Analýza- objekty a třídy

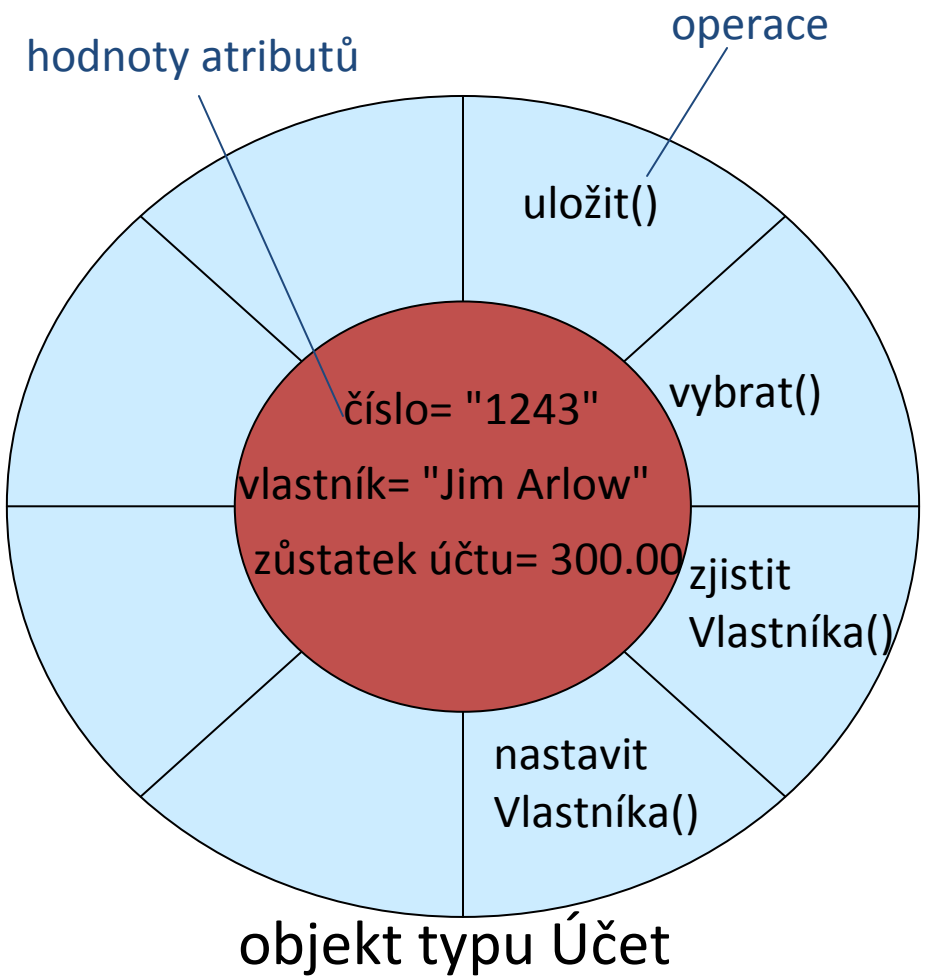
Analýza – část 2

Co jsou to objekty?

- Objekt kombinuje data a funkce do jedné soudržné jednotky. Objekty *zapouzdřují (ukrývají)* data
- Každý objekt je instancí nějaké *třídy,* která definuje společnou *množinu rysů* (atributů a operací či metod), které jsou vlastní všem instancím dané třídy. Objekty mají:
  - Hodnoty atributů – datová část
  - Operace – část chování
- Všechny objekty mají:
  - *Identitu*: Každý objekt lze identifikovat jedinečným způsobem, identita objektu je určitým jedinečným manipulátorem (handle)
  - *Stav*: Je přesně určen hodnotami atributů objektu v určitém okamžiku
  - *Chování*: Nastavení operací, které může objekt vykonávat

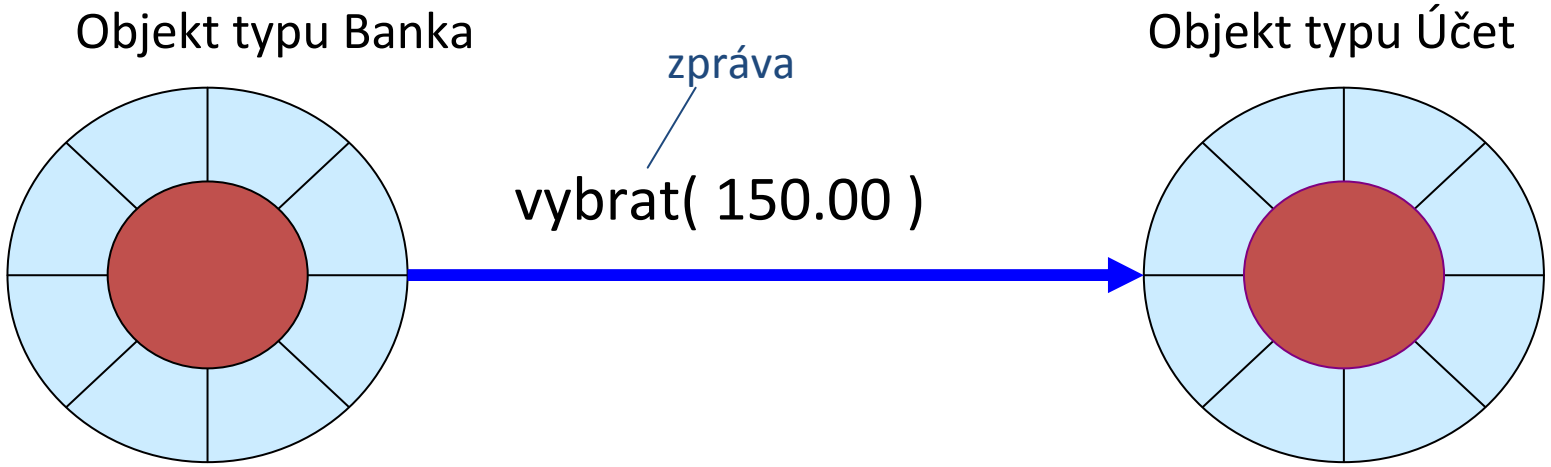
Zapouzdření

- Data jsou ukryta uvnitř v objektu. Jedinou cestou přístupu k datům je přes jednu z operací
- Zapouzdření dat je jednou z hlavních výhod objektově orientovaného programování a může vést k tvorbě robustnějšího a rozšiřitelnějšího softwaru.



Předávání zpráv

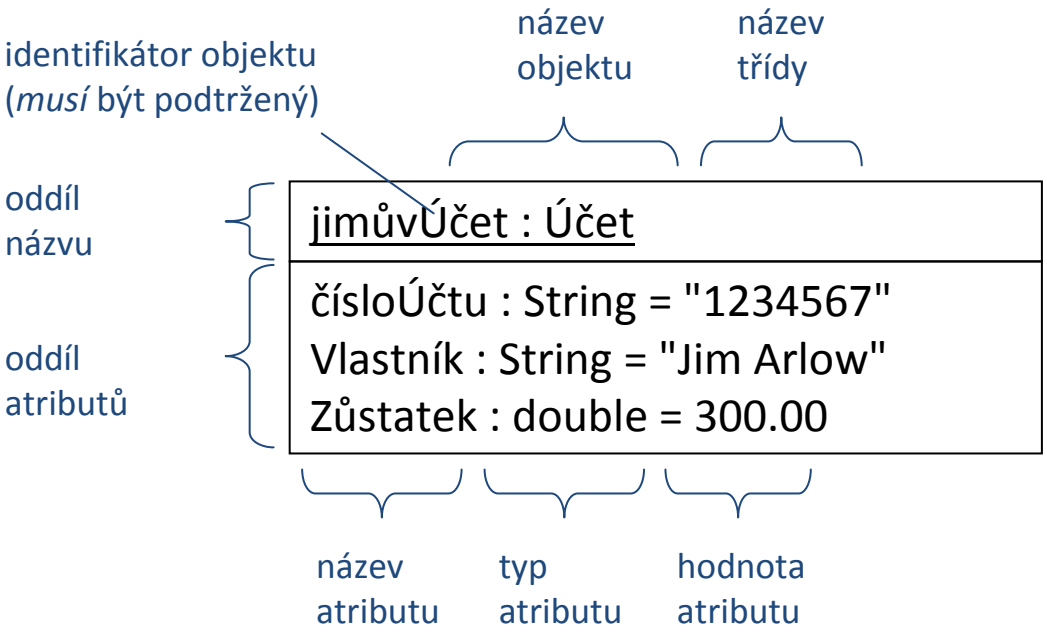
- V OO systému si objekty posílají zprávy navzájem mezi sebou
- Objekty spolupracují proto, aby mohly vykonávat příslušné funkce



Objekt typu Banka odešle objektu typu Účet zprávu „vybrat 150.00“.

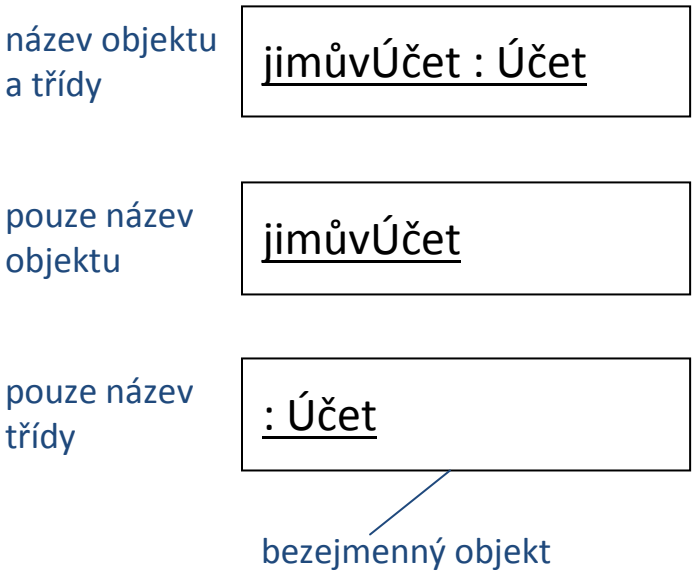
Objekt typu Účet odpoví voláním své operace vybrat. Tato operace sníží zůstatek na účtu o 150.00.

Notace objektů v jazyce UML



varianty

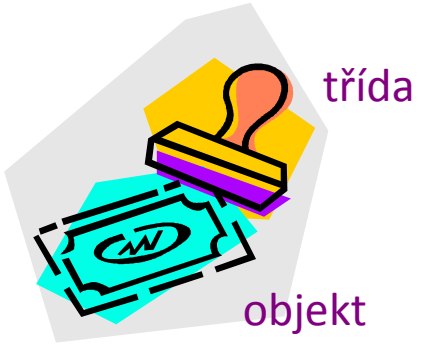
(vynechané části v idetifikátoru objektu)



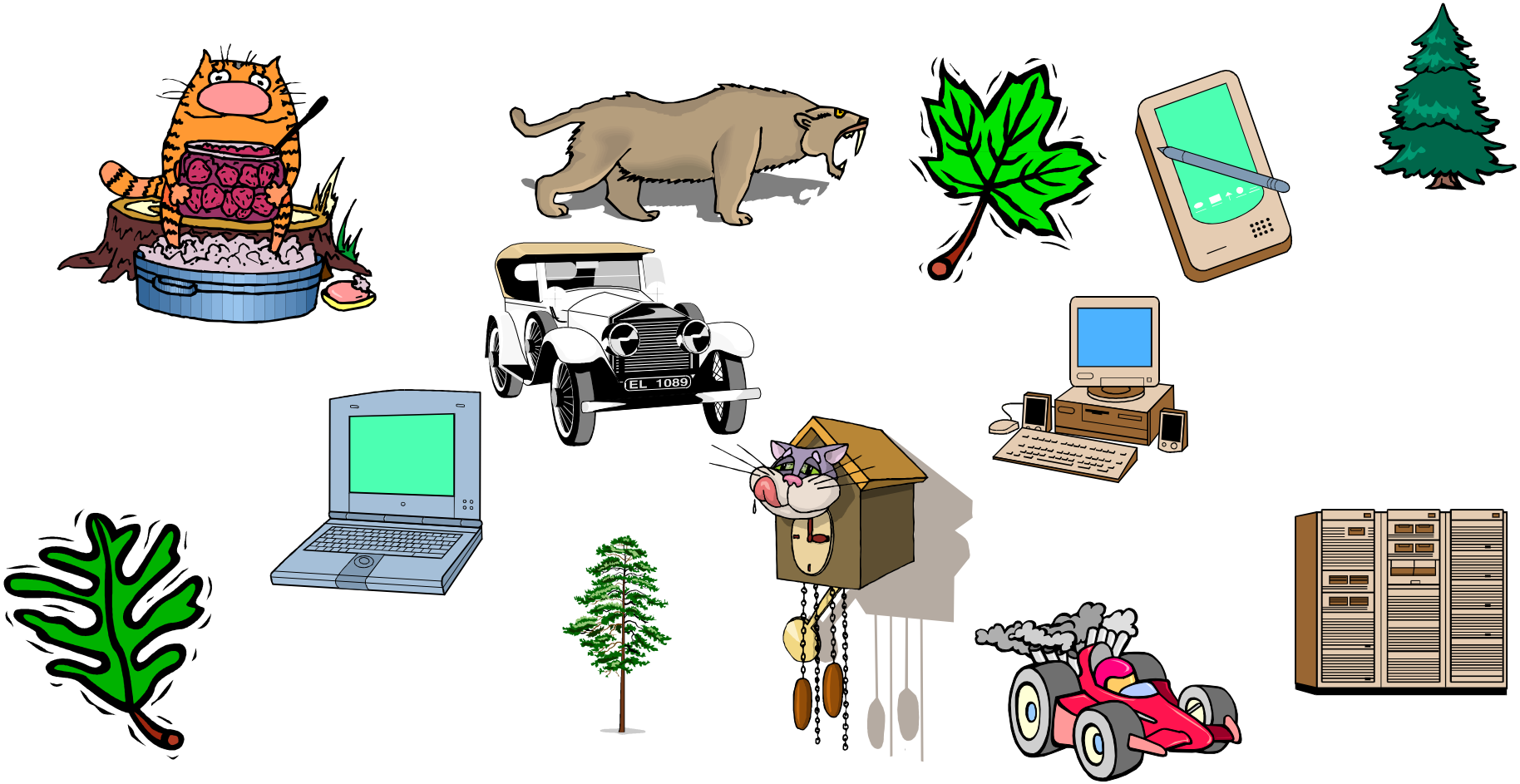
- Objekty stejné třídy poskytují stejné operace a obsahují stejné atributy. Proto jsou operace uvedené v symbolu třídy, nikoliv v symbolu objektu (uvidíme později)
- Typy atributů jsou často vynechány z důvodu zjednodušení diagramu
- Typ zápisu:
  - názvy objektu a atributů v lowerCamelCase (maléVelbloudíPísmo)
  - názvy tříd v UpperCamelCase (VelkéVelbloudíPísmo)

Co jsou to třídy?

- Každý objekt je instancí přesně jedné třídy – třída popisuje „typ“ objektu
- Třída je deskriptorem množiny objektů se *stejnými* charakteristickými vlastnostmi (rysy) – třída je jako šablona objektů:
  - Třída určuje strukturu (nastavení vlastností) všech objektů dané třídy
  - Všechny objekty stejné třídy *musí* obsahovat stejnou množinu operací, stejnou množinu atributů a stejnou množinu relací, jejich atributy však *mohou* obsahovat různé hodnoty
- Klasifikace je jedním z nejdůležitějších způsobů, jímž lidé uspořádávají okolní svět
- Třídu si můžete představit pomocí:
  - Gumové razítko - třída
  - Otisk na papíře - objekt

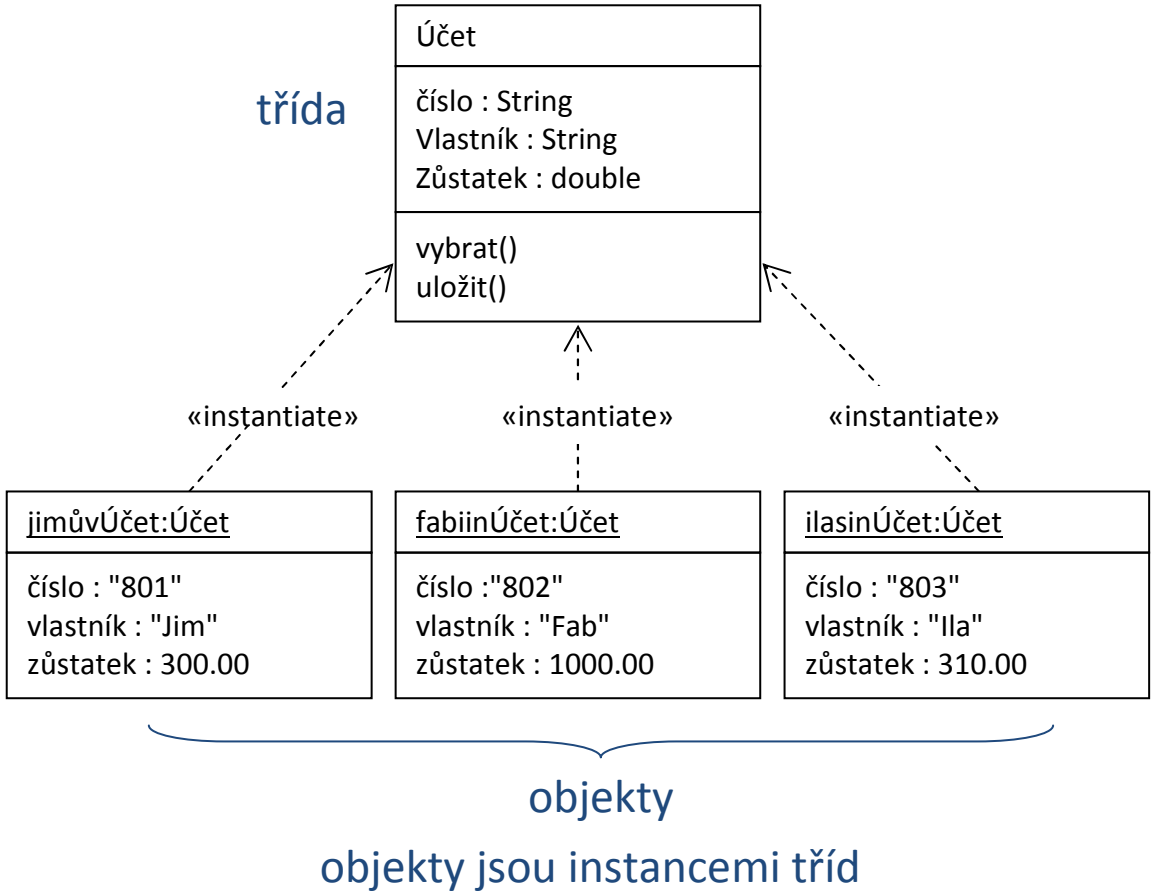


Příklad – kolik je zde tříd?

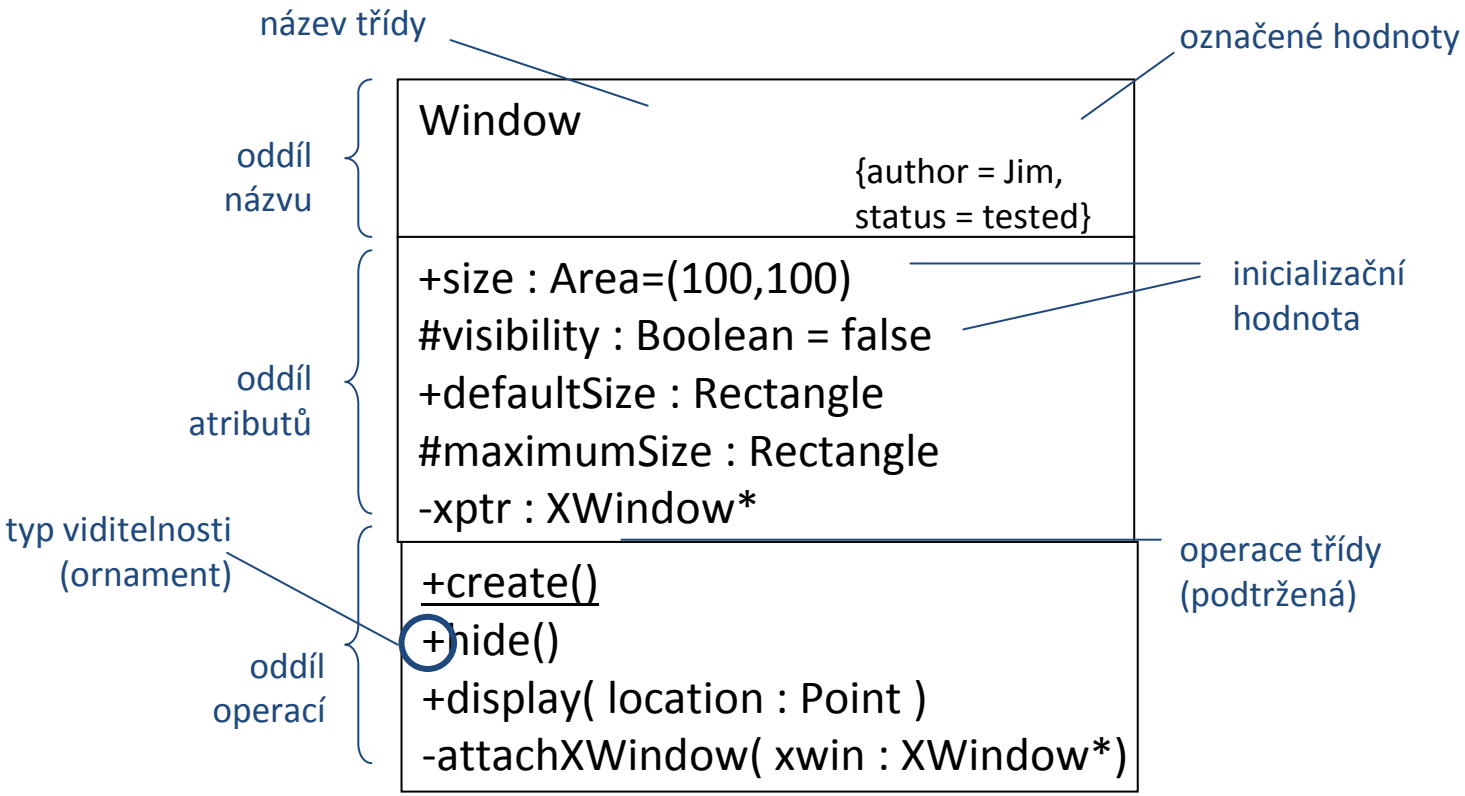


Třídy a objekty

- Objekty jsou instancemi tříd
- UML definuje tvorbu objektu jako “proces tvorby nové instance prvku modelu”
- Speciální metody (operace), tzv. *konstruktory* patří spíše třídě než konkrétním objektům dané třídy. Tyto speciální operace mají platnost třídy a více se o nich dozvíte v oddíle 7.6
- Uvidíte konkrétní využití s dalšími prvky modelování (později)



Notace třídy v UML



- Třídy jsou psány UpperCamelCase (VelkýmVelbloudímPísmem)
- Mohou mít název v podobě podstatného jména nebo jmenné skupiny
- Nikdy nezkracujte názvy tříd, atributů ani operací!

Oddíl atributů

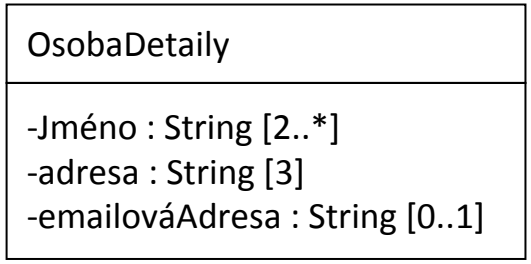
viditelnost název : typ [násobnost] = počátečníHodnota

povinné

- Všechny položky s výjimkou názvu jsou nepovinné
- počátečníHodnota umožňuje nastavit hodnotu atributu v okamžiku tvorby objektu
- Atributy jsou psány lowerCamelCase (malýmVelbloudímPísmem)
  - Mohou mít název v podobě podstatného jména nebo jmenné skupiny
  - Nikdy nezkracujte
- Sémantiku atributů můžete rozšiřovat pomocí prefixů, kde prefix je stereotyp a postfixů, kde postfix je označená hodnota.

Typy viditelnosti

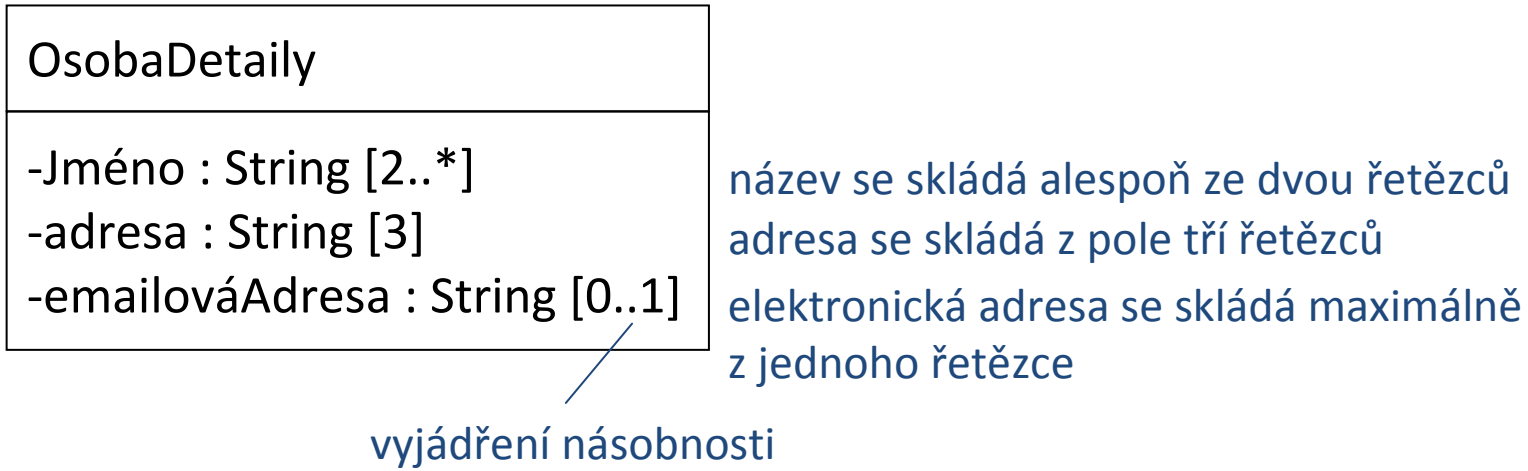
Symbol	Name	Semantics
+	Veřejný (public)	Jakýkoli prvek s přístupem k třídě může používat jakékoli její vlastnosti a funkce deklarované jako veřejné
-	Soukromý (private)	K soukromým vlastnostem a funkcím mají přístup pouze operace uvnitř dané třídy
#	Chráněný (protected)	K chráněným vlastnostem a funkcím mají přístup pouze operace uvnitř dané třídy a uvnitř jejich potomků
~	Balíček (package)	K vlastnostem a funkcím třídy deklarovaným pomocí klíčového slova package mohou přistupovat libovolné prvky z téhož balíčku jako příslušná třída, ale i prvky z vnořených dílčích balíčků



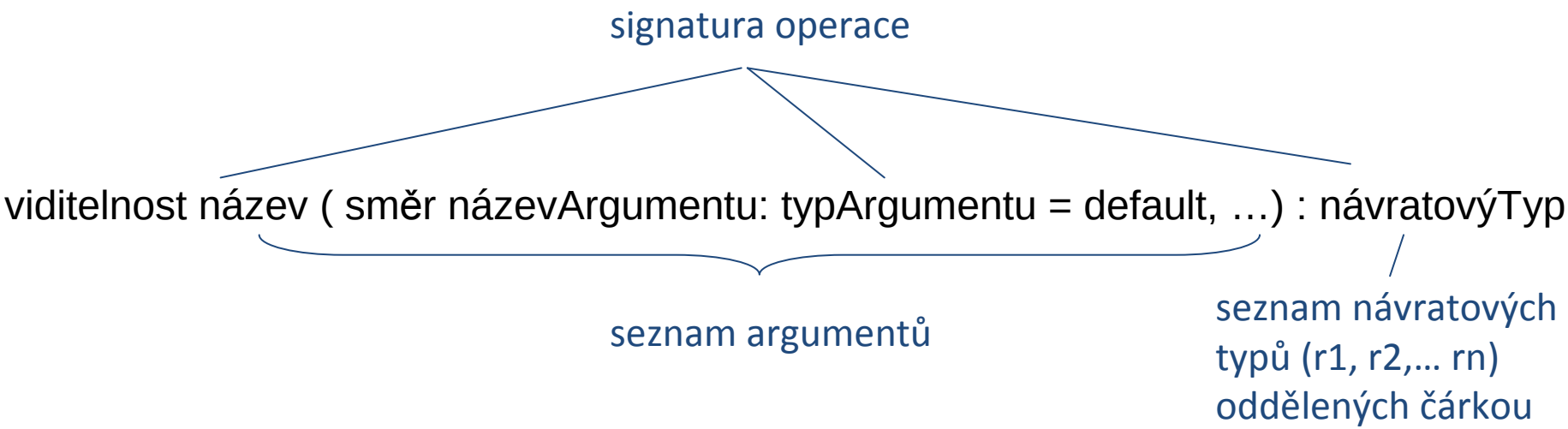
- Při analýze můžete ignorovat viditelnost
- V návrhu mají atributy vždy viditelnost typu soukromý (zapouzdření)

Násobnost

- Násobnost umožňuje modelovat prázdné hodnoty nebo pole předmětů
  - [0..1] znamená, že atributy obsahují hodnotu null (odkaz na chybějící nebo neznámá data)



Oddíl operací



- Operace jsou psány lowerCamelCase (malýmVelbloudímPísmem)
  - V názvech operací se nepoužívají žádné speciální symboly
  - K pojmenování operací se používají slovesa nebo slovesné fráze
- Operace mohou mít více než jeden typ návratové hodnoty
  - Směr argumentu (return) umožňuje modelovat situace, v nichž má operace více návratových hodnot (uvidíte na příštím slide)
- Sémantiku operací můžete rozšiřovat pomocí prefixů v podobě stereotypu a postfixů připojením označených hodnot k názvu operace

Směr argumentu

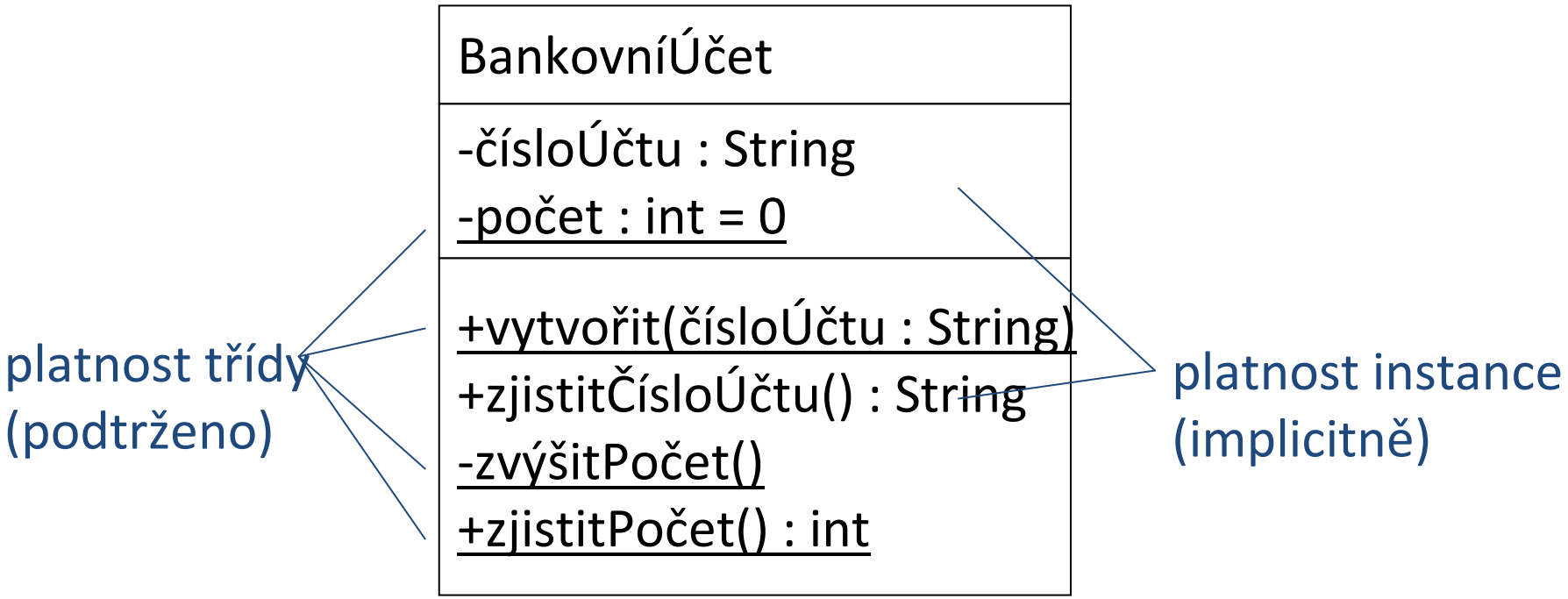
argument	sémantika
in	Operace používá argument jako vstupní. Hodnota není během operace změněna. Implicitní hodnota
out	Argument se používá jako repositář pro příjem výsledné hodnoty operace a může být během operace změněn.
inout	Operace přijímá argument jako vstupní a výstupní. Argument může být během operace změněn.
return	Operace používá argument jako výstupní. Operace vrací argument jako jednu z výstupních hodnot.

Příklad více návratových hodnot:

```
maxMin( in a: int, in b:int, return maxHodnota:int return minHodnota:int )
…
max, min = maxMin( 5, 10 )
```

Rozsah platnosti

- Jsou dva druhy rozsahu platnosti pro atributy a operace:



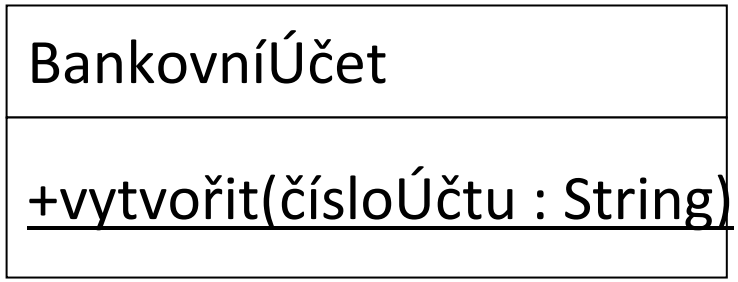
Platnost instance a platnost třídy

	platnost instance	platnost třídy
atributy	Atributy mají implicitně platnost instance	Atributy mohou mít rovněž platnost třídy
	Každý objekt třídy získává vlastní kopii atributů instance	Každý objekt třídy sdílí stejnou kopii atributů třídy
	Každý objekt může obsahovat atributy instance, které nesou jiné hodnoty	Atributy třídy nesou ve všech instancích této třídy stejnou hodnotu
operace	Operace mají implicitně platnost instance	Operace mohou mít rovněž platnost třídy
	Každé volání instanční operace se vztahuje na specifickou instanci příslušné třídy	Volání operace třídy se nevztahuje na žádnou specifickou instanci třídy – operace třídy se vztahují na celou třídu
	Instanční operace nelze volat, dokud nebudete mít k dispozici instanci příslušné třídy. Znamená to, že nelze použít členské metody instance k tvorbě dalších objektů (instancí) dané třídy, neboť před jejím použitím je třeba vždy nejprve vytvořit příslušný objekt	Operace třídy můžete volat, aniž byste měli k dispozici jakoukoli instanci dané třídy – takto mohou ideálně fungovat operace zaměřené na tvorbu objektů

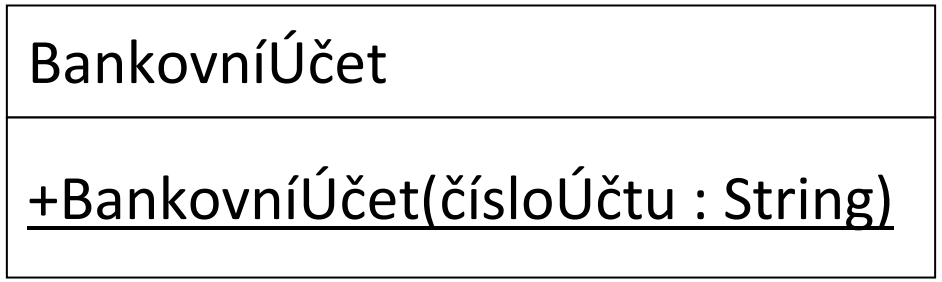
Tabulka s porovnáním platnosti

Tvorba a uvolnění objektů

- Jak vytvoříte instance tříd?
- Každá třída definuje jednu nebo více operací s platností třídy, které jsou nazývány *konstruktory*. Tyto operace vytváří nové instance příslušné třídy.



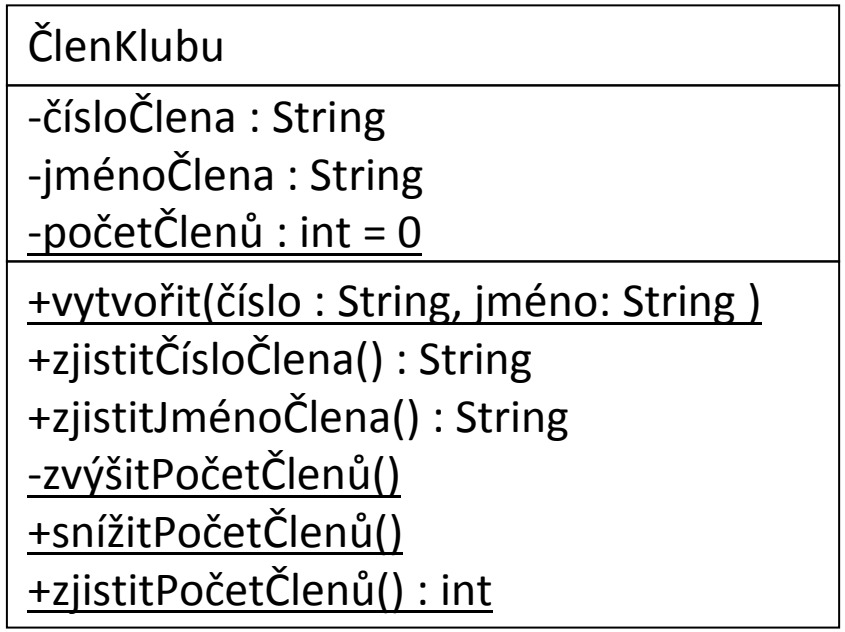
obecný název konstruktoru



Standard v jazycích Java/C++

Konstruktory – ukázková třída ČlenKlubu

- Každý objekt ČlenKlubu má vlastní kopii atributu čísloČlena (platnost instance)
- Atribut početČlenů existuje pouze jednou a je sdílený všemi instancemi třídy ČlenKlubu (platnost třídy)
- Předpokládejme, že zavoláme operaci zvýšitPočetČlenů, která má platnost třídy:
  - Jaký bude početČlenů, když jsme vytvořili 3 objekty?



Shrnutí

- Podívali jsme se na objekty a třídy a vazby mezi nimi
- Prozkoumali jsme syntaxi UML pro modelování tříd zahrnujících:
  - Atributy
  - Operace
- Viděli jsme, že přístup je určen rozsahem platnosti
  - Atributy a operace mají implicitně platnost instance
  - Můžeme použít konstruktory a destruktory s platností třídy
  - Operace instance mohou používat všechny atributy a operace mající platnost třídy