Podprogramy

- zásady: „jednu věc programovat pouze jednou“
- podprogram → logický celek, řešící dílčí část problému
- Příklad velmi špatného zápisu programu na výpočet obsahu obdélníku

```
using System;
class Program
{
	static void Main(string[] args)
	{
		// nacteni strany 1
		double str1;
		Console.Write("Zadej 1.stranu: ");
		str1 = Double.Parse(Console.ReadLine());
		// nacteni strany 2
		double str2;
		Console.Write("Zadej 2.stranu: ");
		str2 = Double.Parse(Console.ReadLine());
		// vypocet a tisk
		Console.WriteLine("Obsah = {0}", str1 * str2);
	}
}
```

prakticky stejné

- řešení s podprogramem

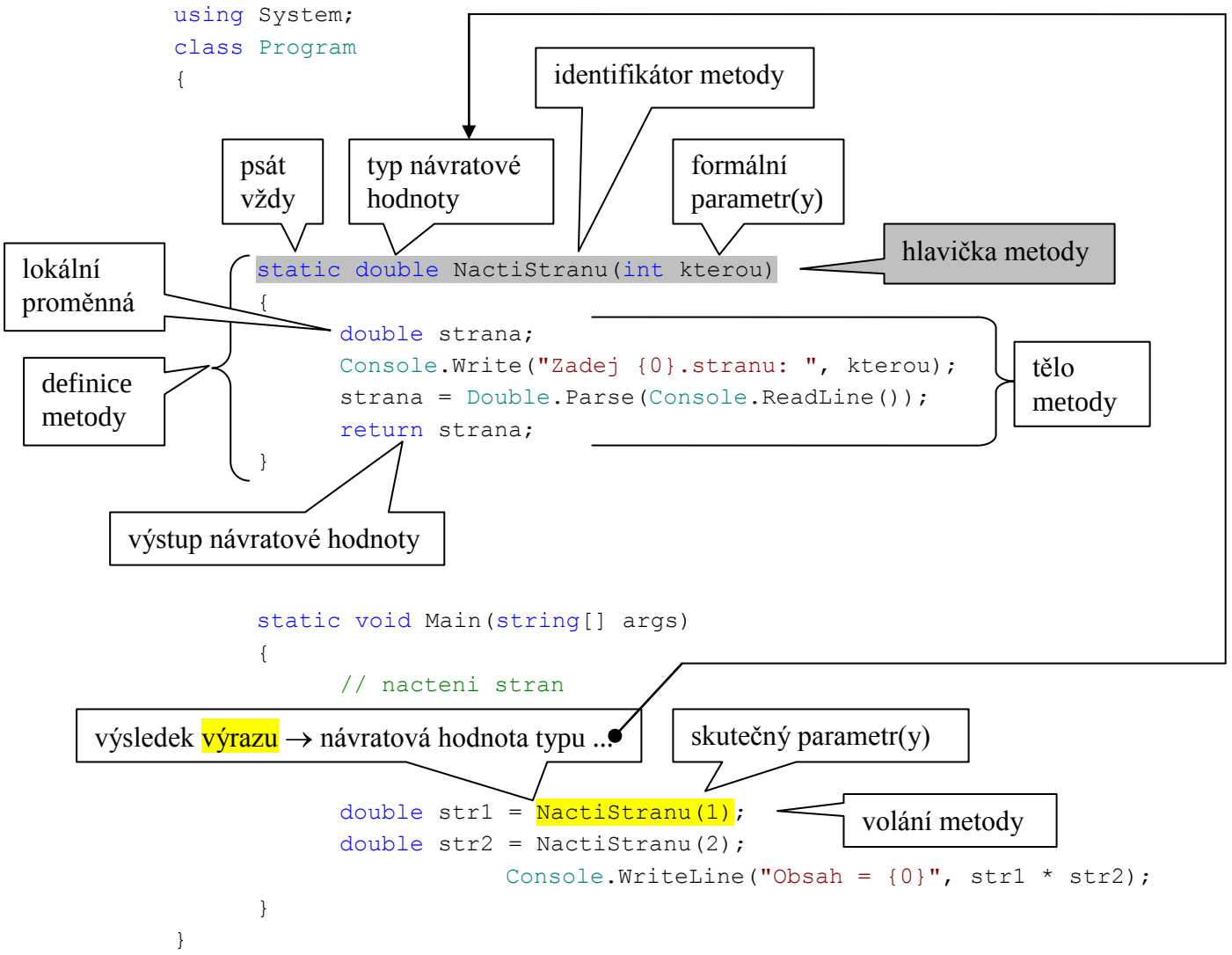
```
using System;
class Program
{
	static double NactiStranu(int kterou)
	{
		double strana;
		Console.Write("Zadej {0}.stranu: ", kterou);
		strana = Double.Parse(Console.ReadLine());
		return strana;
	}

	static void Main(string[] args)
	{
		// nacteni stran
		double str1 = NactiStranu(1);
		double str2 = NactiStranu(2);
		Console.WriteLine("Obsah = {0}", str1 * str2);
	}
}
```

podprogram

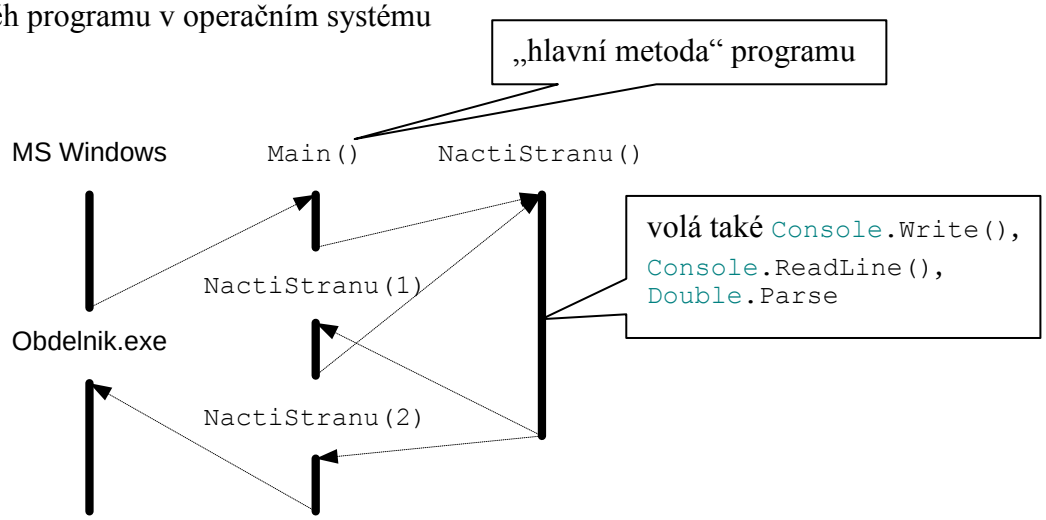
- podprogramy
  - non OOP jazyky (C, Pascal, ...) – procedury, funkce
  - OOP jazyky (C#, Java, ...) – metody

- základní terminologie



Volání metod

- Program v C# → 1 nebo více metod
- běh programu v operačním systému



  - „hloubka volání“ není omezena
- volání a parametry: přiřazení hodnot skutečných parametrů do proměnných formálních parametrů
  - princip skutečného běhu `Main()` min. příkladu

```
static void Main(string[] args)
{
    // double str1 = NactiStranu(1);
    double str1;
    {
        int kterou = 1;
        double strana;
        Console.Write("Zadej {0}.stranu: ", kterou);
        strana = Double.Parse(Console.ReadLine());
        str1 = strana;
    }
    // double str2 = NactiStranu(2);
    double str2;
    {
        int kterou = 2;
        double strana;
        Console.Write("Zadej {0}.stranu: ", kterou);
        strana = Double.Parse(Console.ReadLine());
        str2 = strana;
    }
    Console.WriteLine("Obsah = {0}", str1 * str2);
}
```

„return“

Definice vlastních metod

- analýza:
  1) Co bude předmětem činnosti metody → jméno metody, algoritmus metody
  2) Jaké vstupní informace bude potřebovat → počet a datový typ formálních parametrů
  3) Co bude výsledkem činnosti → typ návratové hodnoty
- počet a typ parametrů neomezen
- Návratová hodnota metody → `return`
  - syntaxe:

    ```
    return vyraz;     // např. return 2*3.14*R
    ```

  - sémantika:
    metoda vrací hodnotu `vyraz`, po tomto příkazu se metoda ihned ukončí

    ```
    static int Metoda()
    {
          return 10;
          Console.WriteLine("Nikdy se neprovede");
    }
    ```

  - `return` v `Main()` → ukončí program
  - datový typ neomezen
- metody dle počtu parametrů
  - s 1 nebo více parametry
  - bez parametrů
  - s proměnným počtem parametrů
- metody dle návratové hodnoty
  - vracející hodnotu
  - nevracející hodnotu

Metody s parametry vracející hodnotu

- nejčastěji
- Napište metodu, která vypočte objem válce

```
using System;
class Program
{
	static void Main(string[] args)
	{
		double polomer = 10.3, vyska = 5.0;
		double objem = ObjemValce(polomer, vyska);
		double dalsiObjem = objem + ObjemValce(2 * polomer, 2.1);
	}

	static double ObjemValce(double R, double v)
	{
		return Math.PI * R * R * v;
	}
}
```

- Další příklady

```
double NactiStranu(int kterou)
bool JePrechodnyRok(int rok)
double Prepona(double odvesna1, double odvesna2)
```

- Příklady z .NET

```
double Math.Sin(double uhel)
```

Metody s parametry nevracející hodnotu

- velmi často
- Příklad: napište metodu, která vytiskne pro osobu daného jména počet vypitých piv. V případě, že bude počet piv menší než nula, vytiskne se chybová zpráva.

```
using System;
class Program
{
	static void Main(string[] args)
	{
		int frantovyPiva = 5;
		TiskPoctuPiv("Franta", frantovyPiva);
		TiskPoctuPiv("Pepa", frantovyPiva - 2);
		TiskPoctuPiv("Martin", -88);
	}

	static void TiskPoctuPiv(string jmeno, int pocetPiv)
	{
		if (pocetPiv < 0)
		{
			Console.WriteLine("{0} nemohl vypit {1} piv!!!",
				jmeno, pocetPiv);
		}
		else
		{
			Console.Write("{0} vypil ", jmeno);
			switch (pocetPiv)
			{
				case 1:
					Console.WriteLine("1 pivo");
					break;
				case 2:
				case 3:
				case 4:
					Console.WriteLine("{0} piva", pocetPiv);
					break;
				default:    // 5 a více
					Console.WriteLine("{0} piv", pocetPiv);
					break;
			}
		}
	}
}
```

nelze ve výrazech, pouze samostatně jako příkaz

nic nevrací

žádné `return`, nebo `return;`

Metody bez parametrů vracející hodnotu

- méně často
- Napište metodu, simulující hod kostkou

```
using System;
class Program
{
	static void Main(string[] args)
	{
		for (int i = 0; i < 10; i++)
		{
			Console.WriteLine(HodKostkou());
		}
	}

	static int HodKostkou()
	{
		Random generator = new Random(DateTime.Now.Millisecond);
		System.Threading.Thread.Sleep(10);
		int nahodneCislo = generator.Next(1, 7);
		return nahodneCislo;
	}

}
```

!!!

žádné `void`

tisk návratové hodnoty

- Příklad z. NET

vrací `string`

```
int prom = Int32.Parse(Console.ReadLine());
```

Metody bez parametrů nevracející hodnotu

- zřídka
- Napište metodu, která vytiskne chybovou zprávu

```
using System;
class Program
{
	static void Main(string[] args)
	{
		TiskChyboveZpravy();
	}

	static void TiskChyboveZpravy()
	{
		Console.WriteLine("************************************");
		Console.WriteLine("Doslo k chybe");
		Console.WriteLine("************************************");
	}
}
```

Závěrečné poznámky

- na pořadí definic metod ve ZK nezáleží

```
class Program
{
      static void Main()
      {
            // kod Main
      }
      static void M1()
      {
            // kod M1
      }
      static void M2()
      {
            // kod M2
      }
}
```

```
class Program
{
      static void M2()
      {
            // kod M2
      }
      static void Main()
      {
            // kod Main
      }
      static void M1()
      {
            // kod M1
      }
}
```

- definice metody musí ležet uvnitř třídy (zatím `class Program`)
- definice metody nesmí ležet uvnitř definice metody jiné

```
using System;
class Program
{
      static void Main(string[] args)
      {
            static void Metoda()
            {
                  // kod Metoda
            }
            // kod Main()
      }
}
```

- volání – počet a datový typ parametrů musí souhlasit s hlavičkou (max. implicitní konverze)

```
static void Main(string[] args)
{
      int vyska= 5;
      double polomer = 10.0;
      double objem = ObjemValce(polomer, vyska, 10.3);
      double objem1 = ObjemValce(3.5);
}
static double ObjemValce(double R, double v)
{
      return Math.PI * R * R * v;
}
```

impl. konverze `int` → `double`

příliš mnoho parametrů

málo parametrů

- identifikátory metod – první písmeno velké

Více return

- Počet není omezen
- Napište metodu, která vrátí větší číslo ze dvou argumentů typu `int`, a pokud jsou shodné, vrátí nulu

klasické řešení

```
static int Porov(int x1, int x2)
{
	int temp;
	if (x1 > x2)
	{
		temp = x1;
	}
	else if (x2 > x1)
	{
		temp = x2;
	}
	else
	{
		temp = 0;
	}
	return temp;
}
```

více `return`

```
static int Porov(int x1, int x2)
{
	if (x1 > x2)
	{
		return x1;
	}
	if (x2 > x1)
	{
		return x2;
	}
	return 0;
}
```