Objektovo-orientované programovanie

Horňák Peter, 5I031 11.2009

Osnova prezentácie

- Základné princípy objektovo orientovaného programovania
- OOP a štruktúra objektovo orientovaného programu
- Zapúzdrenie
- Dedičnosť
- Polymorfizmus

Objektovo orientované prog.

- **Základný princíp**: program pozostáva z množiny objektov, ktoré sú schopné uchovávať a spracovávať dáta a komunikovať s ostatnými objektami
- História: OOP vzniklo v období, keď bežné programy začali presahovať určitú dĺžku a štrukturálne programy sa stali neprehľadnými. Bežne sa začalo využívať začiatkom 90. rokov
- Je to efektívny spôsob organizácie programu
- Hlavná výhoda: dobre navrhnutá objektová štruktúra programu umožňuje oveľa lepšiu orientáciu v kóde

Princíp OOP

- Filozofia OOP je postavená na usporiadaní reálneho sveta
- Základné pojmy OOP: **trieda** a **objekt**
- Tri základné princípy:

  Zapúzdrenie (encapsulation)

  Mnohotvárnosť (polymorphism)

  Dedičnosť (inheritance)

Trieda

- Je to štrukturovaný dátový typ charakterizovaný vlastnosťami (dáta, atribúty) a schopnosťami (metódy)
- Príklad: trieda Človek
- Vlastnosti: meno, vek, výška, váha, …
- Schopnosti: predstaviť sa, povedať svoj vek, mier
  opýtať sa iného človeka na jeho meno, vek, ...

Objekt

- Konkrétny prvok triedy, s jednoznačne danými vlastnosťami

Homer Simpson

39 rokov

hnedé vlasy

Marge Simpson

36 rokov

modré vlasy

Objektovo orient… program

1. Definícia triedy

Trieda **Človek** {

znakový reťazec meno ←

celé číslo vek ←

znakový reťazec farba_vlasov ←

Zapíš_si_svoje_údaje ←

Predstav_sa ←

Povedz_svoj_vek ←

}

Dáta - vlastnosti

Metódy - schopnosti

Objektovo orientovaný program

2. Definícia metód

```
Človek::Zapíš_si_svoje_údaje {
    meno = ...
    vek = ...
    farba_vlasov = ...
}
Človek::Predstav_sa {
    Povedz “Ahoj, ja som”  meno
}
```

Objektovo orient… program

3. Vytvorenie a použitie objektov

Človek Homer, Marge

Homer.Zapíš_si_svoje_údaje

Marge.Zapíš_si_svoje_údaje

Homer.Predstav_sa

Homer.Povedz_svoj_vek

Marge.Predstav_sa

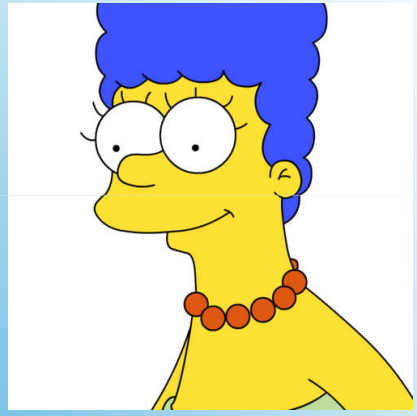
Marge.Povedz_svoj_vek

Objektovo orientovaný program

4. Výstup

Ahoj, ja som Homer Simpson

Mám 39 rokov



Ahoj, ja som Marge Simpson

Mám 36 rokov

Zapúzdrenie

- Mechanizmus, ktorý zväzuje dohromady dáta a kód
- **V úplne objektovo orientovanom programe patria všetky dáta a funkcie nejakej triede**
- Zapúzdrenie (encapsulation) umožňuje lepšiu prehľadnosť programu a najmä môže chrániť dáta pred nežiadúcimi zásahmi zvonku
- Vo vnútri triedy môžu byť všetky dáta alebo metódy definované ako

  súkromné - prístupné len pre triedu samotnú

  verejné – prístupné aj pre ostatné triedy

Zapúzdrenie

Trieda **Človek** {

verejné:

znakový reťazec meno

Predstav_sa

}

Človek Marge

Marge.Predstav_sa

Povedz “Ahoj, ja som” Marge.meno

Zapúzdrenie

Trieda **Človek** {
súkromné:
znakový reťazec meno
verejné:
Predstav_sa
}

Človek Homer

Homer.Predstav_sa
Povedz "Ahoj, ja som" Homer.meno

Technika Data hi

- Je to najbezpečnejší a najbežnejší spôsob návrhu tried
- Základný princíp:

  všetky dáta sú súkromné

  trieda má vytvorený interface, teda metódy, ktoré umožňujú zmenu a sprostredkovanie dát, ak je to potrebné

- Takto sa zabezpečí, že pri použití triedy sú prístupné a meniteľné len tie dáta, ktorým to dovolí interface, čo je dôležité napr. pri programovaní knižníc, ktoré budú používať iní programátori a pod.

Rozhranie objektu (int

- rozhranie objektu definuje operácie, ktoré je objekt schopný vykonať
- je nutné ešte zapísať kód, ktorý danú požiadavku spracuje; kód a skryté dáta spolu tvoria implementáciu
- s každou požiadavkou je v triede spojená funkcia - metóda, ktorá sa vyvolá v okamžiku poslania požiadavky

názov triedy

rozhranie

Konštruktor a deštru

- Sú to funkcie, ktoré sa automaticky spustia pri vytvorení (konštruktor) a pri zániku (deštruktor) objektu danej triedy
- **Konštruktor** sa používa najmä na počiatočné nastavenie hodnoty dát daného objektu a na alokáciu potrebnej pamäte
- **Deštruktor** sa používa na “upratanie”, teda hlavne na dealokáciu vyhradenej pamäte

Konštruktor a deš

Človek::Konštruktor {

Alokuj pamäťové miesto na miery

miery[1]= … (vek)

miery[2]= … (výška)

miery[3]= … (váha)

}

Človek::Deštruktor {

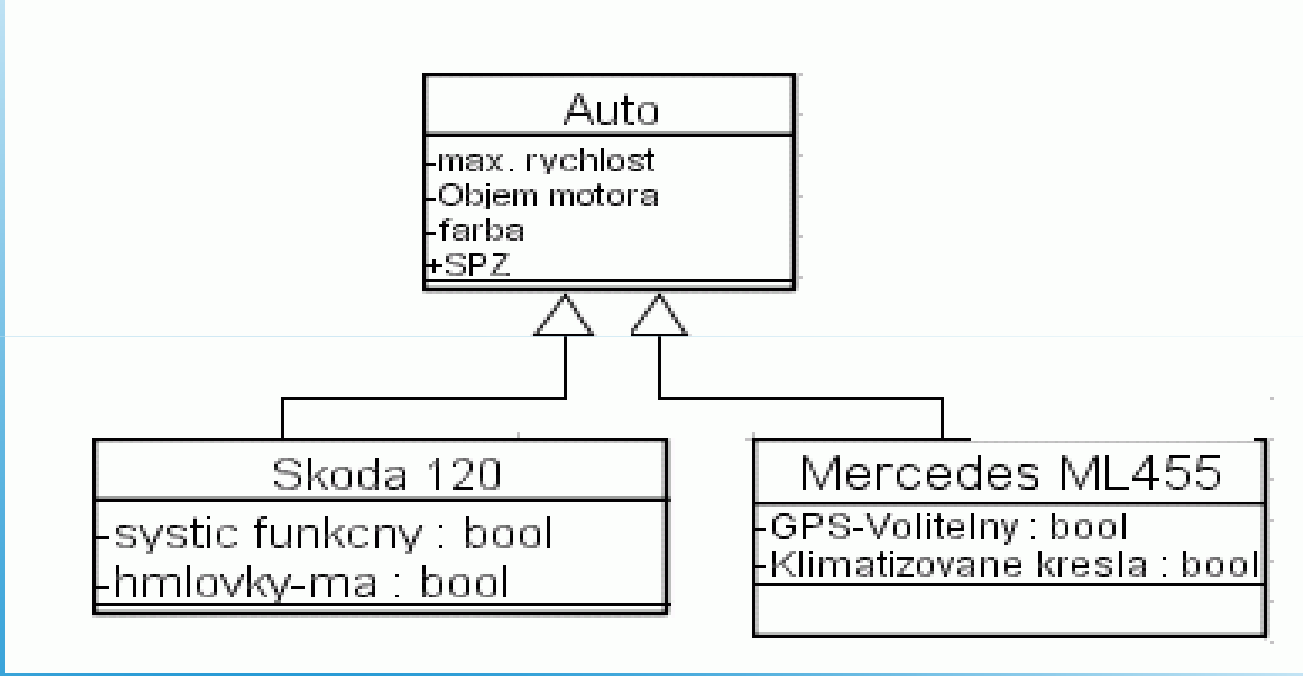
Dealokuj miesto vyhradené pre miery

}

Dedičnosť

- Každá trieda môže mať svoje “dieťa”, teda triedu, ktorá je od nej odvodená, preberá všetky jej dáta a metódy (okrem konštruktora a deštruktora)
- Odvodená trieda môže byť potomkom ľubovoľného počtu tried a každá trieda môže mať ľubovoľný počet potomkov
- Pre rodičovskú triedu je možné sprístupniť svojim potomkom svoje súkromné dáta, takéto dáta sa nazývajú **chránené** a okrem samotnej triedy a jej podtried nie sú inak zvonku prístupné

Dedičnosť


Auto
-max. rychlost
-Objem motora
-farba
+SPZ
Skoda 120
-systic funkcny : bool
-hmlovky-ma : bool
Mercedes ML455
-GPS-Volitelny : bool
-Klimatizovane kresla : bool

Dedičnosť

Trieda **Rodič:**

verejné:

Predstav_sa

Povedz_adresu

chránené:

meno

adresa

súkromné:

PIN

Trieda **Dieťa:**

Predstav_sa

Povedz_adresu

Povedz_zákonného_zástupcu

meno

adresa

zákonný_zástupca

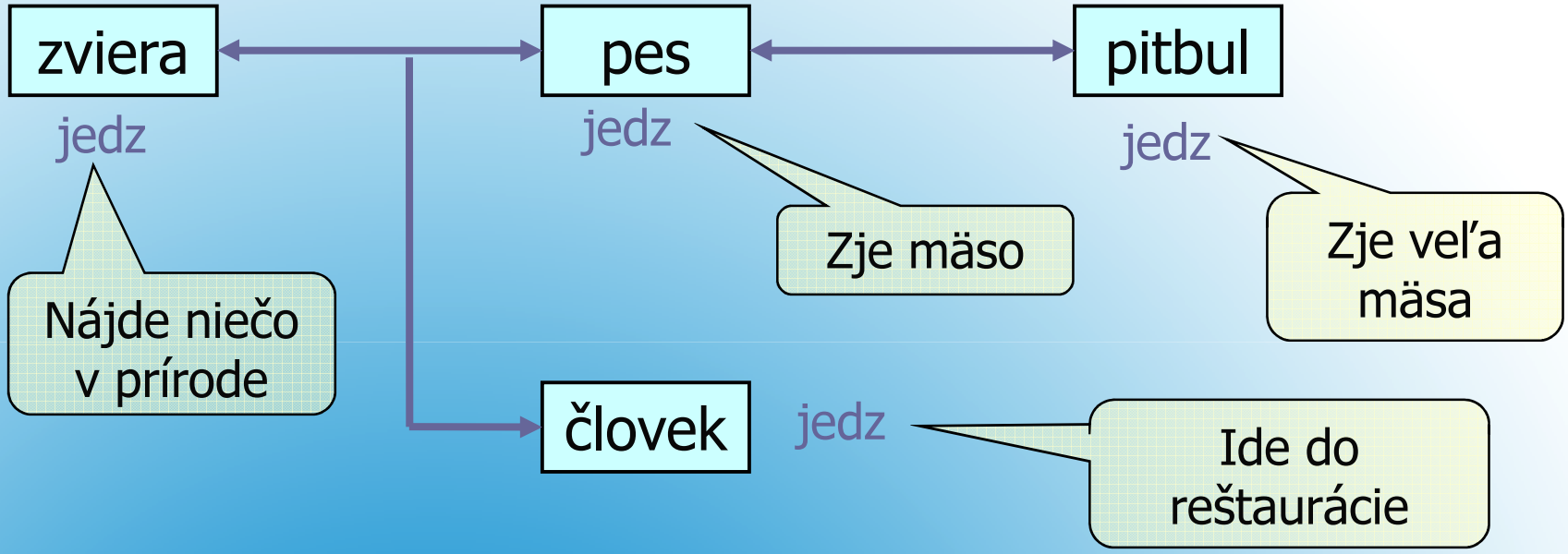
PIN – neprístupné!

Polymorfizmus

- Ide o mnohotvárnosť, resp. viacúčelové využitie metód
- Metóda s jedným názvom môže byť použitá pre rôzne typy dát alebo rôzny počet vstupov, čo uľahčuje orientáciu v programe

Virtuálne metódy

•Ak je metóda rodičovskej triedy virtuálna, znamená to, že potomok, ktorý ju zdedí, si ju môže zmeniť podľa svojich potrieb


zviera
pes
pitbul
jedz
jedz
jedz
Nájde niečo v prírode
Zje mäso
Zje veľa mäsa
človek
jedz
Ide do reštaurácie

Zhrnutie

- Objektovo orientované programovanie je spôsob efektívnej organizácie programu, pri ktorom je program súborom navzájom spolupracujúcich objektov
- Trieda je predloha, objekt je reálny
- Základné princípy objektovo orientovaného programovania sú:

  Zapúzdrenie

  Dedičnosť

  Polymorfizmus

Použité zdroje

Prednášky Ing. Sadloňa z predmetu objektové programovanie

http://cs.wikipedia.org/wiki/Objektově_orientované_programování

http://en.wikipedia.org/wiki/Object-oriented_programming

Ďakujem Vám za pozornosť