Unified Modeling Language (UML)

Jeszenszky Péter
jeszenszky.peter@inf.unideb.hu

Utolsó módosítás: 2024. április 14.

Mi az UML?

- *„The OMG's Unified Modeling Language (UML) helps you specify, visualize, and document models of software systems, including their structure and design [...]. (You can use UML for business modeling and modeling of other non-software systems too.)”*
  - Forrás: *Introduction To OMG's Unified Modeling Language* https://www.omg.org/UML/what-is-uml.htm
- AZ OMG Egységes Modellező Nyelve szoftverrendszerek modelljeinek – ideértve azok felépítését és tervezését – részletes leírását, megjelenítését és dokumentálását segítő nyelv. (Üzleti modellezéshez és más nem szoftver rendszerek modellezéséhez is használható.)

Object Management Group (OMG)

- Technológiai szabványokat fejlesztő nyílt tagságú nemzetközi non-profit konzorcium, melyet 1989-ben alapítottak. https://www.omg.org/
- Szabványok:
  - *Business Process Model and Notation (BPMN)* https://www.bpmn.org/
  - *MetaObject Facility (MOF)* https://www.omg.org/mof/
  - *Systems Modeling Language (SysML)* https://www.omgsysml.org/
  - *Unified Modeling Language (UML)* https://www.uml.org/
  - …

Történet

- Előzménye több objektumorientált szoftverfejlesztési módszer:
  - Booch (Grady Booch)
  - OMT (*object-modeling technique*) (James E. Rumbaugh és mások)
  - OOSE (*object-oriented software engineering*) (Ivar Jacobson)
- *„Three Amigos”*: Booch, Jacobson és Rumbaugh
  - Hármuk vezetésével történik az UML kifejlesztése.
- Lásd még:
  - *The Unified Modeling Language – Versions of UML Versions of UML* https://www.uml-diagrams.org/

Az aktuális szabvány

- *OMG Unified Modeling Language (OMG UML) Version 2.5.1*. December 2017. https://www.omg.org/spec/UML/2.5.1/
- *Diagram Definition (DD) Version 1.1*. June 2015. https://www.omg.org/spec/DD/1.1/
- *XML Metadata Interchange (XMI) Version 2.5.1*. June 2015. https://www.omg.org/spec/XMI/2.5.1/
- *OMG Meta Object Facility (MOF) Core Specification Version 2.5.1*. November 2016. https://www.omg.org/spec/MOF/2.5.1/
- *Object Constraint Language Version 2.4*. February 2014. https://www.omg.org/spec/OCL/2.4/

Object Constraint Language (OCL)

- Formális nyelv kifejezések leírására UML modellekről.
- Nem programozási nyelv, hanem modellező nyelv!
  - Az OCL kifejezések közvetlenül nem végrehajthatók.
- Az OCL kifejezések kiértékelése mellékhatásmentes.
- Típusos nyelv, azaz minden kifejezésnek típusa van.
- Számos különböző célra használható:
  - Lekérdezőnyelvként.
  - Osztályokra és típusokra vonatkozó invariáns feltételek előírására.
  - Műveletek és metódusok elő- és utófeltételeinek leírására.
  - …

XML Metadata Interchange (XMI)

- XML formátum metaadatok alkalmazások közötti cseréjéhez.
- Leggyakrabban UML modellek cseréjéhez használják, de tetszőleges olyan metaadatok sorosítására alkalmas, melyek metamodellje kifejezhető a MOF segítségével.

Modell

- A modell egy rendszer leírása, ahol a rendszer a lehető legszélesebb jelentésben értendő (például szoftverek, szervezetek, folyamatok, …).
- A rendszert egy bizonyos nézőpontból írja le érintettek egy bizonyos csoportjának (például a rendszer tervezői vagy felhasználói) számára egy bizonyos absztrakciós szinten.
- Teljes abban az értelemben, hogy az egész rendszert lefedi, bár csak azon aspektusai kerülnek ábrázolásra benne, melyek lényegesek céljának szempontjából.
  - Forrás: *OMG Unified Modeling Language (OMG UML) Version 2.5.1*. December 2017. https://www.omg.org/spec/UML/2.5.1/

Metamodell

- Egy modell modellje.
- Az UML-ben a metamodell egy olyan modell, mely önmagát modellezi.
  - Nem csupán saját maga, hanem más modellek és metamodellek modellezésére is használható.
  - Például a MOF modell egy metamodell.

Metaosztály

- Egy olyan osztályt jelent egy objektumorientált nyelvben, melyeknek példányai osztályok.
  - **Python**:
    - *PEP 3115 – Metaclasses in Python 3000* https://www.python.org/dev/peps/pep-3115/
    - *Python 3.11.2 documentation – Data model – Metaclasses* https://docs.python.org/3/reference/datamodel.html#metaclasses
  - **Groovy**:
    - *The Groovy programming language – Runtime and compile-time metaprogramming* http://groovy-lang.org/metaprogramming.html
    - `groovy.lang.MetaClass` http://docs.groovy-lang.org/latest/html/api/groovy/lang/MetaClass.html
  - **UML**: a metaosztály egy osztály egy metamodellben (például `Element`, `Classifier`, …).

Szintaxis (1)

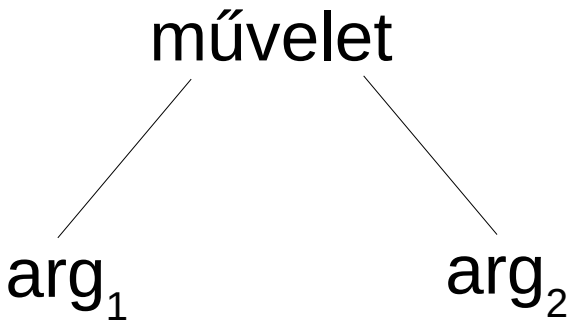
- Nyelvészet:
  - Annak módja, ahogy a nyelvi elemek (mint például a szavak) alkotóelemekké (például szókapcsolatokká és mondategységekké) kerülnek összeillesztésre”.
  - A nyelvtan ezzel foglalkozó része.
    - Lásd: https://www.merriam-webster.com/dictionary/syntax
- Programozási nyelvek:
  - Egy programozási nyelv szintaxisa a programjainak helyes formáját írja le [...].
    - Lásd: Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman. *Compilers: Principles, Techniques, & Tools*. 2nd ed. Addison Wesley, 2006.

Szintaxis (2)

- **Absztrakt szintaxis**: a nyelvi elemek szerkezetének bármiféle ábrázolásmódtól független leírása.
- **Konkrét szintaxis**: az absztrakt szintaxis leképezése egy adott (gépi) ábrázolásmódra.

Szintaxis (3)

- Absztrakt szintaxis:



- Konkrét szintaxis:
  - Infix alak:
    - `(1 + 2) * 3`
  - Prefix alak:
    - `(* (+ 1 2) 3)`
  - Postfix alak:
    - `((1 2 +) 3 *)`

Szintaxis (4)

- Az UML absztrakt szintaxisát egy UML modell segítségével határozzák meg, melyet **UML metamodellnek** neveznek.

Szemantika (1)

- Programozási nyelvek:
  - Egy programozási nyelv szintaxisa a programjainak helyes formáját írja le, míg a nyelv szemantikája határozza meg a programjainak jelentését, vagyis azt, hogy a programok mit csinálnak a végrehajtásukkor.
    - Lásd: Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman. *Compilers: Principles, Techniques, & Tools*. 2nd ed. Addison Wesley, 2006.

Szemantika (2)

- *„Colorless green ideas sleep furiously.” /* „Színtelen zöd eszmék dühödten alszanak.”
  - Forrás: Noam Chomsky. *Syntactic Structures*. Mouton & Co., 1957.
- A mondat a szintaxis és a szemantika közötti különbséget szemlélteti.
  - A mondat nyelvtanilag (szintaktikailag) helyes, azonban értelmetlen.

Szakterület-specifikus nyelvek

- **Szakterület-specifikus nyelv** (***domain-specific language***, **DSL**): Egy bizonyos fajta problémára koncentráló számítógépes nyelv, nem pedig egy általános célú nyelv tetszőleges fajta problémák megoldásához.
  - Példák: BibTeX/LaTeX, CSS, DOT (Graphviz), Gradle DSL, Make, PlantUML, SQL, …
  - Lásd: Martin Fowler. *DomainSpecificLanguage*. https://martinfowler.com/bliki/DomainSpecificLanguage.html

Meta Object Facility (MOF) (1)

- A MOF egy nyílt és platformfüggetlen metaadat kezelő keretrendszert és kapcsolódó metaadat szolgáltatásokat biztosít, melyek lehetővé teszik modell- és metaadat vezérelt rendszerek fejlesztését és együttműködését.
  - A MOF-ot felhasználó rendszerek között vannak modellező és fejlesztőeszközök, adattárház rendszerek, metaadat tárolók, …
  - Forrás: *OMG Meta Object Facility (MOF) Core Specification, Version 2.5.1*

Meta Object Facility (MOF) (2)

- A MOF egy metamodellek definiálására szolgáló szakterület-specifikus nyelv.
  - Önmagának és más metamodellek (például UML, CWM) modellezéséhez használják.
  - Tetszőleges metaadatok (például szoftver konfiguráció vagy követelmény metaadatok) modellezéséhez használható.

Meta Object Facility (MOF) (3)

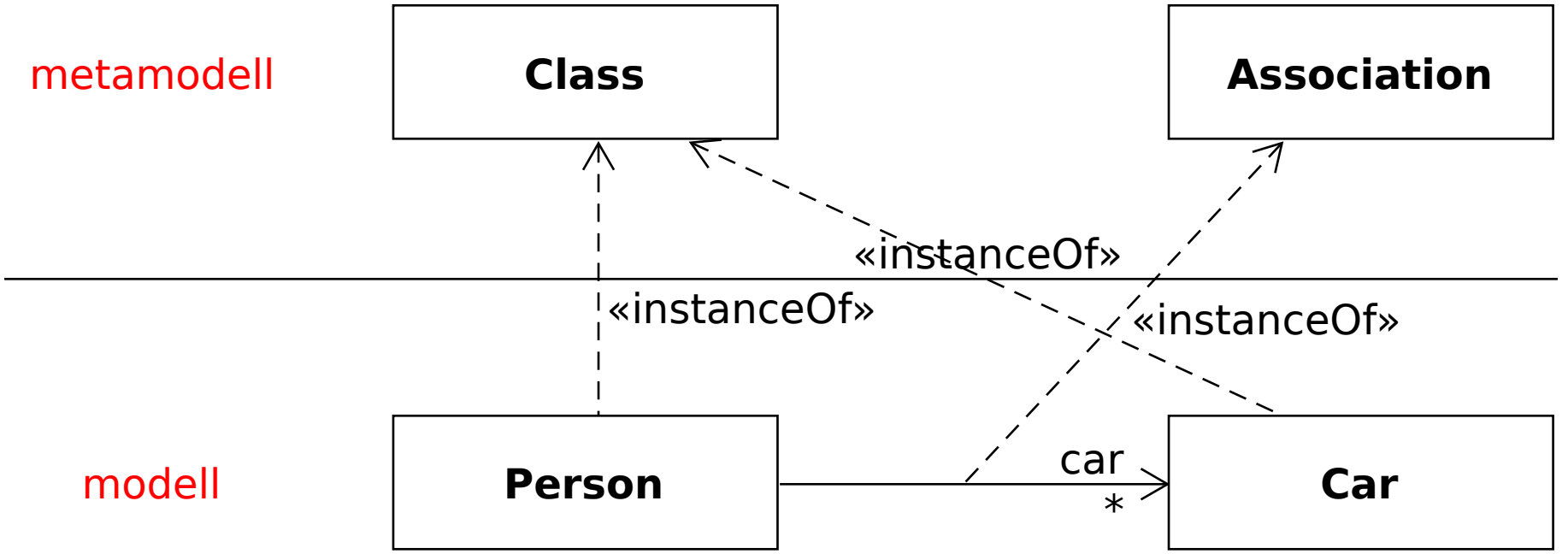
- **UML metamodell**:
  - Az UML absztrakt szintaxisát meghatározó UML modell.
  - Az UML egy olyan részhalmazának konstrukcióit használó metamodell metamodellek létrehozásához, melyet a MOF specifikáció határoz meg.

Rétegzett metamodell architektúra

- Nyelvek definiálásakor általában három réteget kell figyelembe venni:
  - A nyelv specifikációja (metamodell)
  - Felhasználói specifikáció (modell)
  - A modell objektumai

Rétegzett metamodell architektúra

- Metamodellezés:



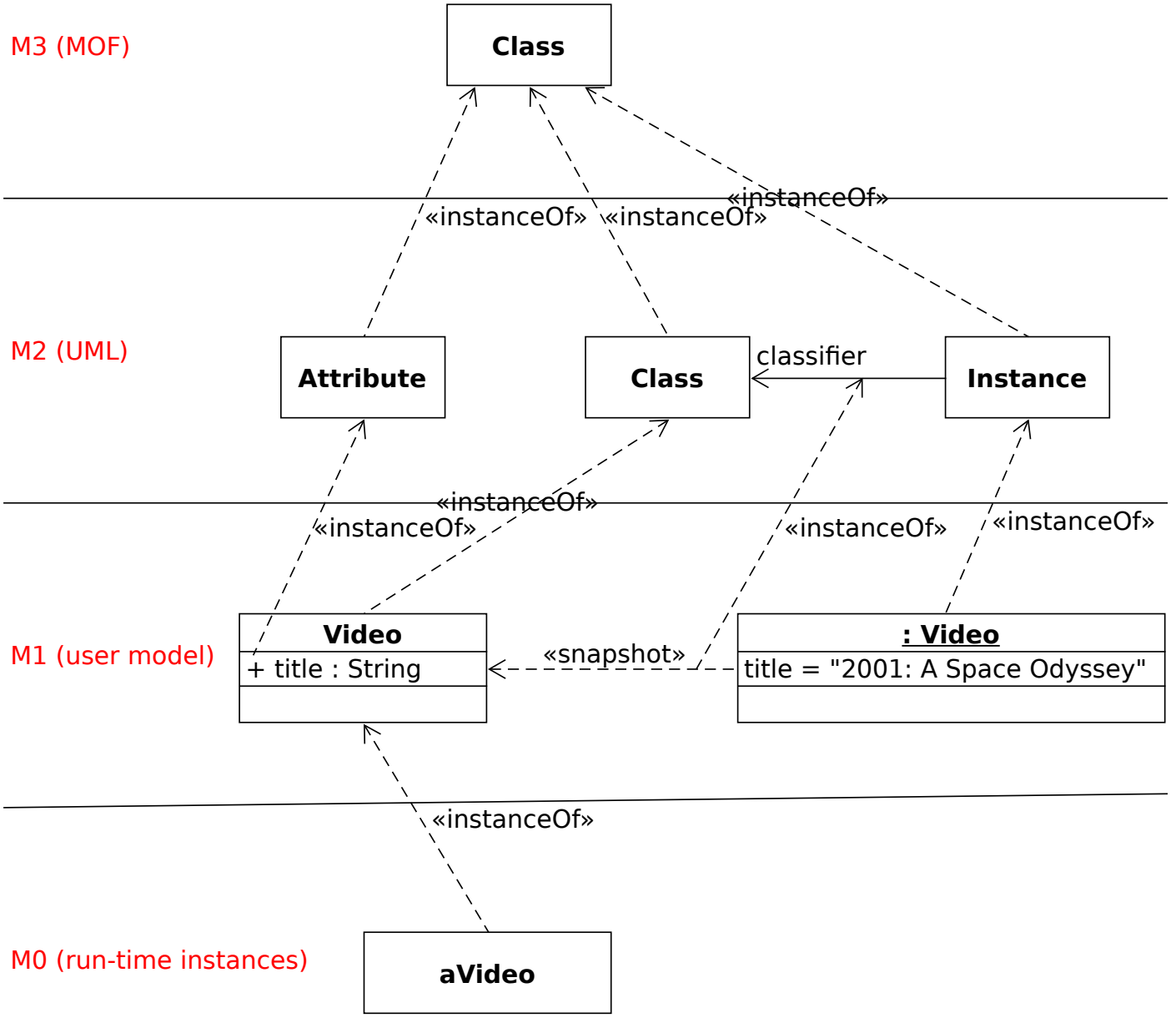
Négyrétegű metamodell architektúra (UML 2.4.1)

- **Meta-metamodell**:
  - A réteg elsődleges feladata egy nyelv definiálása metamodellek megadásához.
  - Egy meta-metamodell általában tömörebb, mint egy általa leírt metamodell.
  - Általában kívánatos, hogy a kapcsolódó metamodellek és meta-metamodellek közös tervezési elvekkel és szerkezetekkel rendelkezzenek.
- **Metamodell**:
  - Egy metamodell egy meta-metamodell példánya, mely azt jelenti, hogy a metamodell minden eleme a meta-metamodell egy elemének példánya.
  - A réteg elsődleges feladata egy nyelv definiálása modellek megadásához.
- **Modell**:
  - Egy modell egy metamodell példánya.
  - A réteg elsődleges feladata különféle problématerületek (például szoftverek, üzleti folyamatok, követelmények) modellezése.
  - Jellemzően modellelemekből áll.
- **Futási idejű példányok**:
  - A réteg a modellben definiált modellelemek futási idejű példányait tartalmazza.

Rétegzett metamodell architektúra

- Ami egy esetben metamodell, egy másik esetben modell, lásd például: UML és MOF.
  - Az UML és a MOF is egy metamodell, melyek révén a felhasználók saját modelleket definiálhatnak.
  - A MOF nézőpontjából az UML egy felhasználói specifikáció (modell), mely a MOF metamodellen alapul.

Négyrétegű metamodell architektúra (UML 2.4.1)



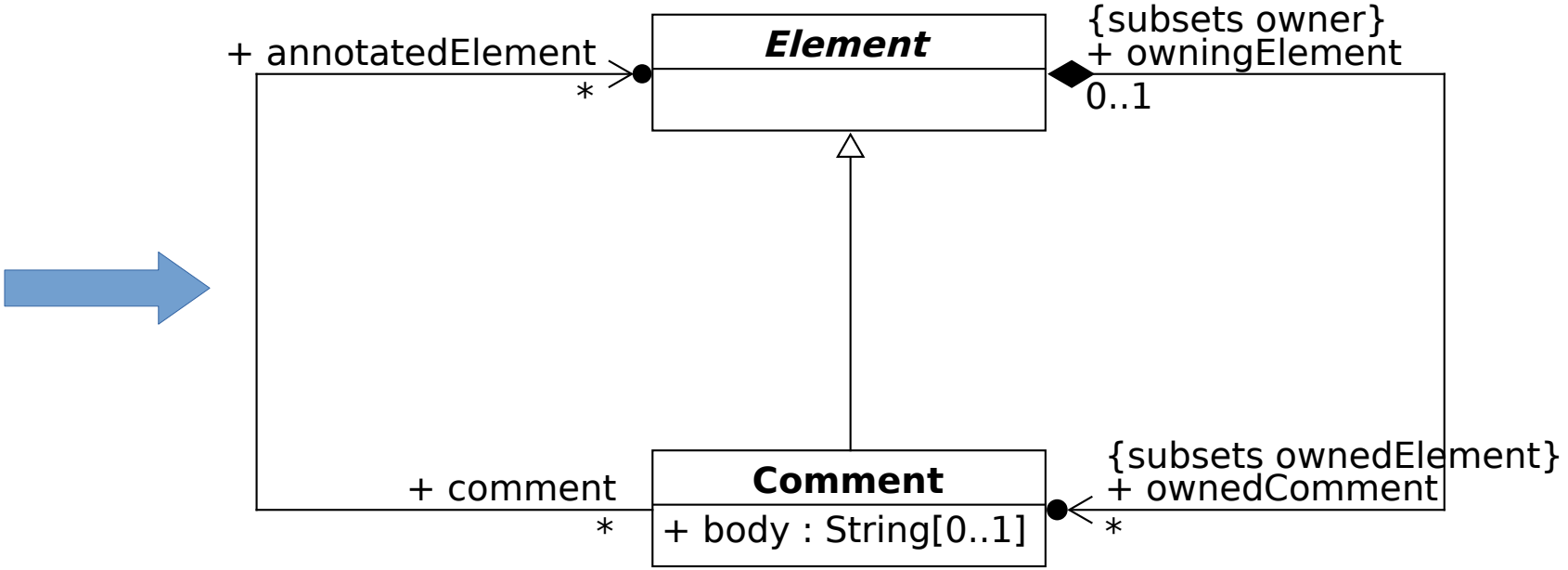
Osztályozók

- Az osztályozó egy modellelem, mely közös jellemzőkkel (tulajdonságokkal, műveletekkel) rendelkező példányok egy halmazát ábrázolja.
- Hierarchiába szervezhetők az általánosítás révén.
- Specializációi: adattípus (`DataType`), asszociáció (`Association`), interfész (`Interface`), osztály (`Class`), ...
- Jelölésmód: mint az osztályoké, a nevük megjelenítéséhez félkövér betűtípust kell használni.

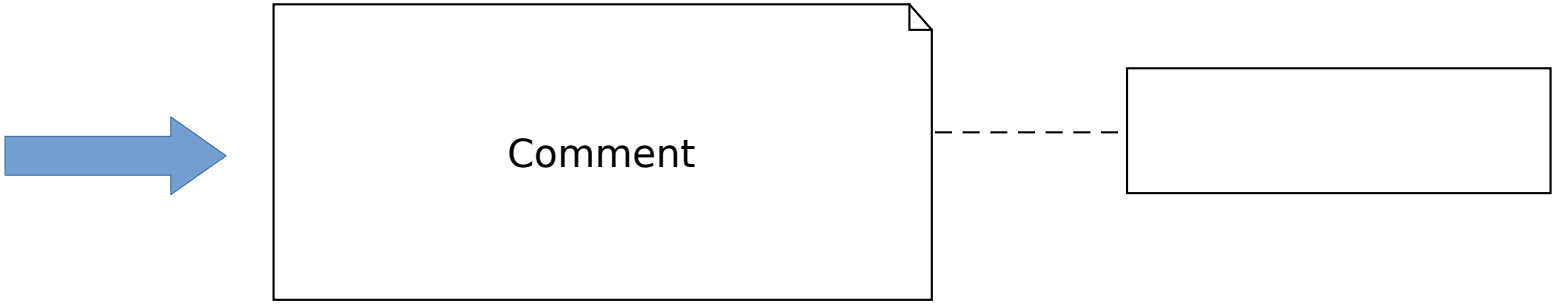
Absztrakt és konkrét szintaxis

- Példa:

Absztrakt szintaxis



Konkrét szintaxis

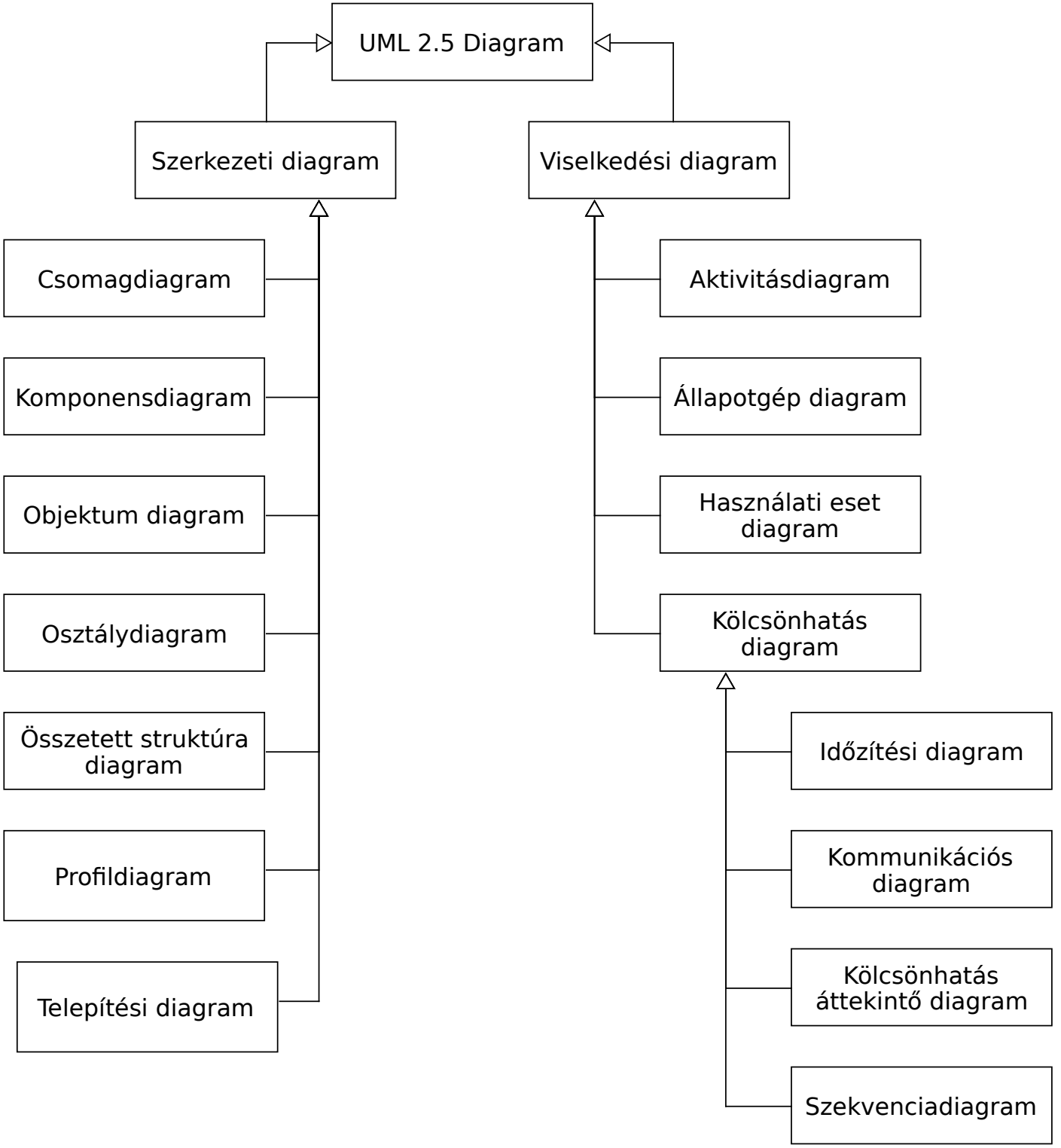


UML diagramok típusai (1)

- A diagramok két fő fajtája:
  - **Szerkezeti diagramok**:
    - Objektumok statikus felépítését mutatják egy rendszerben.
    - Az időtől független elemeket ábrázolnak.
      - Például egy alkalmazás fogalmait, melyek között lehetnek absztrakt, valós világbeli és implementációs fogalmak.
  - **Viselkedési diagramok**:
    - Az objektumok dinamikus viselkedését mutatják egy rendszerben, beleértve az együttműködést, tevékenységeket, állapotváltozásokat.
    - A rendszer dinamikus viselkedése a rendszerben történő időbeli változások sorozataként írható le.
- A felosztás nem zárja ki a különböző fajta diagramtípusok keverését. Kombinálni lehet akár szerkezeti és viselkedési diagramokat is.

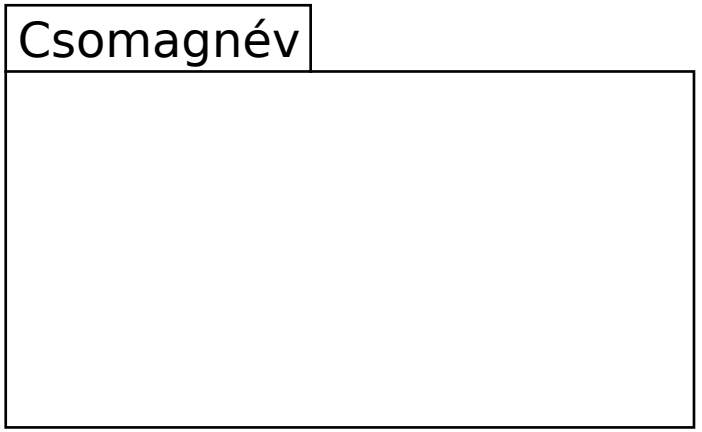
UML diagramok típusai (2)

- Szerkezeti diagramok (*structure diagrams*):
  - Csomagdiagram (*package diagram*)
  - Komponensdiagram (*component diagram*)
  - Objektum diagram (*object diagram*)
  - Osztálydiagram (*class diagram*)
  - Összetett struktúra diagram (*composite structure diagram*)
  - Profildiagram (*profile diagram*)
  - Telepítési diagram (*deployment diagram*)
- Viselkedési diagramok (behavior diagrams):
  - Aktivitásdiagram (*activity diagram*)
  - Állapotgép diagram (*state machine diagram*)
  - Használati eset diagram (*use case diagram*)
  - Kölcsönhatás diagram (*interaction diagram*):
    - Időzítési diagram (*timing diagram*)
    - Kommunikációs diagram (*communication diagram*)
    - Kölcsönhatás áttekintő diagram (*interaction overview diagram*)
    - Szekvenciadiagram (*sequence diagram*)


UML 2.5 Diagram
Szerkezeti diagram
Viselkedési diagram
Csomagdiagram
Komponensdiagram
Objektum diagram
Osztálydiagram
Összetett struktúra diagram
Profildiagram
Telepítési diagram
Aktivitásdiagram
Állapotgép diagram
Használati eset diagram
Kölcsönhatás diagram
Időzítési diagram
Kommunikációs diagram
Kölcsönhatás áttekintő diagram
Szekvenciadiagram

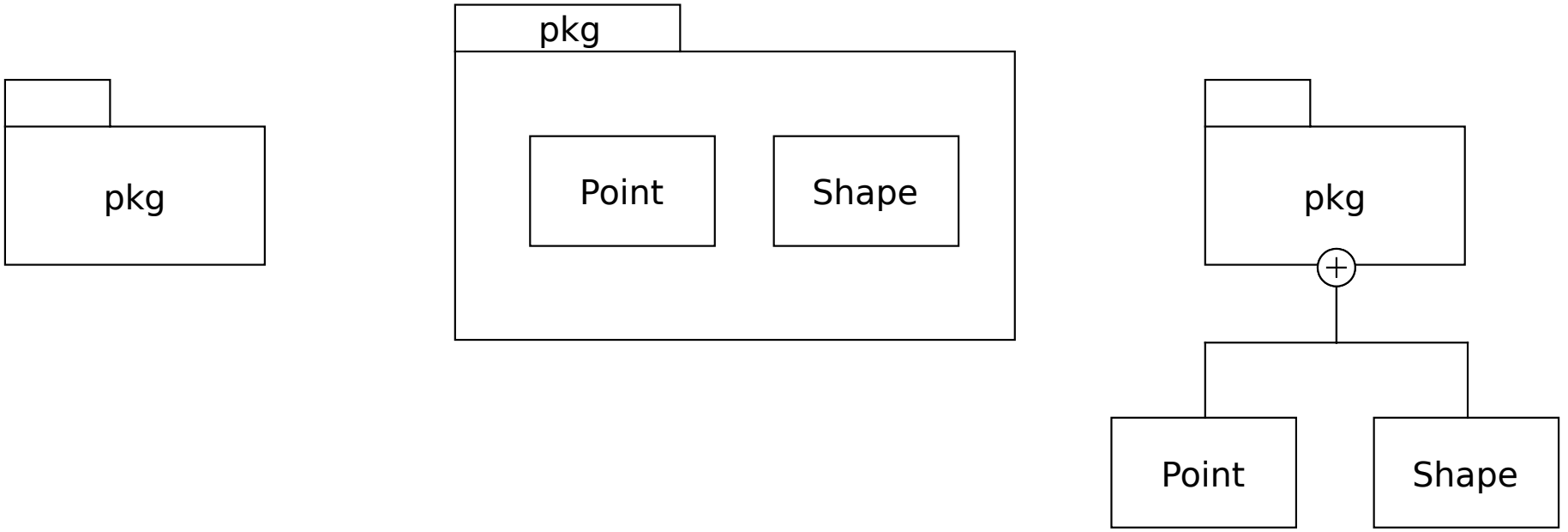
Csomagok (1)

- A csomag egy modellelemek csoportosítására szolgáló konstrukció, mely egy névteret határoz meg a tagjai számára.
- Jelölésmód:



Csomagok (2)

- A tartalmazott elemekre *csomagnév*`::`*elemnév* formájú minősített nevekkel lehet hivatkozni (például `pkg::Point`, `pkg::Shape`).



Kulcsszavak (1)

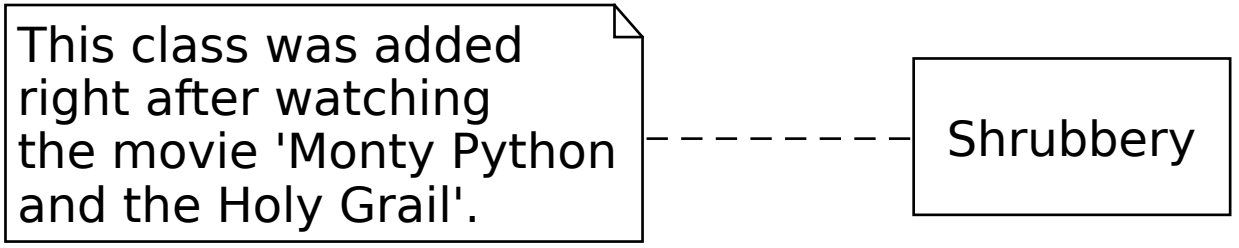
- Az UML jelölésmód szerves részét képző fenntartott szó.
- Szöveges annotációként jelenik meg egy UML grafikus elemhez kapcsolva vagy egy UML diagram egy szövegsorának részeként.
  - Minden egyes kulcsszóhoz elő van írva, hogy hol jelenhet meg.
- Lehetővé teszi azonos grafikus jelölésű UML fogalmak (metaosztályok) megkülönböztetését.
  - Lásd például az osztályokat és interfészeket («`interface`»).

Kulcsszavak (2)

- Megadásuk francia idézőjelek – « és » karakterek – között.
  - Ha a használt betűkészletben nem állnak rendelkezésre a francia idézőjelek, akkor a >> és << karakterekkel helyettesíthetők.
- Egy modellelemre több kulcsszó is vonatkozhat.
  - A kulcsszavak felsorolhatók egymás után, mindegyik külön határolók közé zárva.
  - Több kulcsszó is megadható a határolók között vessző karakterekkel elválasztva.

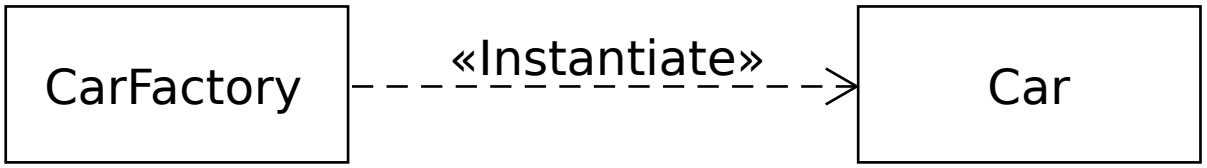
Megjegyzések

- Nincs jelentése, a modell olvasója számára hordozhat hasznos információt.
- Jelölésmód:
  - A jobb felső sarkában „szamárfüles” téglalap ábrázolja. A téglalap tartalmazza a megjegyzés törzsét.
  - Szaggatott vonal kapcsolja a magyarázandó elem(ek)hez. A vonal elhagyható, ha egyértelmű a környezetből vagy nem fontos a diagramon.



Függőségek

- Modellelemek közötti szolgáltató-kliens kapcsolatot jelent, ahol egy szolgáltató módosításának hatása lehet a kliens modellelemekre.
- Jelölésmód:
  - Két modellelem közötti szaggatott nyíl jelöli. A nyíl a függő (kliens) modellelemtől a szolgáltató modellelem felé mutat. A függőséghez megadható egy kulcsszó vagy sztereotípia.

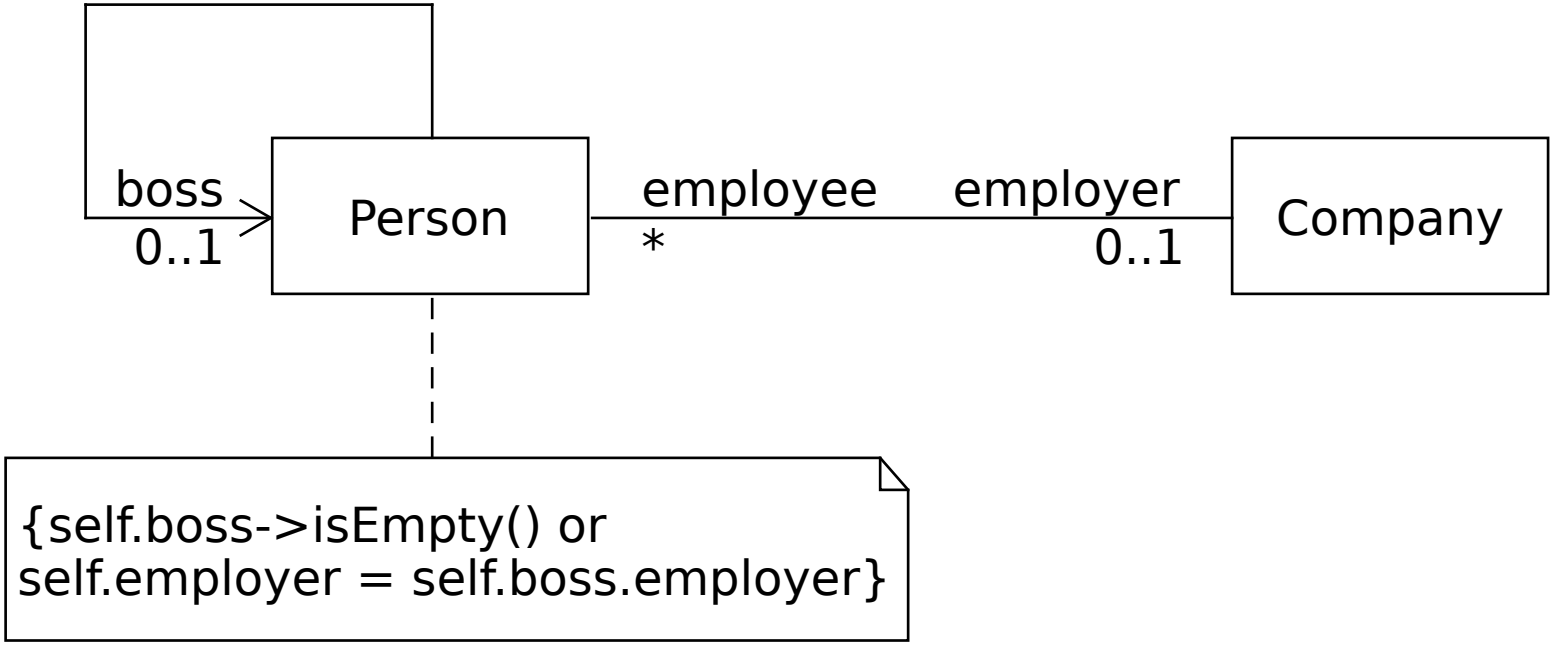


Megszorítások (1)

- Kijelentés, mely egy olyan korlátozást jelez, melyet a megszorítást tartalmazó modell tetszőleges érvényes realizációja ki kell, hogy elégítsen.
- Jelölésmód:
  - { *[név*:*] logikai-kifejezés* }
    - A felhasználói megszorítások megadása tetszőleges nyelvű szövegként. Nyelvként használható egy formális nyelv (például OCL), egy programozási nyelv (például Java) vagy természetes nyelv.

Megszorítások (2)

- Legáltalánosabban a megszorítást egy olyan megjegyzéssel adhatjuk meg, melyet minden olyan modellelemhez hozzákapcsolunk egy szaggatott vonallal, melyre vonatkozik.



Megszorítások (3)

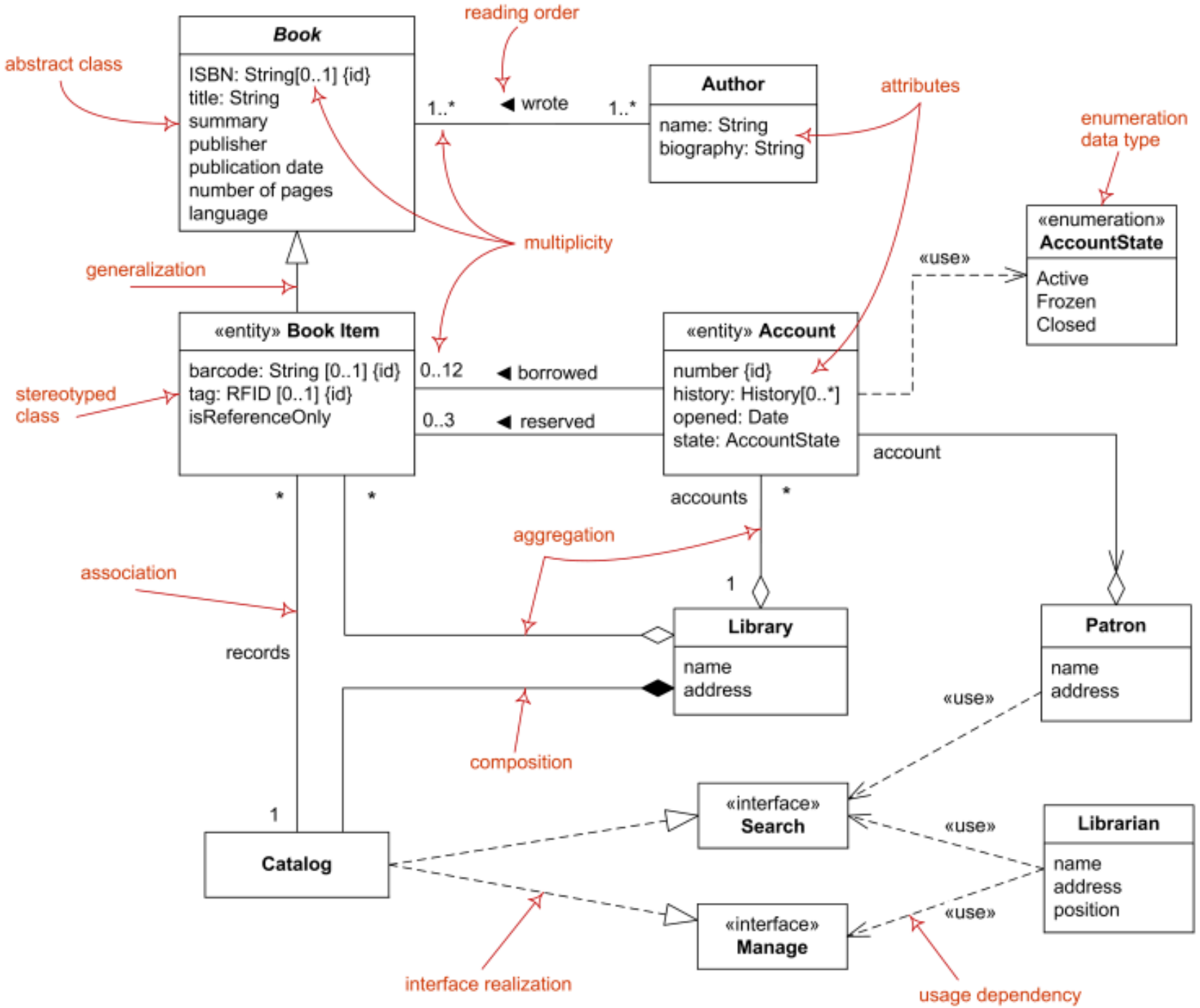
- Egyetlen modellelemre vonatkozó megszorítást a modellelem szimbólumának közelébe (ha van neve, lehetőleg annak közelébe) kell írni.
- Olyan modellelemeknél, melyeket egy karakterlánc jelöl (például az attribútumok), a megszorítást a modellem szöveges reprezentációja után írjuk.

Osztálydiagram (1)

- Egy osztálydiagram az objektumok típusait írja le egy rendszerben és a köztük fennálló különféle statikus kapcsolatokat. Az osztálydiagramok mutatják az osztályok tulajdonságait és műveleteit is, valamint azokat a megszorításokat, melyek az objektumok összekapcsolására vonatkoznak.
  - Forrás: Martin Fowler. *UML Distilled – A Brief Guide to the Standard Object Modeling Language*. Third Edition. Addison-Wesley, 2003.
- *„[…] a class diagram is a diagram where the primary symbols in the contents area are class symbols.”*
  - Forrás: *OMG UML Version 2.5.1*

Osztálydiagram (2)

- Osztálydiagramok fajtái (Störrle):
  - **Elemzési**: Az elemzési szinten az osztályok az alkalmazási szakterület fogalmai, az osztálydiagram a szakterület felépítését modellezi.
  - **Tervezési**: Megjelennek az osztályokban a megvalósítás módjának technikai aspektusai.
  - **Megvalósítási**: Az osztályok egy implementációs nyelv (például C++, Java, …) konstrukcióival ekvivalensek.



Forrás: https://www.uml-diagrams.org/class-diagrams-overview.html

Osztály

- Jelölésmód:

Láthatóság

- + (nyilvános)
- - (privát)
- # (védett)
- ~ (csomagszintű)

Számosság

- Megszorítást fejez ki egy kollekció elemeinek számára.
  - Az elemek száma nem lehet kisebb az adott alsó korlátnál.
  - Az elemek száma nem lehet nagyobb az adott felső korlátnál, ha az nem `*`.
- Jelölésmód:
  - *[alsó_korlát `..`] felső_korlát*
    - Az alsó korlát nemnegatív egész, a felső korlát nemnegatív egész vagy a „korlátlan” jelentésű `*`.
    - Ha az alsó és felső korlát egyenlő, akkor használható önmagában csak a felső korlát.
      - Például `1` ekvivalens az `1..1` számossággal, hasonlóan `5` ekvivalens az `5..5` számossággal.
    - A `0..*` számosság helyett használható az ekvivalens `*` jelölés.

Tulajdonságok

- Egy tulajdonság egy attribútumot vagy egy asszociációvéget ábrázol.
- Jelölésmód:
  - *[^] [láthatóság] [/] név [*`:` *típus] [*`[` *számosság* `]`*]*
    [= alapérték] [`{` *módosító [*`,` *módosító]** `}`*]*
    - A ^ azt jelzi, hogy a tulajdonság örökölt (UML 2.5).
    - A / azt jelzi, hogy a tulajdonság származtatott.
    - A számosság elhagyásakor az alapértelmezés 1.
    - Módosító: például `readOnly`, `ordered`, `unordered`, `unique`, …

Paraméterek, paraméterlisták

- Paraméterlista:
  - *paraméter [`,` paraméter]**
- Paraméter:
  - *[irány] név* `:` *típus [*`[` *számosság* `]`*]*
    [= alapérték] [`{` *tulajdonság [`,` tulajdonság]** `}`*]*
    - Irány: `in` (alapértelmezés), `out`, `inout`, `return`
    - Tulajdonság: `nonunique`, `ordered`, `seq`/`sequence`, `unique`, `unordered`

Műveletek

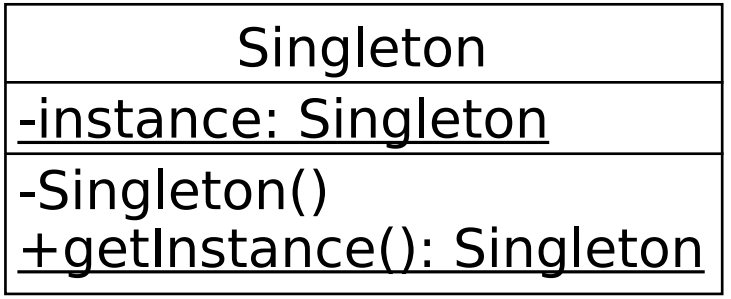
- Jelölésmód:
  - *[^] [láthatóság] név* (*[paraméterlista]*)
    [: *típus] [*[*számosság*]*]*
    [{ *tulajdonság [*, *tulajdonság]** }*]*
    - Tulajdonság: `nonunique`, `ordered`, `query`, `redefines` *név*, `seq`/`sequence`, `unique`, `unordered`, *megszorítás*
      - **query**: azt jelenti, hogy a művelet nem változtatja meg a rendszer állapotát.

Példa osztályra

Person
-title: String[0..1] -name: String -birthDate: Date -/age: int {age >= 0}
+Person(title: String, name: String, birthDate: Date) +Person(name: String, birthDate: Date) +getTitle(): String {query} +setTitle(title: String) +getName(): String {query} +setName(name: String) +getBirthDate(): Date {query} +setBirthDate(birthDate: Date) +getAge(): int {query}

Statikus attribútumok és műveletek

- A statikus attribútumokat és műveleteket aláhúzás jelöli.
  - Példa:

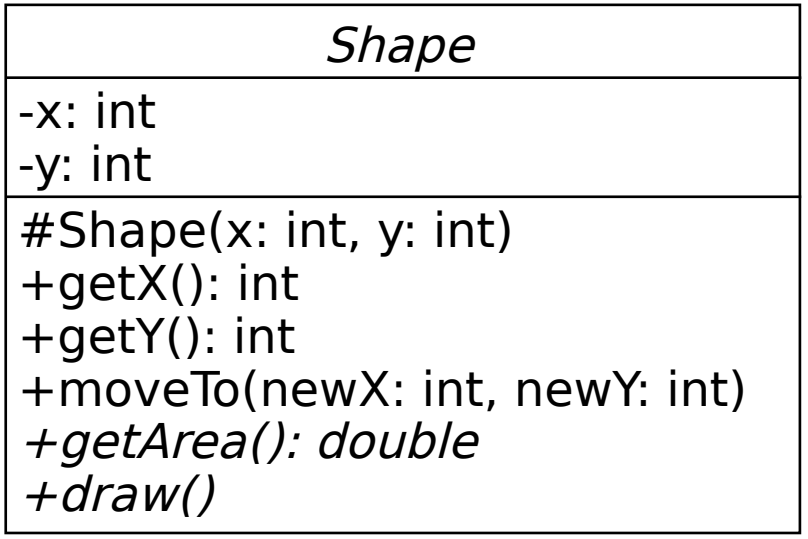


Absztrakt osztályok (1)

- Nem példányosítható osztály (osztályozó).
- Jelölésmód:
  - Szedjük az osztály (osztályozó) nevét dőlt betűvel és/vagy a név után vagy alatt adjuk meg az `{abstract}` szöveges annotációt.
  - Az UML 2.5.1 nem rendelkezik az absztrakt műveletek jelölésmódjáról! (Személyes vélemény: ez valószínűleg hiba.)

Absztrakt osztályok (2)

- Példa:



Asszociációk (1)

- Szemantikus viszonyt jelent, mely osztályozók példányai között állhat fenn.
  - Azt fejezi ki az asszociáció, hogy kapcsolatok lehetnek olyan példányok között, melyek megfelelnek az asszociált típusoknak vagy implementálják azokat.
- Legalább két végük van.
  - Két végű asszociáció: **bináris asszociáció**.
- Egy **kapcsolat** (***link***) egy asszociáció egy példánya.
  - Azaz egy olyan *n*-es, mely minden véghez a vég típusának egy példányát tartalmazza.

Asszociációk (2)

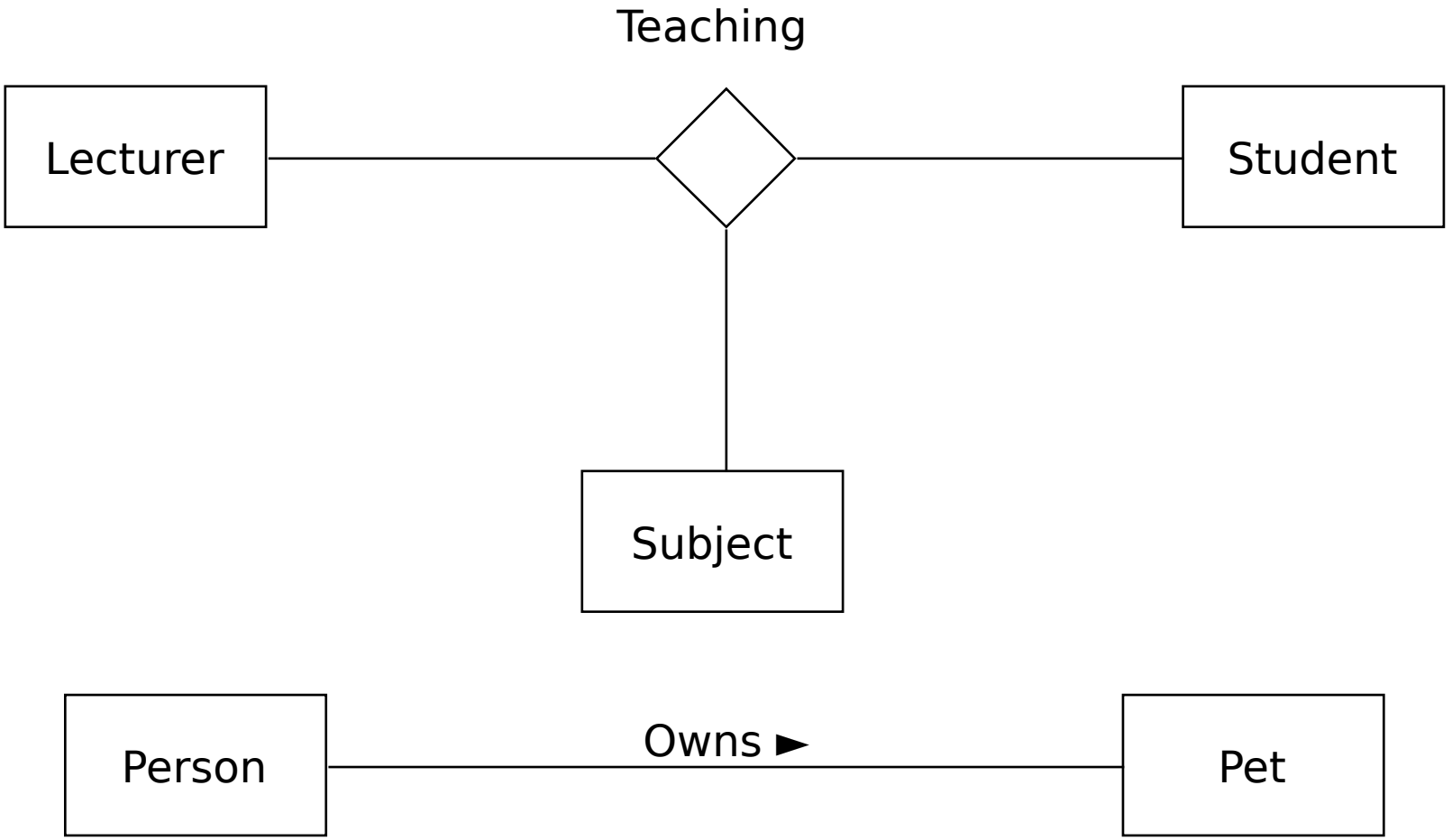
- Jelölésmód:
  - Bármely asszociáció ábrázolható egy csúcsára állított rombusszal, melyet minden egyes vég esetén egy folytonos vonal köt össze azzal az osztályozóval, mely a vég típusa. Kettőnél több végű asszociáció csak így ábrázolható.
  - Egy bináris asszociációt általában két osztályozót összekötő folytonos vonal ábrázol, vagy egy osztályozót önmagával összekötő folytonos vonal.

Asszociációk (3)

- Az asszociáció szimbólumához megadható név (ne legyen túl közel egyik véghez sem).
  - Folytonos vonallal ábrázolt bináris asszociáció neve mellett vagy helyén elhelyezhető egy tömör háromszög, mely a vonal mentén az egyik vég felé mutat és az olvasási irányt jelzi. Ez a jelölés csupán dokumentációs célokat szolgál.

Asszociációk (4)

- Példa:



Asszociációk (5)

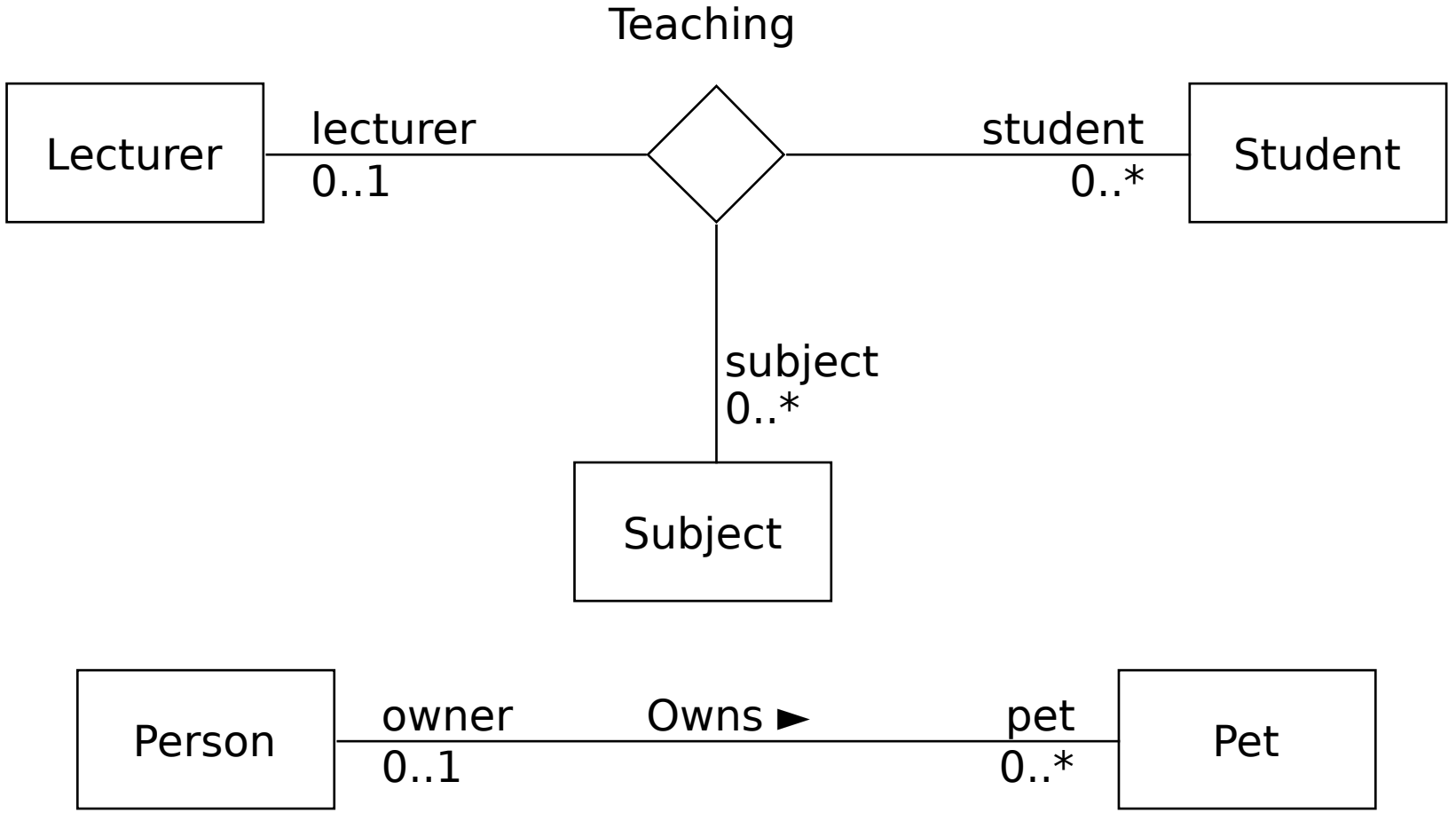
- **Asszociáció vég**: az asszociációt ábrázoló vonal és egy osztályozót ábrázoló ikon (gyakran egy doboz) kapcsolata.
  - A vonal végének közelében elhelyezhető (egyik sem kötelező):
    - Név (gyakran szerepkörnek nevezik)
    - Számosság (ha nincs megadva, akkor semmilyen feltevéssel nem élhetünk a számosságról)
    - Módosító (lásd a tulajdonságoknál)
    - Láthatóság
  - A vonal végén egy nyílt nyílhegy azt jelzi, hogy a vég navigálható, egy $\times$ pedig azt, hogy a vég nem navigálható.

Asszociációk (6)

- Egy asszociáció vég számosságának jelentése:
  - A példányok számát adja meg a végen arra az esetre, amikor a többi (*n* - 1) vég mindegyikén egy-egy értéket rögzítünk.

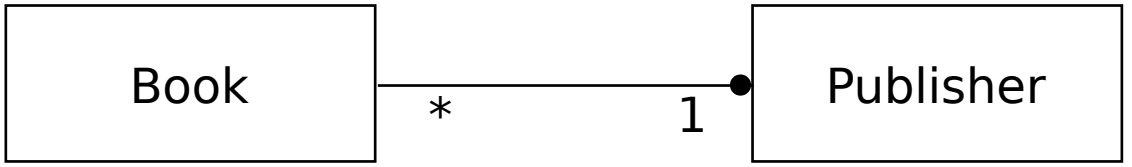
Asszociációk (7)

- Példa:



Asszociációk (8)

- Az osztályozó és a vonal érintkezési pontjában elhelyezhető egy kis tömör kör (a továbbiakban pontnak nevezzük).
  - A pont azt mutatja, hogy a modell tartalmaz egy tulajdonságot, melynek típusát a pont által érintett osztályozó ábrázolja. Ez a tulajdonság a másik végen lévő osztályozóhoz tartozik. Ebben az esetben szokás a tulajdonságot elhagyni az osztályozó attribútum rekeszéből.
  - A pont hiánya azt jelzi, hogy a vég magához az asszociációhoz tartozik.

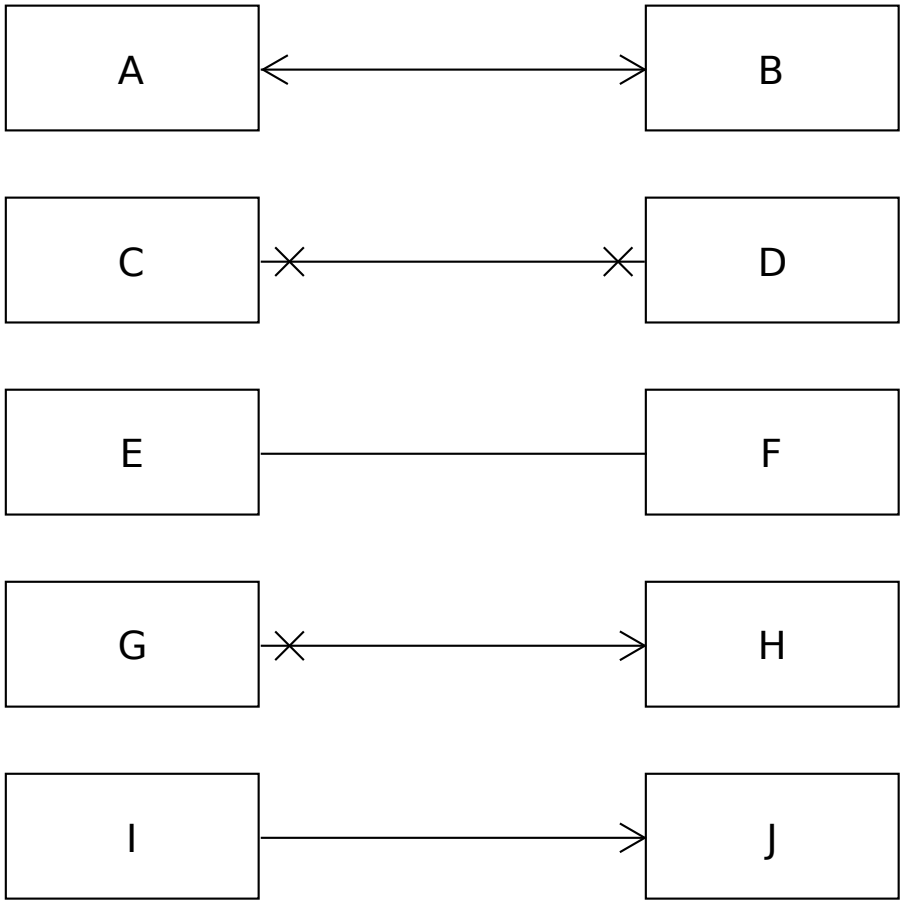


Asszociációk (9)

- A navigálhatóság azt jelenti, hogy a kapcsolatokban résztvevő példányok futásidőben hatékonyan érhetők el az asszociáció többi végén lévő példányokból.
  - Implementáció-specifikus azt a mechanizmus, mely révén hatékony elérés történik.
  - Az osztályokhoz tartozó asszociációvégek mindig navigálhatók, az asszociációkhoz tartozók lehetnek navigálhatók és nem navigálhatók.

Asszociációk (10)

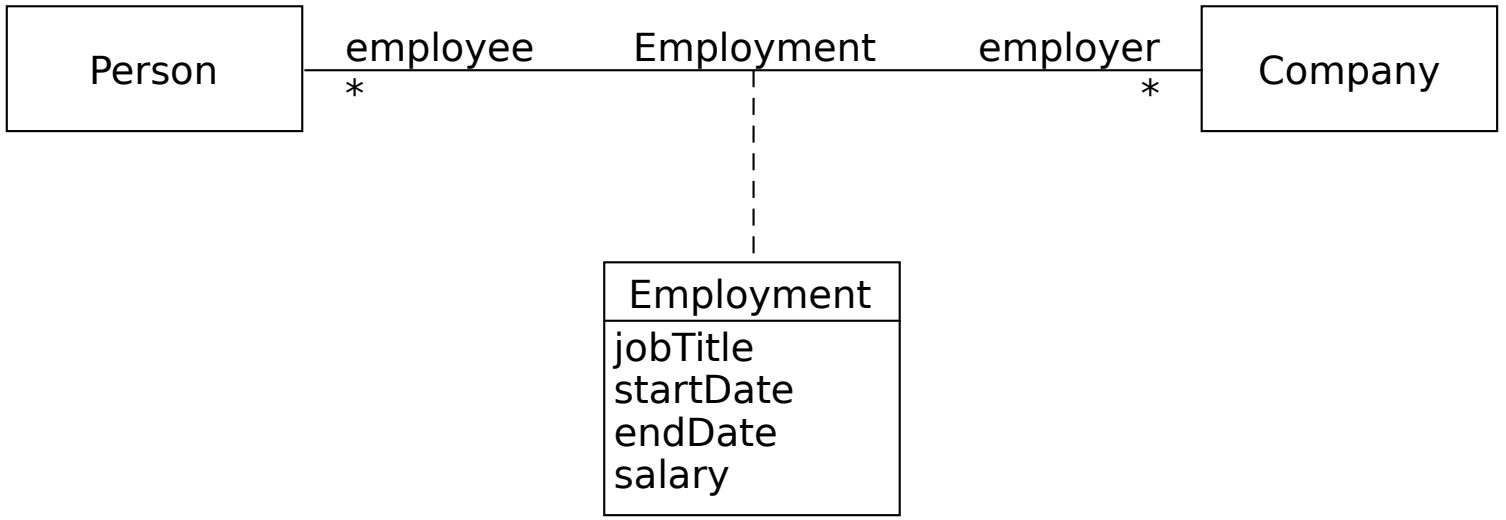
- Mindkét vég navigálható.
- Egyik vég sem navigálható.
- A navigálhatóság nem meghatározott. (Olyan diagramon, mely csak az egyik irányban navigálható asszociációkhoz használ nyilakat, ez valószínűleg kétirányú navigálhatóságot jelent.)
- Az egyik vég navigálható, a másik nem.
- Az egyik vég navigálható, a másik nem. (Olyan diagramon, mely csak az egyik irányban navigálható asszociációkhoz használ nyilakat és nem használ kereszteket.)

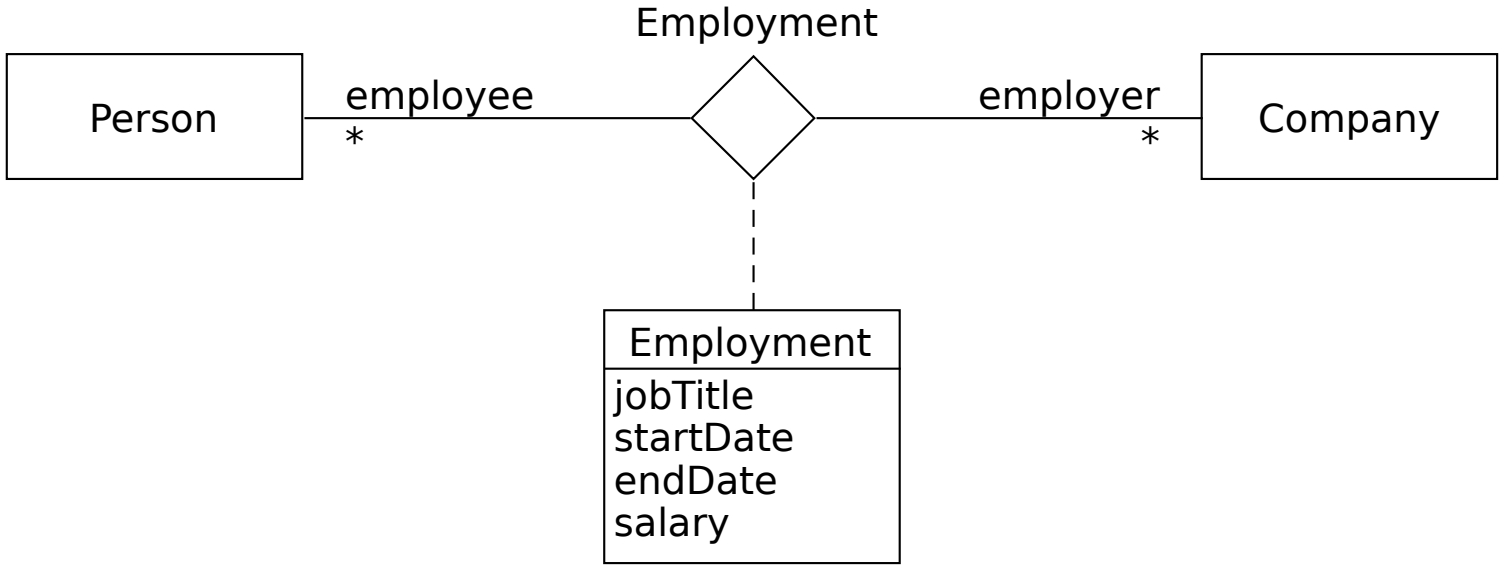


Asszociációs osztályok (1)

- Modellelem, mely osztály és asszociáció is egyben. Jellemzőket definiál magához az asszociációhoz.
- Jelölésmód:
  - Egy osztály szimbólum jelöli, melyet szaggatott vonal köt össze az asszociációt ábrázoló útvonallal vagy rombusszal.
  - Az útvonal/rombusz és az osztály szimbólum ugyanazt a modellelemet jelölik, a nevük is meg kell, hogy egyezzen.

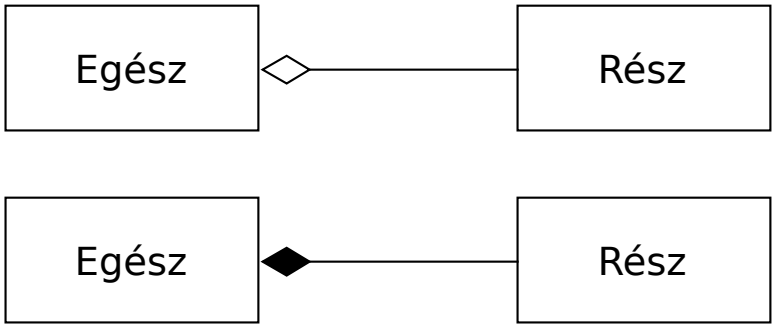
Asszociációs osztályok (2)





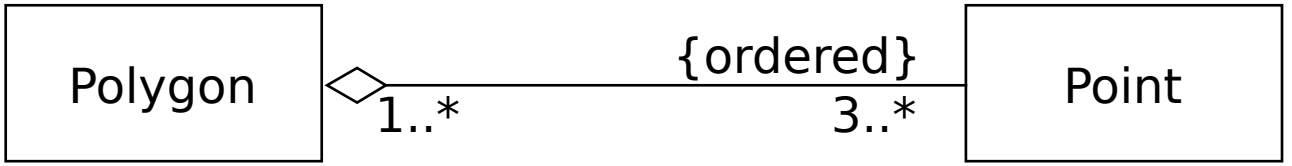
Egész-rész kapcsolat (1)

- A bináris asszociációk egész-rész kapcsolatot kifejező fajtái:
  - **Aggregáció** (***shared aggregation***, ***aggregation***): Egy rész objektum egyidejűleg több aggregációs objektumhoz is tartozhat, a részek és az aggregációs objektum egymástól függetlenül is létezhetnek.
  - **Kompozíció** (***composite aggregation***, ***composition***): Az aggregáció erősebb formája. Egy rész objektum legfeljebb egy kompozit objektumhoz tartozhat. A kompozit objektum törlésekor az összes rész objektum vele együtt törlődik.
- Egy bináris asszociáció egyik vége jelölhető meg csak aggregációként vagy kompozícióként.
- Jelölésmód:



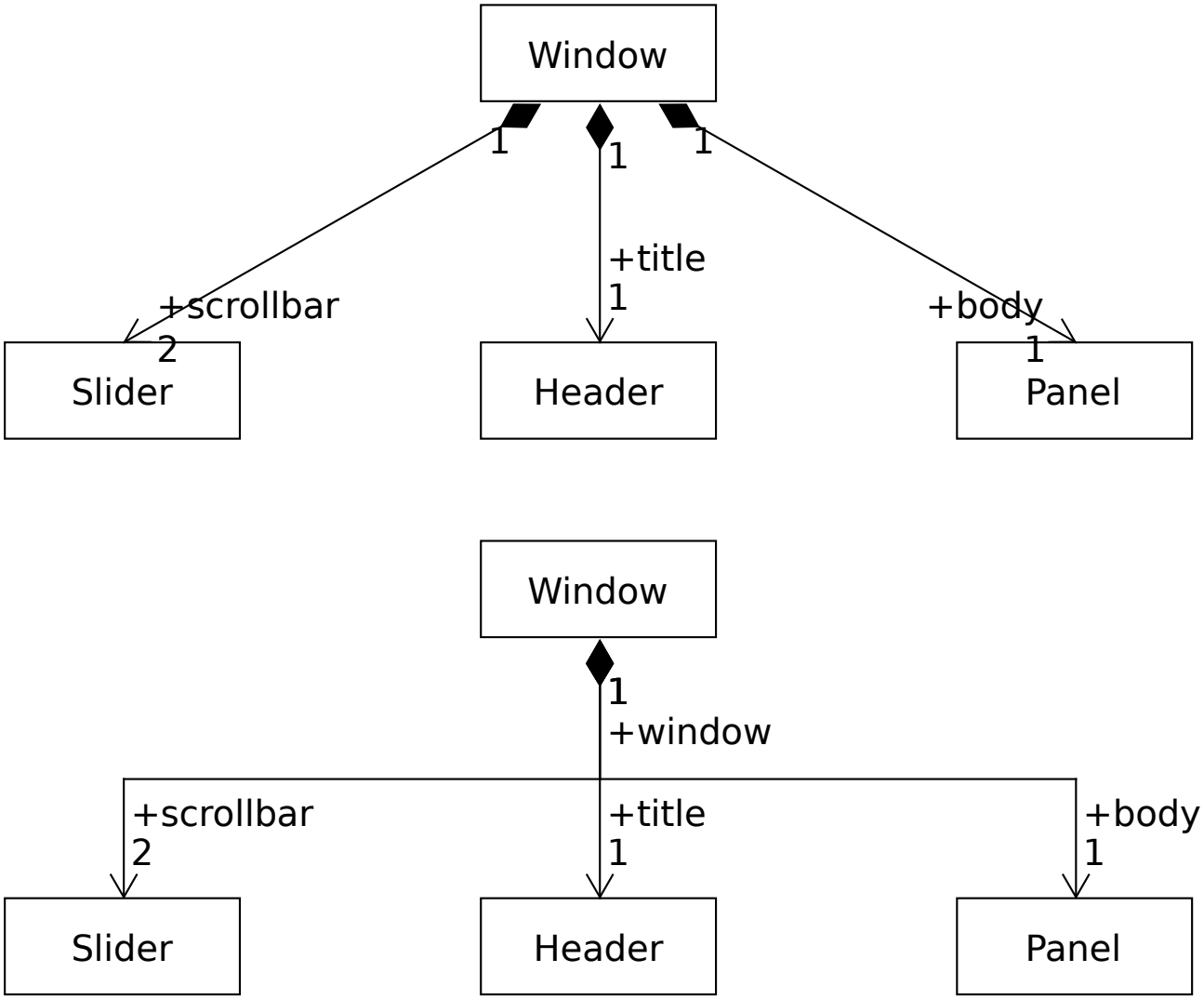
Egész-rész kapcsolat (2)

- Példa aggregációra:



Egész-rész kapcsolat (3)

- Példa kompozícióra:

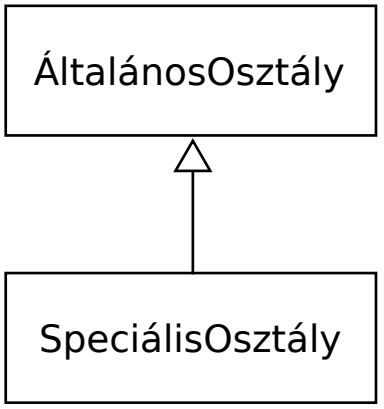


Általánosítás (1)

- Az általánosítás egy általánosítás/specializáció kapcsolatot határoz meg osztályozók között. Egy speciális osztályozót kapcsol össze egy általánosabb osztályozóval.
- Az általánosítás/specializáció reláció tranzitív lezártja szerint értelmezzük egy osztályozó általánosításait és specializációit.
  - A közvetlen általánosításokat a speciális osztályozó **szülő**jének nevezzük, osztályok esetén **ősosztály**nak.

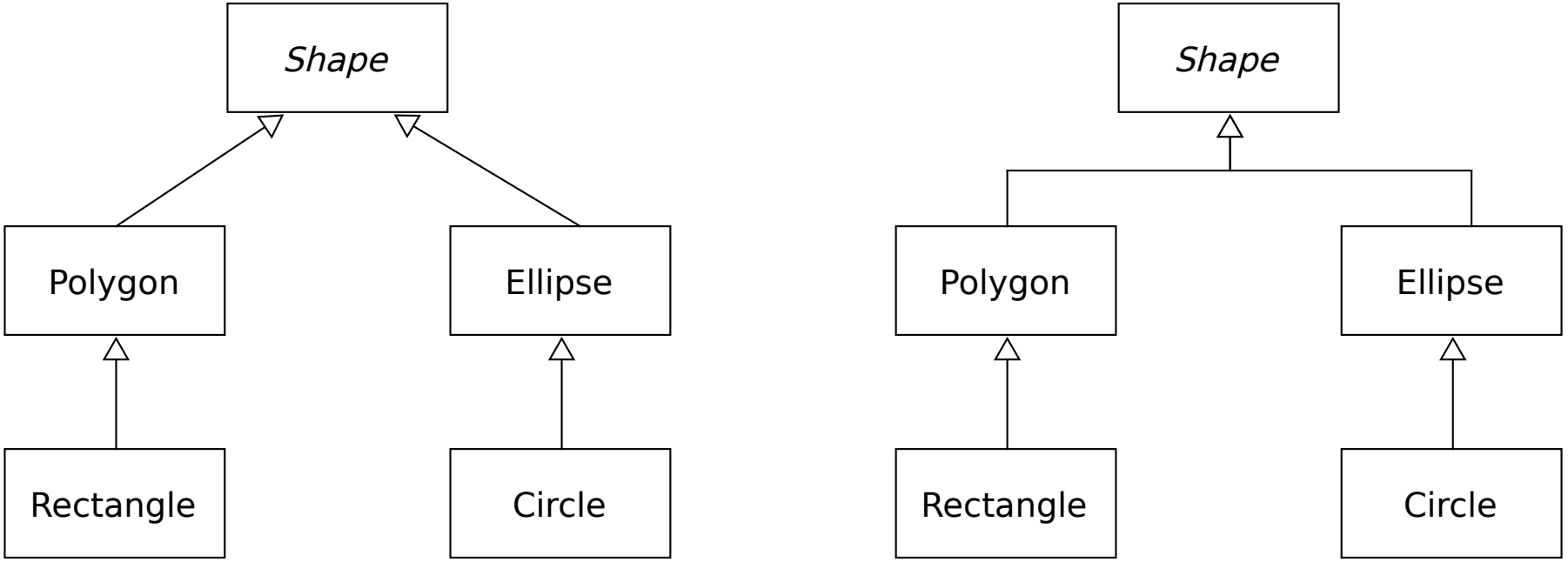
Általánosítás (2)

- Egy osztályozó egy példánya minden általánosításának példánya.
- A speciális osztályozó örökli az általános osztályozó bizonyos tagjait.
- Jelölésmód:



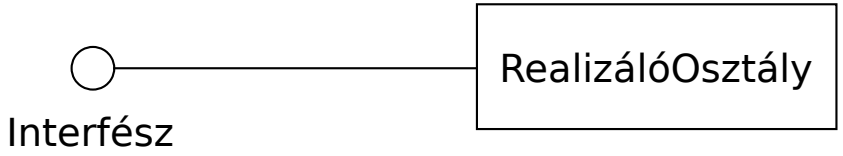
Általánosítás (3)

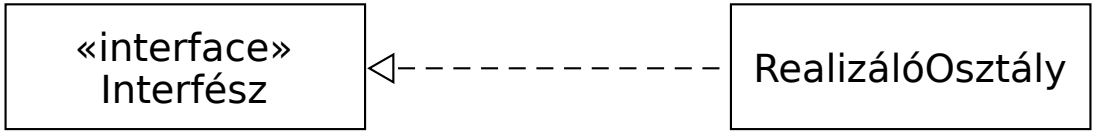
- Példa:



Interfészek (1)

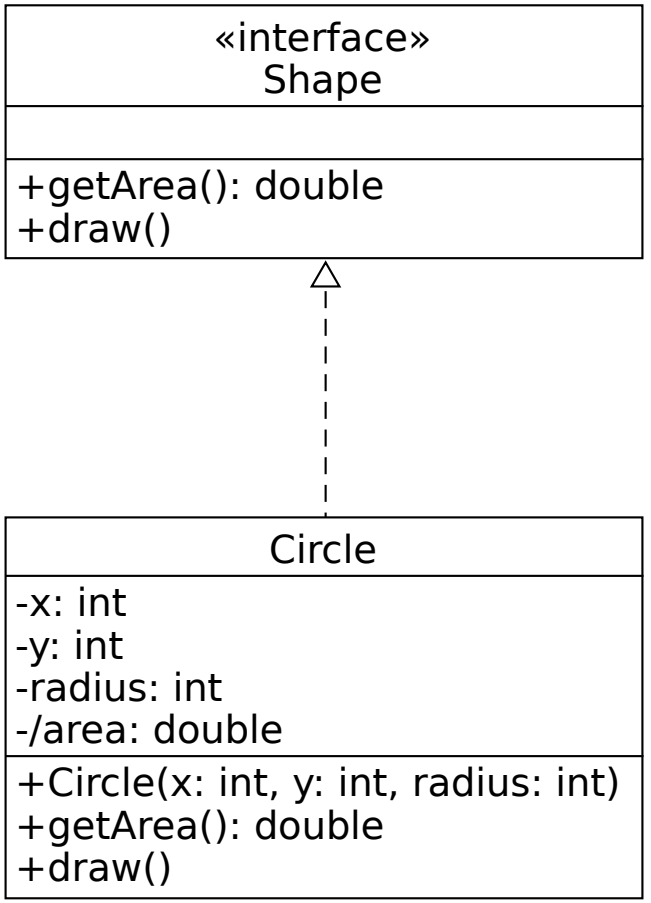
- Az interfész egy olyan fajta osztályozó, mely nyilvános jellemzőket és kötelezettségeket deklarál, melyek együtt egy koherens szolgáltatást alkotnak. Az interfész egy szerződést határoz meg, az interfészt realizáló bármely osztályozó eleget kell, hogy tegyen a szerződésnek.
- Az interfészek nem példányosíthatók. Osztályozók implementálják vagy realizálják az interfész specifikációt, mely azt jelenti, hogy az interfész specifikációnak megfelelő nyilvános felületet nyújtanak.
- Jelölésmód:





Interfészek (2)

- Példa:



UML eszközök

- UML eszközök által nyújtható funkciók:
  - **Előretervezés** (***forward engineering***):
  - **Visszatervezés** (***reverse engineering***):

UML eszközök: szabad szoftverek (1)

- GUI eszközök:
  - *Gaphor* (programozási nyelv: Python; licenc: *Apache License 2.0*) https://gaphor.org/ https://github.com/gaphor/gaphor
  - *Modelio* (operációs rendszer: Linux, macOS, Windows; licenc: GPLv3) https://www.modelio.org/ https://github.com/ModelioOpenSource/Modelio
  - *Papyrus* (platform: Eclipse; licenc: *Eclipse Public License v1.0*) https://eclipse.dev/papyrus/
  - *UMLet* (programozási nyelv: Java; licenc: GPLv3) https://www.umlet.com/ https://github.com/umlet/umlet
    - Eclipse bővítmény: https://marketplace.eclipse.org/content/umlet-uml-tool-fast-uml-diagrams
    - Visual Studio Code kiterjesztés: https://marketplace.visualstudio.com/items?itemName=TheUMLetTeam.umlet
    - Webalkalmazás: *UMLetino* https://www.umletino.com/

UML eszközök: szabad szoftverek (2)

- Diagramok létrehozásához egy szakterület specifikus nyelvet használó eszközök:
  - *Mermaid* (programozási nyelv: JavaScript; licenc: *MIT License*) https://mermaid.js.org/ https://github.com/mermaid-js/mermaid
  - *nomnoml* (programozási nyelv: JavaScript; licenc: *MIT License*) https://nomnoml.com/ https://github.com/skanaar/nomnoml
  - *PlantUML* (programozási nyelv: Java; licenc: GPLv3) https://plantuml.com/ https://github.com/plantuml/plantuml
- UML osztálydiagramokat előállító docletek:
  - *UMLDoclet* (programozási nyelv: Java; licenc: *Apache License 2.0*) https://github.com/talsma-ict/umldoclet

UML eszközök: nem szabad szoftverek

- GUI eszközök:
  - *Altova UModel* (platform: Windows) https://www.altova.com/umodel
  - *IntelliJ IDEA Ultimate Edition* (platform: Linux, macOS, Windows) https://www.jetbrains.com/idea/
    - Lásd: *UML class diagrams* https://www.jetbrains.com/help/idea/class-diagram.html
  - *Microsoft Visio* (platform: Windows) https://www.microsoft.com/en/microsoft-365/visio/flowchart-software
  - *Enterprise Architect* (platform: Windows) https://sparxsystems.com/
  - *StarUML* (platform: Linux, macOS, Windows) https://staruml.io/
  - *Visual Paradigm* (platform: Linux, macOS, Windows) https://www.visual-paradigm.com/
    - *Visual Paradigm Community Edition* https://www.visual-paradigm.com/download/community.jsp

PlantUML (1)

- UML diagramok létrehozására szolgáló szabad és nyílt forrású szoftver. https://plantuml.com/ https://github.com/plantuml/plantuml
  - Programozási nyelv: Java
  - Licenc: GPLv3
- Támogatott diagramok: osztálydiagram, objektum-diagram, szekvencia-diagram, használati eset-diagram, …
- Egy egyszerű nyelvet biztosít a diagramok definiálásához.
  - Osztálydiagram szintaxis: https://plantuml.com/class-diagram
  - Állománynév végződés: `.puml`

PlantUML (2)

- Bizonyos diagramok előállításához az alábbi programot használja:
  - Graphviz (platform: Linux, macOS, Windows; programozási nyelv: C; licenc: *Common Public License v1.0*) https://graphviz.org/ https://gitlab.com/graphviz/graphviz/
- Számos fejlesztőeszköz használja, integrálja.
  - Lásd: *Running* https://plantuml.com/running

PlantUML (3)

- Example:

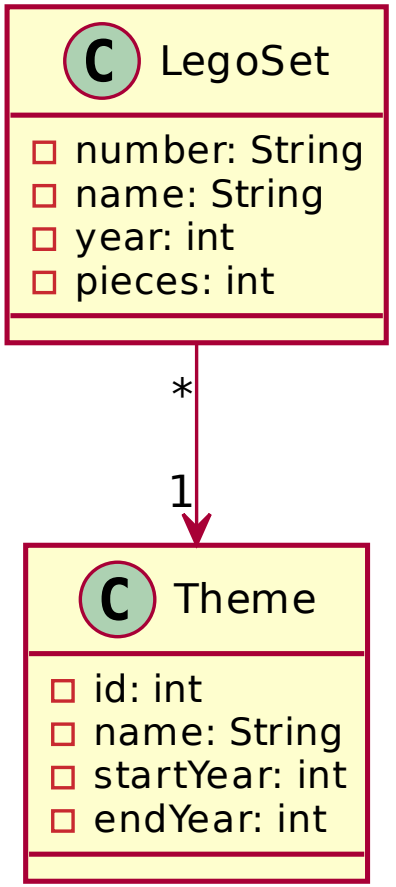
```
@startuml

class LegoSet {
  -number: String
  -name: String
  -year: int
  -pieces: int
}

class Theme {
  -id: int
  -name: String
  -startYear: int
  -endYear: int
}

LegoSet "*" --> "1" Theme

@enduml
```



PlantUML (4)

- Kapcsolódó eszközök:
  - *PlantText* https://www.planttext.com/
  - *UML Reverse Mapper* (licenc: *Apache License 2.0*) https://github.com/iluwatar/uml-reverse-mapper
    - Parancssori eszköz és Apache Maven bővítmény, mely automatikusan egy UML osztálydiagramot állít elő Java kódból.
    - Egy Graphviz `.dot` állományt vagy egy PlantUML `.puml` állományt ír ki eredményként.

PlantUML (5)

- IDE támogatás:
  - **Eclipse**: https://github.com/hallvard/plantuml
  - **IntelliJ IDEA**: *PlantUML integration*
    https://plugins.jetbrains.com/plugin/7017-plantuml-integration
    https://github.com/esteinberg/plantuml4idea/
  - **NetBeans**: *PlantUML-NB*
    https://plugins.netbeans.apache.org/catalogue/?id=58
    https://github.com/matthiasblaesing/plantuml-nb
  - **Visual Studio Code**: *PlantUML*
    https://marketplace.visualstudio.com/items?itemName=jebbs.plantuml
    https://github.com/qjebbs/vscode-plantuml

Ajánlott irodalom

- Angol nyelven:
  - Martin Fowler. *UML Distilled – A Brief Guide to the Standard Object Modeling Language*. Third Edition. Addison-Wesley, 2003. https://martinfowler.com/books/uml.html
  - Robert A. Maksimchuk, Eric J. Naiburg. *UML for Mere Mortals*. Addison-Wesley, 2004.
  - Russ Miles, Kim Hamilton. *Learning UML 2.0*. O'Reilly Media, 2006.
  - Martina Seidl, Marion Scholz, Christian Huemer, Gerti Kappel. *UML @ Classroom – An Introduction to Object-Oriented Modeling*. Springer, 2015. http://www.uml.ac.at/en/
- Magyarul:
  - Robert A. Maksimchuk, Eric J. Naiburg. *UML földi halandóknak*. Kiskapu Kft., 2006.
  - Harald Störrle. *UML 2*. Panem, 2007.
  - Sike Sándor, Varga László. *Szoftvertechnológia és UML*. 2. kiadás. ELTE Eötvös Kiadó, 2008.