5. Metody

- jmenná konvence
  - PascalCase - začíná velkým písmenem

```
static void Soucet(int cisloJedna, int cisloDva) {
    ...
}
```

modifikátor přístupu - static, public, private, internal

návratová hodnota - void, int, bool, double, string, int[], ...
(vracíš nic)

název - PascalCase

parametry - pracujeme s nimi ve funkci (např. součet a další)
- fce může být i bez parametrů

Použití

- nevrací nic, pouze vypisuje

```
static void Main(string[] args)
{
    ReadPerson("Josef", "Smetana" , 50);
    Console.ReadLine();
}

Počet odkazů: 1
static void ReadPerson(string firstname, string surename, int age)
{
    Console.WriteLine(firstname + " " + surename + " " + age);
}
```

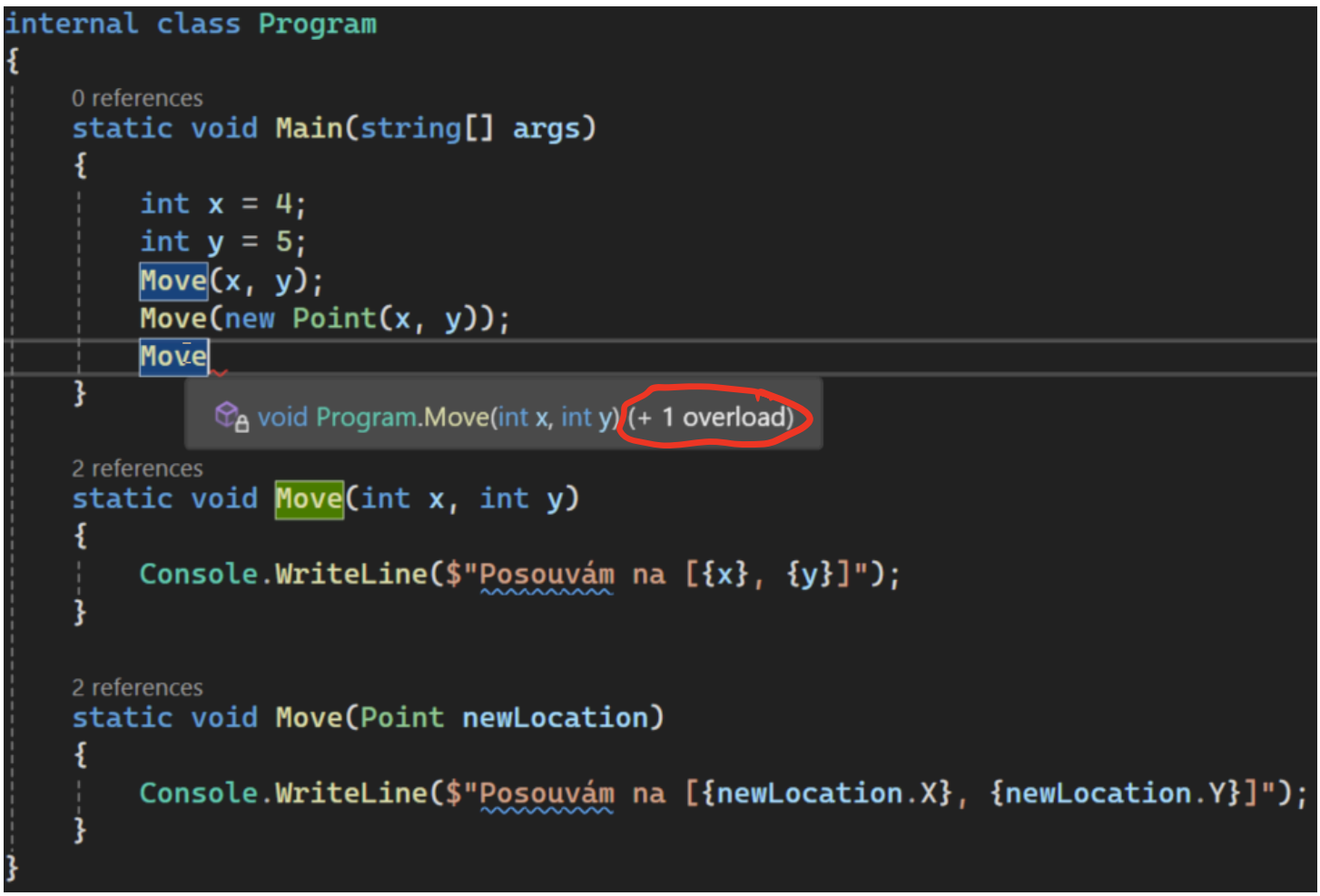
- vrací double

```
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        double vysledek = Soucet(5, 2);
        Console.WriteLine($"Výsledek je {vysledek}");
    }

    1 reference
    static double Soucet(int i, int j)
    {
        return i + j;
    }
}
```

Přetížení funkcí (overloads)

= funkce se stejným jménem, ale jinými parametry



- fce Move lze zavolat
  - s číselným parametrem x a y
  - s objektem typu Point

- přetěžování je ale potřeba využívat efektivně!

```
public int Add(int n1, int n2){}
public int Add(int n1, int n2, int n3){}
public int Add(int n1, int n2, int n3, int n4){}
…
```

- v tomto případě bychom se upsali k smrti

```
public int Add(int[] numbers){}
```

- využijeme pole = lepší přístup

- modifikátor params

```
public class Calculator
{
    public int Add(params int[] numbers){}
}


var result = calculator.Add(new int[]{ 1, 2, 3, 4 });
var result = calculator.Add(1, 2, 3, 4);
```

nemusíme inicializovat

- pouze kratší zápis