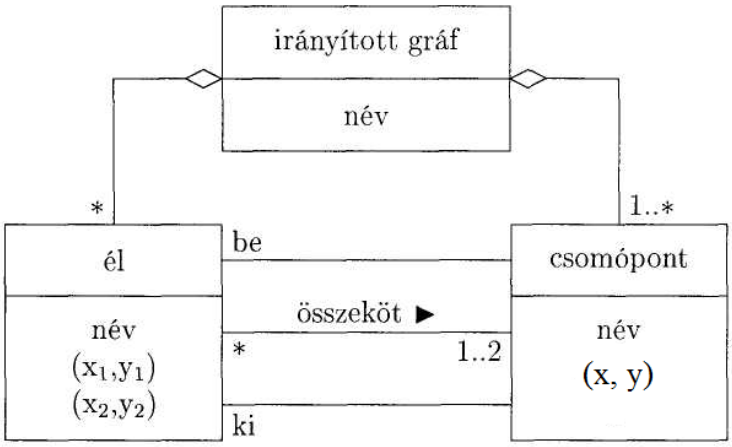
Reláció: él csomópontokat *köt össze.*

Szerep: csomópontba *be* mutat az él, illetve csomópontból *ki* mutat az él.

Multiplicitások: a bemenő és a kimenő élhez egy-egy csomópont tartozik. A hurok élek egy pontot kötnek össze önmagával. Egy izolált ponthoz nem tartozik él. Az irányított gráfnak tetszőleges számú éle és minimum 1 csomópontja van.

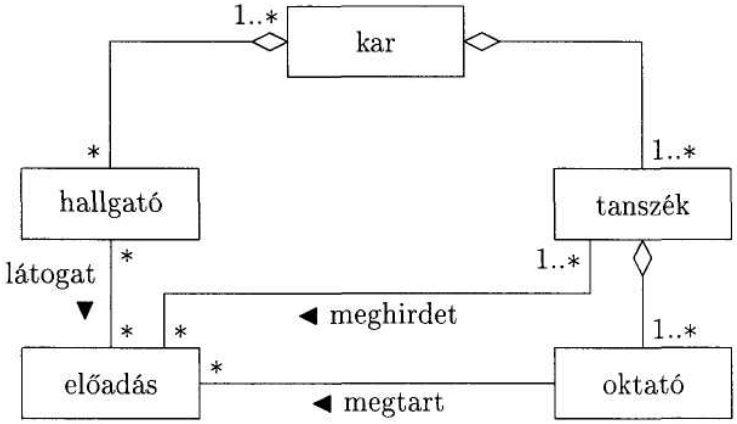
Attribútumok: név (irányított gráf esetén), illetve név és koordináták (él és csomópont esetén).

Ennek alapján egy lehetséges megoldás:



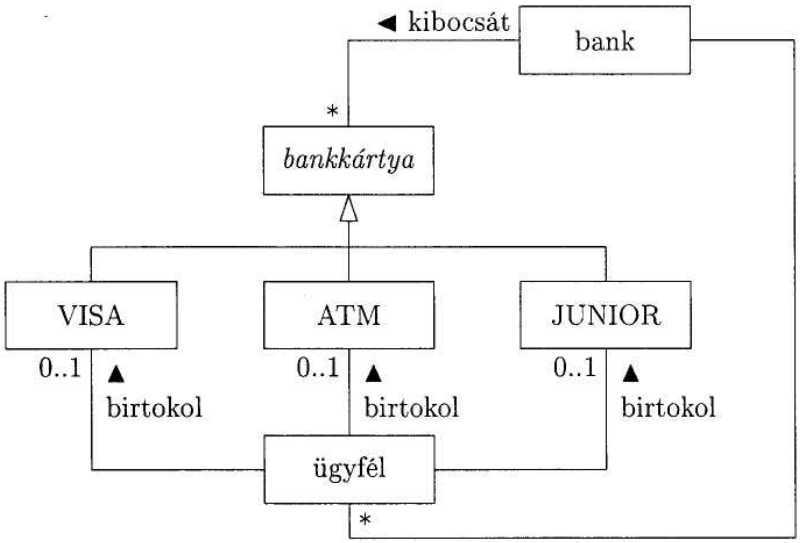
Példa:
Készítsük el az osztálydiagramot a következő leírás alapján. A kar tanszékekből és hallgatókból áll. Az oktatók alkotják a tanszékeket. Egy tanszék csak egy karhoz tartozhat. Minden tanszéknek van legalább egy oktatója. A hallgató több karnak is lehet hallgatója. Az oktató csak egy tanszéknek lehet az oktatója. A tanszékek hirdetik meg az előadásokat, de az oktatók tartják meg azokat. A hallgatók ezeket az előadásokat látogatják. Ugyanazt az előadást több tanszék is meghirdetheti.

Megoldás:
Az osztályok: kar, hallgató, tanszék, oktató, előadás.
A relációk és a multiplicitások adódnak a feladat szövegéből.
Az osztálydiagram:



Példa:
Készítsük el az osztálydiagramot a következő szöveg alapján. A bank bankkártyákat bocsát ki. A bank által kibocsátott bankkártyák háromfélék: VISA, ATM, JUNIOR. Minden ügyfélnek bármelyik kártyából lehet egy kártya a tulajdonában.

Megoldás:
A leírás alapján a bankkártya egy absztrakt osztály, amelynek konkrét formái a VISA, ATM, JUNIOR osztályok objektumai. A banknak a bankkártyákkal és az ügyfelekkel van asszociációs kapcsolata, végül a multiplicitás a szöveg alapján egyértelmű:

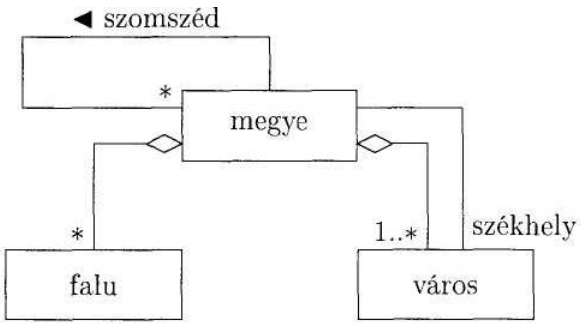


Példa:
Készítsük el az osztálydiagramot a következő leírás alapján. A faluk és városok megyéket alkotnak. A megyének van székhelye, amely város. A feldolgozás szempontjából fontos, hogy mely megyék szomszédosak.

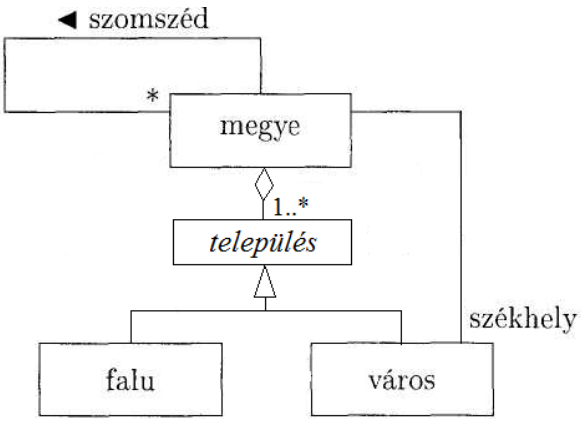
Megoldás:

- A megyék tehát falukból és városokból állnak.
- A városok között van egy, amelynek az a szerepe, hogy székhely.
- Végül a megyék között fontos a szomszédsági reláció.

A fenti elemzés alapján az osztálydiagram:



A feladatban nem szerepelt, de be lehet vezetni a települések osztályát, amely egy absztrakt osztály lenne, és ebből származtatnánk specializációval a faluk és a városok osztályát. A diagram ilyen értelmű módosítása:



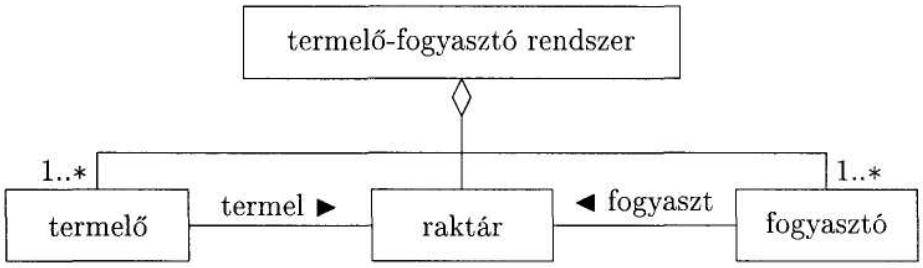
Példa:
Készítsük el a termelő-fogyasztó rendszer osztálydiagramját a következő leírás alapján. A termelő-fogyasztó rendszer egy raktárból, több termelő folyamatból és több fogyasztó folyamatból áll. A raktárba azonban egyszerre csak egy termelő helyezhet el árut, és a raktárból egyszerre csak egy fogyasztó szállíthat ki árut.

Megoldás:
A szöveg elemzése alapján az *osztályok:* termelő-fogyasztó rendszer, raktár, termelő, fogyasztó.

A *relációk:*

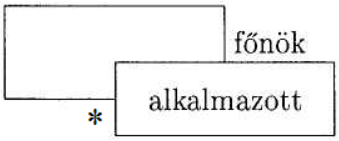
- A termelő-fogyasztó rendszer a raktár, a termelő és a fogyasztó objektumok aggregátuma. A fogyasztó és a termelő multiplicitása többszörös.
- A termelés és a fogyasztás asszociációk, melyek a termelőt és a fogyasztót a raktárral kötik össze. A multiplicitás itt mind a két esetben egyszeres.

Így jutunk el az osztálydiagramhoz:



Példa:
Rajzoljuk fel az osztálydiagramot a következő mondat alapján. Az alkalmazottak között van egy, aki a főnök.

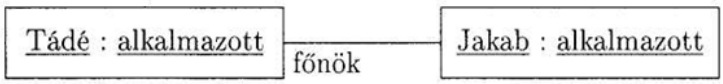
Megoldás:



Példa:

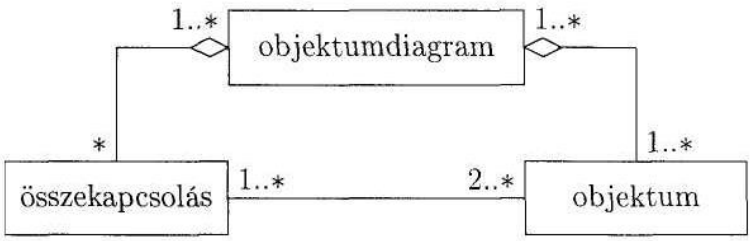
Tegyük fel, hogy *Tádé* a főnök és *Jakab* egy alkalmazott. Rajzoljuk fel a fenti osztálydiagram alapján azt az objektumdiagramot, amely ezekre az objektumokra vonatkozik.

Megoldás:

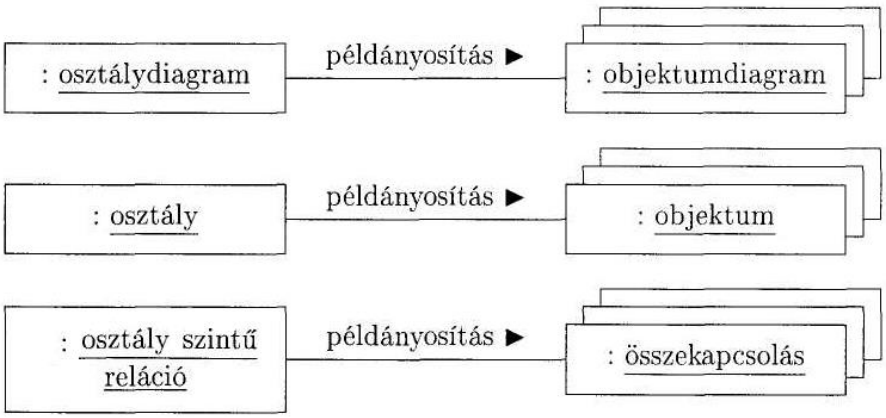


Az osztálydiagramok és az objektumdiagramok kapcsolata

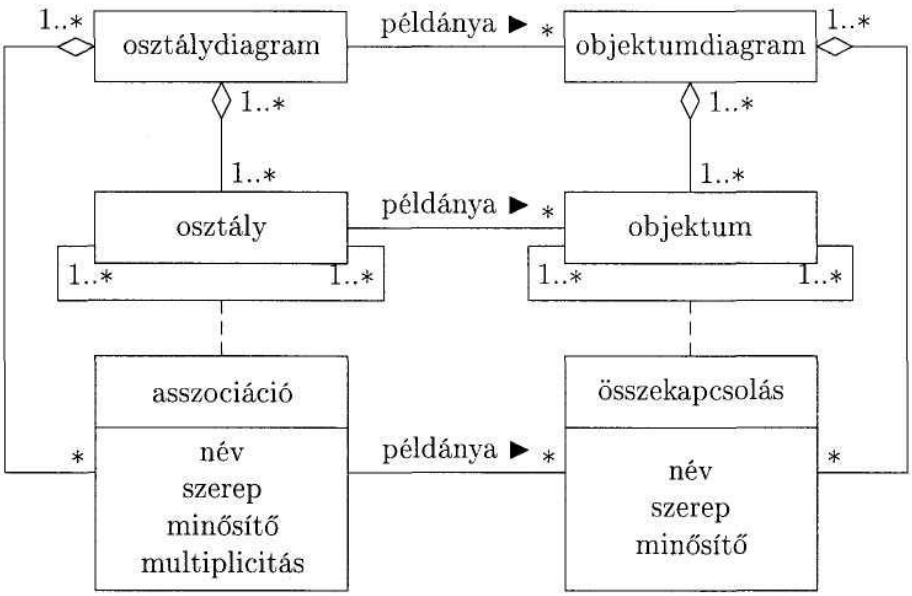
Az objektumdiagram összetevőit és kapcsolataikat leírhatjuk az UML segítségével is:



csakúgy, mint az osztálydiagramból példányosítással történő előállítást is:



Egy osztálydiagramhoz több objektumdiagram tartozhat. Ennek során az osztály helyébe annak példányai, az objektumok; a társítás helyébe pedig a társítás példányai, az összekapcsolások lépnek. Az összekapcsolás átveszi a megfelelő társítás tulajdonságait. Ennek alapján az osztálydiagram és az objektumdiagram kapcsolatának osztálydiagramja:

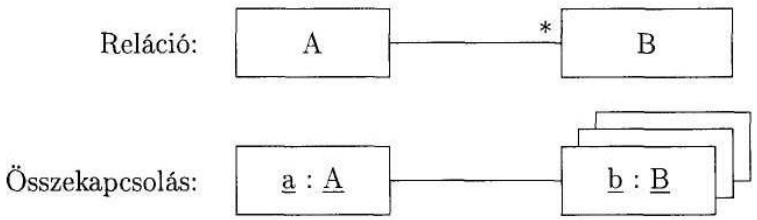


Az osztályszintű relációk példányosításával kapcsolatos tudnivalók:

• A reláció azonosítóját az összekapcsolás átveszi.

• A reláció multiplicitásának értékével megegyező számú objektum jelenik meg az összekapcsolásban.

• Ha a reláció multiplicitásának értéke tetszőleges érték, akkor az összekapcsolásnál ez a következő módon jelenik meg:



• A reláció irányának megfelelően vesznek részt az objektumok aktív, illetve passzív szereplőként az összekapcsolásban. Tehát az irányt az összekapcsolás a relációtól szemantikailag és jelölésben is átveszi.

• A reláció speciális esetének tulajdonságát az összekapcsolás szemantikailag és jelölésben is átveszi (aggregáció, kompozíció).

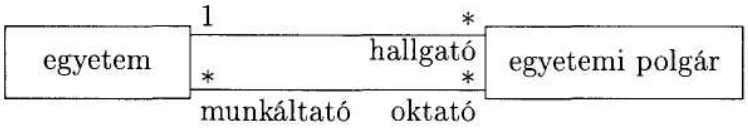
• A reláció „ordered” tulajdonsága az összekapcsolásban részt vevő objektumok sorszámmal kifejezett szerepében nyilvánul meg; az „első” sorszám tetszőlegesen választható meg.

• Minősítéssel ellátott reláció esetén a minősítőnek az adott objektum szempontjából éppen aktuális értékét kapja meg az összekapcsolás (azaz az objektumdiagramban is megjelenik a minősítő).

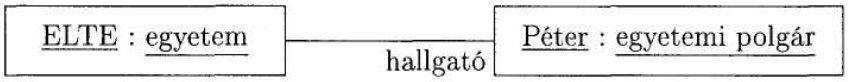
• Az öröklődési reláció az objektumdiagramban nem jelenik meg.

• Megjelennek viszont a konkrét objektumok mellett a szerepnevek.

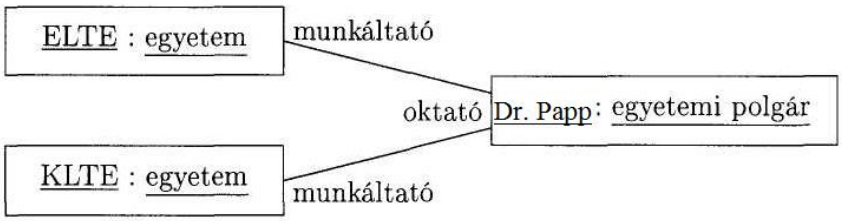
Példa:
Első példaként tekintsük az egyetem és az egyetemi polgárok osztályát. Az egyetemi polgár lehet hallgató vagy oktató. Egy egyetemen sok hallgató és sok oktató lehet. Tegyük fel, hogy egy hallgató csak egy egyetemre járhat, egy oktató pedig több egyetemen is oktathat, és ideiglenesen lehet, hogy akár egy egyetemen sem oktat. Az egyetem az oktatók munkáltatója. Ennek a leírásnak megfelelő osztálydiagram:



Ha Péter az ELTE hallgatója, akkor ebből az objektumdiagram:

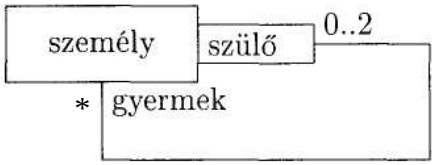


Ha pedig Dr. Papp az ELTE és KLTE oktatója, akkor az ennek megfelelő objektumdiagram:

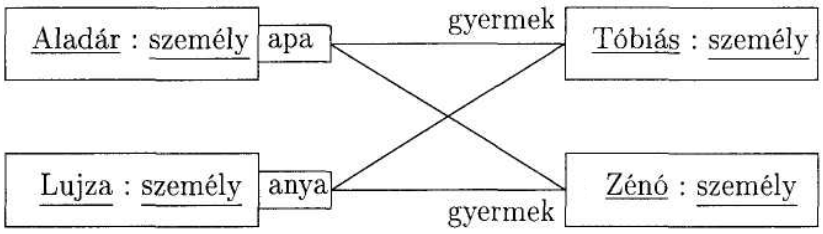


A szerep tehát megjelenik az objektumdiagramban is; a multiplicitás pedig konkretizálódik, azaz annyi objektum jelenik meg az összekapcsolásnál, amennyi a multiplicitás konkrét értéke az adott esetben.

Példa:
A következő példa reflexív asszociáció esetén mutatja meg az osztálydiagram és az objektumdiagram viszonyát. Az osztály legyen az élő személyek osztálya, a reflexív kapcsolat pedig a szülő-gyermek viszony. Egy szülőnek lehet akármennyi (élő) gyermeke, és legfeljebb két élő szülője, akik közül az egyik anya, a másik apa (mint a szülő minősítő lehetséges értékei):

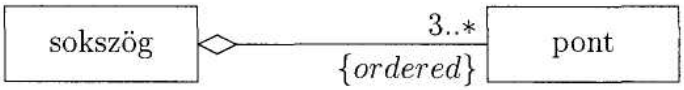


Legyen Aladár és Lujza apja, illetve anyja Tóbiásnak és Zénónak. Ekkor az objektumdiagram:

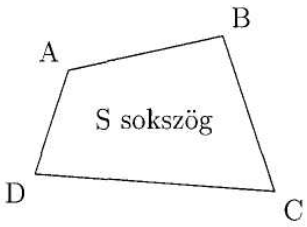


Ez a példa szemlélteti, hogy a minősítő konkrét értéke is megjelenik az objektumdiagramban a konkrét objektum mellett.

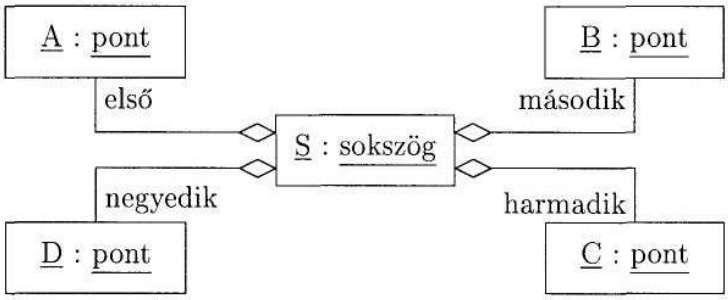
Példa:
A következő példában egy olyan esetet vizsgálunk az osztálydiagram és a konkrét objektumdiagram viszonyára, amikor sorrendiségi szerep is előfordul az osztálydiagramban. Tekintsük a sokszögek és a csúcspontjaik osztályát:



Egy konkrét *S* sokszög:

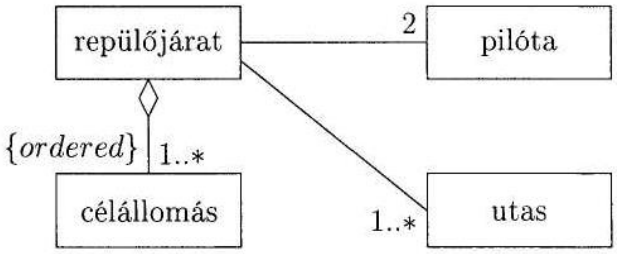


E konkrét sokszög esetén az adott sorrend szerint külön-külön vesszük fel az objektumokat a diagramba:

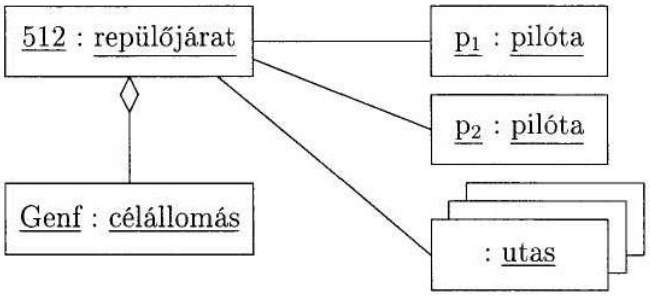


Valójában itt csak az a lényeg, hogy az objektumok milyen sorrendben vesznek részt az aggregátum felépítésében, függetlenül attól, hogy melyik közülük az első sorszámmal jelölt. Az „ordered” szerep tehát egy önkényesen indított sorrendnek megfelelő sorszámozást jelent az objektumdiagramban.

Példa:
Tekintsük a repülőjáratok egy lehetséges modelljét:

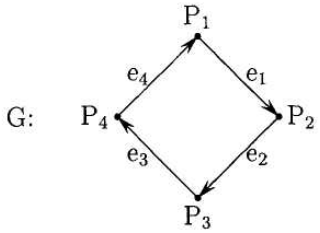


Ebben egy repülőjáratot pontosan két pilóta vezet, legalább egy utas használja, és van valahány célállomása. A célállomások sorrendje lényeges. Egy ehhez tartozó lehetséges objektumdiagram:

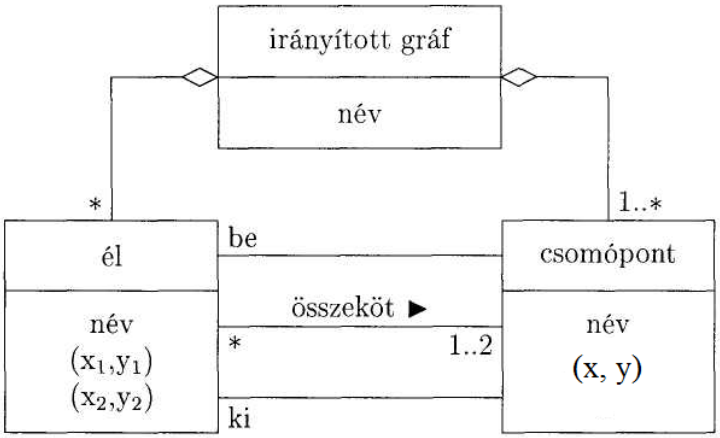


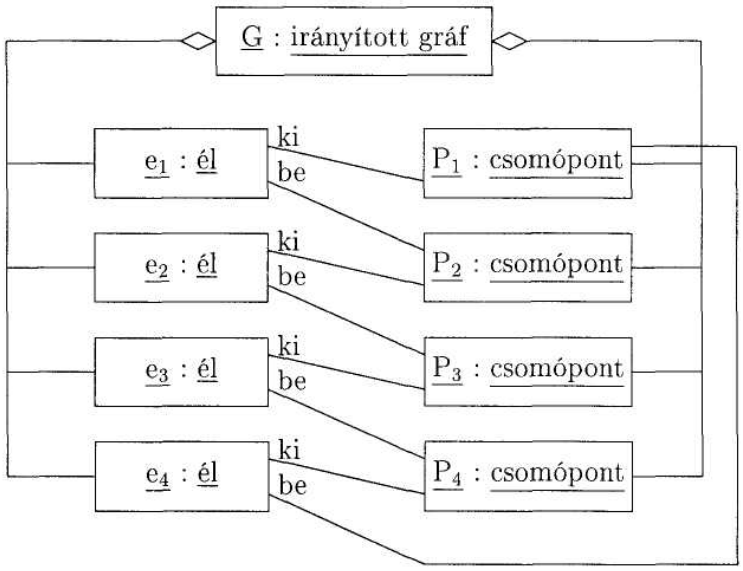
Ebben csak egyetlen célállomás szerepel, ezért a sorrendiséggel nem kell foglalkozni.

Példa:
Tekintsük az alábbi konkrét irányított gráfot:



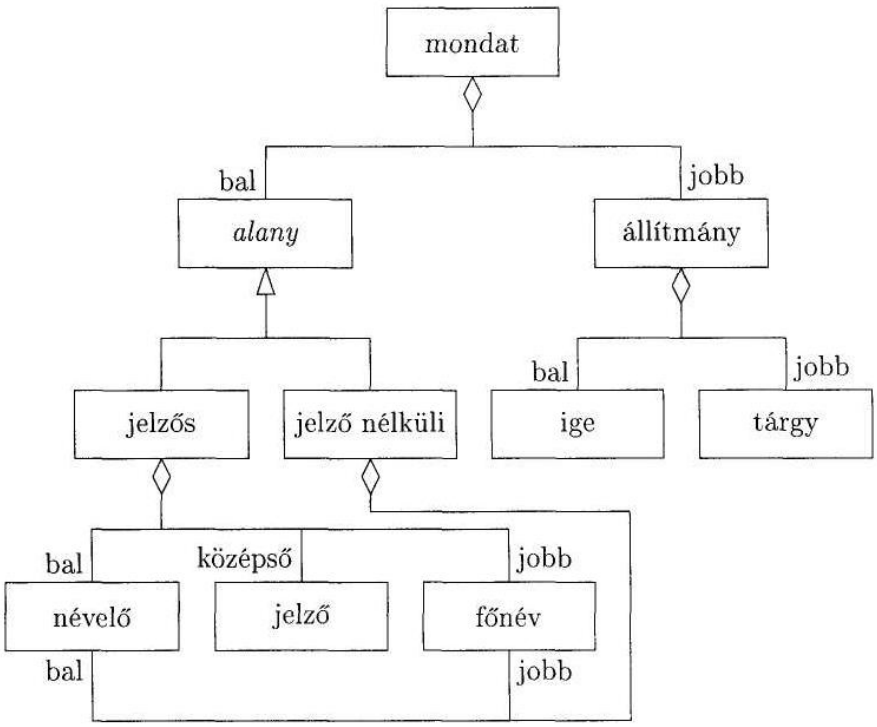
Készítsük el ennek objektumdiagramját osztálydiagramja alapján:





Példa:
Készítsük el az osztálydiagramot a következő leírás alapján. A mondat alanyból és állítmányból áll, ahol az alany az állítmányt megelőzi. Az alany lehet névelővel ellátott, jelzős vagy jelző nélküli főnév. A névelő a főnév előtt áll. Az állítmány igéből és az utána álló tárgyból áll.

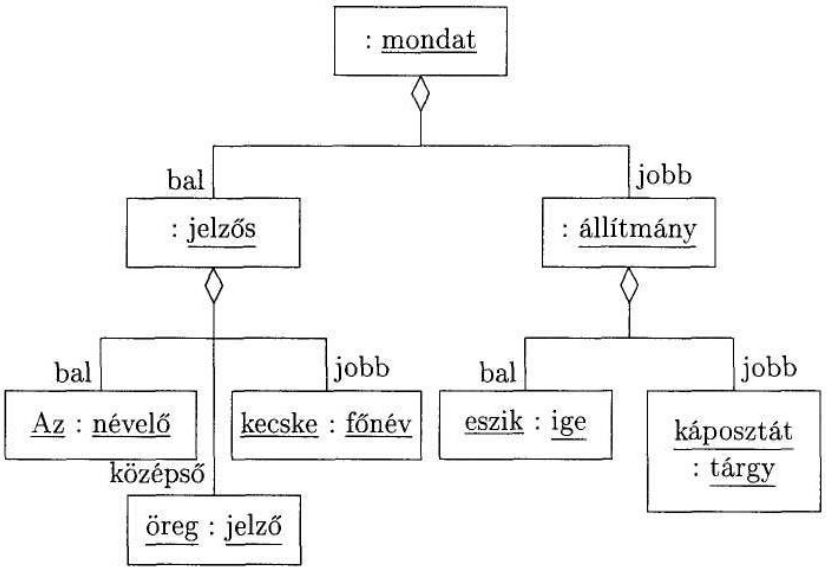
A szöveg elemzése alapján az objektumosztályok: mondat, alany, állítmány, jelzős (alany), jelző nélküli (alany), névelő, jelző, főnév, ige, tárgy. Így az osztálydiagram:



Készítsük el a következő mondat objektumdiagramját: **Az öreg kecske eszik káposztát.**
Először azonosítsuk a mondat alkotóelemeit:

```
Az       : névelő;
öreg     : jelző;
kecske   : főnév;
eszik    : ige;
káposztát : tárgy.
```

Ezt felhasználva az objektumdiagram (figyeljük meg, hogy az absztrakt osztály természetesen nem jelenik meg az objektumdiagramban):



Példa:
Készítsük el az osztálydiagramot a következő leírás alapján. A program blokkokból áll. A blokk utasítások egy véges sorozata. Az utasítás lehet értékadás, feltételes elágazás vagy ciklus. A ciklus egy logikai kifejezésből és az utána álló blokkból áll. Az elágazás egy logikai kifejezésből és az azt követő értékadásból, valamint az értékadást követő blokkból áll. A logikai kifejezéshez és az értékadáshoz változókat és konstansokat használunk fel.

A szöveg alapján a következő osztályokat azonosíthatjuk:
program, blokk, utasítás, feltételes elágazás, értékadás, ciklus, logikai kifejezés, változó, konstans.

Relációk:

- A program és a blokk között aggregáció lelhető fel a szöveg szerint.
- Hasonlóképpen, mivel a blokkok utasításokból állnak, így a blokk és az utasítás között is aggregációs viszony áll fenn.
- Az utasítás egy általánosítás – absztrakt osztály –, amelynek konkrét megjelenési formái a felsorolt utasítások.
- Az értékadás és a változó, illetve az értékadás és a konstans között a szöveg alapján aggregáció azonosítható.

• A logikai kifejezés és a változó, illetve a logikai kifejezés és a konstans között szintén aggregáció áll fenn.

• Az osztályok között fennálló további kapcsolat mind aggregáció, ahol a komponensek sorrendje lényeges.

Szerepek:

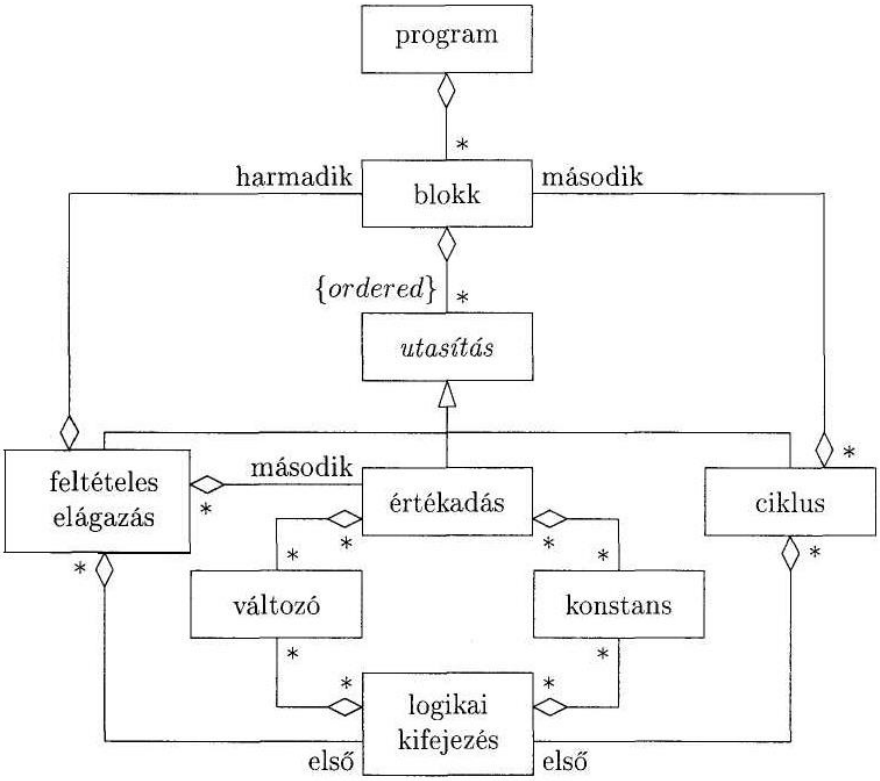
• Az utasítások komponensei sorrendben első, második, illetve harmadik szerepet vállalva alkotják az utasítást.

• A blokk utasítások véges sorozata. Az utasítások tehát sorrendben egymás után állva képeznek blokkot. Nincs megadva, hogy hányan. Tehát csak azt írhatjuk, hogy „sorrendben”. Erre szolgál az „*ordered*”.

Multiplicitások:

A szövegben szereplő többes számok alapján felismerhető, hogy az aggregátumoknak hány komponensük van. Ugyanakkor egy komponens tetszőleges számú aggregátum része lehet a program definíciója miatt.

A fenti elemzés alapján az osztálydiagram:



Készítsük el ezen osztálydiagram alapján az

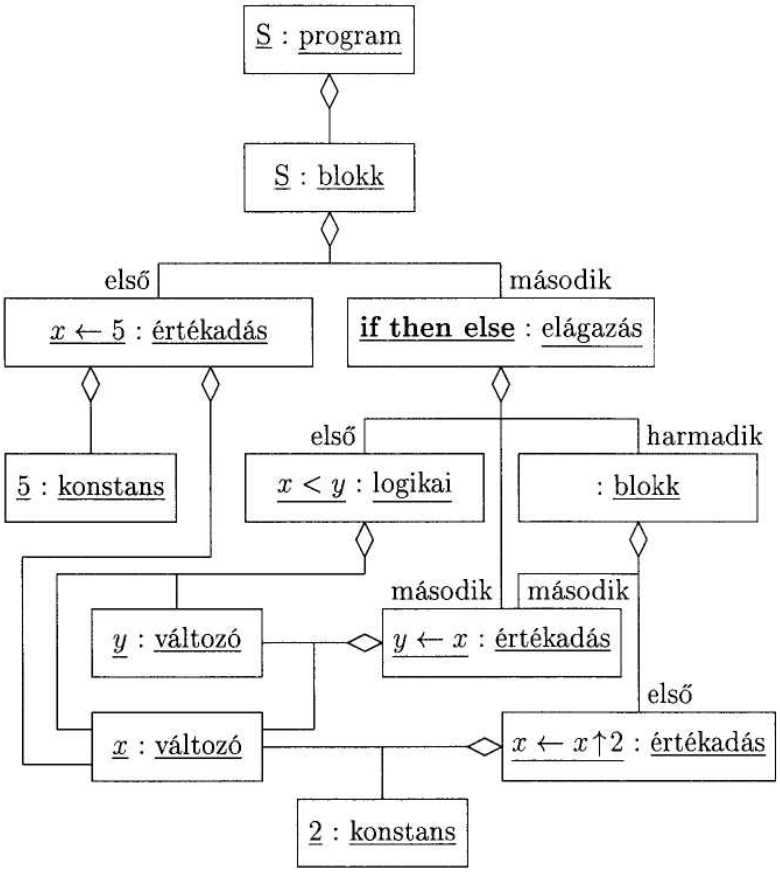
```
S :   begin
          x ← 5;
          if x < y then
              y ← x
          else
              begin
                  x ← x ↑ 2;
                  y ← x
              end
      end;
```

program objektumdiagramját, ahol **begin** ... **end** a blokkot jelenti.

Elemzés:

- A program egyetlen blokkból áll.
- A blokk két utasítást tartalmaz, melyek közül az első az $x \leftarrow 5$ értékadás, a második egy feltételes elágazás.
- A feltételes elágazás logikai kifejezése az $x < y$ feltétel, a második tagot adó értékadás az $y \leftarrow x$, a harmadik tag egy blokk.
- A harmadik tagot alkotó blokk két értékadást tartalmaz, amelyek közül az $x \leftarrow x \uparrow 2$ az első, az $y \leftarrow x$ a második tag. A második tag megegyezik a feltételes elágazás értékadásával.
- A programban az *x, y* változók és a 2, 5 konstansok szerepelnek az értékadásokban, illetve a logikai kifejezésben.

Tehát az adódó objektumdiagram:


S : program
S : blokk
első
második
x ← 5 : értékadás
if then else : elágazás
első
harmadik
5 : konstans
x < y : logikai
: blokk
második
második
y : változó
y ← x : értékadás
első
x : változó
x ← x↑2 : értékadás
2 : konstans