LEKCIA 1: OPAKOVANIE OOP, VS2010, KONVENCIE, JAVA POROVNANIE

Verze 1.1 – poslední změna 21.2.2012 20:45

OBJEKTOVO ORIENTOVANÉ PROGRAMOVANIE

Predmet predpokladá znalosti základných princípov objektového programovania: Object-Oriented Programming (C# and Visual Basic). Pojmy **objekt** a **trieda**, kde trieda predstavuje typ objektov a samotný objekt predstavuje použiteľnú inštanciu triedy. Objekty často predstavujú vhodnú abstrakciu objektov z reálneho sveta. Trieda má svoje zložky, sú nimi **metódy** a **atribúty** (**premenné**, **properties**). Trieda väčšinou predstavuje jednu funkcionalitu alebo inak povedané mala by mať vždy len jeden dôvod ku zmene. Základné koncepty objektového programovania sú:

Dedičnosť

Je možnosť odvodiť nejakú triedu z inej triedy za účelom rozšírenia, modifikácie či znovu použitia. Využívame ju vtedy ak je nejaká trieda špeciálnym prípadom inej triedy (vzťah generalizácia - špecializácia). Podtrieda musí byť vždy schopná zastúpiť svoju nadtriedu. V **C#** môže trieda dediť iba od jednej inej triedy. Trieda však môže implementovať niekoľko rozhraní (viz. nižšie).

Polymorfizmus

Súvisí s dedičnosťou a v podstate hovorí že ak používame potomka (špecializovanú triedu) priradeného do premennej typu predka (čo z definície dedičnosti vždy môžeme), a voláme na ňom operáciu tohto predka (ktorá je v C# navyše označená ako **virtual**), použije sa implicitne implementácia z potomka (za využitia kľúčového slova **override**)

Zapuzdrenie

Dáta a operácie jedného objektu sú považované za jeden nedeliteľný celok. Vnútorná implementácia a reprezentácia objektu je skrytá navonok a z vonkajšieho pohľadu na objekt sú dostupné len nevyhnutné verejné zložky (public).

ĎALŠIE POJMY OBJEKTOVÉHO PROGRAMU A ČASTO VYUŽÍVANÉ PRVKY

Konštruktor je metóda vyvolaná (iba) pri vytváraní objektu a má rovnaký názov ako trieda.

Rozhranie reprezentuje kontrakt operácií, ktoré musí každá trieda implementovať syntakticky presne podľa definície.

Pole je dátová štruktúra obsahujúca určitý počet premenných rovnakého typu indexovaných od nuly.

Kolekcie sú špecializované triedy pre ukladanie dát a ich získavanie. Kolekcie najčastejšie využívajú koncept generík, umožňujúci definovať triede typovú parametrizáciu. Základné často využívane kolekcie sú najmä **`Dictionary<TKey, TValue>`** (ekvivalent Map z Javy) a **`List<T>`**.

V C# tiež využívame funkcionalitu výnimiek, pomocou ktorej je zabezpečené reagovanie na výnimočné a často chybové situácie.

ZÁSADY DOBRÉHO PROGRAMOVANIA

DRY - Don't repeat yourself

KISS - Keep it simple, stupid!

SOLID - ďalšie prednášky.

JMENNÉ KONVENCE

Při vývoji software je jednou z největších položek údržba. Jednotné jmenné konvence dovolují rychlejší čtení a porozumění kódu programátory, a také jim poskytují dodatečné informace o kódu.

Dokument Design Guidelines for Developing Class Libraries shrnuje pokyny pro vývoj knihoven na platformě .NET tak, aby byly konzistentní s knihovnami samotného Frameworku. První částí tohto dokumentu jsou právě pokyny pro pojmenování (Guidelines for Names).

VELIKOST PÍSMEN - KAPITALIZACE

Identifikátory - názvy objektů, proměnných a metod - v jazyce C# jsou citlivé na velikost písmen (case-sensitive). Pro zápis identifikátorů se používají jen následující dvě konvence:

Pascal Case

Víceslovné identifikátory jsou psány dohromady, bez mezer mezi slovy. Každé ze slov začíná velkým písmenem: `ThisIdentificatorIsWrittenInPascalCase`.
Pascal case se používá na **názvy všech metod**, tříd, vlastností. Jinak řečeno pro všechno, co je veřejné.

Camel Case

Stejně jako Pascal Case, ale první písmeno identifikátoru je malé: `camelCaseIdentificator`.
V C#/.NET je camel case vyhrazen jen pro lokální **proměnné**, **parametry** metod, **soukromé datové položky** třídy.

VOLBA JMEN

"You can't give a variable a name the way you give a dog a name – because it's cute or it has a good sound."
– Steve McConnell

Používejte vhodné názvy:

- snadno čitelné, úplné a přesné, nezkracujte na úkor čitelnosti a srozumitelnosti;
- používejte jen alfanumerické znaky anglické abecedy, ostatním znakům se vyhněte;
- používejte používejte pro názvy identifikátorů angličtinu;
- využívejte kontext pro zkrácení názvů (např.: `class Customers { public void Add~~Customer~~(...); }`);
- volte názvy z oblasti problému, ne řešení (`inputString` x `employeeRecord`).

jmenné prostory (`namespace`)	Název jmenného prostoru se tvoří: `Firma.Produkt(Technologie).Komponenta`. `Komponenta` může být dále dělena tečkovou notací. Jmenný prostor odpovídá názvu projektu + hiearchie složek v projektu. `Microsoft.VisualStudio.Editor, System.Collections.Generics`.
assembly	Názvy assembly obvykle kopírují název největšího jmenného prostoru (+přípona .exe|.dll).

třídy a struktury	Název třídy je tvořen podstatným jménem nebo jmennou frází.
	Pokud se chce zdůraznit hiearchie typů, přidává se za jméno typu název společného rodiče. Ze standardních typů to platí zejména pro výjimky, atributy, proudy:
	`class IO`**`Exception`**` : Exception { ... }`
	`class FileNotFound`**`Exception`**` : IOException { ... }`

rozhraní	Před názvem rozhraní je vždy velké písmeno `I`. Používá se podstatné jméno, jmenná fráze nebo přídavné jméno. `IDataRecord`, `IAnimal`, `IEnumerable`, `ICloneable`, `IComparable`.
metody	Slovesa v rozkazovacím tvaru a slovesné fráze (sloveso+předmět): `Kill()`, `KillCat()`, `OpenWindow()`, `CloseForm()`.
vlastnosti, proměnné	Pokud to jde použijte stejný název jako má typ (~pokud nění třeba další rozlišení). Volte podstatné jméno nebo jmennou frázi.
konstanty	Konstanty na úrovni tříd se zapisují Pascal case, konstanty v metodách pak camel case.
události	Používají se slovesa a slovesné fráze (`Click`, `Load`). Souslednost je vyjádřena časem slovesa (před zavřením `Closing`, po zavření `Closed`).
typové proměnné	Písmeno `T`, je-li více typových proměnných `T` + popisný název. `IEnumerable<T>, Dictionary<TKey, TValue>`
výčtové typy (`enum`)	Jednotné číslo, pro kombinační výčty (flags) množné číslo: `DayOfWeek`, `AccessRights`.
pole a kolekce	Použijte množné číslo.
booleanovské typy	Pokud to má význam, použijte předpony `Has`, `Can`, `Is`: `IsReady`, `HasValue`, `CanRead`.
zkratky	Užívejte jen velmi dobře známé zkratky, nebo pokud je plný název opravdu velmi dlouhý. Dvouznakové zkratky se zapisují stejnou velikostí písmen, delší zkratky se zapisují jako by to bylo jedno slovo. `GetWindow()` x ~~`GetWin()`~~. třídy: **`Html`**`Document`, **`UI`**`Element` x proměnné: **`xml`**`Element`, **`html`**`Document`, **`ui`**`Control`.

DOKUMENTACE

DOKUMENTACE A KOMENTÁŘE

Co kód dělá by mělo být patrné z kódu samotného – měl by být samodokumentující. Komentáře v kódu by měli zdůvodňovat, proč je kód napsán právě takto. Samodokumentujícího se kódu dosáhnete volbou vhodných názvů a rozdělením metod do menších metod a tříd (názvy tříd a metod pak nahrazují komentář).

Pro potřeby dokumentace aplikačního rozhraní (API, třídy a jejich veřejné členy) se C#/.NET používá XML dokumentace – obdoba JavaDocu/Doxygen.

XML DOKUMENTACE

XML dokumentace je zvláštní forma komentáře, pomocí které píšete dokumentaci přímo k příslušnému zdrojovému kódu. Zcela překvapivě se pro popis používá syntaxe jazyka XML. XML dokumentaci dokáže využít editor vývojového prostředí Visual Studio, který ji zobrazuje v tooltipech.

Komentáře XML dokumentace jsou buď

a) řádkové: začínající značkou `///`,
b) blokové: ohraničené `/**` a `*/`.

Tagy XML dokuentace zahrnují tagy pro popis kódu (`<summary>`, `<return>`) a tagy pro strukturování textu (`<c>`,`<para>`,`<list>`):

```
/// <summary>Souhrný popis třídy. Zobrazuje se v ToolTipech Visual Studia.</summary>
/// <remarks>Dodatečné informace k popisu třídy.</remarks>
public class TestClass
{
    /// <summary>Popis metody DoWork, která patří do třídy<see cref="TestClass" />.</summary>
    /// <param name=" numberOfGummyBears">Popis parametru metody.</param>
    /// <return>Popis návratové hodnoty metody.</return>
    public static void DoWork(int numberOfGummyBears)
    { … }
}
```

Seznam doporučený tagů pro dokumentaci je k nalezení v knihovně MSDN: Recommended Tags for Documentation Comments.

VISUAL STUDIO 2010

Visual Studio 2010 vo verzii Professional si môžete stiahnuť a nainštalovať z webu DreamSpark, prípadne z MSDN Academic Alliance na FI.

Visual Studio 2010 je nainštalované v učebni **B116, B117, A104 a B311** a na vybraných počítačoch v počítačovej hale na FI.

KLÁVESOVÉ SKRATKY

Súhrn najpoužívanejších klávesových skratiek:

http://blogs.msdn.com/b/lisa/archive/2010/04/16/vs-2010-keyboard-shortcut-posters-now-available-for-vb-c-f-c.aspx

POROVNANIE S JAVOU

Pre zabehnutých Java programátorov a študentov s absolvovaným PB162 a PV168 môže byť zaujímavý zdroj:

- Moving to C# and the .NET Framework, for Java Developers
- Špeciálne potom Cheat Sheet: C# for Java Developers.
- Ďalšie vyčerpávajúce porovnanie sa nachádza v zdroji C# From a Java Developer's Perspective.
- C Sharp from Java Orange Book

POROVNÁNÍ S C++

- C# FAQ for C++ programmers
- MSDN: C# for C++ Developers