Programovanie v jazyku C#

Správanie sa objektov

Ing. Martina Szabóová, PhD.

Zapuzdrenie (Encapsulation)

Proces ,,zaškatuľkovania" elementov abstrakcie, ktoré tvoria jej štruktúru a správanie. Účelom zapúzdrenia je oddeliť rozhranie s kontraktom abstrakcie od jej implementácie. (Grady Booch)

- objekt ma verejné rozhranie (**interface**) a to sa používa na interakciu s ním
- triedy majú byť „nepriehľadné" (t.j. nešpecifikujú svoje implementačné detaily) ale zároveň majú mať dobré rozhranie aby sa s nimi dobre pracovalo

Zapuzdrenie (Encapsulation)

Interface, Contract

- **Interface** (rozhranie) - formálny popis toho, čo môžeme zadať
- **Contract** (kontrakt) je dohoda, čo tam dáme a nedáme, obvykle je popísaný v dokumentácii
  - keď sa na metódu pozeráme ako matematickú funkciu, tak kontrakt definuje jej vlastnosti:
    - definičný obor – parametre a ich typy
    - obor hodnôt – návratová hodnota
    - definícia správania, ak príde hodnota mimo definičného oboru

Zapuzdrenie (Encapsulation)
Interface, Contract

- Kalkulacka - sqrt; double Math.Sqrt(double d)
  - interface ?
  - contract ?
- Tankovanie
  - interface ?
  - contract ?

Zapuzdrenie (Encapsulation)

Interface, Contract

- Kalkulacka - sqrt; double Math.Sqrt(double d)
  - interface - môžeme zadať hocijaké číslo
  - contract - vieme, že môžeme zadať len nezáporne číslo
- Tankovanie
  - interface ?
  - contract ?

Zapuzdrenie (Encapsulation)

- **abstrakcia** - formálny popis, výsledkom abstrakcie je trieda a jej interface
  - oddeľuje vnútorné (skryté) detaily objektov a ich vonkajšie detaily
  - hlavným nástrojom abstrakcie je zapuzdrenie

Proces ,,zaškatuľkovania" elementov abstrakcie, ktoré tvoria jej štruktúru a správanie. Účelom zapúzdrenia je oddeliť rozhranie s kontraktom abstrakcie od jej implementácie.

Zapuzdrenie (Encapsulation)

- **členské premenné** triedy predstavujú stav objektu
  - stav objektu - definovaný a môže byť menený iba členskými premennými
  - stav objektu môže byť menený iba procesmi v triede
- **metódy** triedy predstavujú správanie objektu (procesy v objekte)
  - niektoré procesy su vnútorne a menené iba v rámci objektu
  - niektoré procesy predstavujú interakciu objektu s inými objektmi

Proces ,,zaškatuľkovania" elementov abstrakcie, ktoré tvoria jej štruktúru a správanie. Účelom zapúzdrenia je oddeliť rozhranie s kontraktom abstrakcie od jej implementácie.

Skrývanie informácií

- členská premenná = dátový člen = inštančná premenná
- všeobecné pravidlo: Používajte **privátn**e dátové členy, a k nim vhodné **gettre** (accessors) a **settre** (mutators)
- *Ukážka programu*

Použitie gettrov a settrov

- môžeme obmedziť hodnoty
- *Ukážka programu*
- ak by užívatelia pristupovali k premenným priamo = sú zodpovední za overenie obmedzení
- ak užívatelia pristupovali k premenným priamo = sú zodpovední za vykonanie vedľajších efektov

Modifikátor viditeľnosti

- prístupnosť každého člena triedy je definovaný modifikátor viditeľnosti
- **privátny** (private)
  - obmedzuje prístup k atribútom a metódam na tie metódy triedy, ktoré sú prvkami tej istej triedy (definované v tej istej triede)
- **verejný** (public)
  - dostupnosť z vonkajška triedy
  - hocaký iný objekt môže pristúpiť k členovi triedy
- **chránený** (protected)

Zapuzdrenie (Encapsulation)

- nezvereňujte vnútornú štruktúru triedy
- nezvereňujte detaily o internom stave triedy
- nezvereňujte rozdiel medzi uloženým a vypočítaným stavom
- nezvereňujte implementačné detaily triedy
- dáta a operácie nad nimi nech sú v ronakej triede

Použitie kľúčového slova *this*

- v rámci inštancie metódy alebo konštruktora - *this* odkazuje na aktuálny objekt (objekt, ktorého metóda alebo konštruktor je volaný)
  - môžeme referovať na akéhokoľvek člena triedy aktuálneho objektu z vnútra inštancie metódy alebo konštruktora
- OOP jazyky implementujú referencovanie aktuálneho objektu
  - k pristúpeniu „shadowed“ člena triedy
  - k pristúpeniu seba samého

Použitie *this* s členskou premennou

Použitie *this* v konštruktore

- explicitné volanie konštruktora
  - v rámci jedného konštruktora voláme ďalší konštruktor tej istej triedy

Statické inicializátory

- existencia statických členov triedy nám narúša všetko čo, o čom sme rozprávali
- statický člen je volateľný i keď neexistuje inštancia triedy
- vždy prístupný pod menom triedy (nie inštancie)
- existuje len jedna kópia statického člena
- ak vaša trieda obsahuje statické polia, nezabudnite na statický konštruktor, ktoých ich inicializuje, keď je trieda načítaná
- podpora globálnych premenných a globálnych metód (bez potreby vytvorenia triedy)

Členovia triedy

- statické premenné
  - používajte **statické premenné** na definovanie statických premenných triedy
  - statické premenné používaju statický memory mód
- statické metódy
  - používajte statické kľúčove slovo na definovanie statickej premennej triedy
  - môžu byť volané priamo z inštancie triedy

Prečo statické premenné?

- ako globálna premenná
- hodnota je uložená v statickej pamäti, ktorá je spoločná pre celú triedu
- výhodné ak chceme premennú, pre všetky inštancie triedy

Členské premenné vs statické premenné

- v rámci triedy (všetkých jej inštancií) máme jednu kópiu … premennej
- v rámci objektu máme jednu kópiu … premennej
- každý objekt môže mať vlastnú … premennú
- //každý objekt má prístupnú tú istú … premennú

Statické metódy

- trieda môže mať metódy definované ako static
- k statickým metódam môžeme pristupovať bez vytvorenia objektu
  - (nie je potrebné vytvárať inštancie objektov)
- nevedia pristupovať k dátovým členom svojej triedy
- príklad použitia: pseudotriedy (ktoré sú iba akási zbierka užitočných metód a konštánt) library class, factory method (statická metóda vracajúca inštanciu triedy)

Obmedzenia statických metód

- môžu volať len ostatné statické metódy
- môžu pristupovať len k statickým premenným
- nemôžu referovať na *this*

Kedy použiť statické metódy a statické premenné?

- je nutné dobre zvážiť, ich použitie môže viesť k zlému návrhu a mnohým ďalším problémom
- naozaj potrebujeme danú hodnotu pre danú triedu iba raz? nebudeme ich v budúcnosti potrebovať viac?
- v praxi existuje len niekoľko konkrétnych situácií, keď majú statické premenné svoj zmysel (napr. konštanty, počet inštancií tried, …)

Singleton

- návrhový vzor, ktorý zamedzí vytváraniu inštancií a dovolí vytvoriť iba jednu inštanciu triedy hocikde v programe
- Factory method (ďalší návrhovy vzor), ktorý overí, či už existuje inštancia triedy, v prípade že existuje vráti triedu inak vytvorí inštanciu novej triedy
- Originál (ďalší návrhový vzor), ktorý dovolí vytvoriť len jednu inštanciu triedy s rovnakými parametrami
- *Príklad*

Ďakujem za pozornosť