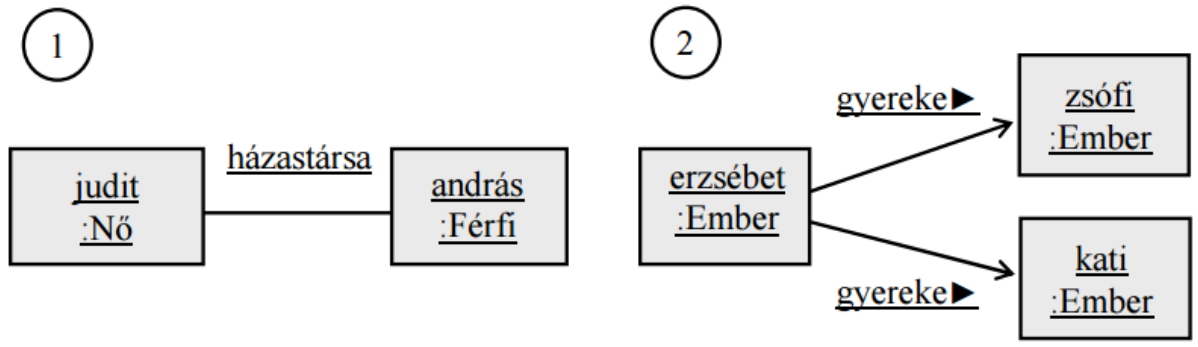
1. Objektumok közötti társítási kapcsolatok

Objektumdiagram / példánydiagram:
Az objektumot és kapcsolatait ábrázoló diagram

Ismertségi – használati kapcsolat:
Egymástól függetlenül létező objektumok között

Kapcsolatok jelölése két objektum között:

- Navigálási irány:
  Melyik irányítja melyiket?
  Ha nem ismert, vagy még nem tudható, akkor nem tesszük ki a nyilat
- Kapcsolat neve és iránya:
  Információs céllal használjuk. Ha két objektumról van szó, akkor aláhúzzuk.



Tartalmazási – rész-egész kapcsolat:
Egyik objektum része egy másiknak

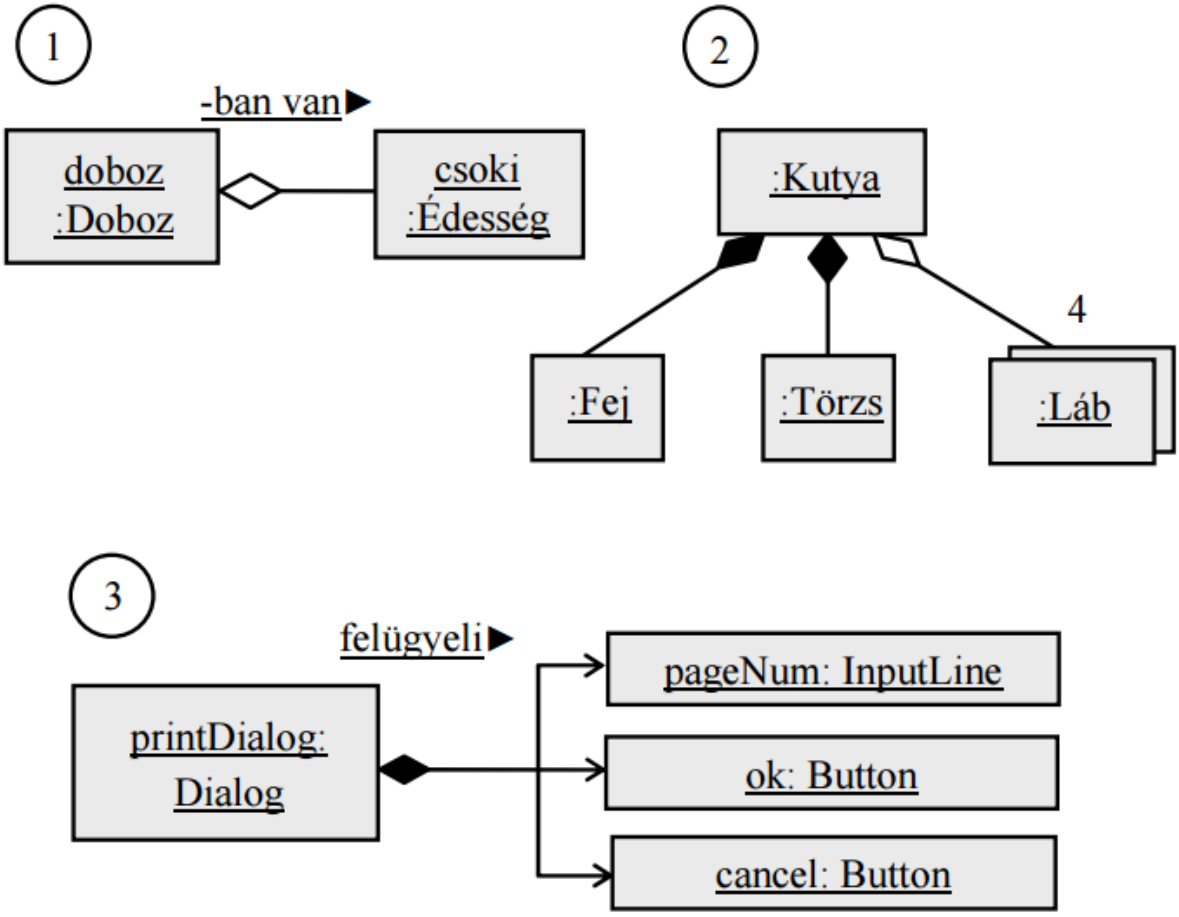
A tartalmazó objektum az **összetett** vagy egész objektum, a benne levő pedig a **részobjektum.**

Tartalmazási kapcsolatok fajtái:

- **Gyenge tartalmazás**: ha a rész kivehető az egészből;
- **Erős tartalmazás**: ha a rész nem vehető ki az egészből.
  - **Kompozíció**: az egész minden részével erős kapcsolatban áll

Kapcsolatok jelölése két objektum között:

- Az ismertségi kapcsolat jelöléseit itt is használjuk
- Kis rombuszt teszünk a tartalmazó objektum oldalára
- A gyenge tartalmazást üres, míg az erős tartalmazást tömör rombusz jelöli
- **Multiobjektum** ábrázolása: ha egy objektum több egyforma (egy osztályhoz tartozó) objektummal áll kapcsolatban, akkor azokat objektumcsoportként, egymást majdnem takaró téglalapokkal ábrázolhatjuk



A tartalmazási kapcsolat erősebb, mint az ismeretségi: az egész objektum mindig ismeri a részét

2. Osztályok közötti társítási kapcsolatok

Osztálydiagram:

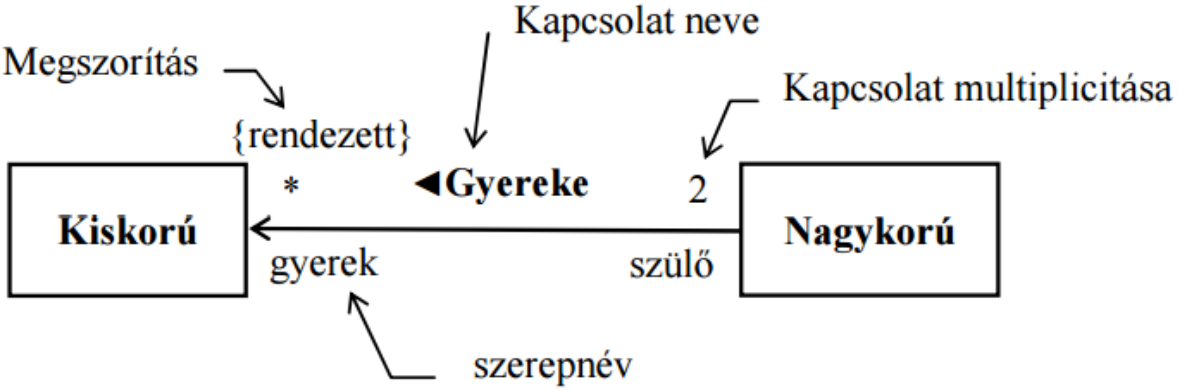
Az osztályt és kapcsolatait ábrázoló diagram.

Egy feladat kapcsán általában nem konkrét objektumok közötti kapcsolatokról van szó, hanem osztályok közötti kapcsolatokról, hiszen a szabályokat legtöbbször általánosan kell meghatározni.

Két osztály kapcsolatát a következők jellemzik

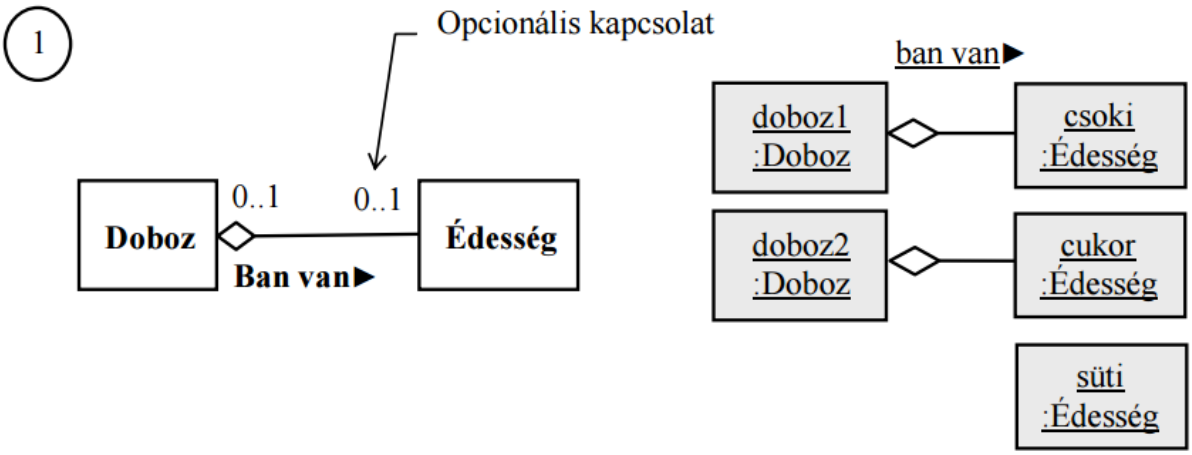
- **Kapcsolat**: Vonalat húzunk a két osztály közé. Az ismeretségi, illetve tartalmazási (erős vagy gyenge) kapcsolatokat ugyanúgy jelöljük, mint objektumok esetében. Opcionális, vagyis nem kötelező megadni.
- **Navigálási irány:** A vonal végein levő nyilak mutatják. Opcionális.
- **Kapcsolat neve és iránya**: A vonalra tesszük nagy kezdőbetűvel, vastag betűsen, aláhúzás nélkül. Opcionális.
- **Multiplicitás** (kardinalitás): A vonal két végére egy-egy számot vagy számhalmazt írunk: az osztályhoz közel eső szám azt jelenti, hogy a szemközti osztály egy objektumához hány objektum tartozhat ebben az osztályban. Számhalmazt számok és számintervallumok felsorolásával adhatunk meg. A * jelentése: akárhány. Ha nem írunk a vonalra semmit, az 1-et jelent.
  Például:
  - 3 : pontosan három
  - 10..20 : 10 és 20 közé eső szám
  - * : 0..∞, vagyis akárhány
  - 1..* : 1..∞, vagyis legalább egy
  - 1,3,10..* : 1 vagy 3, vagy 10-től kezdve akárhány.

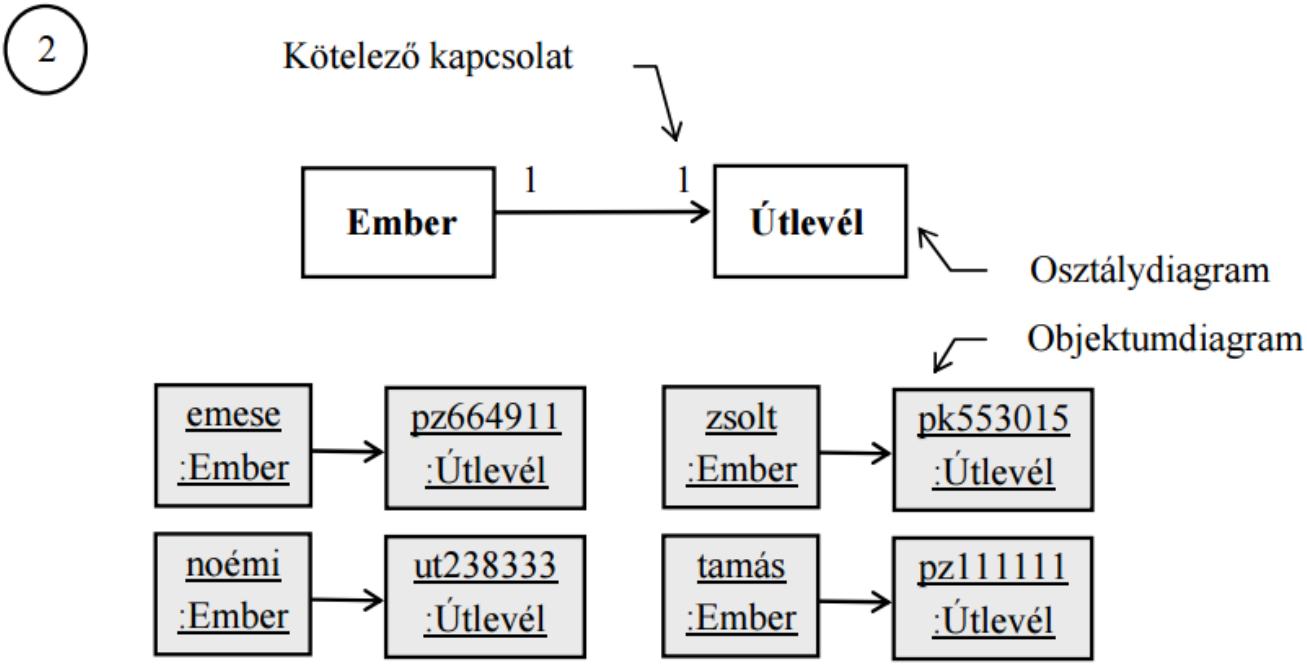
  Fajtái:
  1. Egy–egy kapcsolat (1:1)
  2. Egy–több kapcsolat (1:∞)
  3. Több–Több kapcsolat (∞:∞)
- **Szerepnév**: Két osztály kapcsolata nagyon jól jellemezhető azzal, hogy mi az osztályok szerepe ebben a kapcsolatban. A szerepnevet kisbetűvel írjuk. Opcionális.
- **Megszorítások** (kitételek): Ilyen például a kapcsolat rendezett volta (ha egy objektummal kapcsolatban álló objektumok valamilyen szempont szerint rendezve vannak). A megszorításokat kapcsos zárójelbe tesszük, például: {rendezett}. Opcionális



Egy kapcsolat lehet **kötelező** vagy **opcionális** jellegű aszerint, hogy a túloldalon kell-e egyáltalán társ, vagy nem.

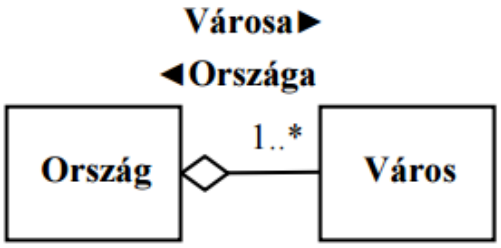
Egy-egy kapcsolat

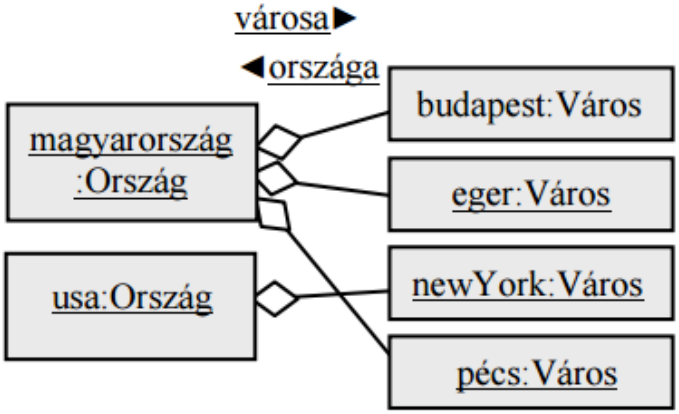




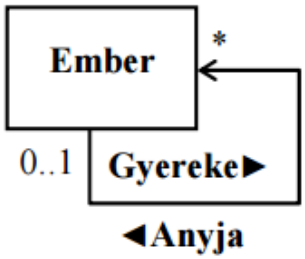
Egy-több kapcsolat

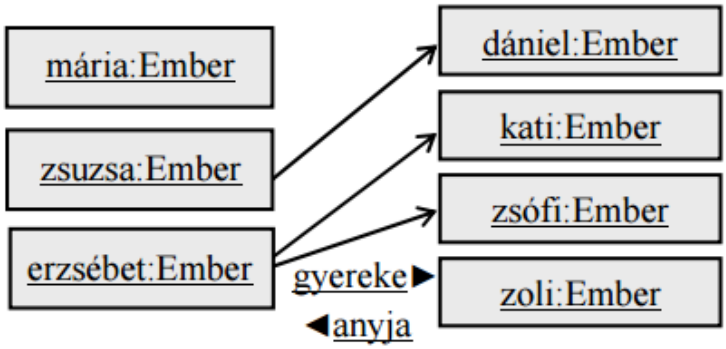
1





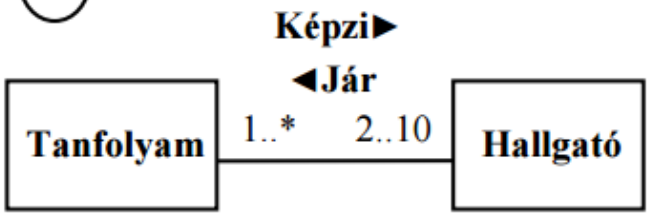
2

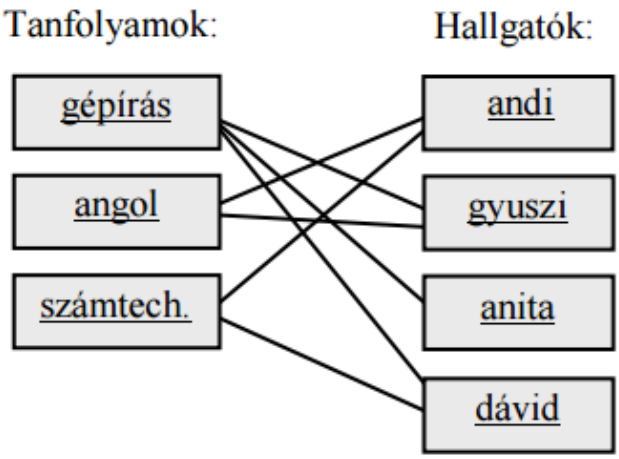




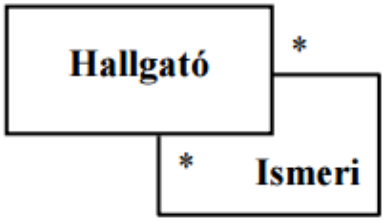
Több-több kapcsolat

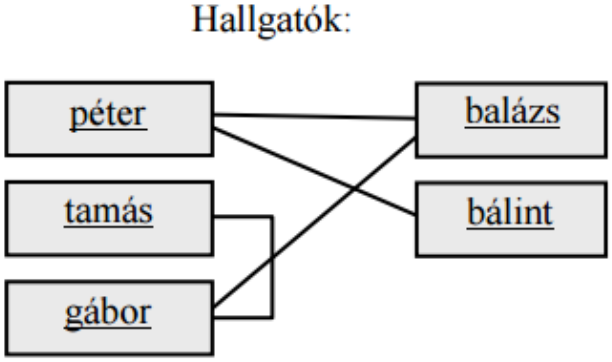
1





2

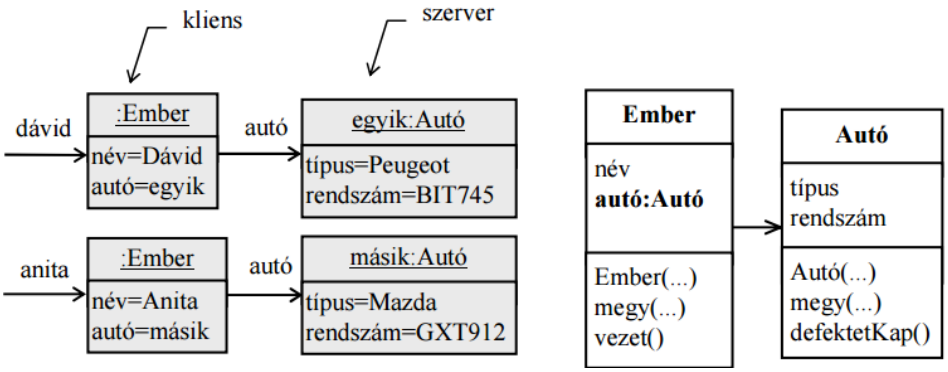




3. Társítási kapcsolat megvalósítása

A kapcsolat minden jellemzőjét (multiplicitás, tartalmazási/ismeretségi stb.) a programozónak kell megvalósítania: a programot úgy kell kódolni, hogy érvényesüljenek a megadott szabályok.

Az egy–egy kapcsolatot úgy valósítjuk meg, hogy az egyik osztályban felveszünk egy referenciatulajdonságot.



A kapcsolatokat megadó tulajdonságokat csak a program megvalósításakor, a kódban kell felvennünk.

```
dávid.autó.defektetKap();
anita.autó.megy();
```

DE! `anita.autó = dávid.autó; - ez nem felelne meg az 1:1 kapcsolatnak`

Az egy–sok kapcsolatot tipikusan konténerobjektumokkal valósítjuk meg, de megvalósítható több referenciatulajdonság definiálásával is.