METODY

význam metod
statická metoda
deklarace metody
volání metody
příkaz return
přetížené metody

Metody

- metoda je pojmenovaná posloupnost příkazů
  (v některých jazycích se nazývá funkce nebo procedura)
- každá metoda má
  - název
    - volte smysluplně, např. VypocetPlochy (všimněte si, že první písmeno názvu je velké, tím se liší od pojmenovávání proměnných)
  - tělo
    - obsahuje vlastní příkazy, které jsou provedeny po zavolání metody
- většina metod přebírá a předává data
  - vstupní parametry (vstupní data, vstupní argumenty)
  - výstupní parametry (výstupní data, návratová hodnota)

Význam metod

- něco opakovaně počítáme
  - pokud si představíte výpočet složený např. z dvaceti řádků kódu, veškerý kód bychom museli při opakovaném výpočtu znovu zapisovat (popř. kopírovat)
  - pokud bychom však chtěli daný výpočet upravit, nezbývalo by nám nic jiného, než najít všechny jeho výskyty a přepsat je
- lepším řešením by bylo mít kód pro výpočet na jednom místě a tam, kde bychom ho chtěli provést, se na něj odkázat
  - takovým kódem je funkce třídy, v jazyce objektově orientovaného programování hovoříme spíše o **metodě třídy / objektu**
- provedení metody zařídíme tzv. **voláním metod**
- metoda je tedy jakási černá skříňka, u které okolní program ví, jaké má vstupy, jaké výstupy, tzn. jak ji volat, ale vůbec nepotřebuje vědět, co je vevnitř, tj. jakým způsobem se provádí daný výpočet

Statická metoda

- pro jednoduchý výpočet nepotřebujeme objekt, který by obsahoval i nějaké údaje (stav)
- pokud požadujeme metodu, kterou by bylo možné volat bez tvorby objektu, použijeme **statickou metodu**
- statická metoda je metodou třídy, ne objektu. Deklarujeme tedy
  - např. třídu s názvem **Matematika** a
  - její statickou metodu **ObsahCtverce**, této metodě předáme jeden parametr, kterým je strana čtverce, metoda provede vlastní výpočet a nakonec na místo volání vrátí vypočtenou hodnotu

Deklarace metody

příklad:

```
class Matematika
  {
    public static double ObsahCtverce(double a)
    {
      return a * a;
    }
  }
```

- je deklarována třída Matematika pomocí klíčového slova **class**
- následuje **hlavička metody**

Klíčové slovo **public** říká, že je metoda **veřejná** a že ji tedy bude možné volat i zvenčí, nejen uvnitř třídy.

Soukromé metody – **private** -není možné volat zvenku, slouží pouze pro pomocné výpočty uvnitř třídy

Deklarace metody

- klíčové slovo **static** právě označuje, že se jedná o statickou metodu
- následuje klíčové slovo **double**, které udává návratový typ - datový typ hodnoty, kterou funkce vrací
- pokud metoda nevrací žádnou hodnotu, pak je potřeba uvést klíčové slovo **void**
- následuje **název metody**, který podléhá známým konvencím, první písmeno však budeme používat velké (oproti velbloudímu stylu, jinak platí stejná pravidla jako pro názvy proměnných, nezapomeňte, že se rozlišují malá a velká písmena)
- v závorce za názvem metody je **seznam formálních parametrů**, obsahuje datové typy a názvy proměnných, více parametrů se odděluje čárkou, formální parametry jsou lokální proměnné, do kterých se ukládají hodnoty skutečných parametrů při volání metody

Deklarace metody

- následuje tělo metody, které provádí vlastní výpočet pomocí formálních parametrů (pokud metoda vůbec nějaké parametry má)
- kód se píše mezi otevírací a uzavírací složené závorky
- vrací se klíčovým slovem **return**

Příkaz **return**

- pokud metoda nevrací žádnou hodnotu (metoda je typu void), příkaz **return** není povinný (metoda skončí na poslední uzavírací složené závorce)
- pokud metoda vrací nějakou hodnotu, pak je příkaz **return** uvnitř těla metody povinný
- skládá se z
  - klíčového slova **return**
  - výrazu, který musí být stejného datového typu jako v hlavičce metody (pokud je v hlavičce metody uvedeno, že metoda vrací hodnotu typu **int**, pak i výraz v příkazu **return** musí být datového typu **int**)
  - středníku
- př. **return a * a;**

Volání metody

- např. **volání metody** ObsahCtverce
- můžeme volat z kteréhokoli místa programu, např. v metodě Main

Voláním je příkaz:

Matematika.ObsahCtverce(a)

Matematika.ObsahCtverce(strana)

Matematika.ObsahCtverce(5)

- danou metodu voláme přímo u třídy, neboť se jedná o statickou metodu
- můžeme ji zavolat, neboť je veřejná
- **skutečný parametr** může mít zcela jiné jméno než formální parametr v definici metody (!!!)

Volání metody

- parametry předávané metodě musí odpovídat parametrům v deklaraci
  1. svým počtem
  2. pořadím
  3. datovým typem (jinak nastane chyba při překladu)
- pokud metoda vrací nějakou hodnotu, měli bychom ji nějakým způsobem zachytit
- př. **obsah=Matematika.ObsahCtverce(a);**
- metodě můžeme předat jako parametr také pole, což naznačíme dohodnutou konvencí **[]**
- pole je referenční typ, tudíž se nekopíruje hodnota ale pouze odkaz (reference), pracuje se s původní pamětí

Předávání parametrů

- předávání parametrů hodnotou
  - předává proceduře kopii původní proměnné, tzn. že pokud uvnitř procedury provedete nějaké změny parametru, neovlivní to původní proměnnou
- předávání parametrů odkazem
  - předá funkci adresu proměnné v paměti

Parametrem metody je pole

```
class PraceSPolem
    {
      public static void ZadaniPole(double[] pole)
      {
        Console.WriteLine("Zadání pole:");

        for (int i = 0; i < pole.Length; i++)
        {
          Console.Write("Zadejte {0}. prvek: ", i+1);
          pole[i] = double.Parse(Console.ReadLine());
        }
      }
    }
```

Předání pole

```
static void Main()
        {
          Console.Write("Zadejte počet prvků: ");
          int n = int.Parse(Console.ReadLine());

          double[] pole = new double[n];

          PraceSPolem.ZadaniPole(pole);


          Console.ReadLine();
        }
```

Lokální proměnné

- uvnitř metody je možné deklarovat a používat proměnné
- takové proměnné **zaniknou** po skončení provádění metody
- obor platnosti je dán umístěním deklarace
- obor platnosti **určují otevírací a uzavírací složené závorky**
- lokální proměnné nelze používat pro sdílení údajů mezi metodami nebo metodou a hlavním programem

Přetížené metody (overloaded)

- užitečná věc
- příklad: metoda **WriteLine** objektu **Console**
  - přebírá různé typy parametrů (žádný parametr, jeden parametr např. číslo, logickou hodnotu, atd.)
  - při kompilaci překladač sleduje počet a typy parametrů metody a zavolá její správnou variantu
- přetížení je šikovné, když potřebujete provádět stejnou operaci
  - na různých datových typech
  - nebo s různým počtem parametrů
- volání přetížené metody tedy provede tu verzi, která souhlasí signaturou se seznamem skutečných parametrů (co do jejich počtu i datových typů)

Přetížené metody

příklad

```
class Matematika
    {
        public static int Objem (int x)
          {
          return x*x*x ;
          }
        public static int Objem (int x, int y, int z)
          {
          return x*y*z ;
          }
    }
```

Rekurzivní volání metody

- metoda volá samu sebe
- zavolaná rekurzivní metoda zpravidla rozdělí problém na více stejných problémů nižšího řádu
- **rekurze** tedy vytváří řetěz volání téže metody

nevýhoda:

- možnost přetečení zásobníku (výjimka StackOverflowException)

Příklad 1: rekurzivní volání

```
public class Matematika
  {
    public static double Fak (int n)
    {
      if (n <= 1)
        return 1;
      else
        return n * Fak (n - 1);
    }
  }
```

příští týden: složitost algoritmů