OOT
Objektově orientované technologie

Logická struktura systému (Diagram tříd)

Daniela Szturcová
Institut geoinformatiky, HGF

Osnova

- Třídy
- Statický pohled na systém
- Atributy a operace, řízení přístupu
- Vazby mezi třídami

Definice třídy

- **Třída** je popis množiny objektů mající společnou strukturu, chování, vztahy a sémantiku [Vondrák 2004].
- Objektům vytvořeným podle tohoto popisu (šablony) se říká **instance.**
- Vždy platí, že objekt může být instancí maximálně jedné třídy.
- Třída je jakási „šablona“ pro tvorbu objektů.

Třída filosoficky

- Třídu můžeme definovat extenzivně, nebo intenzivně
  - Výčtem **vlastností** (extenzivně).
  - Výčtem **objektů** (intenzivně).
- Například: třídu všech koček můžeme definovat
  - buď pomocí vlastností – savec, 4 nohy, mňouká, hodně spí, atd...,
  - nebo tak, že vyjmenujeme všechny kočky – Míca, Mourek, Zrzek, Májový, ...

Třídní diagram

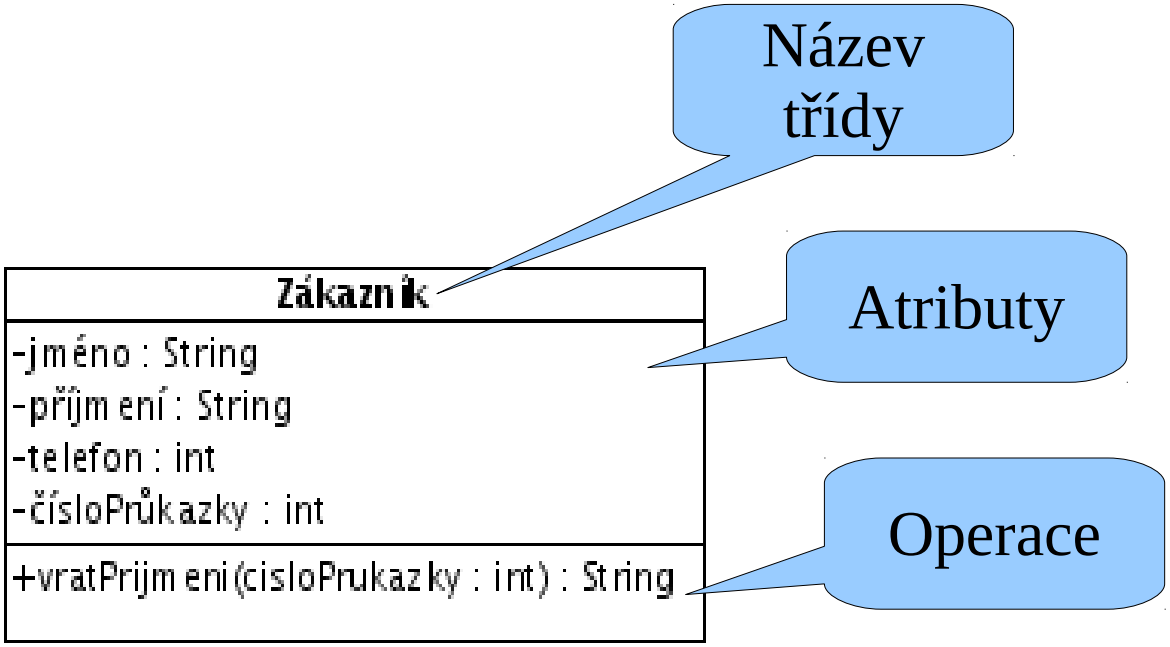
- Popisuje typy objektů a vazby mezi nimi.
- Zachycuje:
  - vlastnosti třídy,
  - operace třídy,
  - omezení plynoucí ze vztahů.

} feature

Třída – grafický zápis

- Třída se zakresluje obdélníkem s třemi oblastmi
  - název třídy,
  - seznam atributů,
  - seznam operací (metod).

Jak vypadá třída?



Název třídy

- Je důležitý pro pochopení smyslu třídy.
- Obvykle se vyjadřuje podstatným jménem (Místo, Událost, Zaměstnanec atd.).
- Nikdy se nesnažíme do názvu „vpašovat“ vlastnosti třídy: ZaměstnanecNaPůlÚvazku, PropuštěnýZaměstnanec - to jsou většinou atributy.
- Rovněž se vyhýbáme názvům, které reprezentují nějaké činnosti: Jízda, Pohyb, AktualizaceÚčtu – to vyjadřujeme operacemi.

Atributy – vlastnosti třídy

- Popisují informace o třídě.
- Plná syntaxe:

  [viditelnost][/] název [:typ][násobnost]

Viditelnost atributů

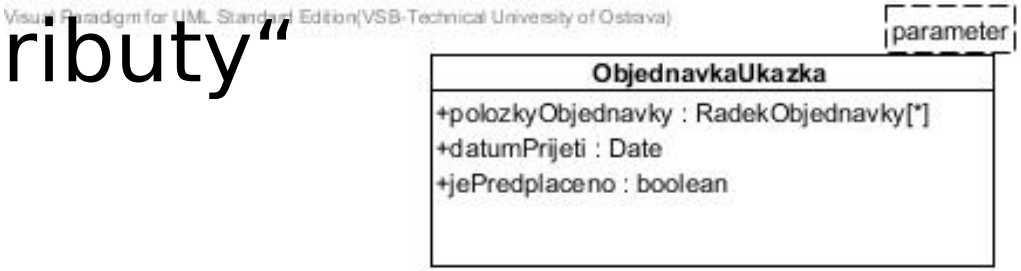
- Existují 4 základní typy viditelnosti:
  - soukromé – atribut je viditelný pouze pro členy téže třídy (mohou ho využívat jen operace dané třídy).
  - veřejné – atribut je viditelný v rámci celého systému.
  - chráněné – atribut je viditelný v rámci téže třídy a všech potomků.
  - v rámci balíčku – viditelný v rámci jednoho balíčku tříd.

Viditelnost (atributů, metod)

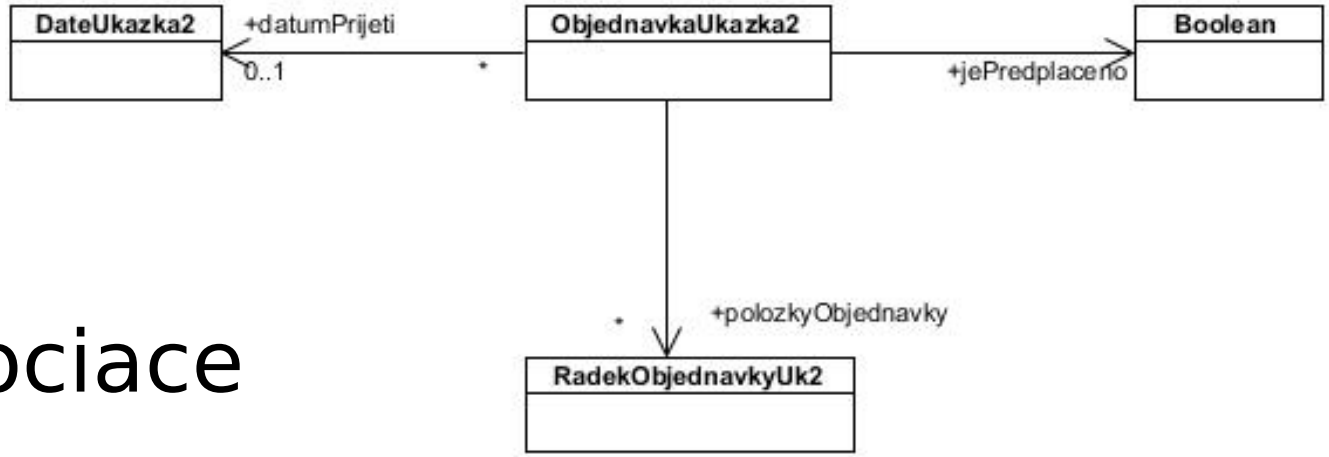
Viditelnost	**Symbol**
Soukromý	-
Veřejný	+
Chráněný	#
V rámci balíčku	~

Zachycení vlastností

- „atributy“



- asociace



Operace

- Definují chování a služby, které objekt poskytuje.
- Operace poskytuje všechny informace potřebné ke spuštění daného chování.
- Implementace operace (která se nemodeluje diagramem tříd) se nazývá **metoda**. (Tyto pojmy se často zaměňují).

Operace – syntaxe

- Popis operace (název, parametry, návratová hodnota atd.) se nazývá signatura. Syntaxe:

[viditelnost] název ([seznam_parametrů]) ':' [návratová hodnota]

Příklady:

+secti(int a, int b) : int

-zvysPocetClenuFK()

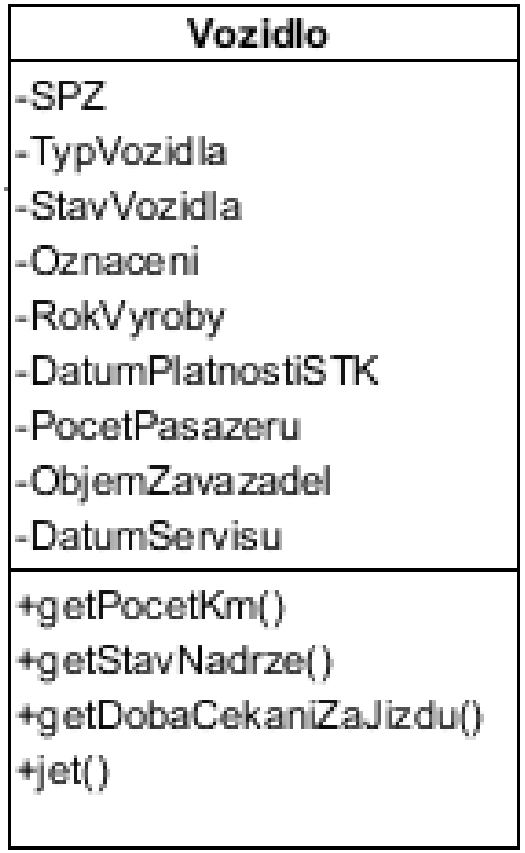
+zjistiZustatek(): double

Viditelnost operací

- Typy viditelnosti operací jsou stejné jako u atributů a vztahují se na ně stejná pravidla:
  - veřejné
  - soukromé
  - chráněné
  - viditelné v rámci balíčku

Příklad třídy

- Název třídy se obvykle zapisuje s velkým počátečním písmenem.
  - Doporučeno užívat CamelCase přístup.
- Atributy i metody začínají malým písmenem.
  - camelCase označení



Vztahy mezi třídami

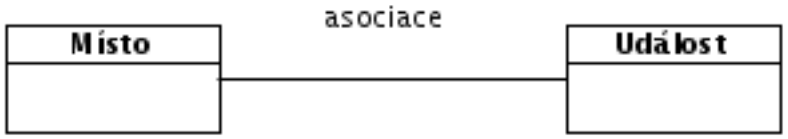
- Asociace – definuje, v jakém vzájemném vztahu se nacházejí dvě třídy.
- Generalizace – zobecnění. Třídy s podobnými vlastnostmi je možné zobecnit a vytvořit nadtřídu (rodičovskou třídu).
- Závislost – určuje vztah, kdy třídy jsou na sobě závislé a změna v jedné třídě způsobí změnu ve druhé. Může také vyjadřovat dočasný vztah mezi třídami (jedna třída využije funkci druhé).

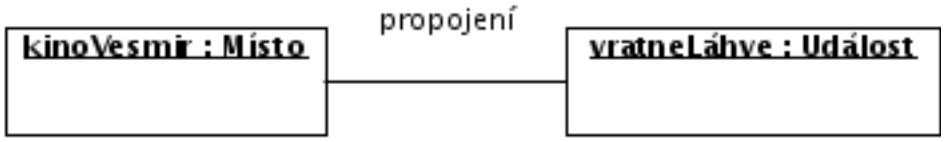
Účel asociace

- Smyslem asociace je popsat důvod, proč dvě třídy (objekty) musí o sobě vzájemně vědět.
- Asociace rovněž popisuje typ vztahu mezi třídami.

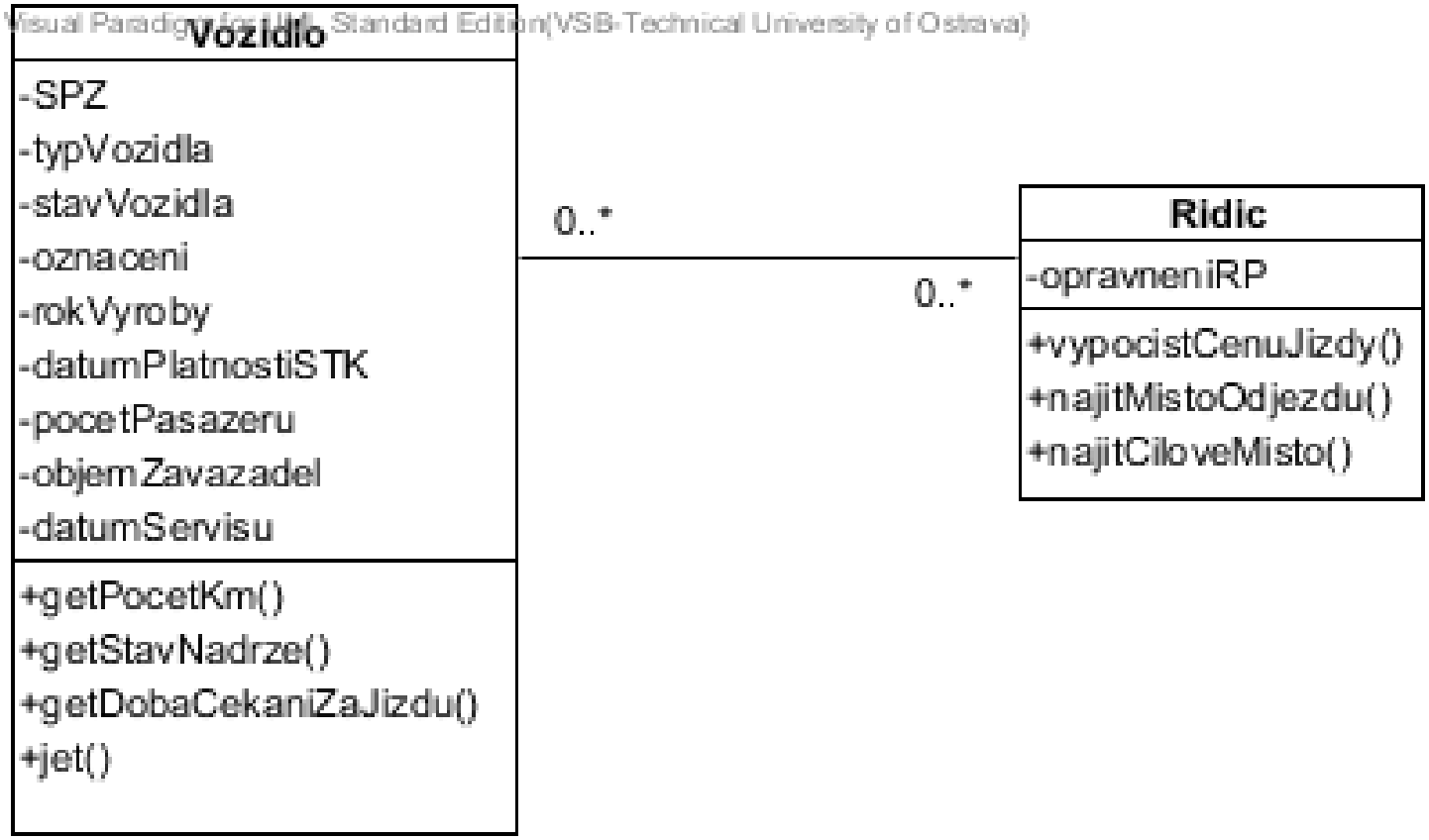
Modelování asociací

- Prvním krokem je identifikace tříd, které jsou vůči sobě v nějakém vztahu.
  - binární asociace – vztah se týká pouze dvou tříd.
  - n-ární asociace – vztah se může týkat několika tříd.





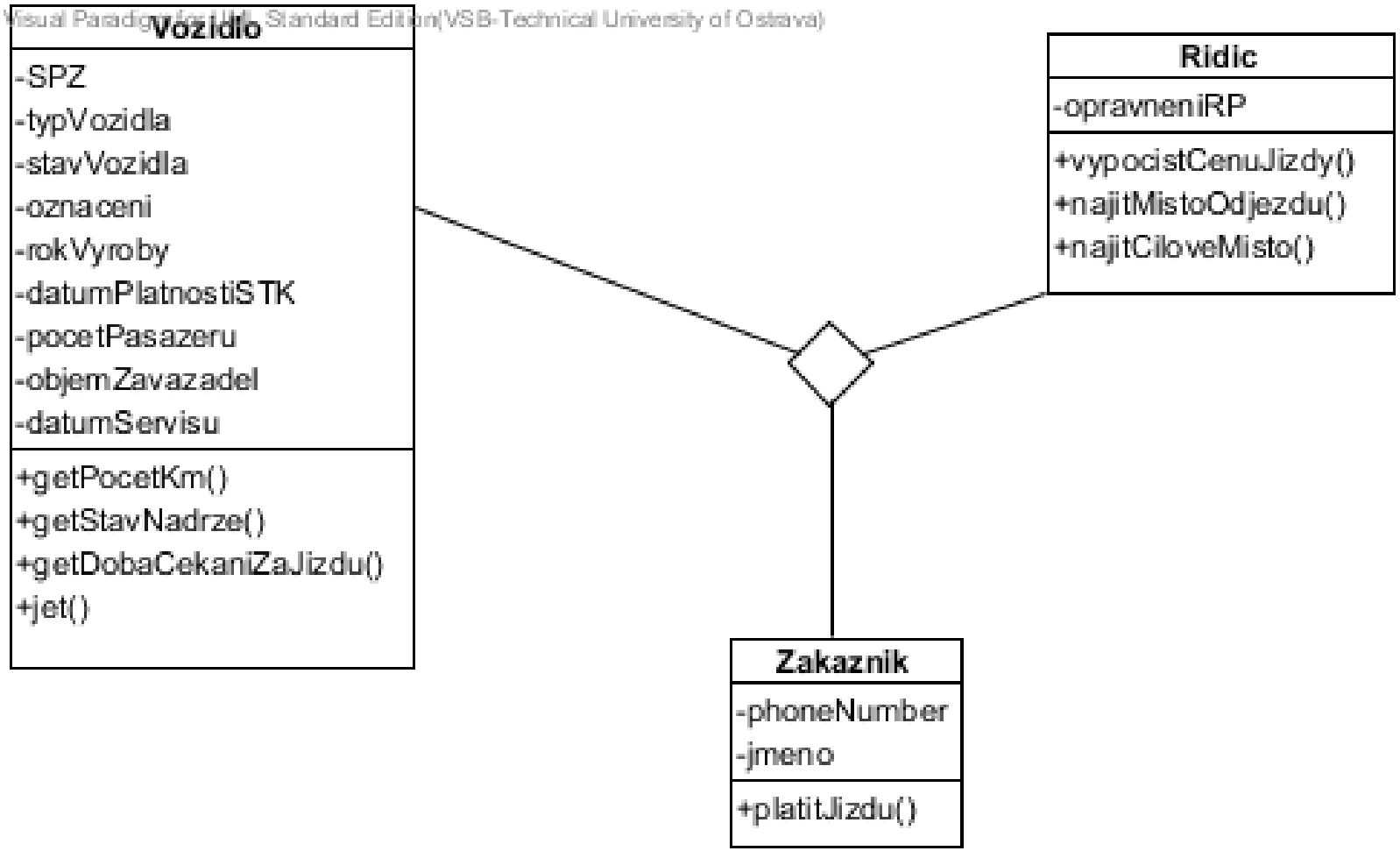
Binární asociace



N-ární asociace

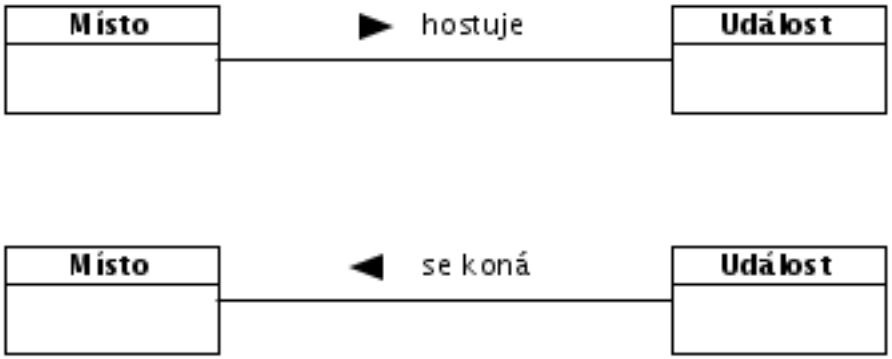
- Modeluje vztah, do kterého vstupuje více než dvě třídy.
- Většinou je přehlednější použít několik binárních asociací.

N-ární asociace



Názvy asociací

- Názvy asociací jsou velmi důležité, mohou ovlivňovat sémantický význam vazby.
- Obvykle je názvem nějaké sloveso, nebo slovesné spojení.
- Může mít směr (označuje se šipkou).



Názvy asociací

- Pokud je mezi dvěma třídami více typů vztahu, je nutné výstižně pojmenovat každý z nich.

Role

- Účelem rolí je vysvětlit, význam objektu v daném vztahu.
- Důležité pro generování kódu.

Násobnost asociací

- Násobnost asociací popisuje počet objektů, které se mohou této asociace účastnit.

Reflexivní asociace

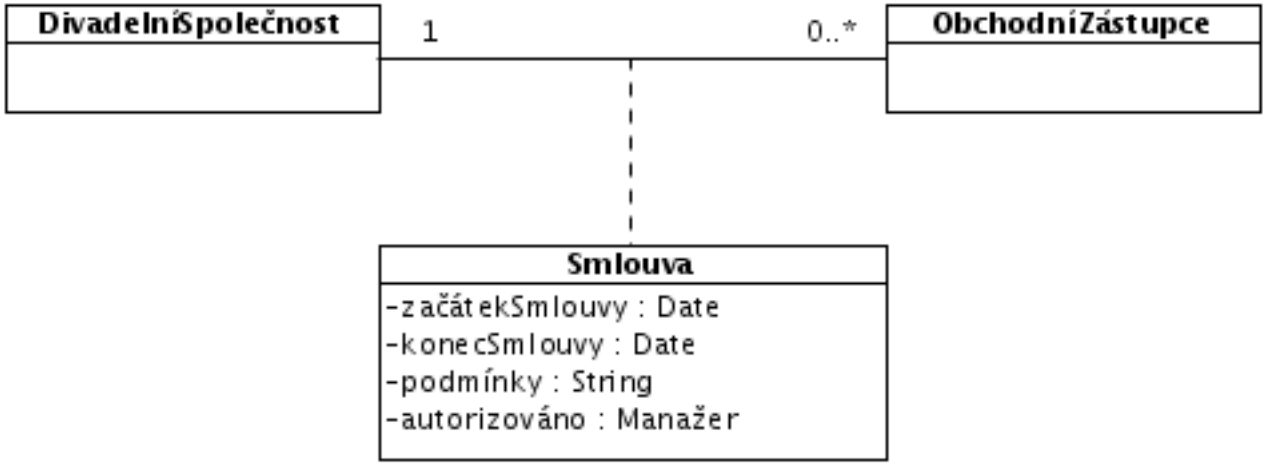
- Asociace mezi objekty téže třídy.
- Příklad k rozmyšlení – rodinné vztahy mezi Osobami téže třídy.

Obousměrná asociace

- Jedná se o inverzní vztah, tedy jde o obousměrnou vlastnost.

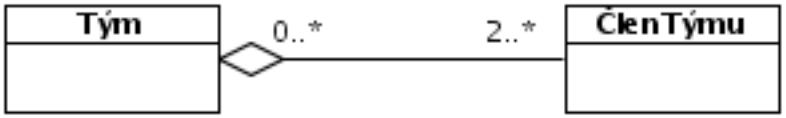
Asociační třída

- Nepopisuje pouze vztah mezi dvěma třídami, ale navíc i vlastnosti tohoto vztahu (např. kdy končí platnost asociace?, kdy začala? Podmínky platnosti asociace? atd.).
- Je to běžná třída, může být propojena s dalšími třídami.



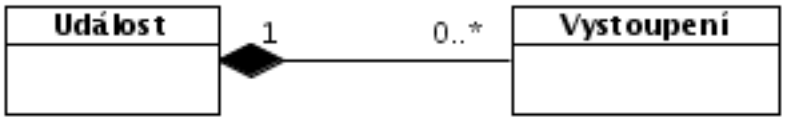
Agregace

- Speciální typ asociace.
- Modeluje hierarchický vztah, kdy jeden objekt objekt se skládá z jiných objektů (Celek – komponenty).
- Celek funguje jako jeden objekt.
- Komponenty mohou existovat samostatně (slabé spojení).



Kompozice

- Speciální typ agregace.
- Komponenta může být součástí pouze jednoho celku.
- Komponenta nemůže existovat samostatně mimo celek (pevné spojení).

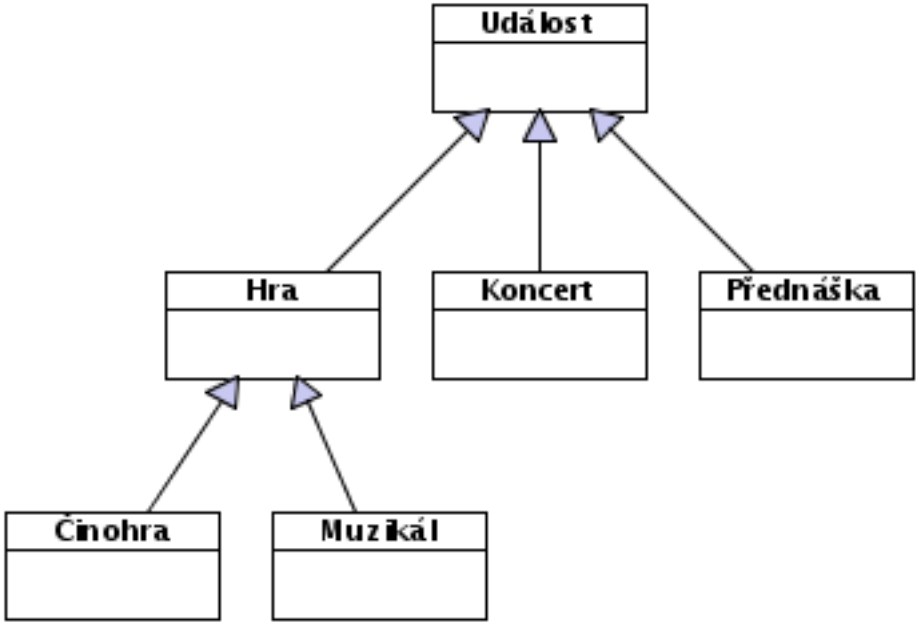


Generalizace

- Generalizace (zobecnění) je proces sdružování množin objektů podle podobných vlastností nebo účelu (opak je specializace).
- Například jablka, hrušky a švestky je možné generalizovat a říkat jim Ovoce.
- Naopak speciálním typem Ovoce jsou jablka, hrušky, švestky apod.

Generalizace

- Hierarchie může mít libovolné množství úrovní (do hloubky i do šířky).



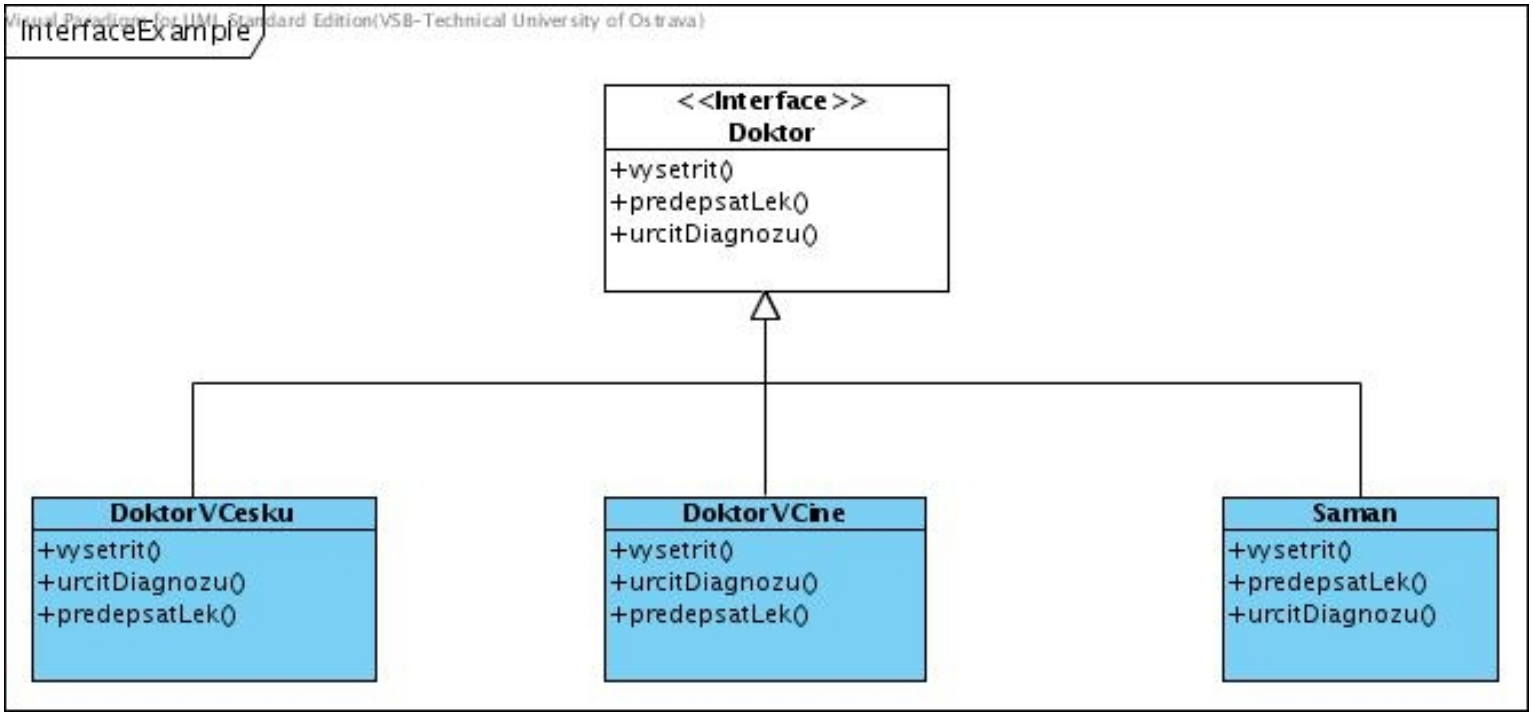
Generalizace

- Generalizace není asociace.
- Není nutné popisovat role, násobnosti, omezení apod.
- Každý potomek může mít pouze jednoho rodiče (rodič může mít více potomků).

Rozhraní

- Je seznam operací, které definují chování tříd implementujících toto chování.
- Nelze vytvořit instanci rozhraní.
- Třída, která implementuje rozhraní musí obsahovat implementaci definovaných metod.
- Tento princip umožňuje systému pracovat s rozhraním nezávisle na implementaci třídy.

Rozhraní


InterfaceExample
<<Interface>>
Doktor
+vysetrit()
+predepsatLek()
+urcitDiagnozu()
DoktorVCesku
+vysetrit()
+urcitDiagnozu()
+predepsatLek()
DoktorVCine
+vysetrit()
+urcitDiagnozu()
+predepsatLek()
Saman
+vysetrit()
+predepsatLek()
+urcitDiagnozu()

Použité zdroje

- Tom Pender.: UML Bible
- Jim Arlow, Ila Neustadt.: *UML2 a unifikovaný proces vývoje aplikací.*
- Fowler, M*.: Destilované UML*
- *přednášky P. Děrgela*

Děkuji za pozornost