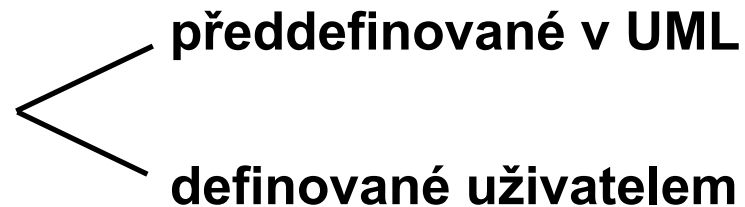


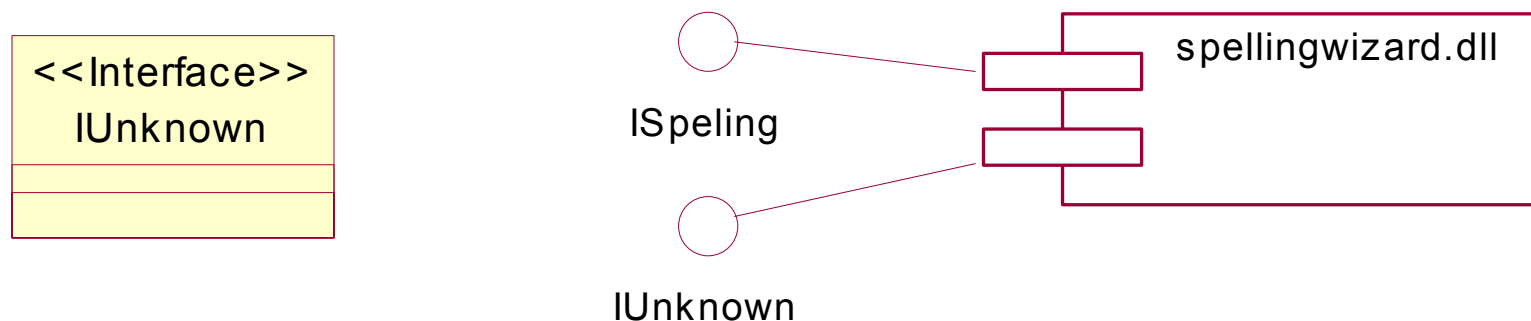
## 7.5 Diagram tříd – pokročilé techniky

- Stereotypy

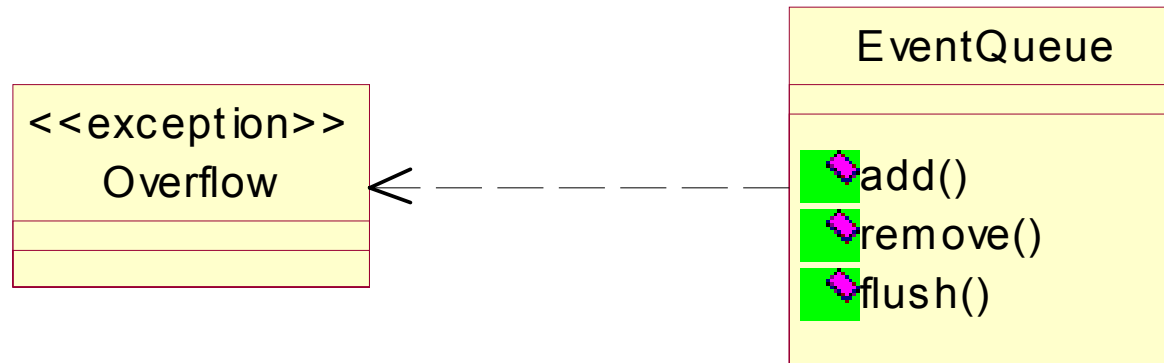
- jeden ze základních prostředků rozšiřitelnosti UML - pro modelovací konstrukce neexistující v UML, ale podobné



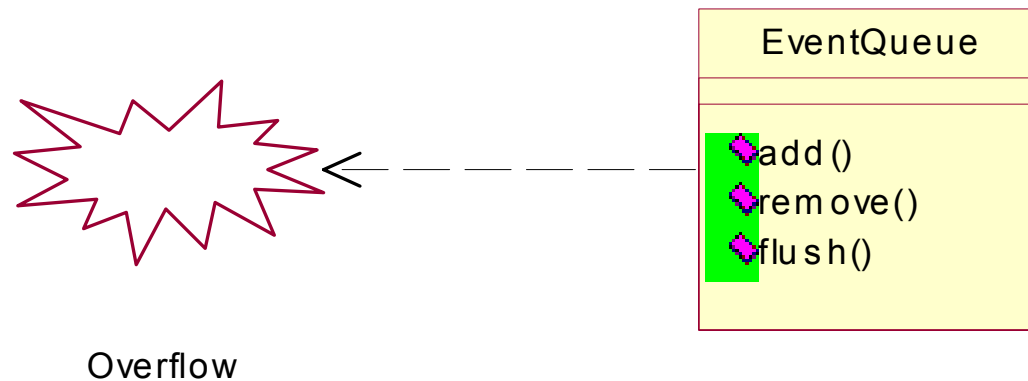
**Př) rozhraní (interface)**– specifikuje pouze externě viditelné (veřejné) operace třídy nebo komponenty bez vlastní implementace, typicky specifikuje pouze omezenou část chování třídy nebo komponenty (např. Java, COM, CORBA) → speciální druh třídy v UML



## Př) uživatelem definovaný stereotyp pro výjimku



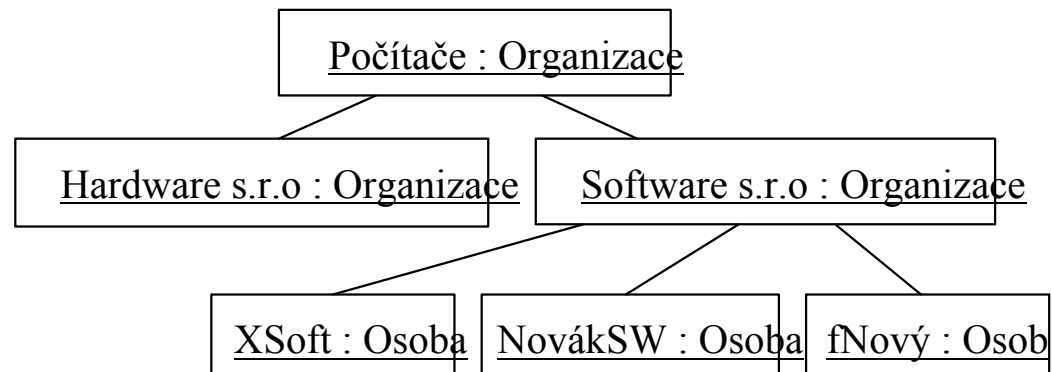
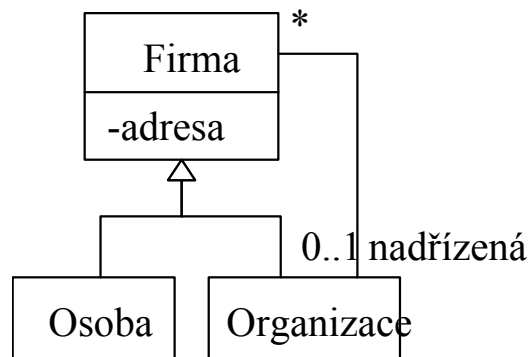
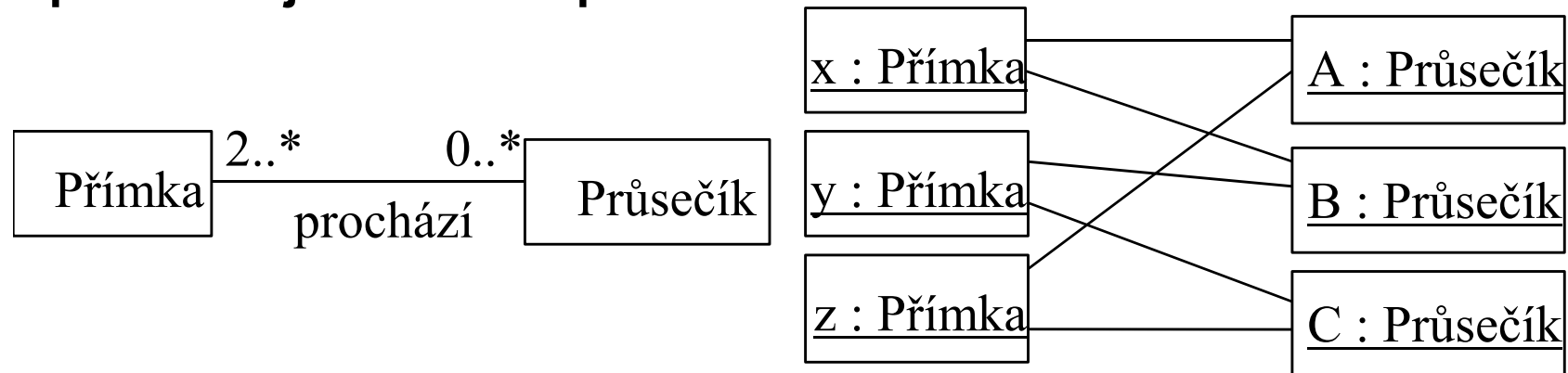
### - lze definovat vlastní symbol (ikonu)



- lze vytvářet stereotypy řady prvků diagramů, u diagramu tříd stereotypy třídy, asociace, generalizace, operace,...
- OMG využívá stereotypy k vytváření profilů (pro RT, CORBA IDL)

- **Objektový diagram**

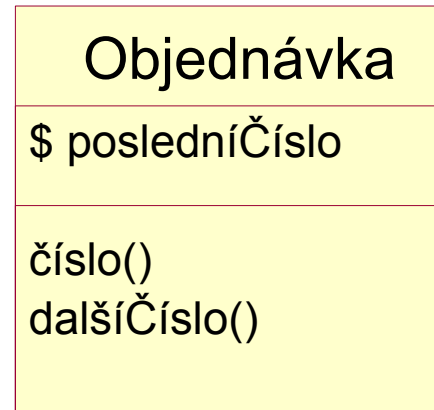
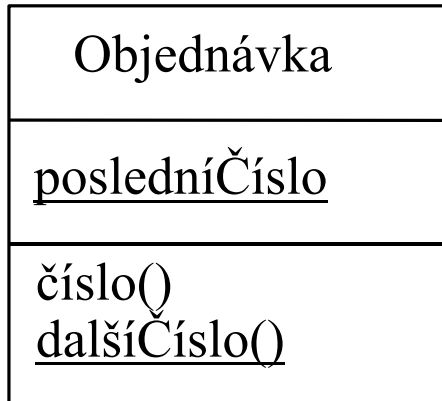
- **diagram tříd** - schéma, vzor nebo šablona popisující obecně řadu možných objektů (instancí), představuje obecný případ
- **diagram instancí** - popisuje konkrétní objekty a vztahy mezi nimi, představuje konkrétní příklad



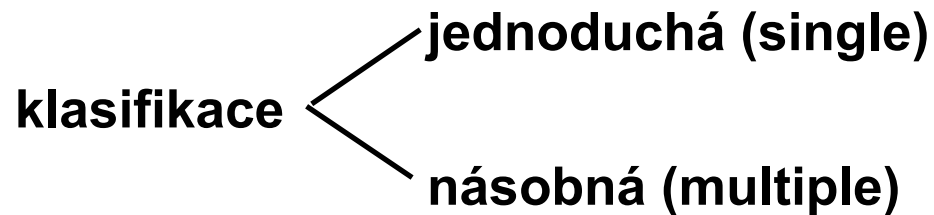
- lze uvádět jen jméno objektu nebo třídy
- lze považovat za diagram spolupráce bez zpráv

- Operace a atributy třídy (class scope)

- statické členy v C++ a Java, proměnné a metody třídy ve Smalltalku

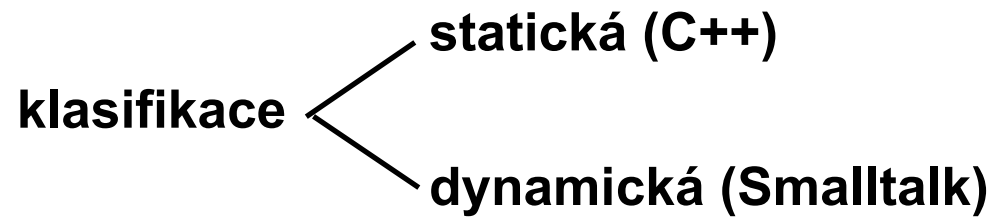


- Násobná a dynamická klasifikace



**Jednoduchá klasifikace** – objekt je jediného typu (s případnou generalizací)

**Násobná klasifikace** – objekt může být popsán několika typy, které nemusí být spojeny generalizací, tj. objekt může přímo patřit do více než jedné třídy



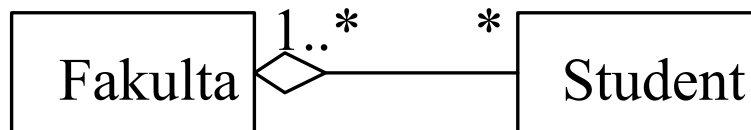
**Dynamická klasifikace** – v průběhu života může objekt měnit svůj typ v rámci struktury podtypů

- užitečné při konceptuálním modelování, při specifikačním již nutno přihlížet k technice implementace

- **Agregace a kompozice**

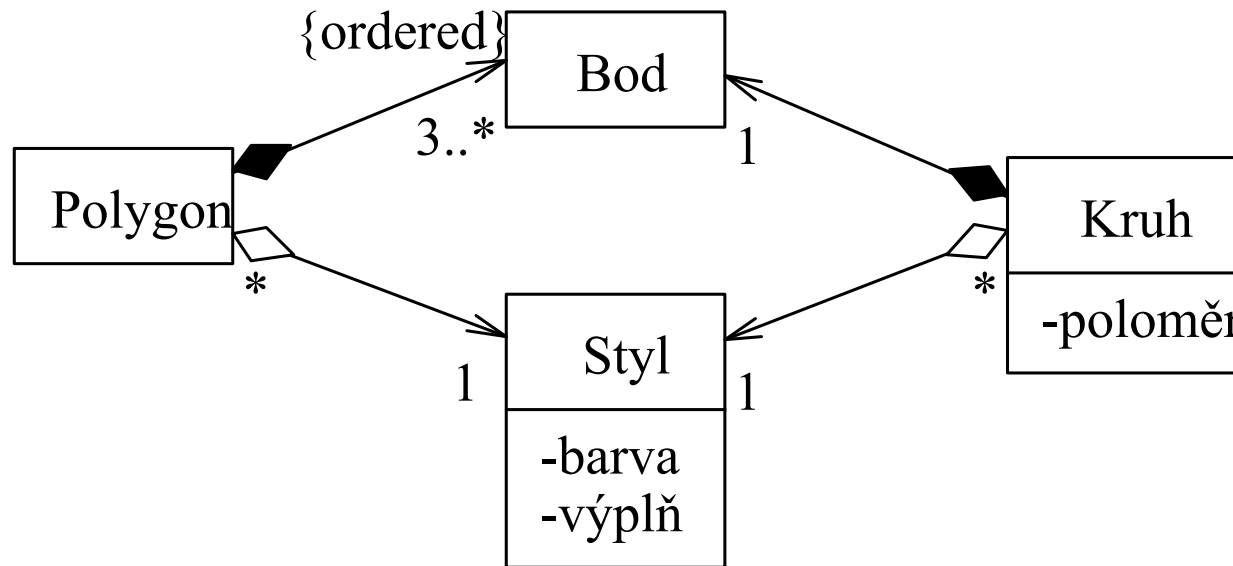
**Agregace** – zvláštní druh asociace k modelování vztahu „celek/část“

**Př)**

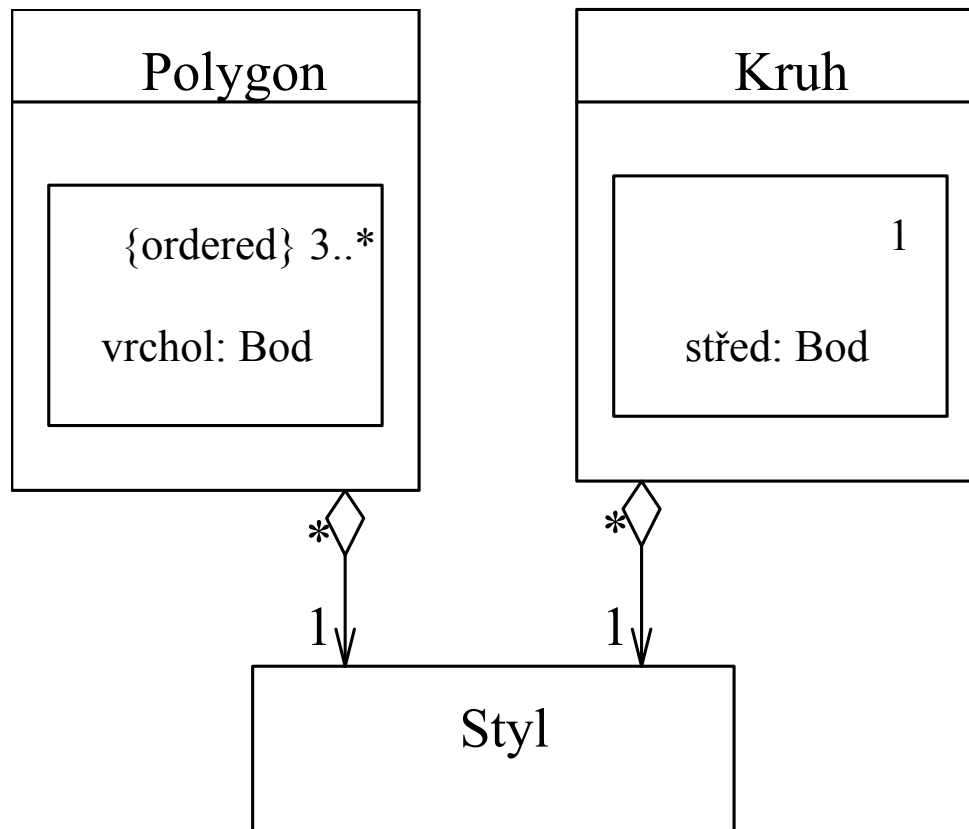


**Kompozice** – silnější varianta agregace, objekt „část“ patří jen jednomu celku, typicky je rovněž existenční závislost.

Př)



- implikuje rovněž, že objekt „část“ je „hodnotovým“ objektem
- alternativní notace kompozice v UML – viz další strana
- vztah atribut - kompozice

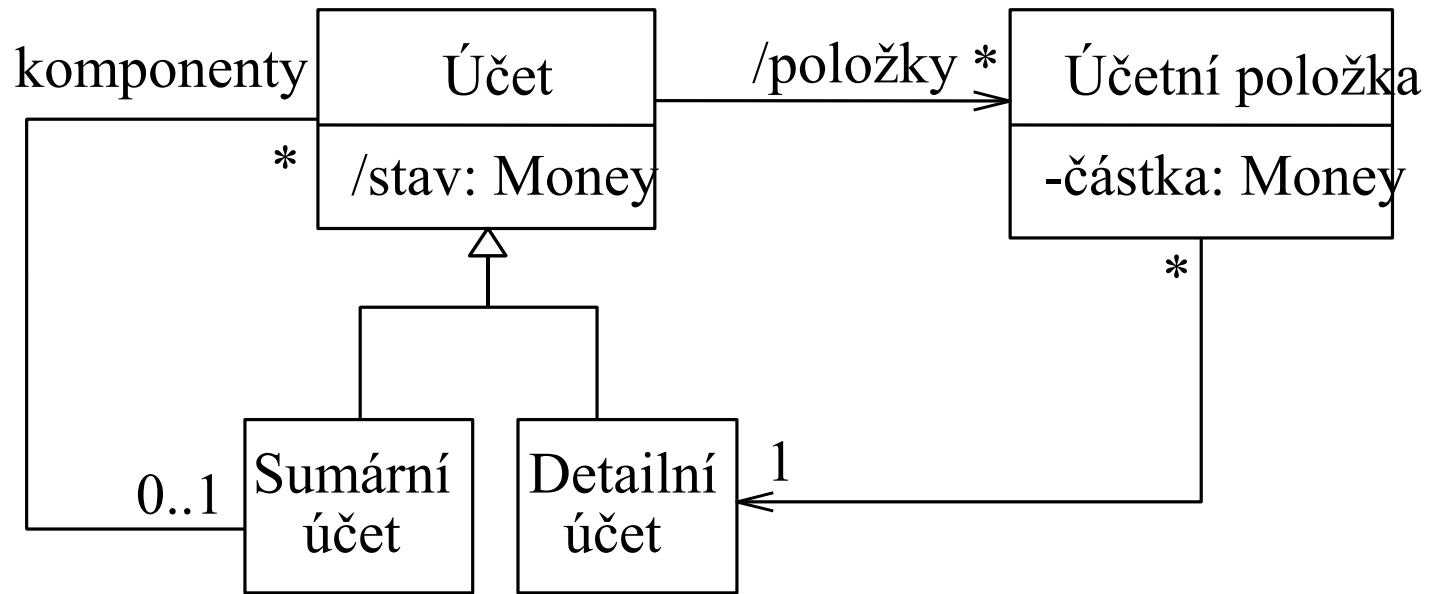


- **Odvozené asociace a atributy**

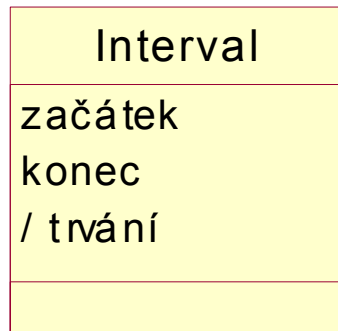
- mohou být počítány z jiných asociací, resp. atributů
- každý pohled přináší vlastní interpretaci odvozených vlastností
- specifikační pohled: omezení mezi hodnotami, ne způsob výpočtu; neříká nic o implementaci

Př)

{stav = součet účetních položek}



Př)





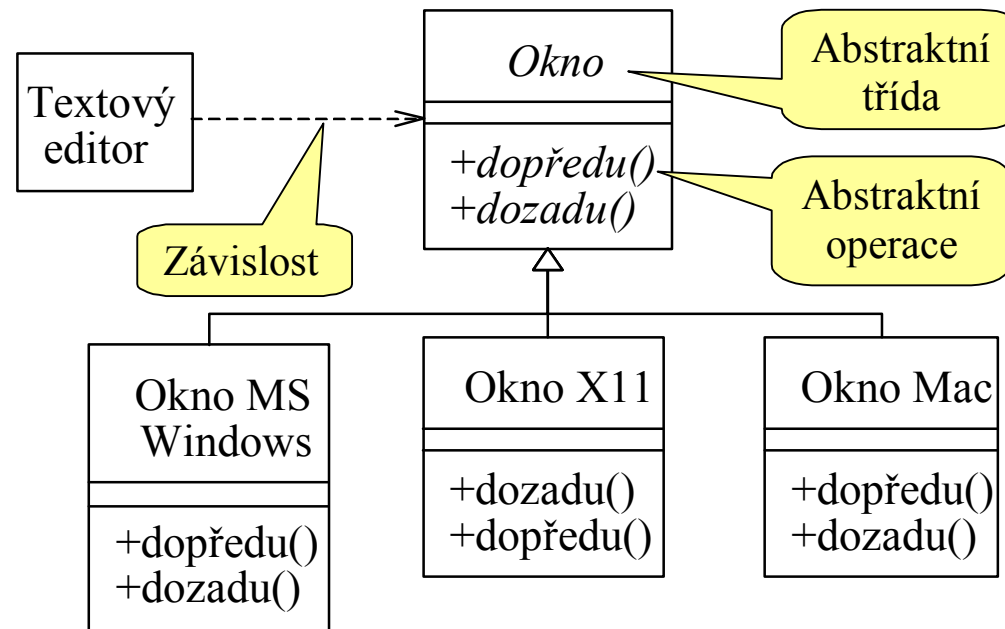
- Rozhraní a abstraktní třídy

- možnost změny rozhraní tříd nezávislého na implementaci je jednou z předností OO vývoje

**Rozhraní** – pouze deklarace operací bez implementace (metod), implementaci poskytuje třída, která rozhraní zdědí. Klient třídy vidí pouze rozhraní.

- programovací jazyky nabízí často jediný konstrukt – třídu (rozhraní+implementace), děděním se dědí oboje; rozhraní se zde deklaruje pomocí abstraktních tříd

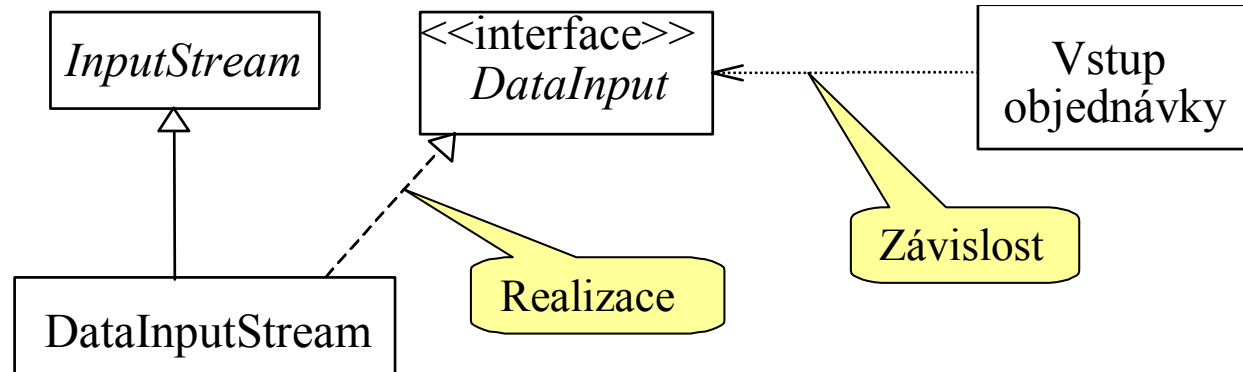
Př)



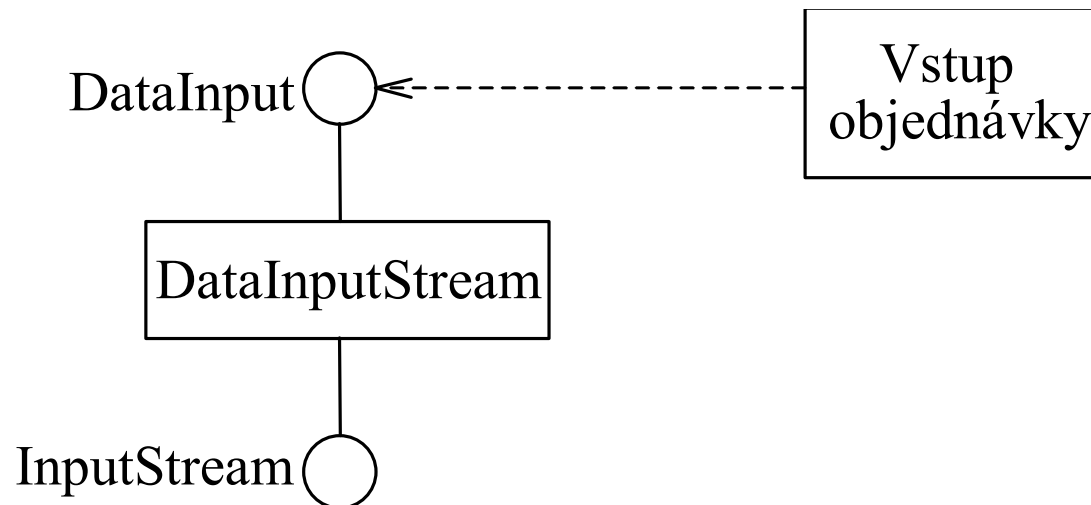
## Notace pro abstraktní třídu: kurzíva nebo {abstract}

- některé jazyky (např. Java) nabízí rozhraní jako konstrukt

Př)



- jiná notace pro rozhraní a podtřídy abstraktních tříd:



- rozdíl abstraktní třída x rozhraní

- Referenční a hodnotové objekty

- objekt má identitu

**Referenční objekty** (reference objects) – typicky jediný objekt v programu, reprezentující objekt reálného světa, odkaz prostřednictvím identity (Objednávka, Zákazník, ...), porovnání pomocí identit.

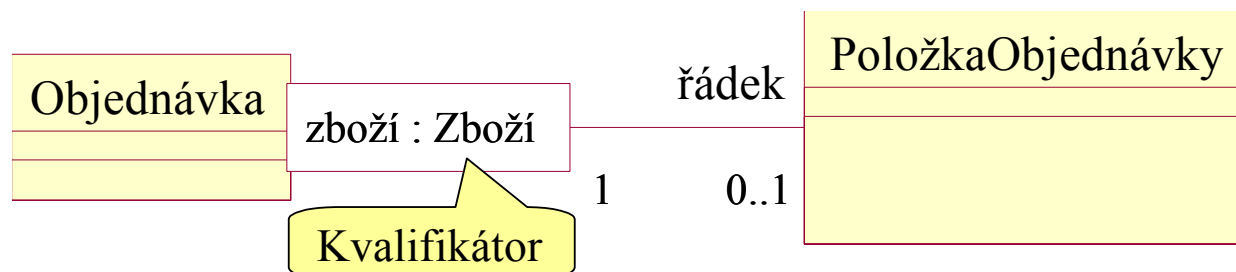
**Hodnotové objekty** (value objects) – typicky řada objektů (kopií), reprezentujících tentýž objekt reálného světa (Integer, Datum, ...), porovnání na základě hodnot.

- v UML: atributy a kompozice pro hodnotové, asociace pro referenční (při konceptuálním modelování možno i pro hodnotové)

- Kvalifikovaná asociace

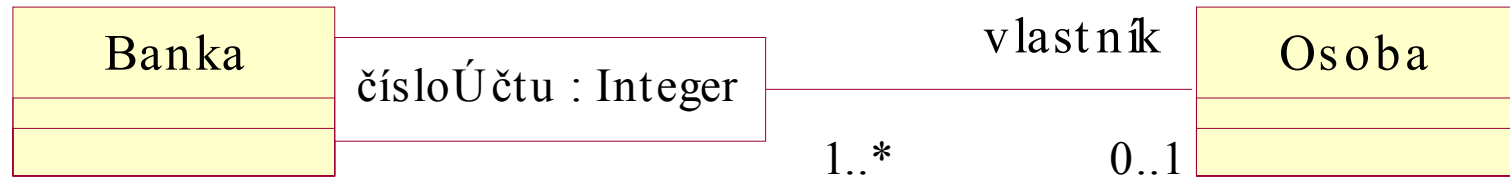
**Kvalifikátor** – atribut nebo seznam atributů asociace sloužící k rozčlenění množiny objektů spojených s daným objektem v rámci dané asociace.

Př)



- násobnost 0..1, 1, \*

Př)



Konceptuální pohled: ukazuje omezení.

Specifikační pohled: rozhraní s vyhledáváním pomocí klíče.

Př) class Objednavka {  
    public PolozkaObjednavky ctiRadek(**Zbozi** zboziPolozky);  
    public void pridejRadek(Number pocet, **Zbozi** zboziPolozky);  
    ...}

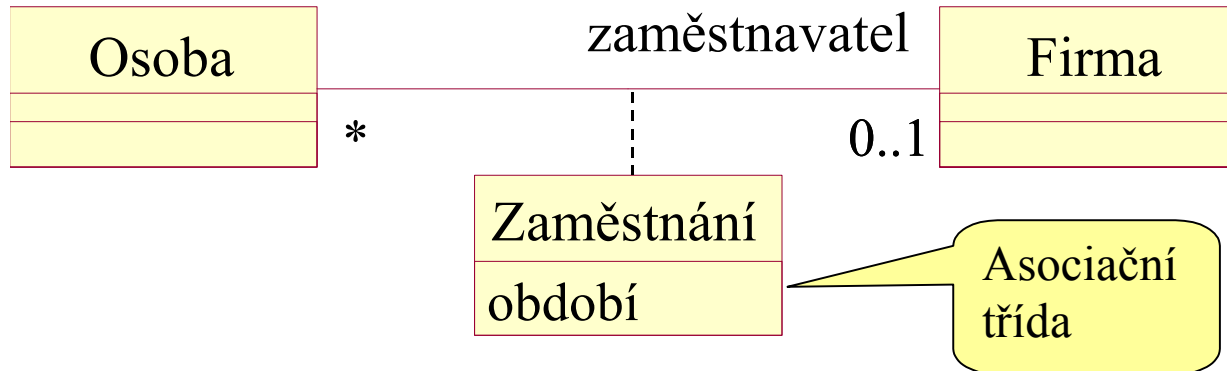
Implementační pohled: použití asociativního pole, apod.

Př) class Objednavka {  
    private **Map** polozkyObjednavky; ...}

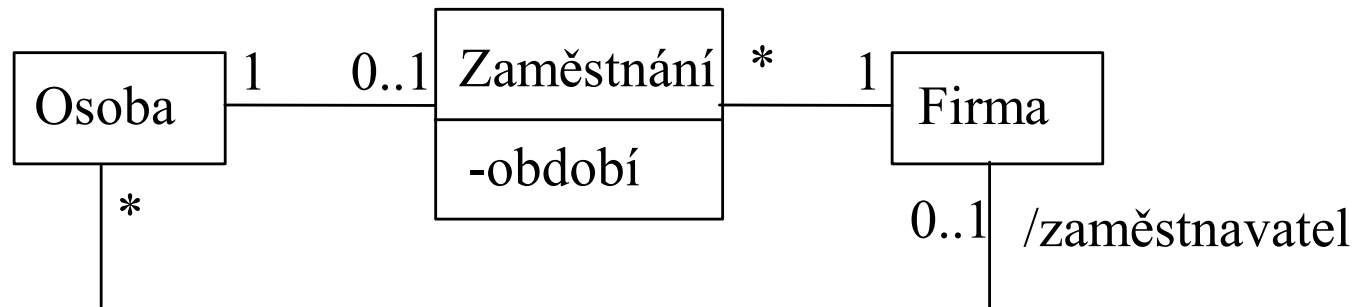
- **Asociační třída**

- asociace s vlastnostmi třídy (atributy, operace, asociace, ...)

**Př)**



- vyjádření pomocí plnohodnotné třídy:



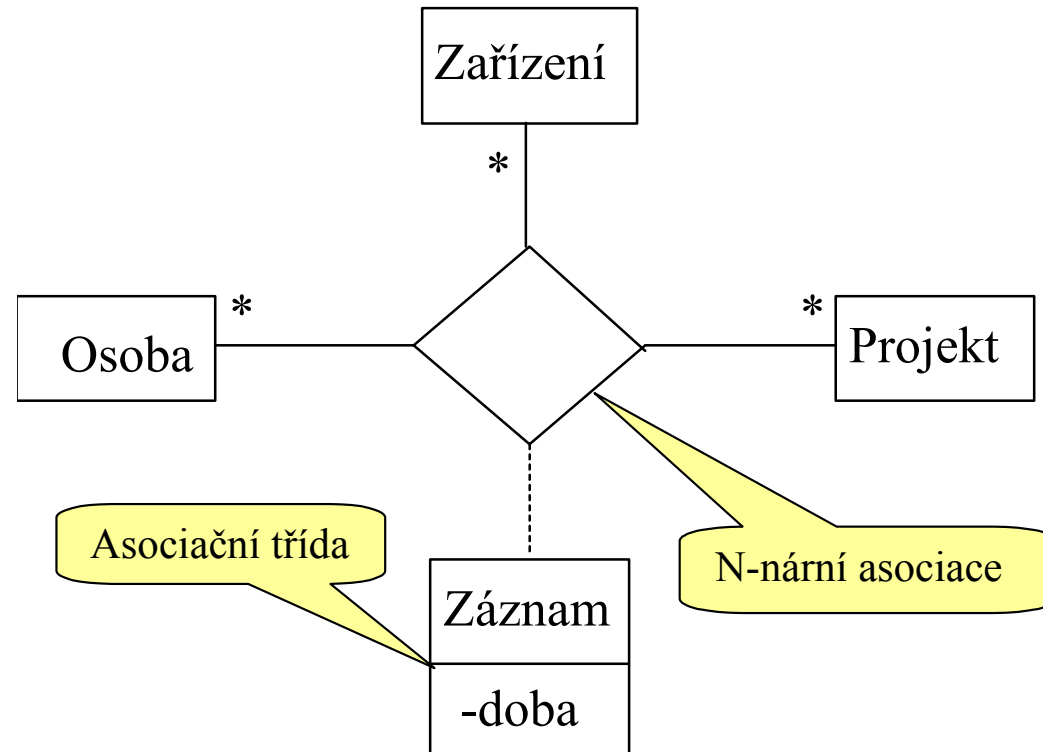
- mezi dvěma participujícími objekty může být jen jedna asociční třída (stejně jako jedno spojení v rámci dané asociace)

**Př)** Osoba – Firma s historií zaměstnání

- často u „historické“ informace

- **N – nární asociace**

Př)



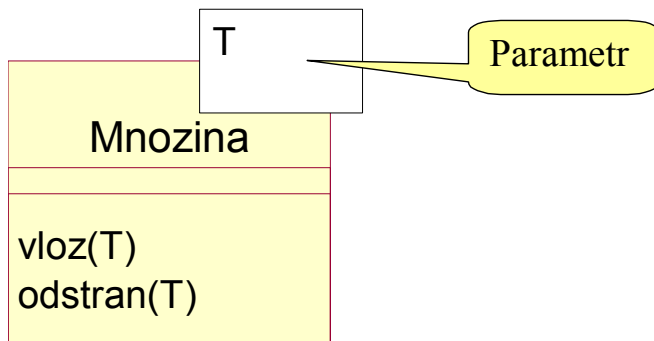
- **obecně nelze nahradit N-nární asociaci binárními beze ztráty informace**

- **Parametrizované třídy**

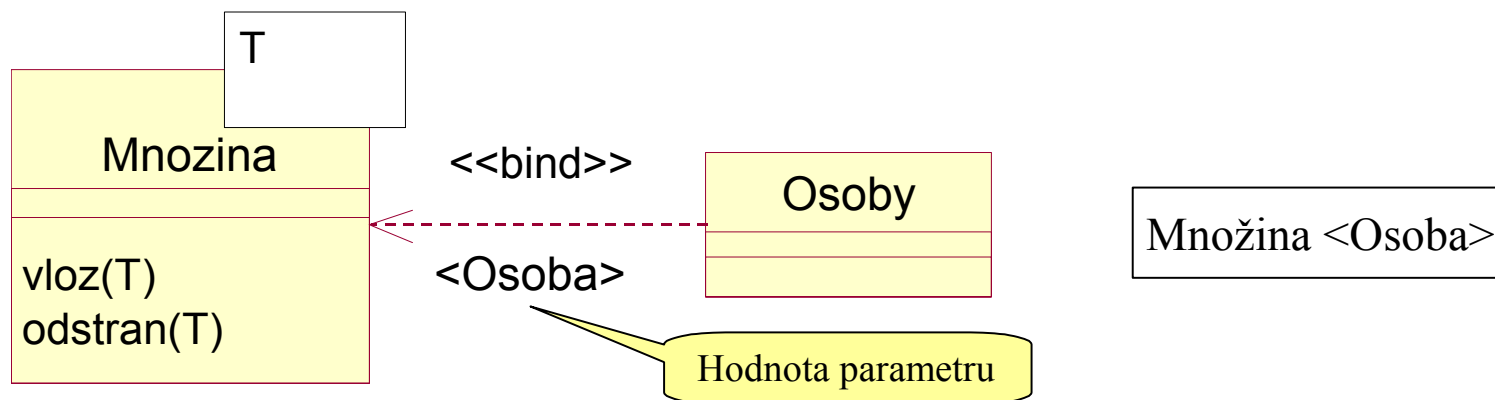
- podpora v některých jazycích (C++ - šablona (template))
- vhodné zejména pro práci s kolekcemi

**Př) class Mnozina <T>**  
**void vloz (T novyPrvek);**  
**void odstran (T prvek); ...}**

**Mnozina <Osoba> osoby;**



**- použití šablony – tzv. vázaný prvek (bound element)**



- **Viditelnost**

vlastnost  $\left\{ \begin{array}{l} \text{veřejná (+, public) - použitelná libovolnou jinou třídou} \\ \text{privátní (-, private) - použitelná jen vlastnickou třídou} \\ \text{chráněná (#, protected) - použitelná třídou a následníky} \end{array} \right.$

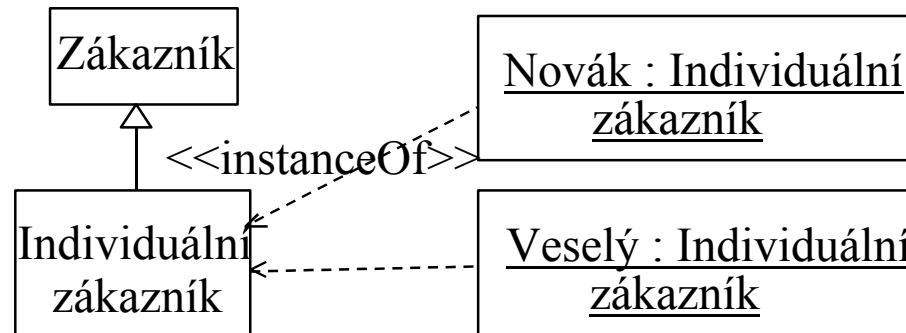
- rozdílný význam u různých jazyků:

**C++:** public – viditelná kdekoliv v programu

private – jen ve třídě, která vlastnost definuje

protected - ve třídě, která vlastnost definuje, a následníky

**Př)**



**Smalltalk:** všechny instanční proměnné (atributy) privátní (přístupné ale i následníkům) a operace veřejné

**Java:** podobně jako C++ (přístup k členům jiných objektů téže třídy), navíc viditelnost „package“ a význam „protected“ rozšířen i na libovolné třídy téhož modulu (package)

**C++:** navíc „friend“ pro třídu nebo metodu