ADATSZERKEZETEK ÉS ALGORITMUSOK

OOP – Elvek, alapfogalmak

A valós világ modellezése

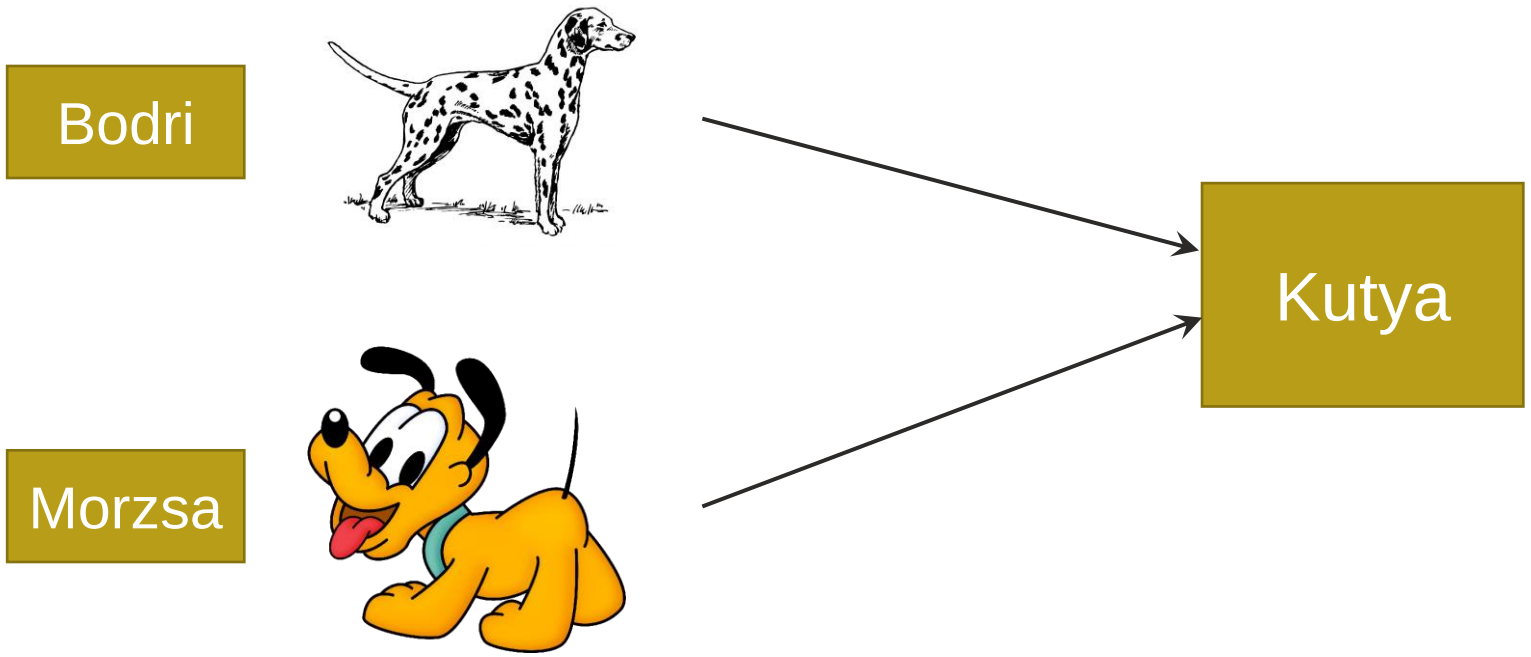
- Az ember a világ megértéséhez modelleket épít
- Modellezési alapelvek
  - Absztrakció
    - Szemléletmód, amelynek segítségével a valós világot leegyszerűsítjük
      - Csak a lényegre, a cél elérése érdekében feltétlenül szükséges részekre összpontosítunk
    - Elvonatkoztatunk a számunkra pillanatnyilag nem fontos, közömbös információktól
      - Kiemeljük az elengedhetetlen fontosságú részleteket.
  - Megkülönböztetés
    - Az objektumok a modellezendő valós világ egy-egy önálló egységét jelölik.
    - Az objektumokat a számunkra lényeges tulajdonságaik, viselkedési módjuk alapján megkülönböztetjük.

A valós világ modellezése

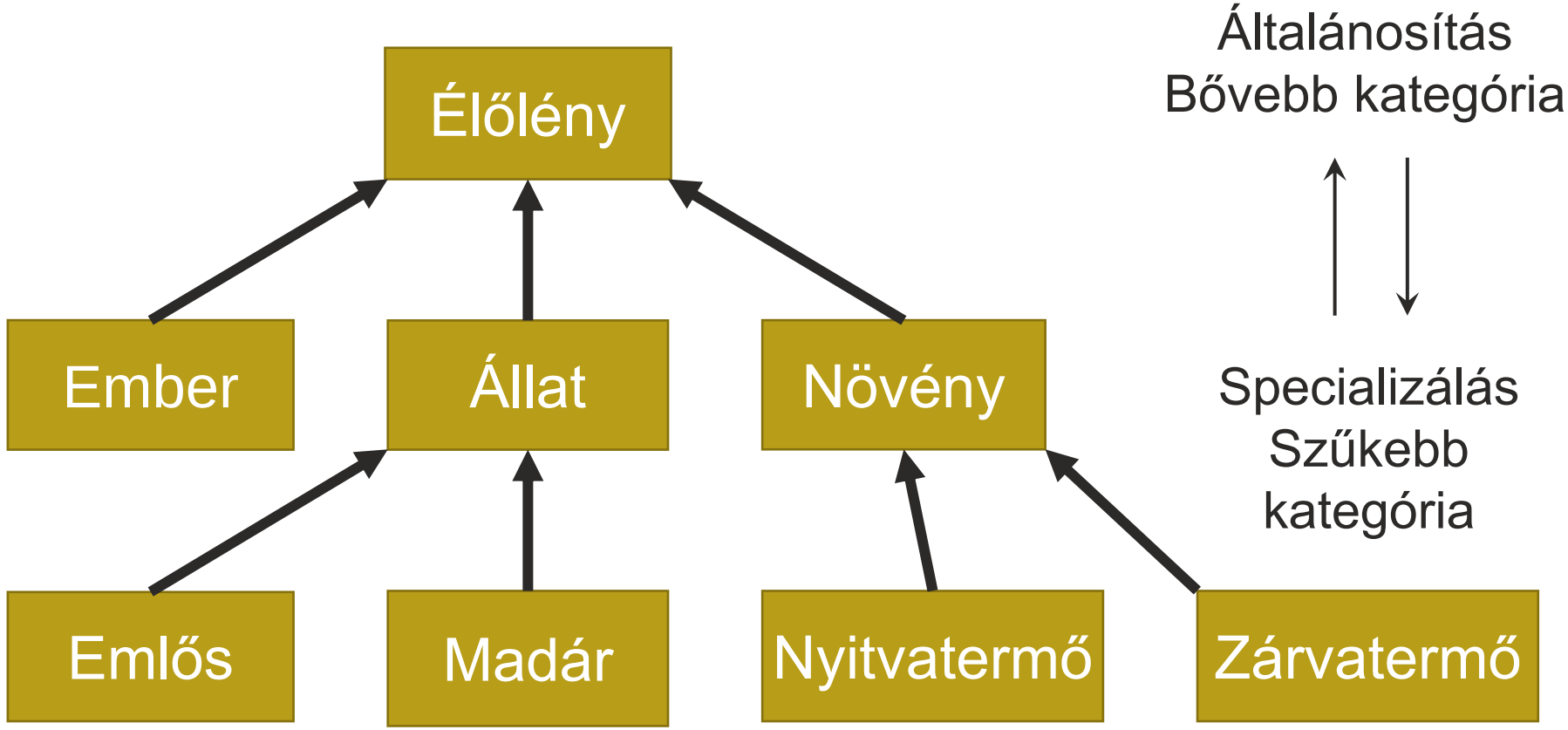
- Modellezési alapelvek
  - Osztályozás
    - Az objektumokat kategóriákba, osztályokba
      - a hasonló tulajdonságokkal rendelkező objektumok egy osztályba kerülnek.
      - a különböző vagy eltérő tulajdonságokkal rendelkező objektumok külön osztályokba kerülnek.
    - Az objektum-osztályok hordozzák a hozzájuk tartozó objektumok jellemzőit, objektumok mintáinak tekinthetők.
  - Általánosítás, specializálás
    - Az objektumok között állandóan hasonlóságokat vagy különbségeket keresünk
      - Ezáltal bővebb vagy szűkebb kategóriákba, osztályokba soroljuk őket.

A valós világ modellezése

- Osztályozás



A valós világ modellezése



OOP – alapelvek

- OOP Alapelvek (Benjamin C. Pierce)
  - Dynamic binding (dinamikus kötés)
    - Egy objektum esetén dinamikusan, futási időben dől el, hogy egy metódus melyik implementációja kerül futtatásra
  - Encapsulation (egységbe zárás)
    - Adatok és rajtuk végrehajtható műveletek egységet alkotnak
    - Praktikusan ez a modern típus-definícióval konzisztens
  - Subtype polymorphism (altípusos polimorfizmus)
    - Egy rögzített típusú változó több a típus altípusának példányára is hivatkozhat
    - Altípus
      - Az eredeti típus megszorításával létrehozott új típus
  - Inheritance, vagy delegation (öröklődés, delegáció)
    - Egy adott osztályból lehetőség van képezni egy másik osztályt
      - Ez az ős tulajdonságait megtartja
      - Azonban módosíthatja, bővítheti
  - Open recursion (nyílt rekurzió)
    - Speciális változó, amely egy metódus esetén lehetővé teszi az aktuális példány elérését

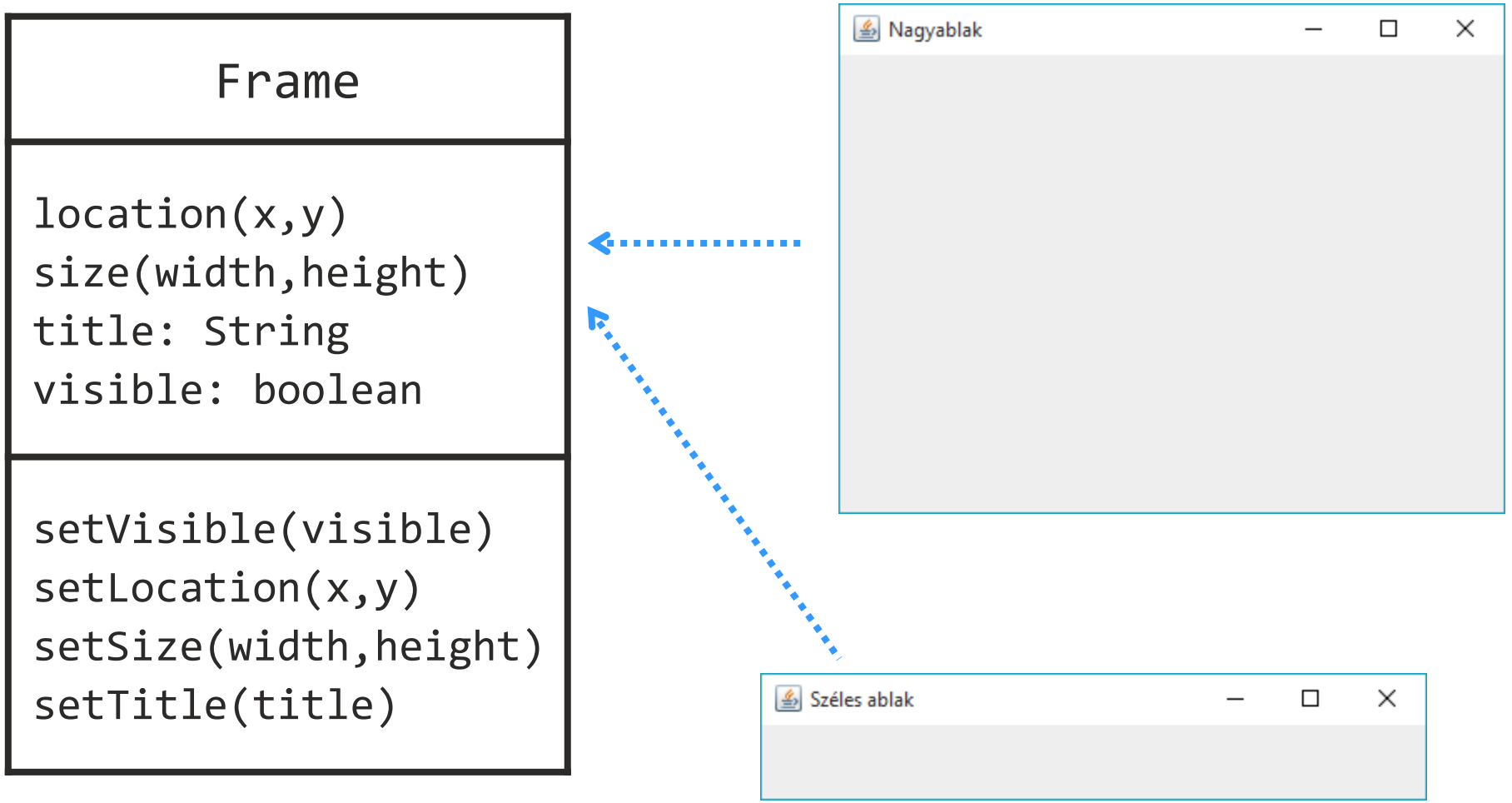
Objektum

- Belső állapota van
  - Ebben információt tárol
  - Változókkal valósítjuk meg
    - Ezek az adattagok, vagy tulajdonságok.
- Kérésre feladatokat hajt végre
  - Metódusok – melyek hatására állapota megváltozhat
- Üzeneteken keresztül lehet megszólítani
  - Kommunikál más objektumokkal
- Minden objektum egyértelműen azonosítható

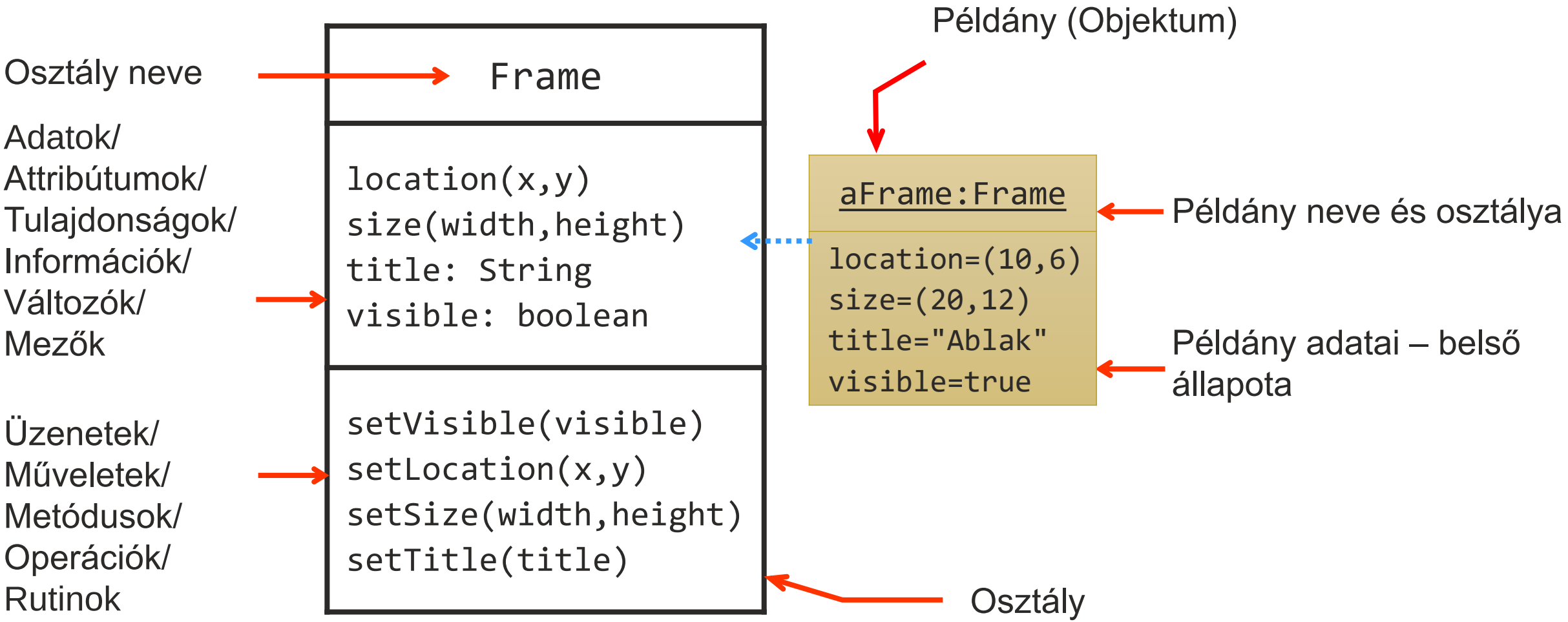
Osztály, példány

- Osztály (class)
  - Olyan objektumminta vagy típus
  - Ez alapján példányokat (objektumokat) hozhatunk létre
- Példány (instance)
  - Egy osztály (minta) alapján létrejött konkrét példány
  - Minden objektum születésétől kezdve egy osztályhoz tartozik

Frame osztály és példányai



Osztály és példány az UML-ben



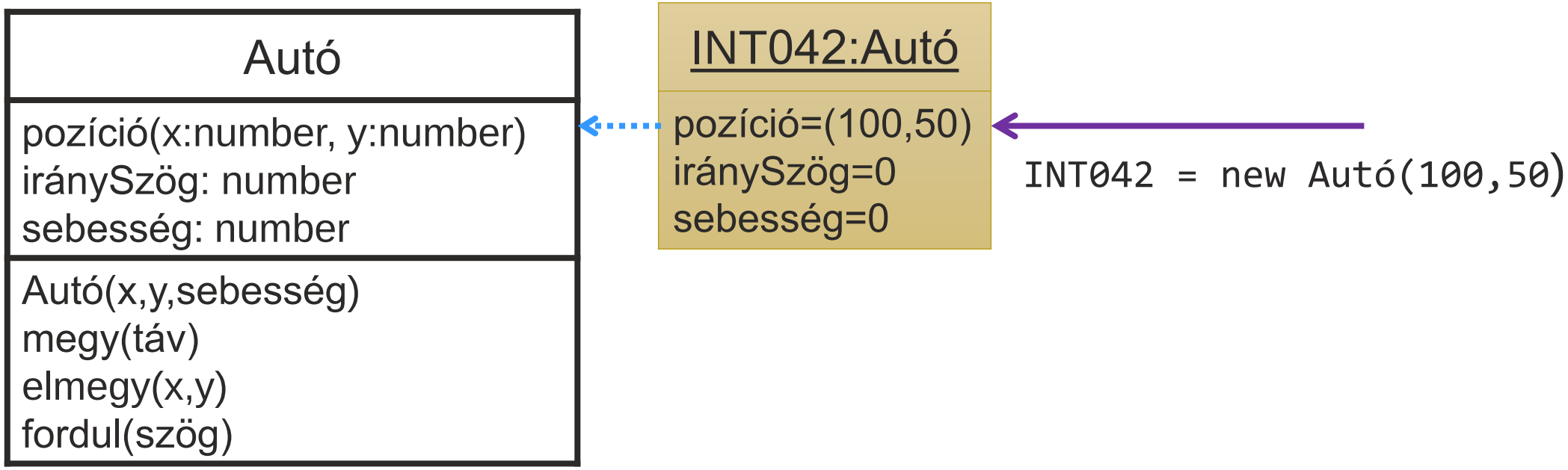
Objektum életciklusa

- Objektum életciklusa
  - Létrejön – „megszületik”
  - Állapotváltozásokon esik keresztül – „él”
  - Megszűnik – „meghal”

Objektum létrehozása, inicializálása

- Az objektumot létre kell hozni és inicializálni kell!
- Objektum inicializálása
  - Konstruktor (constructor) végzi
  - Adatok kezdőértékadása
  - Objektum működéséhez szükséges tevékenységek végrehajtása
  - Típusinvariáns beállítása

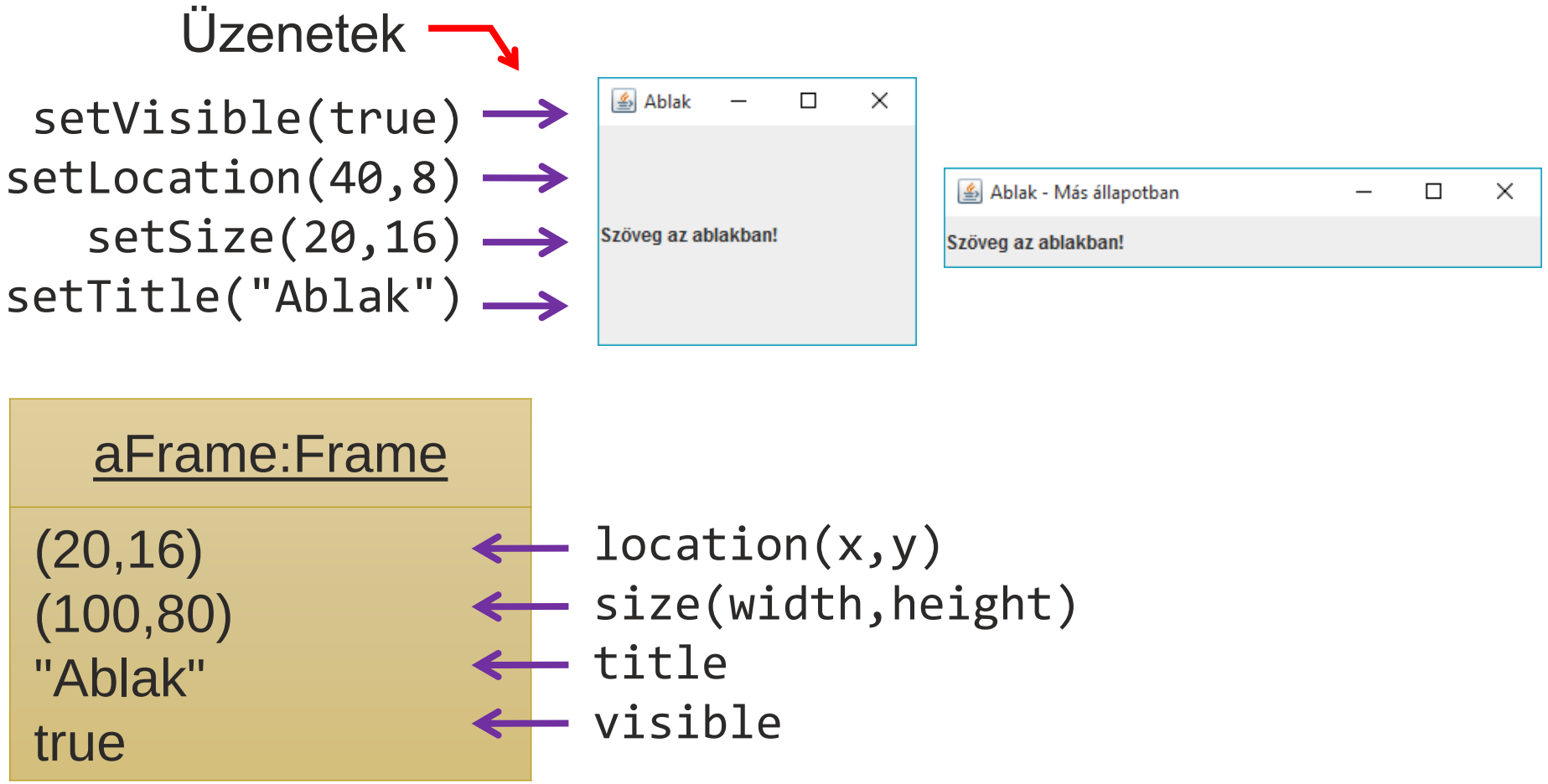
Objektum létrehozása, inicializálása



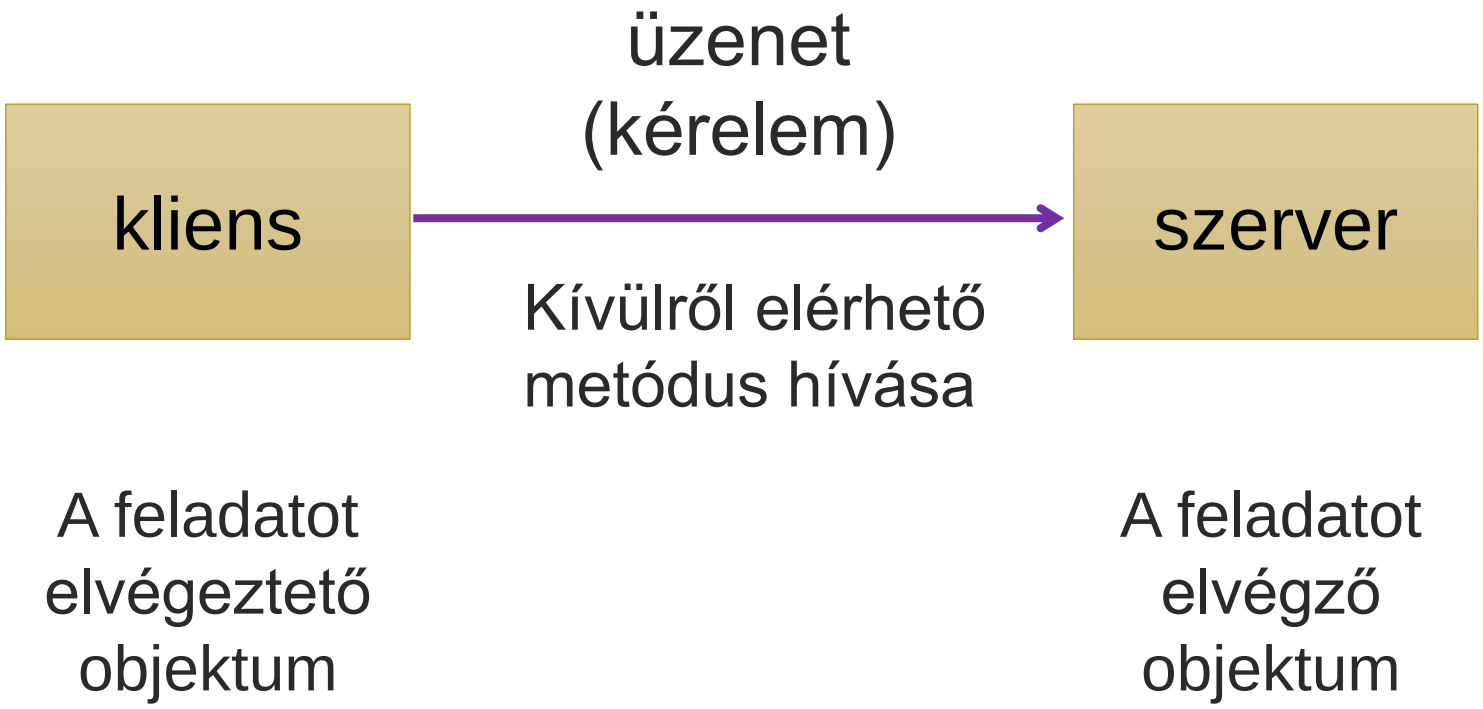
Objektum állapotváltozása

- Az objektum kérésre feladatokat hajt végre
- Életciklusa során üzeneteket fogad és feldolgozza
  - Hatására az állapot megváltozik
  - Ezek során üzeneteket küldhet más objektumoknak

Állapotváltozás



Üzenetküldési modell



Üzenetküldési modell

- Kliens
  - Aktív objektum, másik objektumon végez műveleteket, de rajta nem végeznek
  - Nincs export felülete
  - Például óra (órajel)
    - Meghatározott időközönként művelet egy regiszteren
- Szerver
  - Passzív objektum
  - Csak export felülete van
  - Másoktól érkező üzenetekre vár, mások szolgáltatását nem igényli
- Ágens
  - Általános objektum, van export és import felülete

Objektum műveletei

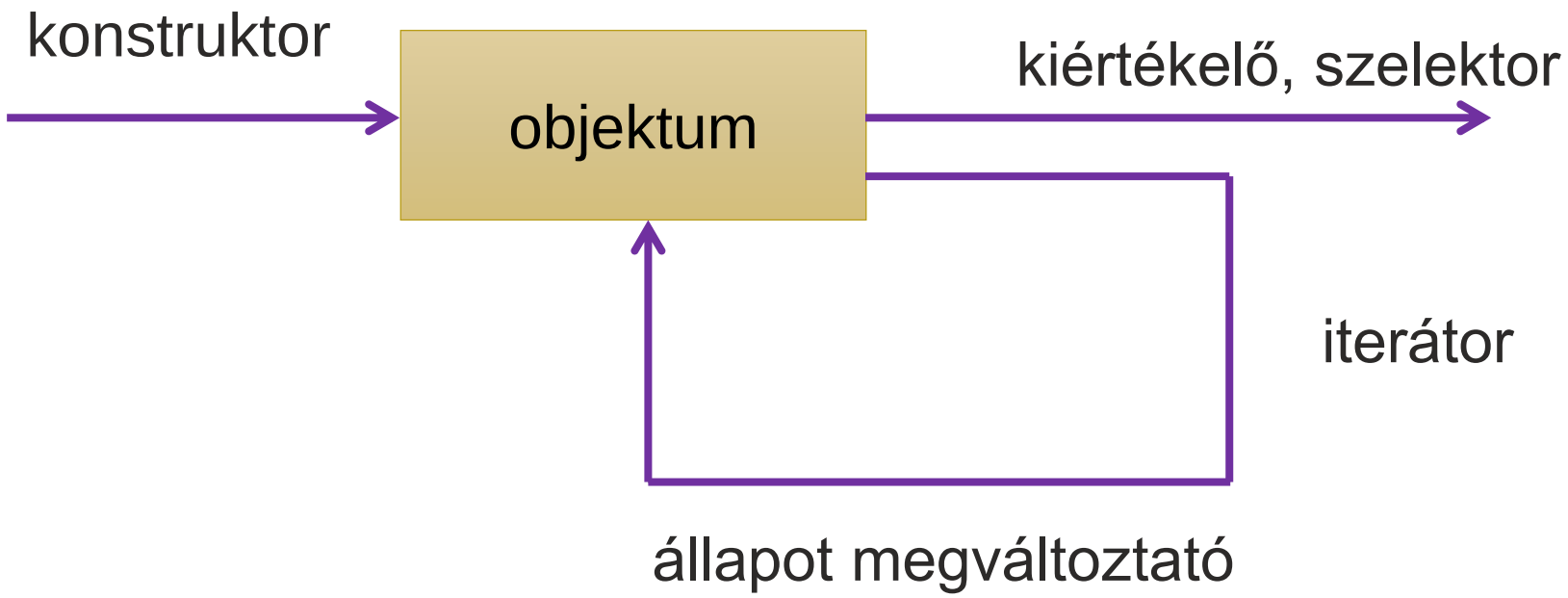
- Export műveletek
  - Amelyeket más objektumok hívhatnak
  - Például verem
    - `push`, `pop`, `top` stb.
- Import műveletek
  - Amelyeket az objektum igényel ahhoz, hogy az export szolgáltatásait nyújtani tudja
  - Például verem, ha fix méretű (vektoros) reprezentáció
    - Vektorműveletek

Objektum műveletei

- Export műveletek csoportosítása:
  - Létrehozó (konstruktor) az objektum létrehozására, felépítésére
    - Példa veremnél
      - `create`: $\rightarrow \mathrm{Verem}$
  - Állapot megváltoztató
    - Példa verem esetén
      - `pop`: $\mathrm{Verem} \rightarrow \mathrm{Verem} \times \mathrm{Elem}$
      - `push`: $\mathrm{Verem} \times \mathrm{Elem} \rightarrow \mathrm{Verem}$
  - Szelektor – kiemeli az objektum bizonyos részét
    - Például
      - vektor adott indexű elemét
      - `access`: $\mathrm{Vektor} \times \mathrm{Index} \rightarrow \mathrm{Elem}$
  - Kiértékelő – objektum jellemzőit lekérdező műveletek (`size`, `has`, stb.)
  - Iterátor – bejáráshoz

Objektum műveletei

- Export műveletek csoportosítása:

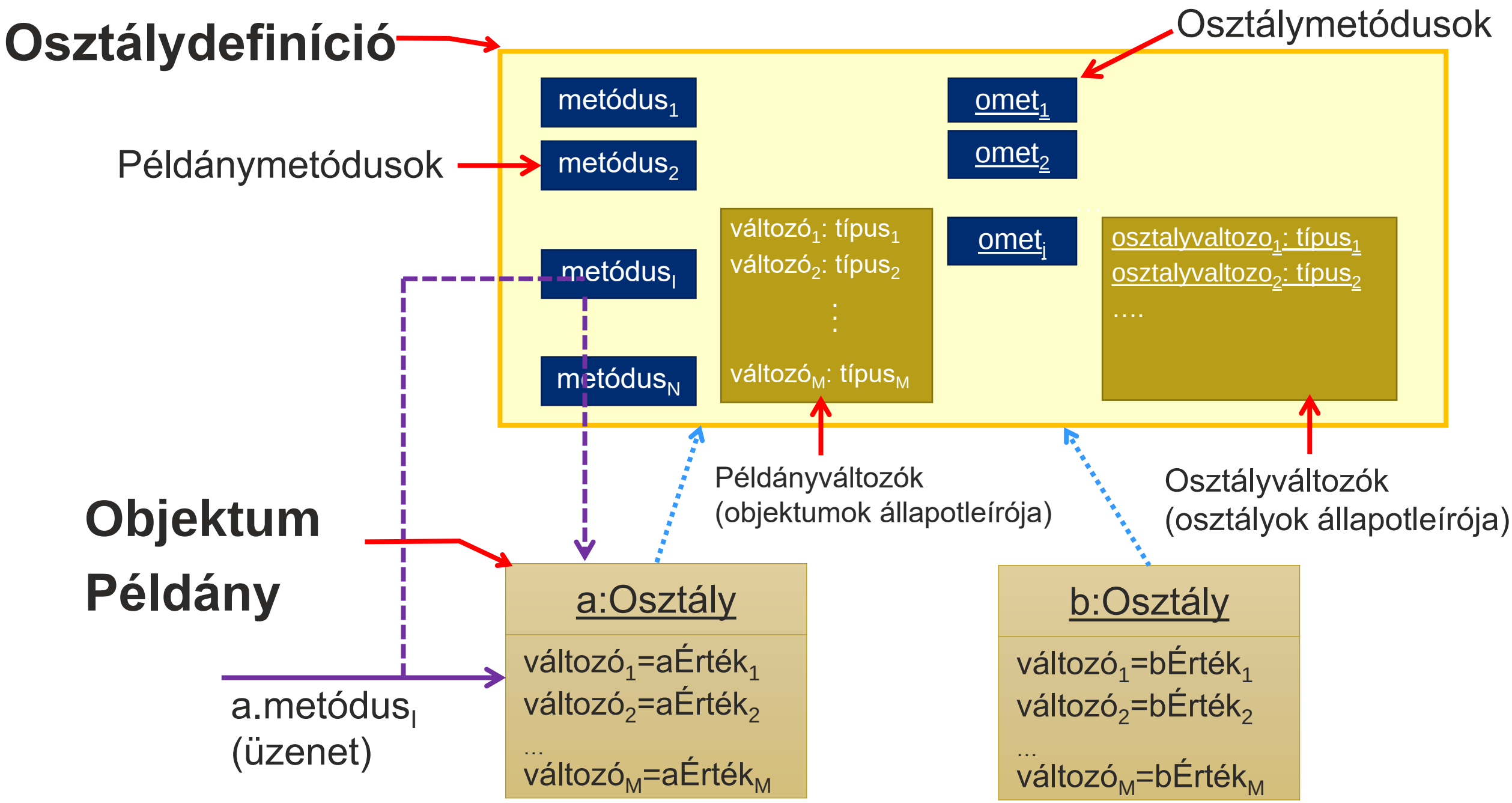


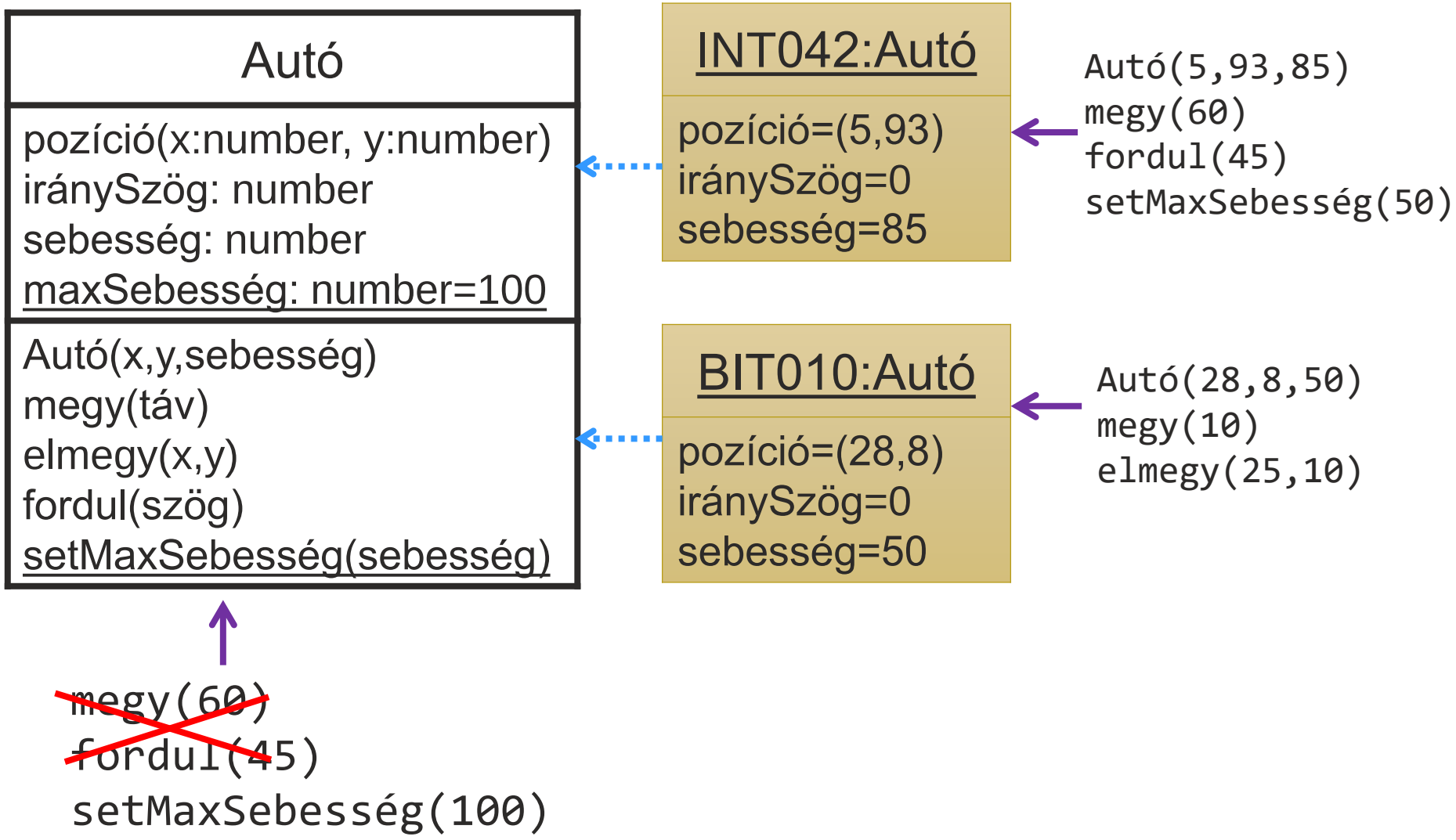
Objektum megszűnése

- Az objektum életciklusának végén:
  - Erőforrások (memória) felszabadítása
  - Adatok, állapot mentése
- Ha a nyelv támogatja, akkor a destruktor (destructor) hajtja végre.

Osztály, példány

- Osztálydefiníció:
  - Példányváltozó
    - Példányonként helyet foglaló változó
  - Példánymetódus
    - Példányokon dolgozó metódus
  - Osztályváltozó
    - Osztályonként helyet foglaló változó
  - Osztálymetódus
    - Osztályokon dolgozó metódus


Osztálydefiníció
Osztálymetódusok
Példánymetódusok
metódus1
metódus2
metódusI
metódusN
omet1
omet2
ometJ
változó1: típus1
változó2: típus2
változóM: típusM
osztalyvaltozo1: típus1
osztalyvaltozo2: típus2
....
Objektum
Példány
Példányváltozók
(objektumok állapotleírója)
Osztályváltozók
(osztályok állapotleírója)
a:Osztály
változó1=aÉrték1
változó2=aÉrték2
...
változóM=aÉrtékM
b:Osztály
változó1=bÉrték1
változó2=bÉrték2
...
változóM=bÉrtékM
a.metódusI
(üzenet)


Autó
pozíció(x:number, y:number)
iránySzög: number
sebesség: number
maxSebesség: number=100
Autó(x,y,sebesség)
megy(táv)
elmegy(x,y)
fordul(szög)
setMaxSebesség(sebesség)
INT042:Autó
pozíció=(5,93)
iránySzög=0
sebesség=85
Autó(5,93,85)
megy(60)
fordul(45)
setMaxSebesség(50)
BIT010:Autó
pozíció=(28,8)
iránySzög=0
sebesség=50
Autó(28,8,50)
megy(10)
elmegy(25,10)
megy(60)
fordul(45)
setMaxSebesség(100)

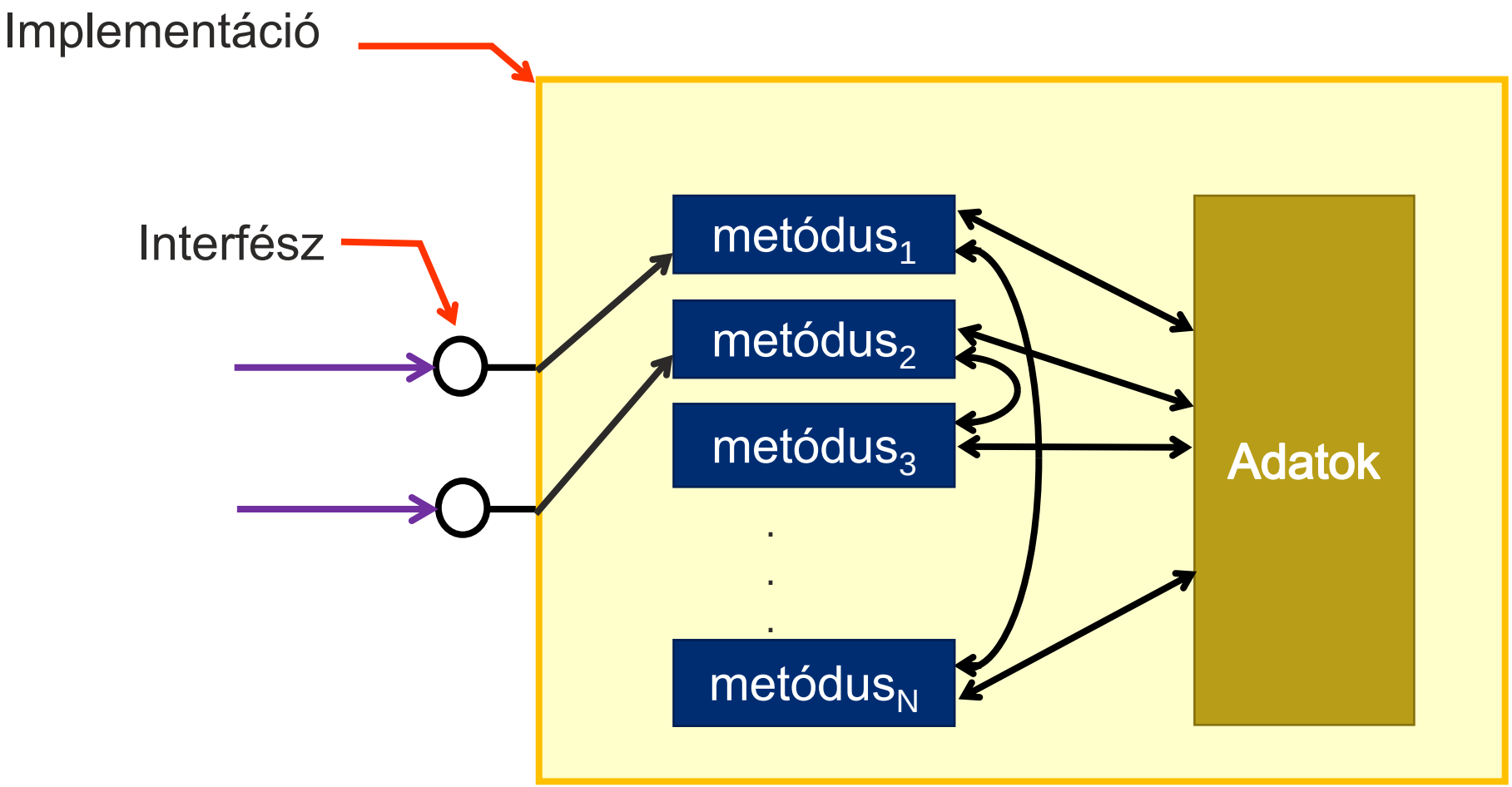
A „this”

- Adott osztályhoz tartozó példányok esetén
  - Honnan tudjuk, hogy éppen melyik objektum hívta meg a megfelelő metódust
  - Melyik objektum adataival fog dolgozni a metódus?
- Szükségünk van egy olyan mutatóra, amely mindig a metódust meghívó objektumpéldányra mutat.
  - Ezt szolgálja a „`this`” pszeudóváltozó
    - Néha explicit formális paraméter
  - Ez metódushíváskor egyértelműen rámutat azokra az adatokra, amelyekkel a metódusnak dolgoznia kell.
- Az objektum a saját magának küldött üzenet esetén a `this.Üzenet(Paraméterek)` formát kell, hogy használja.
  - A metódustörzsekben az adott példányra mindig a `this` segítségével hivatkozhatunk.
    - Ez számos nyelvben alapértelmezett.

OOP elvárások

- Bezárás (encapsulation)
  - Adatok és metódusok összezárása
  - Egybezárás, egységbezárás – osztály (class)
- Információ elrejtése (information hiding)
  - Az objektum „belügyeit” csak az interfészen keresztül lehet megközelíteni (láthatóságok!)
- Kód újrafelhasználása (code reuse)
  - Megírt kód felhasználása példány létrehozásával vagy osztály továbbfejlesztésével

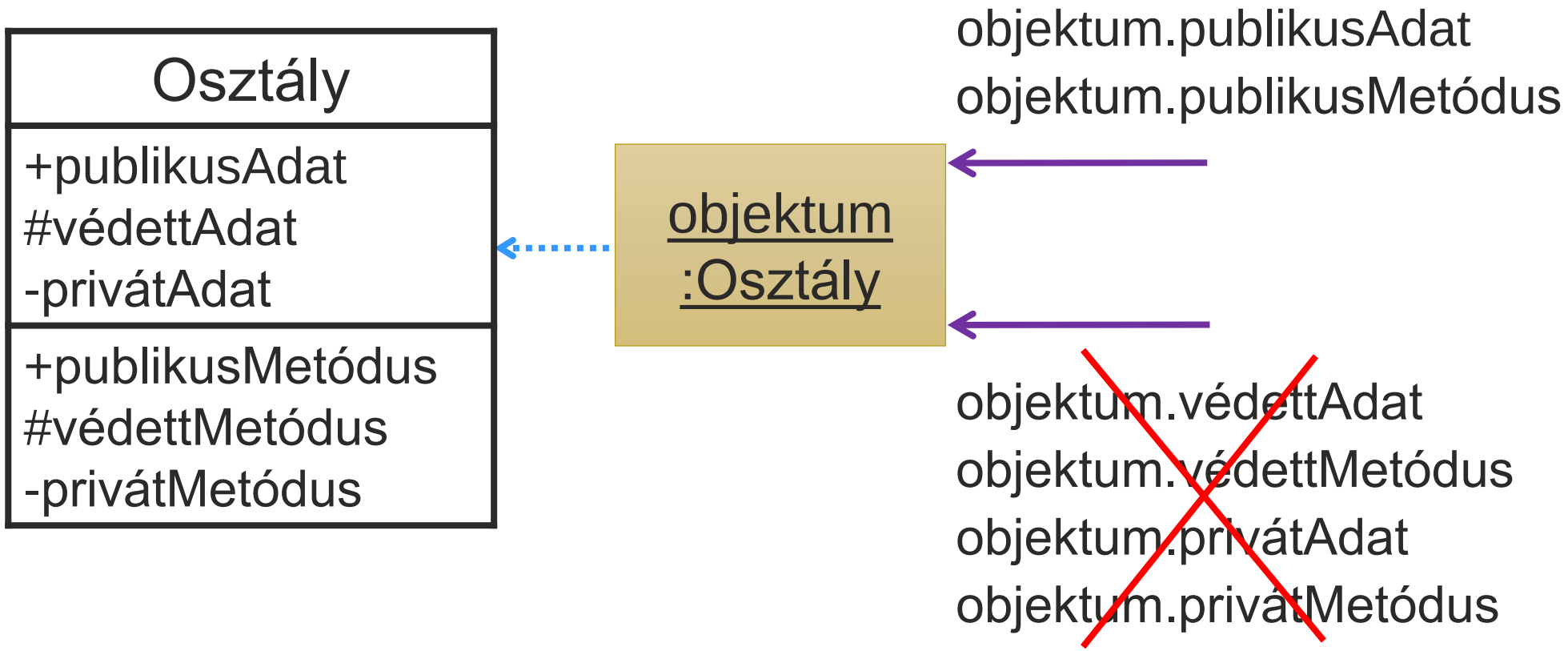
Információ elrejtése



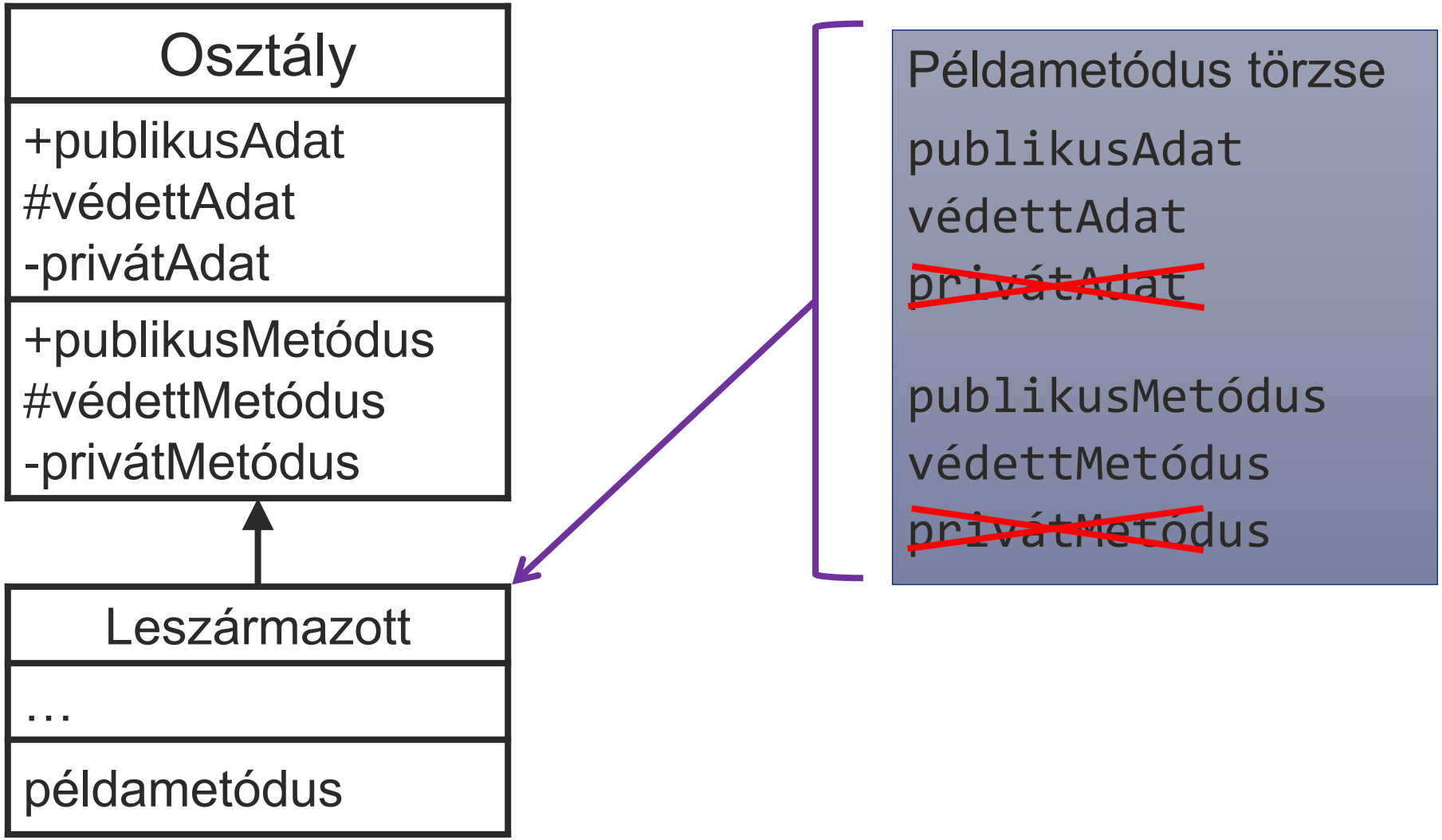
Információ elrejtése

- Az objektum adatait csak interfészen keresztül lehet elérni
  - Az elv szigorú követése esetén, minden adattag elrejtésre kerül, közvetlenül nem hozzáférhető
    - Dedikált függvények használatával lehetséges a hozzáférés.
- A láthatóság minősítője leggyakrabban
  - public
    - a külső felhasználók elérik
  - protected
    - csak a leszármazottak érhetik el
  - private
    - csak az adott osztály számára elérhető

Láthatóság – Objektum védelme



Láthatóság – Objektum védelme



OOP Öröklés

Következő téma