

## Metode

Metoda je organizacijska jedinica koja je zadužena za implementaciju željenog algoritma

Vrste metoda

1. tipa void, ne prima vrijednost
2. tipa void, prima vrijednost
3. određenog tipa podatka koji vraća pozivatelju, ne prima vrijednost
4. određenog tipa podatka koji vraća pozivatelju, prima vrijednost
5. konstruktor – kasnije u OOP

Potpis metode

**način pristupa** **tip podatka** **naziv** (**lista parametara**)

Izvor:

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/methods>

## Metode

1. tipa void ne prima vrijednost

Deklaracija metode

```
void Tip1()  
{  
    Console.WriteLine("Metoda tipa void ne vraća vrijednost");  
}
```

Poziv metode

```
Tip1();
```

Rezultat izvođenja

>- Terminal – Metode

```
Metoda tipa void ne vraća vrijednost
```

## Metode

### 1. tipa void, prima vrijednost

#### Deklaracija metode

```
void Tip2(string tekst)
{
    Console.WriteLine(tekst);
}
```

#### Poziv metode

```
Tip2("Ovo je vrijednost parametra koji šaljem");
string s = "Ovo je druga vrijednost parametra koji šaljem";
Tip2(s);
```

#### Rezultat izvođenja

Terminal – Metode

```
Ovo je vrijednost parametra koji šaljem
Ovo je druga vrijednost parametra koji šaljem
```

## Metode

3. određenog tipa podatka koji vraća pozivatelju, ne prima vrijednost

Deklaracija metode

```
int Tip3()  
{  
    return new Random().Next();  
}
```

Poziv metode

```
Console.WriteLine(Tip3());  
  
int slucajniBroj = Tip3();  
  
Console.WriteLine(slucajniBroj);
```

Rezultat izvođenja

Terminal – Metode

```
303036459  
1264990036  
█
```

## Metode

4. određenog tipa podatka koji vraća pozivatelju, prima vrijednost

Deklaracija metode

```
int Tip4(int odBroja, int doBroja)
{
    return new Random().Next(odBroja, doBroja);
}
```

Poziv metode

```
Console.WriteLine(Tip4(12,19));
int slucajniBroj = Tip4(-98,-23);
Console.WriteLine(slucajniBroj);
```

Rezultat izvođenja

> Terminal – Metode

```
12
-25
```

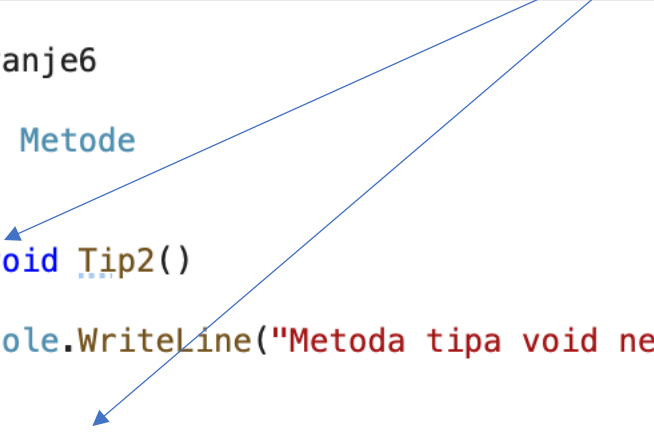
## Metode

Metodama je mjesto u klasi

Deklaracija metoda unutar klase

Potpis metode

```
1  using System;
2  namespace Predavanje6
3  {
4      public class Metode
5      {
6
7          public void Tip2()
8          {
9              Console.WriteLine("Metoda tipa void ne vraća vrijednost");
10         }
11
12         public void Tip2(int i)
13         {
14             Console.WriteLine("Primio {0}, ne vraća vrijednost",i);
15         }
16     }
17 }
18
19
```



Deklaracija instance klase i poziv metode

```
var m = new Metode();
m.Tip2();
m.Tip2(7);
```

Rezultat izvođenja

**Terminal – Predavanje6**

```
Metoda tipa void ne vraća vrijednost
Primio 7, ne vraća vrijednost
```

## Načini pristupa

Vrste

**[bez navođenja]:** Dostupno unutar klase ili strukture u kojoj je deklarirana

**private:** Dostupno unutar klase ili strukture u kojoj je deklarirana

**protected:** Dostupno u trenutnoj klasi i u svim klasama koje nasljeđuju trenutnu klasu

**internal:** Dostupno unutar trenutnog projekta – dll

**public:** Dostupno svima (trenutni projekt – dll ili u drugom projektu – dll-u)

protected internal

private protected

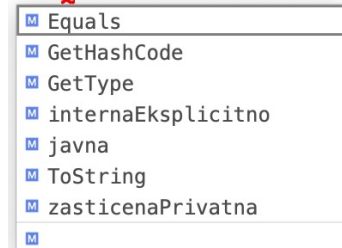
O načinima pristupa još u detalje u dijelu nasljeđivanja

Izvor:

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/access-modifiers>

```
1  using System;
2  namespace Predavanje6
3  {
4      public class NaciniPristupa
5      {
6          private void privatna()
7          {
8          }
9      }
10     internal void internaEksplicitno()
11     {
12     }
13     void internaImplicitno()
14     {
15     }
16     protected void zasticena()
17     {
18     }
19     public void javna()
20     {
21     }
22     protected internal void zasticenaPrivatna()
23     {
24     }
25     private protected void privatnaZasticena()
26     {
27     }
28     }
29 }
30
31
32
33
34
35
36 }
```

```
var n = new NaciniPristupa();
n.
```



## Ključna riječ static

static označava mogućnost korištenja metode na klasi bez instanciranja klase  
koristimo ju kada ne želimo pratiti stanje već izvesti određeni algoritam

```
1 using System;
2 using System.Text.RegularExpressions;
3
4 namespace Predavanje6
5 {
6     public class KljucnaRijecStatic
7     {
8         public static bool ValjaniOIB(string oib)
9         {
10             if (string.IsNullOrEmpty(oib) || !Regex.IsMatch(oib, "^([0-9]{11})$"))
11                 return false;
12
13             var oibSpan = oib.AsSpan();
14             var a = 10;
15             for (var i = 0; i < 10; i++)
16             {
17                 a += int.Parse(oibSpan.Slice(i, 1));
18                 a %= 10;
19
20                 if (a == 0)
21                     a = 10;
22
23                 a *= 2;
24                 a %= 11;
25             }
26
27             var kontrolni = 11 - a;
28
29             if (kontrolni == 10)
30                 kontrolni = 0;
31
32             return kontrolni == int.Parse(oibSpan.Slice(10, 1));
33         }
34     }
35 }
```

```
Console.WriteLine(KljucnaRijecStatic.ValjaniOIB("65220720635"));
Console.WriteLine(KljucnaRijecStatic.ValjaniOIB("65220720634"));
```

Terminal – Predavanje6

```
True
False
```

```
public bool ValjaniOIB(string oib)
```

```
Console.WriteLine(KljucnaRijecStatic.ValjaniOIB("65220720635"));
Console.WriteLine(KljucnaRijecStatic.ValjaniOIB("65220720634"));
```

class Predavanje6.KljucnaRijecStatic  
CS0120: An object reference is required for the non-static field, method, or property 'KljucnaRijecStatic.ValjaniOIB(string)'

```
var kr = new KljucnaRijecStatic();
Console.WriteLine(kr.ValjaniOIB("65220720635"));
Console.WriteLine(kr.ValjaniOIB("65220720634"));
```

```
public static bool ValjaniOIB(string oib)
```

```
var kr = new KljucnaRijecStatic();
Console.WriteLine(kr.ValjaniOIB("65220720635"));
Console.WriteLine(kr.ValjaniOIB("65220720634"));
```

bool KljucnaRijecStatic.ValjaniOIB(string oib)  
CS0176: Member 'KljucnaRijecStatic.ValjaniOIB(string)' cannot be accessed with an instance reference; qualify it with a type name instead

Izvor:

<https://github.com/domagojpa/oib-validation>



## Metode

rekurzija – metoda poziva samu sebe

```
1  using System;
2  namespace Predavanje6
3  {
4      public class Rekurzija
5      {
6          public static int Faktorijel(int i)
7          {
8              if (i == 0)
9              {
10                 return 1;
11             }
12             return i * Faktorijel(i - 1);
13         }
14     }
15 }
```

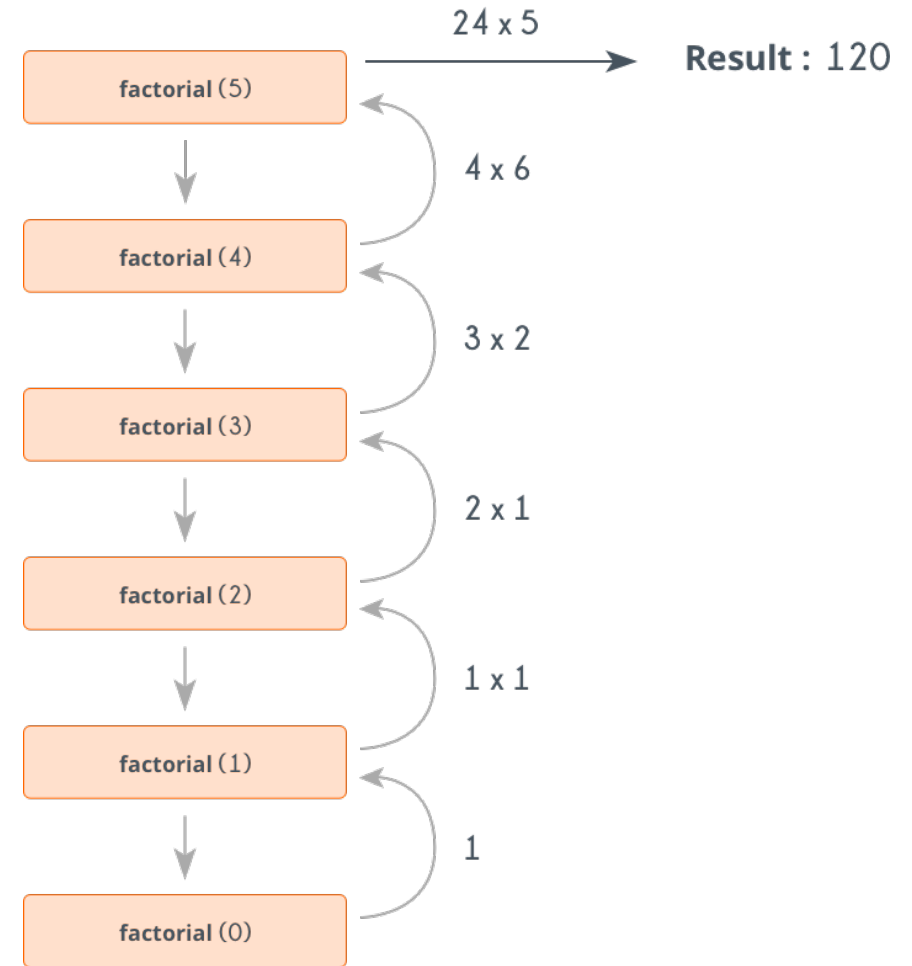
```
Console.WriteLine(Rekurzija.Faktorijel(5));
```

> Terminal – Predavanje6

120

Izvor:

<https://www.hackerearth.com/practice/basic-programming/recursion/recursion-and-backtracking/tutorial/>



## Obrada pogrešaka (iznimki)

Osmišljavanje kôda na način da ima sposobnost „bacanja” iznimke s jedne strane i „hvatanja” s druge

### Bacanje iznimke

```
public class Obrada
{
    /// <summary>
    /// Metoda vraća true za Muško
    /// ili false za Žensko
    /// </summary>
    public static bool Musko(Osoba o)
    {
        if (o == null)
        {
            throw new ArgumentNullException("Primljeni parametar je null");
        }
        return o.Equals("Muško");
    }
}
```

### Hvatanje iznimke

```
Student student=null;

try
{
    Obrada.Musko(student);
} catch (ArgumentNullException e)
{
    Console.WriteLine(e.Message);
}
```

Terminal – Predavanje6

Value cannot be null. (Parameter 'Primljeni parametar je null')

Izvor:

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/try-catch>

## Obrada pogrešaka (iznimki)

```
try
{
    Console.WriteLine("try: Uvijek se izvodi, hvata izminke");
    string[] niz = { "12", "a4" };
    foreach(string ulaz in niz)
    {
        Int16.Parse(ulaz);
    }
    Int16.Parse(niz[2]);
}
catch (IndexOutOfRangeException e)
{
    Console.WriteLine("catch se izvodi ako kod baci iznimku");
}
catch (FormatException e)
{
    Console.WriteLine("catch se izvodi ako kod baci iznimku");
}
catch (Exception e)
{
    Console.WriteLine("catch se izvodi ako kod baci iznimku");
}
finally
{
    Console.WriteLine("finally: Uvijek se izvodi");
}
```