Vyšší odborná škola obalové techniky
a střední škola, Štětí

Digitální učební materiály

Programování - Programování C#

Ivan Pomykacz

esf evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Licence

Disclaimer

Tento PDF dokument byl strojově vygenerován z HTML stránek

http://dumy.odbornaskola.cz/pomykacz/.

Je tedy možné, že sazba textu může obsahovat chyby. Jde převážně o vizuální a typografické chyby, které mohou narušit plynulou čitelnost textu. V některých případech může být ohrožena i funkčnost některých komponent (jako vnitřní odkazy).

Vzhledem k tomu, že vypracované materiály nebyly nikdy určeny pro výstupní formát PDF, autor se zříkává jakékoli odpovědnosti za nalezené chyby. Nesnažte se proto v této souvislosti autora kontaktovat.

Programování

Programování C#

Obsah

Metody

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_459		
Název tématické oblasti (sady)	Programování		
Název materiálu	Metody		
Anotace	Text seznamuje s pojmem metoda. Ukazuje na příkladu použití metod, jejich argumentů a návratových hodnot.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Chápe významu použití metod. Rozumí, co je vstupní argument metody. Chápe, co je návratová hodnota metody.		
Klíčová slova	metody, vstupní a výstupní argumenty, návratová hodnota		
Druh výukového zdroje	Výklad	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	20.11.2013	**Celková velikost**	

Obsah

Definice

Metoda v objektově orientovaném programování (nebo funkce v klasickém, procedurálním programování) je tzv. znovu-použitelný blok kódu, který vykonává specifickou úlohu v programu. Proč psát oddělené bloky kódu, když to funguje i bez nich?

- Pokud se v kódu objeví chyba, je mnohem rychlejší její nalezení a opravení, pokud je program (zdrojový kód) dobře organizován. To že mají různé funkce různou úlohu pomáhá v organizaci programu.
- To že přidělíme specifické úlohy rozdílným funkcím má za následek méně nadbytečného kódu. Rovněž vzniká tzv. znovu použitelný kód. Funkce můžeme opakovaně spouštět (volat) a nebo je

dokonce využít i v jiném programu.
- V objektově orientovaném programování (a C# je objektově orientovaný jazyk) se s metodami dá provádět mnoho zajímavých věcí.
- D.R.Y. - znamená **D**ont **R**epeat **Y**ourself (Neopakuj se). Při používání funkcí (metod) se tomuto dá lehce předejít.
- Rekurze je velmi mocný nástroj. Umožňuje spustit opakovaně kód, který spouští sám sebe. Existuje několik programovacích problémů, které nelze bez rekurze vyřešit. Díky funkcím, můžeme provádět rekurzi.

Příklad

Mějme metody `WriteLine()` a `ReadLine()`. Obě jsou svázány s objektem `Console`, proto je používáme tak jak doposavad.

```
Console.WriteLine();
Console.ReadLine();
```

Používali jsme i metodu `Parse()` z objektu `int`.

Tuto metodu jsme kombinovali společně s metodou `Console.ReadLine()`, díky čemuž jsme z řetězce `string` udělali číslo `int`.

```
int.Parse(Console.ReadLine());
```

Platí, že nejprve se výraz vyhodnocuje od těch nejvnořenějších metod (zde `Console.ReadLine()`) směrem vzhůru (zde `int.Parse()`).

Tzn. jako první se provede `Console.ReadLine()`, která čeká na stisk klávesy Enter a ihned poté předá získaná data z klávesnice metodě `int.Parse()`.

Vstupní argumenty (parametry)

Metoda může a nemusí mít vstupní argumenty.

Jelikož metoda je oddělený (samostatný) blok kódu, byl definován způsob, jak těmto metodám předat nějaká vstupní data, se kterými by mohly pracovat.

Např. metoda `Console.WriteLine()`. Metoda vypíše do konzole předaný argument a vloží řádkový zlom (jako byste napsali Enter). Přičemž řádkový zlom vloží vždy, i když nepředáte metodě žádný argument.

Argumenty jsou konkrétního datového typu. Není tedy možné libovolné metodě předat proměnnou typu `int` nebo `string`, pokud to nepodporuje.

`Console.WriteLine()` je tak trochu výjimka. Je jí totiž celkem jedno, co jí předáte.

```
string jmeno = "Lorem Ipsum";
int cislo = 12345;
Console.WriteLine(jmeno);
Console.WriteLine(cislo);
```

Jak je vidět výše, předali jsme metodě nejprve `string` a potom `int` a kompilátor žádnou chybu nezahlásil.

O tom, jaké vstupní argumenty metoda podporuje, se lze dočíst v referenční příručce jazyka (knihovny).

Návratové hodnoty

Každá metoda má definovaný tzv. návratový typ. Jde vlastně o datový typ (např. `int`, `string`). Proč mají metody návratové hodnoty?

Účelem metody je něco dělat, a pokud to má smysl, tak vrátit nějaký výsledek.

Pokud se bavíme o metodě `Console.ReadLine()`, tak tato metoda čeká na vstup z klávesnice. Je-li stisknuta klávesa Enter, čtení vstupu se ukončí, a to, co bylo napsáno, se předá na výstupu metody.

```
string jmeno;
jmeno = Console.ReadLine();
```

Metoda `Console.ReadLine()` vrací vždy datový typ `string`. Proto, když chceme uložit data z klávesnice do proměnné, musí být tato proměnná datového typu `string`.

O tom, jaké datové typy metoda vrací, se lze dočíst opět v referenční příručce jazyka (knihovny).

Prázdnota

Některé metody nevrací nic, resp. vrací "nic". Např. metoda `Console.WriteLine()` nic nevrací. Přesto musí něco vracet. Programátoři to vymysleli šibalsky a přidali návrátový typ `void` (prázdnota).

Pokud tedy existuje metoda, která nemá nic vracet, pak musí vrátit typ `void`.

Proměnnou typu `void` se vám však deklarovat nepodaří.

Metody a objekty

C# je objektově orientovaný jazyk. Proto nemůže metoda existovat jen tak, bez svého objektu.

Patrně jste si všimli, že některé metody se používají trochu jinak, než bylo popsáno v předchozích kapitolách.

Např. metoda `string.ToLower()`. Tato metoda nemá žádný vstupní argument, a přesto vrací hodnotu typu `string`.

```
string jmeno = "Lorem Ipsum";
string mala = jmeno.ToLower();
```

Je tomu tak proto, že metoda `ToLower()` je svázána se svým objektem (typu `string`). Pokud zavoláme tuto metodu, pak operaci převodu na malá písmena provede nad objektem, ze kterého byla volána (zde `jmeno`). A nepotřebuje tak vstupní argument.