

Unified Modeling Language (UML)

Jeszenszky Péter
jeszenszky.peter@inf.unideb.hu

Utolsó módosítás: 2018. december 13.

Object Management Group (OMG)

- Technológiai szabványokat fejlesztő nyílt tagságú nemzetközi non-profit konzorcium, melyet 1989-ben alapítottak. <http://www.omg.org/>
- Szabványok:
 - Business Process Model and Notation (BPMN) <https://www.omg.org/spec/BPMN/>
 - Common Object Request Broker Architecture (CORBA) <http://www.omg.org/corba/>
 - MetaObject Facility (MOF) <http://www.omg.org/mof/>
 - Model Driven Architecture (MDA) <http://www.omg.org/mda/>
 - Systems Modeling Language (SysML) <https://www.omg.org/spec/SysML/>
 - Unified Modeling Language (UML) <http://www.uml.org/>
 - XML Metadata Interchange (XMI) <http://www.omg.org/spec/XMI/>
 - ...

3

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* <http://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.)

2

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* <http://www.uml-diagrams.org/>

4

Az aktuális szabvány

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

5

UML 2.5.1

- Kevés változás a 2.5 verzióhoz képest, de ezek sem lényegesek az átlagos felhasználó számára.
 - Lásd: *Specification changebar*
<http://www.omg.org/spec/UML/2.5.1/PDF/changebar>

7

UML 2.5

- *OMG Unified Modeling Language (OMG UML) Version 2.5*. March 2015. <http://www.omg.org/spec/UML/2.5/>
 - A specifikáció formailag az előző verzió kismértékű átdolgozásának tekinthető (*minor revision*).
 - A specifikáció előző verzióját nagymértékben átírták, a könnyebb olvashatóság érdekében eltávolítva a redundáns részeket és pontosításokkal élve.
 - A specifikációt például úgy szervezték át, hogy a lehető legkevesebb előre hivatkozás legyen benne.
 - A teljes UML specifikációt egyetlen dokumentum tartalmazza.
 - Korábban két specifikáció volt (*Infrastructure*, *Superstructure*), lásd: <http://www.omg.org/spec/UML/2.4.1/>

6

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.
- Számos különböző célra használható:
 - Lekérdezőnyelvként.
 - Osztályokra és típusokra vonatkozó invariáns feltételek előírására.
 - Műveletek és metódusok elő- és utófeltételeinek leírására.
 - ...

8

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.

9

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. <http://www.omg.org/spec/UML/2.5.1/>

11

Diagram Definition (DD)

- Lehetővé teszi grafikus jelölésmódok modellezését és cseréjét, speciálisan gráf jellegű diagramokét, melyeket például az UML, a SysML és a BPMN használ, ahol a jelölésmód a MOF-fal definiált absztrakt szintaxishoz kötődik.
- Egy keretrendszert biztosít más modellező nyelvek specifikációinak a saját diagramjaik definiálásához.

10

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.

12

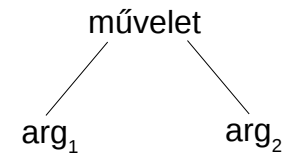
Metaosztály

- 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 3.6.4 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 metamodelben (például Element, Classifier, ...).

13

Szintaxis (2)

- Absztrakt szintaxis:
- Konkrét szintaxis:



- Infix alak:
 - $(1 + 2) * 3$
- Prefix alak:
 - $(* (+ 1 2) 3)$
- Postfix alak:
 - $((1 2 +) 3 *)$

15

Szintaxis (1)

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

14

Szintaxis (3)

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

16

Szemantika

- „*Colorless green ideas sleep furiously.*” / „Színtelen zöd eszmék dühödten alszanak.”
 - Forrás: Noam Chomsky. *Syntactic Structures*. Mouton & Co., 1957.

17

Meta Object Facility (MOF)

- **Szakterület-specifikus nyelv (DSL, *domain-specific language*)**: Egy bizonyos fajta problémára koncentrálnó számítógépes nyelv, nem pedig egy általános célú nyelv tetszőleges fajta problémák megoldásához.
 - Példák: BibTeX/LaTeX, CSS, DOT (Graphviz), Gradle DSL, SQL, ...
 - Lásd: Martin Fowler. *DomainSpecificLanguage*.
<https://martinfowler.com/bliki/DomainSpecificLanguage.html>
- A MOF egy metamodellek definiálására szolgáló szakterület-specifikus nyelv.

19

Meta Object Facility (MOF)

- 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.
 - Forrás: *OMG Meta Object Facility (MOF) Core Specification, Version 2.5.1*

18

Meta Object Facility (MOF)

- Újrafelhasználja az UML modellezési lehetőségeit.
 - Az UML-lel közös metamodell, melyet kiterjeszt (például a reflexióval).
- Önmagának és más metamodellek (például UML, CWM) modellezéséhez használják.
 - Tetszőleges metaadatok (például szoftver konfiguráció vagy követelmény metaadatok) modellezéséhez használható.

20

Meta Object Facility (MOF)

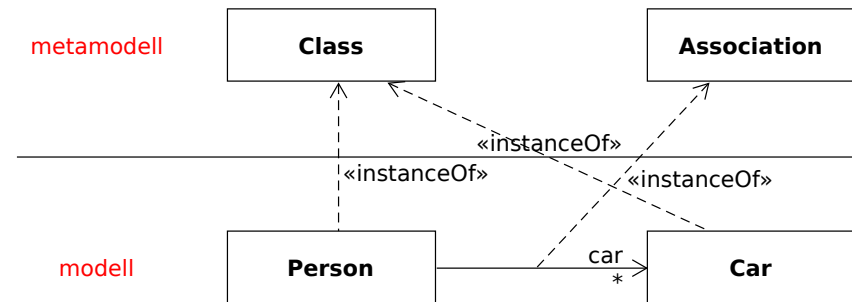
- **UML metamodel:**

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

21

Négyrétegű metamodel architektúra (UML 2.4.1)

- Metamodellezés:



23

Négyrétegű metamodel architektúra (UML 2.4.1)

- Nyelvek definiálásakor általában három réteget kell figyelembe venni:

- A nyelv specifikációja (metamodel)
- Felhasználói specifikáció (modell)
- A modell objektumai

22

Négyrétegű metamodel architektúra (UML 2.4.1)

- **Meta-metamodel:**

- A réteg elsődleges feladata egy nyelv definiálása metamodellek megadásához.
- Egy meta-metamodel általában tömörebb, mint egy általa leírt metamodel.
- Általában kívánatos, hogy a kapcsolódó metamodellek és meta-metamodellek közös tervezési elvekkel és szerkezetekkel rendelkezzenek.

- **Metamodel:**

- Egy metamodel egy meta-metamodel példánya, mely azt jelenti, hogy a metamodel minden eleme a meta-metamodel egy elemének példánya.
- A réteg elsődleges feladata egy nyelv definiálása modellek megadásához.

- **Modell:**

- Egy modell egy metamodel 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.

24

Négyrétegű metamodel architektúra (UML 2.4.1)

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

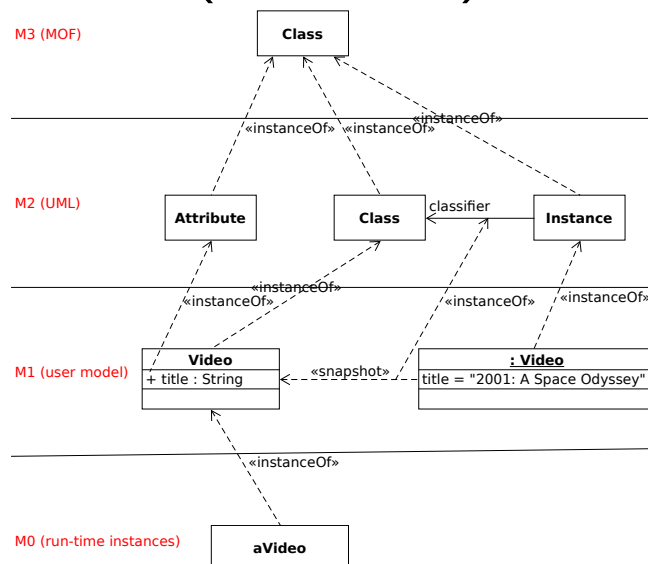
25

Meta Object Facility (MOF)

- A kulcsfogalmak az osztályozó és példány vagy osztály és objektum, valamint egy példánytól annak meta-objektumához (osztályozójához) való navigálás képessége.
 - Utóbbi a **reflexió (reflection)** révén történik.
- Ezek révén tetszőleges számú (meta-rétegnek is nevezett) réteg kezelhető.
- A legtöbb rendszer csak néhány szintet használ, általában négyet vagy kevesebbet (a minimális szám 2).
 - Például az általános reflektív rendszerek 2 réteget (osztály, objektum) használnak.

27

Négyrétegű metamodel architektúra (UML 2.4.1)



26

Meta Object Facility (MOF)

- Két fő csomagból áll:
 - **Essential MOF (EMOF):** az objektumorientált programozási nyelvekének megfelelő metamodellezési képességeket biztosít egyszerű metamodellek alkotásához.
 - **Complete MOF (CMOF):** teljes metamodellezési eszköztár biztosítása.
 - Általa történik például az UML metamodel megadása.

28

Modellelemek (1)

- Az UML modellelemek három fő kategóriája:
 - **Osztályozók:** Objektumok egy halmazát írják le. Egy objektum egy egyed, melynek állapota van és kapcsolatai vannak más objektumokkal.
 - **Események:** Lehetséges történések egy halmazát írják le. A történéseknek valamilyen következménye van a rendszerre nézve.
 - **Viselkedések:** Lehetséges végrehajtások egy halmazát írják le. Egy végrehajtás bizonyos tevékenységek teljesítése (potenciálisan valamilyen időtartam alatt), melyek eseményeket válthatnak ki/eseményekre válaszolhatnak.

29

Osztályozó

- UML modellelem, mely közös jellemzőkkel rendelkező példányok egy halmazát ábrázolja.
- Az osztályozóknak jellemzői (tulajdonságai, műveletei) vannak.
- 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.

31

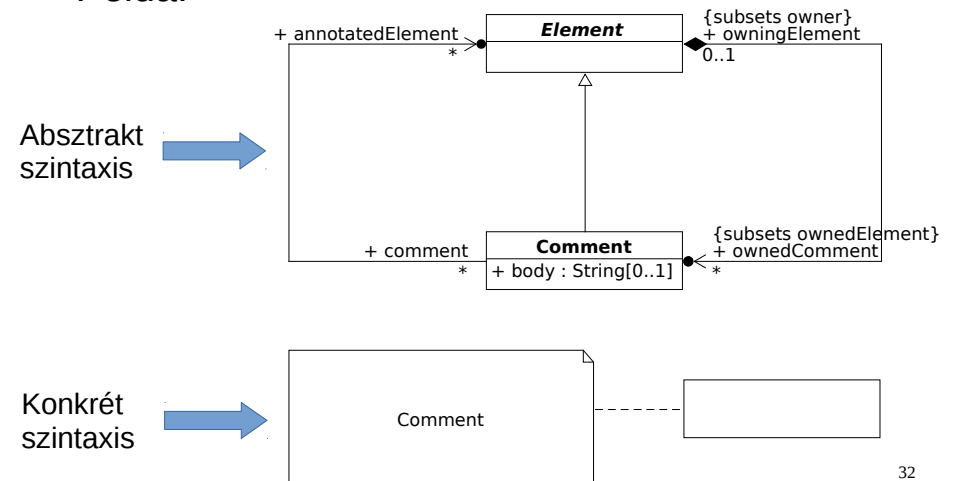
Modellelemek (2)

- Az UML modellek nem tartalmazznak objektumokat, történéseket és végrehajtásokat, mivel az ilyen egyedek a modellezett szakterület részei, nem pedig maguknak a modelleknek!
 - Az UML-nek vannak egyedek modellezésére szolgáló konstrukciói, mint például a példány-specifikációk, de ezek modellelemek.

30

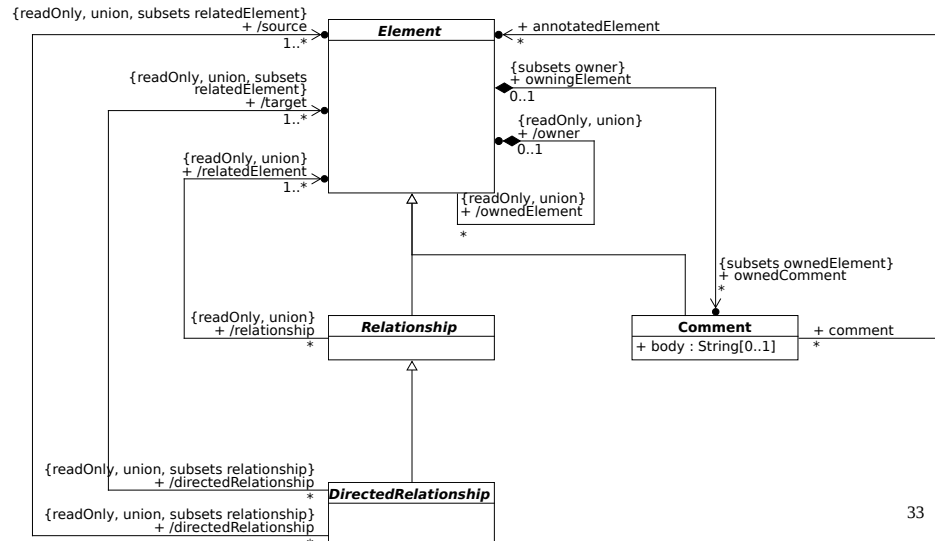
Absztrakt és konkrét szintaxis

- Példa:



32

UML alapfogalmak



33

UML diagramok típusai (2)

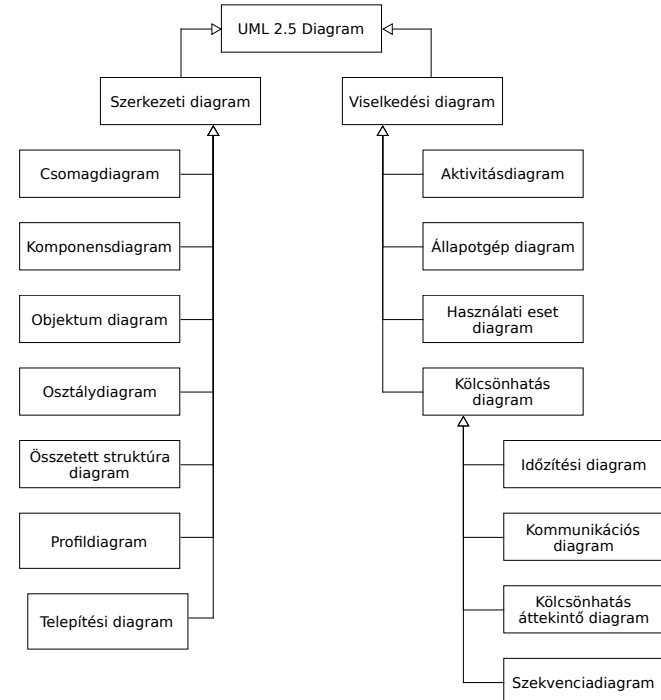
- Szerkezeti diagramok (*structure diagrams*):
 - Csomagdiagram (*package diagram*)
 - Komponensdiagram (*component diagram*)
 - Objektum diagram (*object diagram*)
 - Osztálydiagram (*class diagram*)
 - Összetett struktúra diagram (*composite structure diagram*)
 - Profildíagram (*profile diagram*)
 - 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*)

35

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.

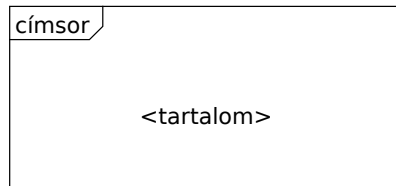
34



36

Diagram

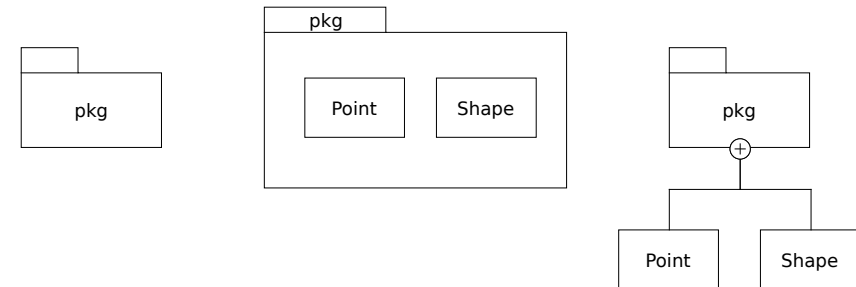
- Opcionálisan lehet egy keretük és címsoruk.



37

Csomag (2)

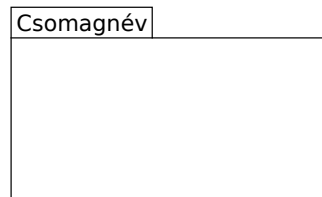
- A tartalmazott elemekre *csomagnév :: elemnév* formájú minősített nevekkel lehet hivatkozni (például `pkg::Point`, `pkg::Shape`).



39

Csomag (1)

- Modellelemek csoportosítására szolgál. A csomag egy névtér.
- Jelölésmód:



38

Kulcsszó, sztereotípa

- 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 és/vagy sztereotípa is vonatkozhat.
 - A kulcsszavak és/vagy sztereotípiák felsorolhatók egymás után, mindegyik külön határolók közé zárva.
 - Több kulcsszó és/vagy sztereotípa név is megadható a határolók között vessző karakterekkel elválasztva.

40

Kulcsszó

- Az UML jelölésmód szerves részét képező 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 (például «interface»).

41

Sztereotípa (2)

- A sztereotípiát alkalmazásakor a modellelem neve fölött vagy előtt kell megadni, vagy ott ahol a név megjelenne (ha elhagyásra kerül).
- A sztereotípa megváltoztathatja ikonok használatával a kiterjesztett modellelem grafikus megjelenítését.
- A sztereotípiák neve általában nagybetűvel kezdődik.
 - Például: «Create», «Instantiate», «Metaclass», ...
- A szabvány számos előre definiált szabványos sztereotípiát biztosít.

43

Sztereotípa (1)

- Egy vagy több metaosztály kiterjesztését definiálja, lehetővé teszi platform- vagy területspecifikus terminológia vagy jelölésmód használatát a kiterjesztett metaosztályhoz használt helyett, vagy annak kiegészítéseként.
- A metaosztályok egy korlátozott fajtája, mely önmagában nem használható, csak valamelyik kiterjesztett metaosztállyal együtt.

42

Megjegyzés

- 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.

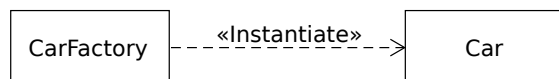
This class was added right after watching the movie 'Monty Python and the Holy Grail'.

Shrubbery

44

Függőség

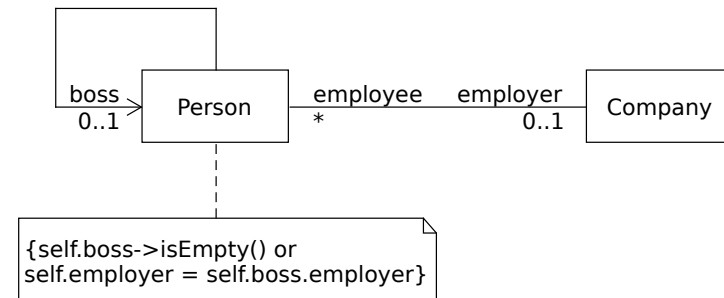
- Modellelemek közötti szolgáltató-kliens kapcsolatot jelent, ahol egy szolgáltató módosításának hatása lehet a kliens modellelemekre.
- Jelölésmód:
 - Két modellelem közötti szaggatott nyíl jelöli. A nyíl a függő (kliens) modellelemtől a szolgáltató modellelem felé mutat. A függőséghez megadható egy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



45

Megszorítás (2)

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



47

Megszorítás (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 szöveggént. Nyelvként használható egy formális nyelv (például OCL), egy programozási nyelv (például Java) vagy természetes nyelv.

46

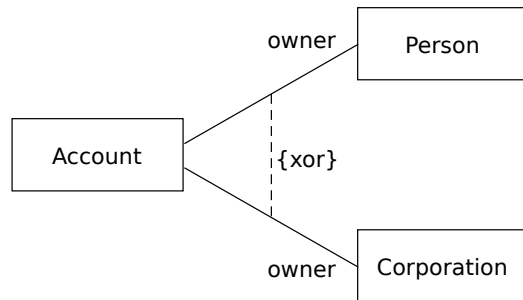
Megszorítás (3)

- 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.

48

Megszorítás (4)

- Két modellelemre vonatkozó megszorítás ábrázolható a két elemet összekötő szaggatott vonallal, melyre rá kell írni a megszorítást.
 - Példa:



49

Osztálydiagram (2)

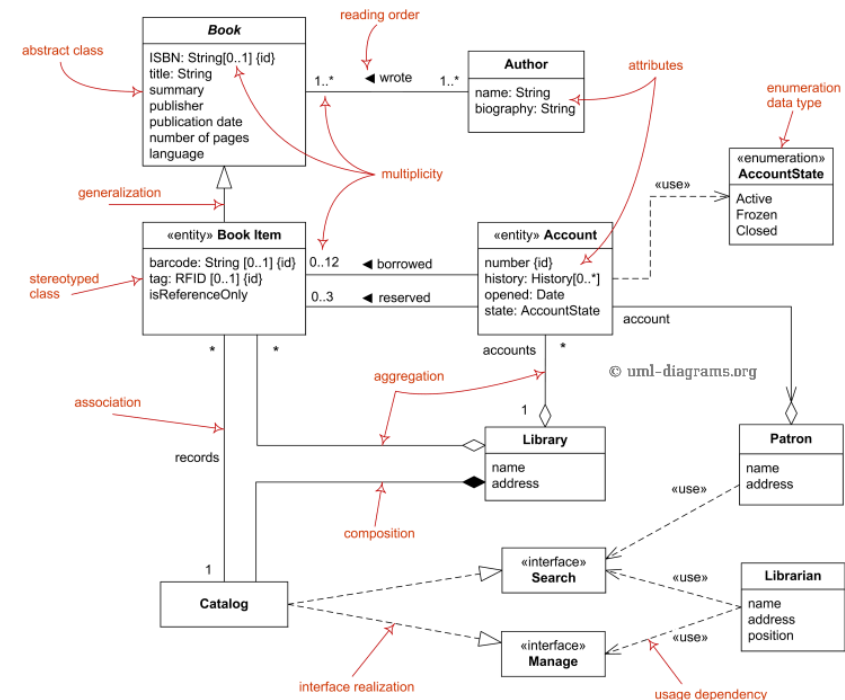
- Osztálydiagramok fajtái (Större):
 - Elemzési:** Az elemzési szinten az osztályok az alkalmazási szakterület fogalmai, az osztálydiagram a szakterület felépítését modellezi.
 - Tervezési:** Megjelennek az osztályokban a megvalósítás módjának technikai aspektusai.
 - Megvalósítási:** Az osztályok egy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

51

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

50



52

Forrás: <http://www.uml-diagrams.org/class-diagrams-overview.html>

Osztály

- Jelölésmód:

Név
Attribútumok
Műveletek

53

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:
 - $[alsó_korlát \dots] felső_korlát$
 - 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 $1 \dots 1$ számossággal, hasonlóan 5 ekvivalens az $5 \dots 5$ számossággal.
 - A $0 \dots *$ számosság helyett használható az ekvivalens $*$ jelölés.

55

Láthatóság

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

54

Tulajdonság (1)

- Egy attribútum vagy egy asszociációvég.
- 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`, ...

56

Tulajdonság (2)

- Módosítók (nem teljes felsorolás):
 - **id**: azt jelenti, hogy a tulajdonság az osztály azonosítójának része.
 - **nonunique**: azt jelenti, hogy lehetnek ismétlődések egy többértékű tulajdonságnál.
 - **ordered**: azt jelenti, hogy a tulajdonság rendezett.
 - **readOnly**: azt jelenti, hogy a tulajdonság csak olvasható.
 - **redefines** *név*: azt jelenti, hogy a tulajdonság újradefiniálja az adott nevű örökölt tulajdonságot.
 - **seq** vagy **sequence**: azt jelenti, hogy a tulajdonság egy rendezett multihalmazt ábrázol.
 - **unordered**: azt jelenti, hogy a tulajdonság nem rendezett.
 - **unique**: azt jelenti, hogy nincsenek ismétlődések egy többértékű tulajdonságnál.
 - **megszorítás**: a tulajdonságra vonatkozó megszorítás.

57

Művelet

- Jelölésmód:
 - $[^]$ [*láthatóság*] *név* (*[paraméterlista]*)
[: típus] [*[számosság]*]
[{ tulajdonság [, tulajdonság] }]*
 - A $^$ azt jelzi, hogy a művelet örökölt (UML 2.5).
 - Tulajdonság: nonunique, ordered, query, redefines *név*, seq/sequence, unique, unordered, megszorítás
 - **query**: azt jelenti, hogy a művelet nem változtatja meg a rendszer állapotát.

59

Paraméter, paraméterlista

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

58

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}

60

Statikus attribútumok és műveletek

- A statikus attribútumokat és műveleteket aláhúzás jelöli.

– Példa:

Singleton
<u>-instance: Singleton</u>
<u>-Singleton()</u>
<u>+getInstance(): Singleton</u>

61

Absztrakt osztály (2)

- Példa:

Shape
-x: int
-y: int
#Shape(x: int, y: int)
+getX(): int
+getY(): int
+moveTo(newX: int, newY: int)
+getArea(): double
+draw()

63

Absztrakt osztály (1)

- Nem példányosítható osztály.
- Jelölésmód:
 - Szedjük az osztály 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.)

62

Asszociáció (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.

64

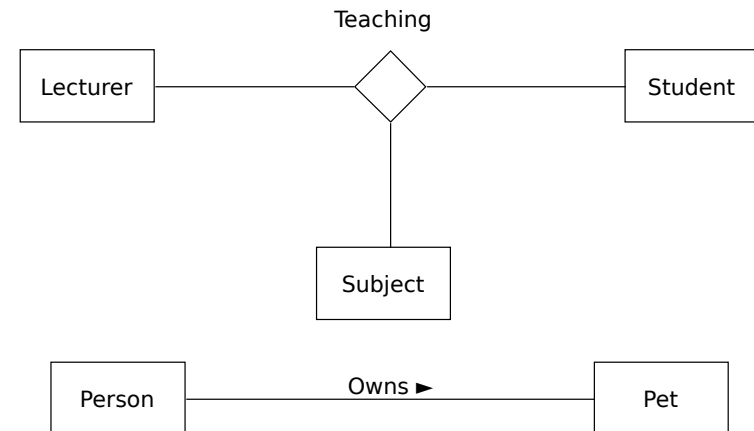
Asszociáció (2)

- Jelölésmód:
 - Bármely asszociáció ábrázolható egy csúcsára állított rombuszsal, 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.

65

Asszociáció (4)

- Példa:



67

Asszociáció (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.

66

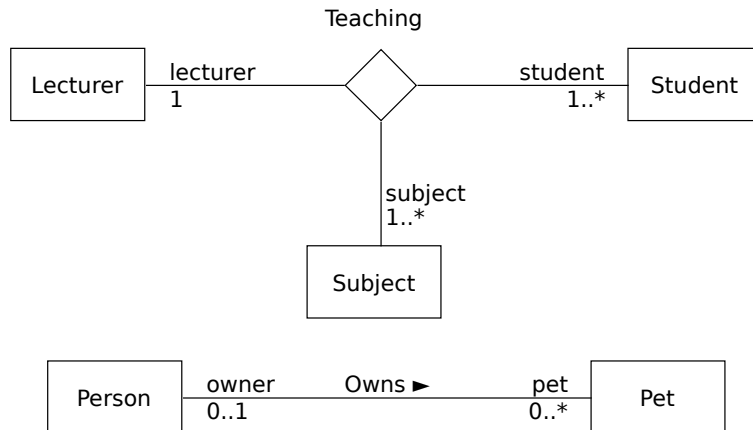
Asszociáció (5)

- **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 × pedig azt, hogy a vég nem navigálható.

68

Asszociáció (6)

- Példa:



69

Asszociáció (8)

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

71

Asszociáció (7)

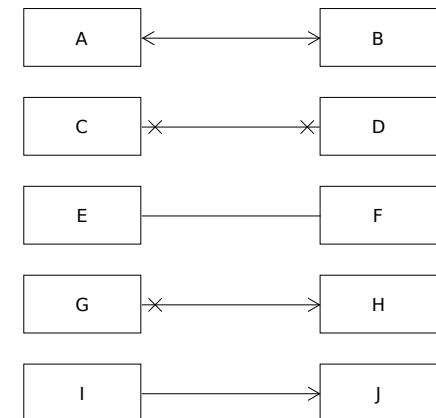
- Az osztályozó és a vonal érintkezési pontjában elhelyezhető egy kis tömör kör (a továbbiakban pontnak nevezzük).
 - A pont azt mutatja, hogy a modell tartalmaz egy tulajdonságot, melynek típusát a pont által érintett osztályozó ábrázolja. Ez a tulajdonság a másik végen lévő osztályozóhoz tartozik.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



70

Asszociáció (9)

- 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.)



72

Asszociációs osztály (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 rombuszsal.
 - Az útvonal/rombusz és az osztály szimbólum ugyanazt a modellelemet jelölik, a nevük is meg kell, hogy egyezzen.

73

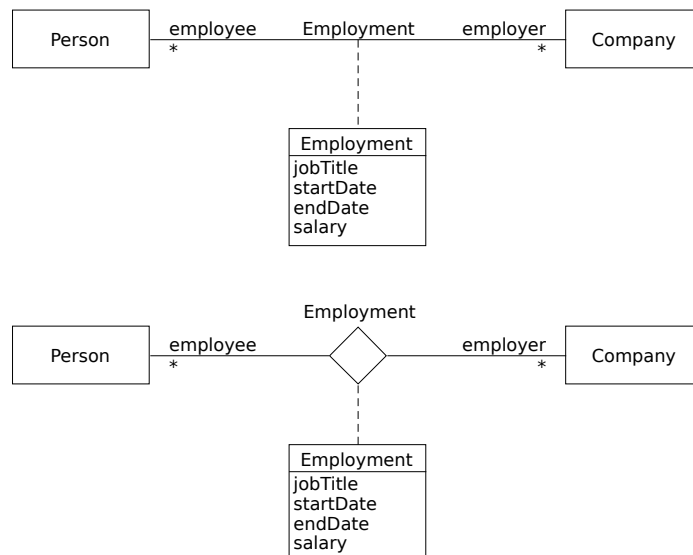
Minősített asszociáció (1)

- A minősítő egy asszociációvéghez tartozó tulajdonság(ok).
- Jelölésmód:
 - Egy kis doboz ábrázolja az asszociáció útvonalának végén a végső útvonalszakasz és az osztályozó között.
 - Egy vagy több minősítő attribútum adható meg a dobozban, jelölésmódjuk megegyezik az attribútumokéval, azzal a kivétellel, hogy nincs értelme kezdőértéknek.



75

Asszociációs osztály (2)



74

Minősített asszociáció (2)

- A minősítő a végen egy adott példányhoz (a minősített példányhoz) tartozó példányok halmazát diszjunkt partíciókra osztja.
 - Minden egyes partíciót egy minősítő érték jelöl ki, mely egy olyan vektor, mely minden egyes minősítő attribútumhoz egy értéket tartalmaz.
 - A multiplicitások az asszociáció többi végén a példányok számát határozzák meg az egyes partíciókban. Így például a $0..1$ azt jelenti, hogy legfeljebb egy példány van minősítő értékenként.

76

Minősített asszociáció (3)

- Példa:
 - A diagram azt fejezi ki, hogy egy adott bank esetén egy bizonyos számlaszám 0 vagy 1 személyt azonosít.
 - A minősítő az accountNo tulajdonság, a minősített objektum a Bank. A minősítő a nem megnevezett Bank típusú tulajdonsághoz tartozik.

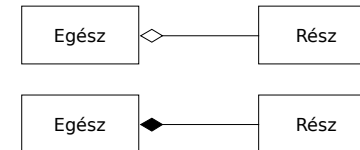


77

Egész-rész kapcsolat (1)

- A bináris asszociációk egész-rész kapcsolatot kifejező fajtái:
 - **Aggregáció (*shared 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 asszociáció egy vége jelölhető meg aggregációként vagy kompozícióként, a másik nem.

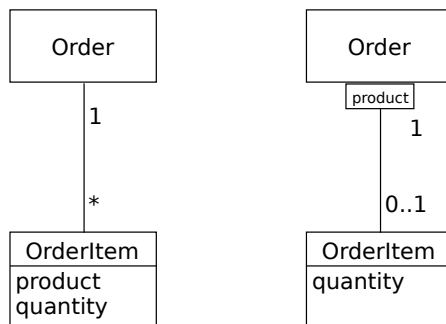
- Jelölésmód:



79

Minősített asszociáció (4)

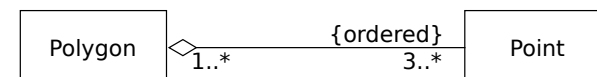
- Példa:



78

Egész-rész kapcsolat (2)

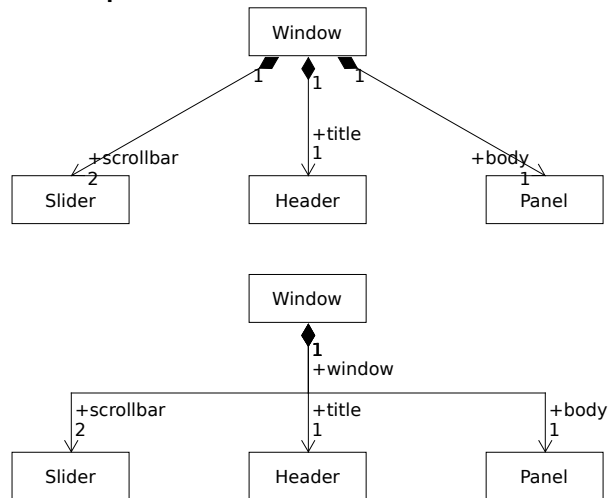
- Példa aggregációra:



80

Egész-rész kapcsolat (3)

- Példa kompozícióra:



81

Általánosítás (2)

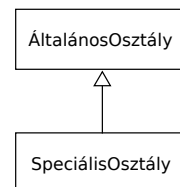
- Az általánosítás/specializáció reláció tranzitív lezártja szerint értelmezzük egy osztályozó általánosításait és specializációit.
- Egy osztályozó egy példánya minden általánosításának példánya.

83

Általánosítás (1)

- Az általánosítás egy általánosítás/specializáció kapcsolatot határoz meg osztályozók között. Egy speciális osztályozót kapcsol össze egy általánosabb osztályozóval.
 - Az általános osztályozót a speciális osztályozó **szülőjének** nevezzük, osztályok esetén **ősosztálynak**.
 - A speciális osztályozó örökli az általános osztályozó bizonyos tagjait.

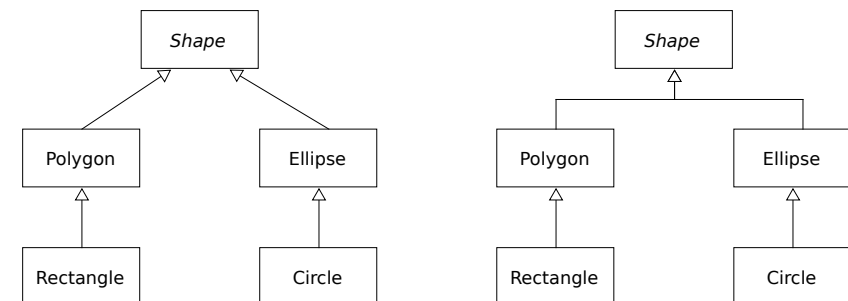
- Jelölésmód:



82

Általánosítás (3)

- Példa:



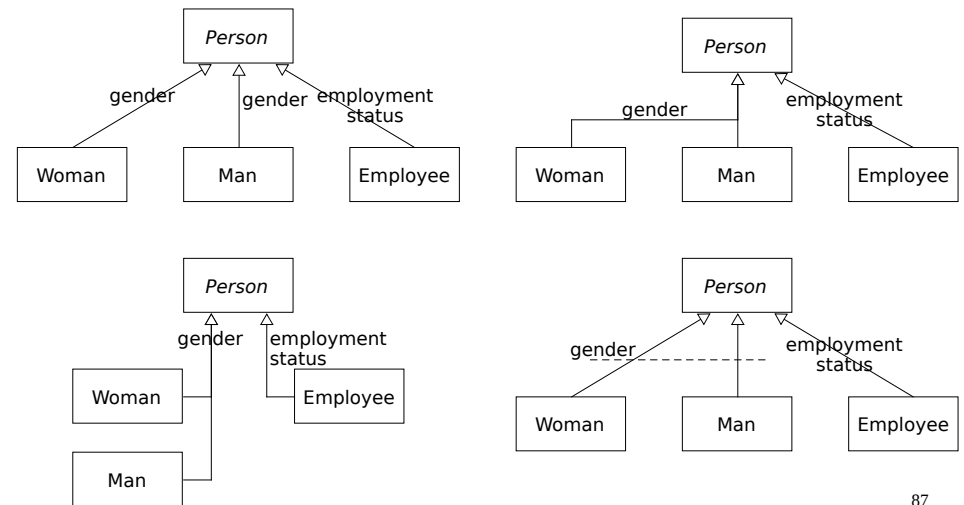
84

Általánosítás-halmaz (1)

- Általánosítások csoportosítására szolgálnak.
 - Céljuk az általánosítás ortogonális dimenzióinak ábrázolása.

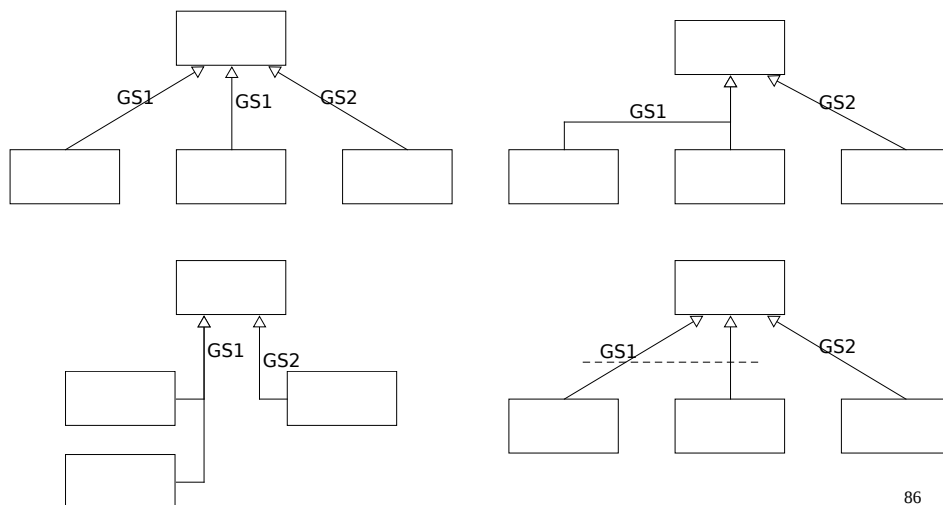
85

Általánosítás-halmaz (3)



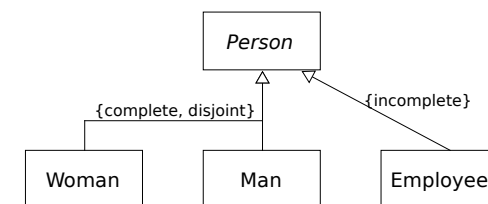
87

Általánosítás-halmaz (2)



86

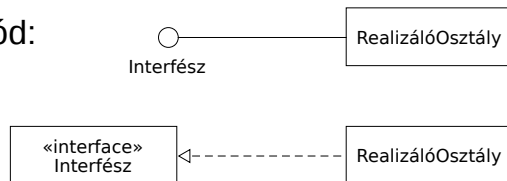
Általánosítás-halmaz (4)



88

Interfész (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:



89

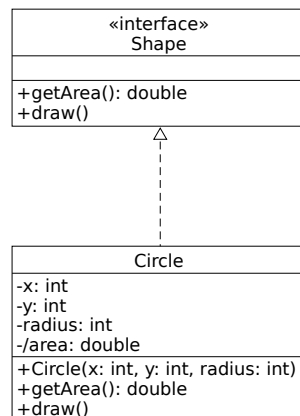
Példány-specifikáció (1)

- A példány-specifikációk osztályozók lehetséges vagy ténylegesen létező példányait ábrázolják a modellezett rendszerben, melyeket teljesen vagy részlegesen írnak le.
- Egy példány-specifikáció ábrázolhat egy példányt egy adott időpillanatban (egy pillanatfelvétel).
- A példány-specifikáció egy modellelem, mely nem tévesztendő össze azzal a példánnyal, melyet modellez!
 - Lehet, hogy csak részlegesen határozza meg egy példány tulajdonságait, a modellezett rendszerben ténylegesen több olyan példány lehet, melyek kielégítik a példány-specifikáció követelményeit.

91

Interfész (2)

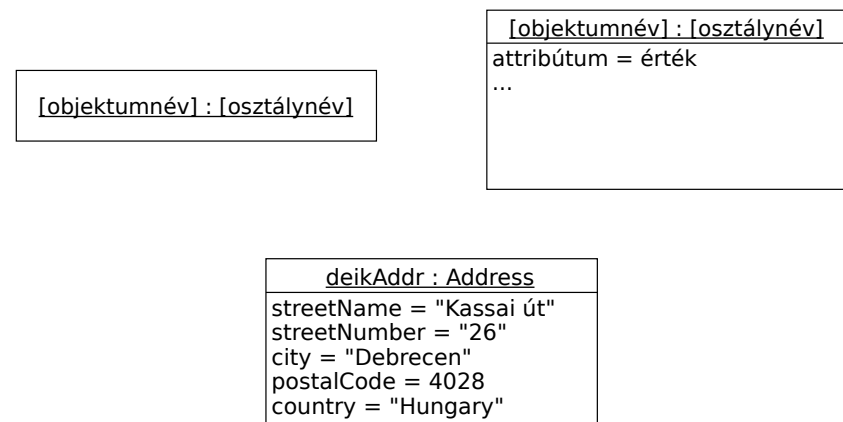
- Példa:



90

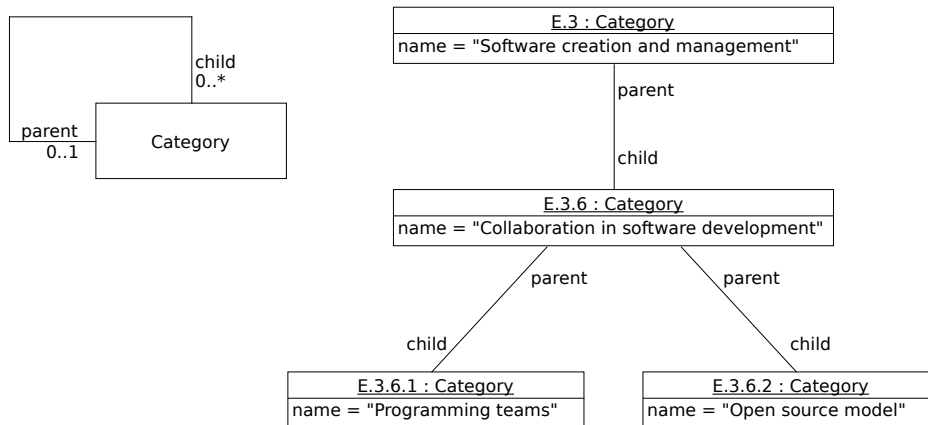
Példány-specifikáció (2)

- Jelölésmód:



92

Objektum-diagram



93

UML eszközök: szabad szoftverek

- *Modelio* (operációs rendszer: Linux, macOS, Windows; licenc: GPLv3) <https://www.modelio.org/>
- *Papyrus* (platform: Eclipse; licenc: *Eclipse Public License v1.0*) <http://www.eclipse.org/papyrus/>
- *PlantUML* (platform: Java; licenc: GPLv3) <http://plantuml.com/>
- *UML Designer* (platform: Eclipse; licenc: *Eclipse Public License v1.0*) <http://www.uml-designer.org/> <https://github.com/ObeoNetwork/UML-Designer>
- *UMLet* (platform: Java; licenc: GPLv3) <http://www.umlet.com/> <https://github.com/umlet/umlet>
 - Eclipse bővítmény: <http://marketplace.eclipse.org/content/umlet-uml-tool-fast-uml-diagrams>
 - Webalkalmazás: *UMLetino* <http://www.umlet.com/umletino/>

95

UML eszközök

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

94

UML eszközök: nem szabad szoftverek

- *Altova UModel* (platform: Windows) <https://www.altova.com/umodel.html>
- *IntelliJ IDEA Ultimate Edition* (platform: Linux, macOS, Windows) <https://www.jetbrains.com/idea/>
- *MagicDraw* (platform: Linux, macOS, Windows) <https://www.nomagic.com/products/magicdraw>
- *Microsoft Visio* (platform: Windows) <http://visio.microsoft.com/>
- *ObjectAid UML Explorer* (platform: Eclipse) <http://www.objectaid.com/>
- *Sparx Systems Enterprise Architect* (platform: Windows) <http://www.sparxsystems.com/>
- *StarUML* (platform: Linux, macOS, Windows) <http://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>

96

PlantUML (1)

- UML diagramok létrehozására szolgáló szabad és nyílt forrású szoftver.
<http://plantuml.com/>
 - Programozási nyelv: Java
 - Licenc: GPLv3
- Támogatott diagramok: osztálydiagram, objektum-diagram, szekvencia-diagram, használati eset-diagram, ...
- Egy egyszerű nyelvet biztosít a diagramok definiálásához.
- Bizonyos diagramok előállításához az alábbi programot használja:
 - Graphviz (platform: Linux, macOS, Windows; programozási nyelv: C; licenc: *Common Public License v1.0*) <http://www.graphviz.org/>
- Számos fejlesztőeszköz használja, integrálja.
 - Lásd: *Tools using the PlantUML language* <http://plantuml.com/running>

97

PlantUML (3)

- Eszköztámogatás:
 - **Eclipse:** <https://github.com/hallvard/plantuml>
 - **IntelliJ IDEA:** *PlantUML integration*
<https://plugins.jetbrains.com/plugin/7017-plantuml-integration>
 - **NetBeans:** *PlantUML-NB – Netbeans Plugin for PlantUML*
<http://plugins.netbeans.org/plugin/49069/plantuml>
<https://sourceforge.net/projects/plantumlnb/>

99

PlantUML (2)

- Eszköztámogatás:
 - *Atom* (platform: Linux, OS X, Windows; licenc: *MIT License*) <https://atom.io/>
 - *PlantUML Viewer*
<https://atom.io/packages/plantuml-viewer>
 - *PlantUML language package*
<https://atom.io/packages/language-plantuml>
 - *PlantText* <https://www.planttext.com/>

98

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.

100