

Prava pristupa članovima
klase

Definisanje prava pristupa članovima klase

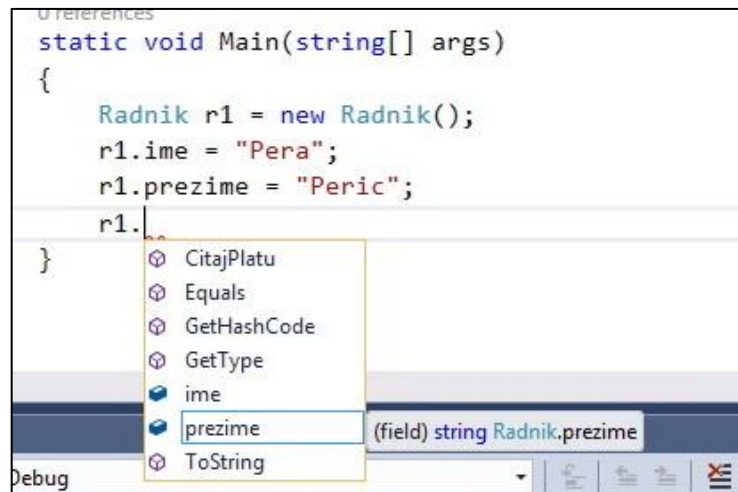
Deklaracija	Definicija
public	pristup nije ograničen
private	pristup je ograničen na klasu koja sadrži tog člana
internal	pristup je ograničen na aplikaciju
protected	Pristup je ograničen na članove klase i klase izvedenih iz te klase

Ilustracija private prava pristupa

```
class Radnik
{
    public string ime;
    public string prezime;
    private decimal plata;
}
```

```
static void Main(string[] args)
{
    Radnik r1 = new Radnik();
    r1.ime = "Pera";
    r1.prezime = "Peric";
    r1.plata = 57898.56m;
}
```

'Radnik.plata' is inaccessible due to its protection level



Čitanje i setovanje privatnog člana unutar klase

```
internal class Radnik
{
    public string ime;
    public string prezime;
    private decimal plata;

    public decimal CitajPlatu()
    {
        return plata;
    }

    public void SetujPlatu(decimal plata)
    {
        this.plata = plata;
    }

    public void Stampaj()
    {
        Console.WriteLine($"Radnik {ime} {prezime} ima platu od {plata} dinara");
    }
}
```

Indirektno setovanje privatnog člana klase

```
static void Main(string[] args)
{
    Radnik r1 = new Radnik();
    r1.ime = "Pera";
    r1.prezime = "Peric";
    r1.SetujPlatu(345567.678m);
    r1.Stampaj();

    Console.ReadLine();
}
```

Svojstva (Properties)

```
internal class Radnik
{
    public string ime;
    public string prezime;

    private decimal plata;
    public decimal Plata
    {
        get { return plata; }
        set { plata = value; }
    }

    public void Stampaj()
    {
        Console.WriteLine($"Radnik {ime} {prezime} ima platu od {plata} dinara");
    }
}
```

propfull Code Snnipet

Upotreba svojstva

```
static void Main(string[] args)
{
    Radnik r1 = new Radnik();
    r1.ime = "Pera";
    r1.prezime = "Peric";
    r1.Plata = 345567.678m; // set property

    decimal iznos = r1.Plata; // get property
    Console.WriteLine(iznos);
    Console.ReadLine();
}
```

Automatska svojstva

```
internal class Radnik
{
    public string ime;
    public string prezime;

    // automatsko svojstvo
    public decimal Plata { get; set; }

    public void Stampaj()
    {
        Console.WriteLine($"Radnik {ime} {prezime} ima platu od {Plata} dinara");
    }
}
```


Definisanje klase korišćenjem automatskih svojstava

```
internal class Radnik
{
    public string Ime { get; set; }
    public string Prezime { get; set; }

    // automatsko svojstvo
    public decimal Plata { get; set; }

    public void Stampaj()
    {
        Console.WriteLine($"Radnik {Ime} {Prezime} ima platu od {Plata} dinara");
    }
}
```

Inicijalizator objekta

```
static void Main(string[] args)
{
    Radnik r1 = new Radnik { Ime = "Pera", Prezime = "Peric", Plata = 345678.56m };
    r1.Stampaj();
    Console.ReadLine();
}
```

Inicijalizator objekta

- Kada kompajler naiđe na kod sa inicijalizatorom objekta, on ga interno prevodi u sekvencu koraka:
- **Poziva se konstruktor**, ako se ne navede eksplicitno koji konstruktor da se pozove poziva se podrazumevani (default) konstruktor bez argumenata
- Nakon što je objekat kreiran (alocirana memorija i pozvan konstruktor), **kompajler postavlja vrednosti za svaki član objekta pojedinačno**

Postavljanje privatnog set aksesora

propg Code Snnipet

```
internal class Osoba
{
    public Guid Id { get; private set; }
    public string Ime { get; set; }
    public string Prezime { get; set; }

    public Osoba()
    {
        this.Id = Guid.NewGuid();
    }
}
```

Guid (Globaly Unique Identifier) je standardni tip za kreiranje jedinstvenih 128-bitnih vrednosti.

c450228a-3e1c-43b0-8e72-ad4ab089714d

Pristup svojstvu koje može samo da se čita

```
static void Main(string[] args)
{
    Osoba os1 = new Osoba
    {
        Ime = "Pera",
        Prezime = "Peric"
    };
    //os1.Id = 1;
    Console.WriteLine(os1.Id);
    Console.ReadLine();
}
```

Anonimni tipovi podataka

- Anonimni tipovi su klase koje kompajler automatski generiše
- Anonimni tipovi se najčešće koriste za privremeno grupisanje skupa svojstava za koje se ne mora kreirati nova formalna klasa
- Anonimni tip je tip koji nema svoje ime ali ima definisana svojstva
- Ključna reč **var** koristi se kada kompajler mora da zaključi stvarni tip promenljive u vreme kompajliranja

Kreiranje anonimnog objekta

```
static void Main(string[] args)
{
    var v = new { Ime = "Pera", Prezime = "Peric" };
    Console.WriteLine(v.Ime + " " + v.Prezime);
    Console.ReadLine();
}
```

Ključna reč var ukazuje kompajleru da odredi tip na osnovu desne strane izraza za inicijalizaciju. To može biti ugrađeni tip, anonimni tip ili korisnički definisani tip podataka.

Vrednosni i referentni tipovi podataka

Vrednosni tipovi podataka

- Pri kopiranju vrednosnog tipa u memoriji se kreira nova promenljiva
- Promena vrednosti kod originala se ne odražava na kopiju i obratno
- Vrednosni tipovi se čuvaju na steku

```
static void Main(string[] args)
{
    int x = 10;
    int y = x;
    x++;
    Console.WriteLine("x={0}", x);
    Console.WriteLine("y={0}", y);
}
```

Referentni tipovi

- Kreiraju se u memoriji koja se naziva hip
- Kada promenljivoj dodelimo referencu, ona jednostavno počinje da pokazuje na objekat u memoriji
- Ako dvema promenljivama pridružimo istu referencu, obe pokazuju na isti objekat
- Ako promenimo podatak u objektu, promene će se odnositi na sve promenljive koje referenciraju objekat

Demonstracija referentnih tipova

```
class Osoba
{
    public string Ime { get; set; }
    public string Prezime { get; set; }

    public void Stampaj()
    {
        Console.WriteLine(Ime + " " + Prezime);
    }
}
```

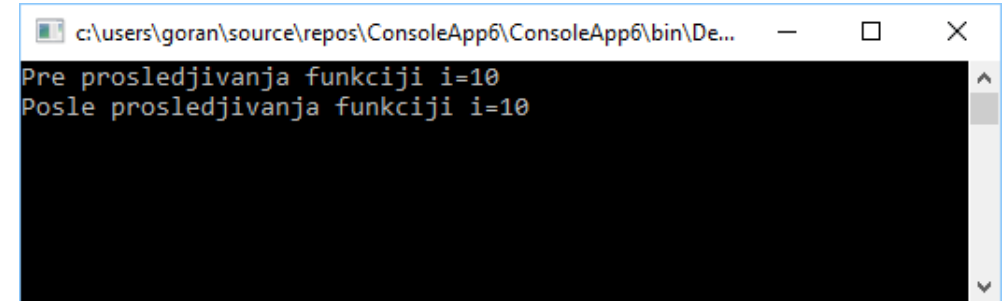
```
static void Main(string[] args)
{
    Osoba os1 = new Osoba {Ime = "Pera", Prezime="Peric" };
    os1.Stampaj();

    Osoba os2 = os1;
    os2.Ime = "Mika";

    os1.Stampaj();
    Console.ReadLine();
}
```

Prenos parametara po vrednosti

```
public static void Promeni(int a)
{
    a++;
}
```

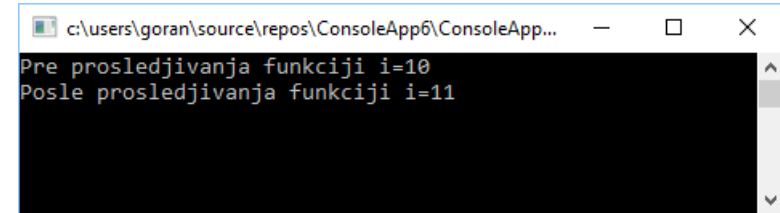
A screenshot of a Windows console window. The title bar shows the file path 'c:\users\goran\source\repos\ConsoleApp6\ConsoleApp6\bin\De...'. The console output consists of two lines: 'Pre prosledjivanja funkciji i=10' and 'Posle prosledjivanja funkciji i=10'.

```
c:\users\goran\source\repos\ConsoleApp6\ConsoleApp6\bin\De...
Pre prosledjivanja funkciji i=10
Posle prosledjivanja funkciji i=10
```

```
static void Main(string[] args)
{
    int i = 10;
    Console.WriteLine($"Pre prosledjivanja funkciji i={i}");
    Promeni(i);
    Console.WriteLine($"Posle prosledjivanja funkciji i={i}");
    Console.ReadLine();
}
```

Prenos vrednosnih tipova po referenci

```
public static void Promeni(ref int a)
{
    a++;
}
```

A screenshot of a Windows console application window. The title bar shows the file path 'c:\users\goran\source\repos\ConsoleApp6\ConsoleApp...'. The console output displays two lines of text: 'Pre prosledjivanja funkciji i=10' and 'Posle prosledjivanja funkciji i=11', demonstrating the effect of passing a variable by reference to a function that increments its value.

```
c:\users\goran\source\repos\ConsoleApp6\ConsoleApp...
Pre prosledjivanja funkciji i=10
Posle prosledjivanja funkciji i=11
```

```
static void Main(string[] args)
{
    int i = 10;
    Console.WriteLine($"Pre prosledjivanja funkciji i={i}");
    Promeni(ref i);
    Console.WriteLine($"Posle prosledjivanja funkciji i={i}");
    Console.ReadLine();
}
```

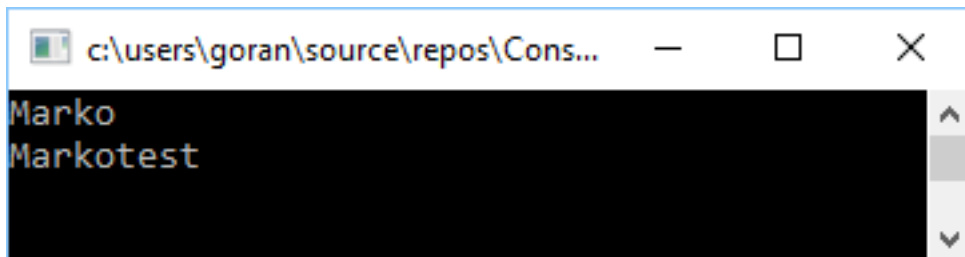
Referenca kao ulazni parameter metode

```
class Osoba
{
    public string Ime { get; set; }
    public string Prezime { get; set; }
}
```

Referenca kao ulazni parameter metode

```
static void PromeniIme(Osoba os)
{
    os.Ime += "test";
}
```

```
static void Main(string[] args)
{
    Osoba os1 = new Osoba { Ime = "Marko", Prezime = "Markovic" };
    Console.WriteLine(os1.Ime);
    PromeniIme(os1);
    Console.WriteLine(os1.Ime);
    Console.ReadLine();
}
```

A screenshot of a Windows console window. The title bar shows the file path 'c:\users\goran\source\repos\Cons...'. The console output displays two lines: 'Marko' on the first line and 'Markotest' on the second line, demonstrating that the name was successfully modified by the method call.

```
c:\users\goran\source\repos\Cons...
Marko
Markotest
```

Izlazni parametri

- Koristi se ključna reč out da se naglasi da je parametar metode izlazni
- Sličan je ref parametru ali se koristi više za vraćanje vrednosti iz metode a ne za prosleđivanje parametara metodi
- Korišćenjem out parametara metoda može vratiti više od jedne vrednosti

Metoda sa izlaznim parametrima

```
static void Dodeli(out int a, out int b)
{
    Console.WriteLine("Unesite prvi broj");
    a = int.Parse(Console.ReadLine());

    Console.WriteLine("Unesite drugi broj");
    b = int.Parse(Console.ReadLine());
}
```

Poziv metode sa izlaznim parametrima

```
static void Main(string[] args)
{
    Dodeli(out int x, out int y);

    Console.WriteLine($"x = {x}");
    Console.WriteLine($"y = {y}");

    Console.ReadLine();
}
```

Nullable tip

- Null vrednost se koristi za inicijalizaciju referentnog tipa podataka i ne može se dodeliti vrednosnom tipu
- Kada promenljivoj dodelimo null referencu znači da ona nije inicijalizovana (tj. ne pokazuje ni na šta)
- Nije dozvoljeno `int x = null;`
- C# definiše modifikator koji se koristi da se promenljiva definiše kao nullable vrednost
- Koristi se modifikator `?` koji naznačava da vrednosni tip može biti nullable
- Nullable tip se ponaša kao originalni vrednosni tip ali mu se može pridružiti null vrednost
- Dozvoljeno `int? x = null;`
- Nullable tipovi imaju `HasValue` i `Value` svojstva

Upotreba nullable tipa

```
static void Main(string[] args)
{
    int? x = null;
    Console.WriteLine("Unesite broj x");

    if (int.TryParse(Console.ReadLine(), out int x1))
    {
        x = x1;
    }

    if (x.HasValue)
    {
        Console.WriteLine(x.Value);
    }
    else
    {
        Console.WriteLine("X nije definisano");
    }

    Console.WriteLine("Pritisni ENTER za izlazak");
    Console.ReadLine();
}
```

Pitanje 1

Svojstvo pridruženo privatnom članu klase mora imati i get i set akcesor:

- a. Da
- b. Ne

Odgovor: b

Pitanje 2

Definisanjem automatskog svojstvo zamenjuje se definicija:

- a. privatnog polja klase i propertija koji poseduje get i set akcesor
- b. privatnog polja i get akcesora
- c. get i set akcesora nekog privatnog polja klase

Odgovor: a

Pitanje 3

Ako dve promenljive referentnog tipa upućuju na isti objekat i promeni se svojstvo kroz jednu od njih, šta se dešava?

- a. Promena se vidi samo kroz tu promenljivu
- b. Promena se vidi kroz obe promenljive
- c. Druga promenljiva ostaje nepromenjena

Odgovor: b

Pitanje 4

Koja ključna reč u C# označava da kompajler sam zaključi tip promenljive na osnovu dodeljene vrednosti?

- a. auto
- b. var
- c. dynamic

Odgovor: b

Pitanje 5

Šta radi izraz `Guid.NewGuid()`?

- a. Vraća jedinstvenu 128-bitnu vrednost
- b. Generiše slučajan broj tipa `double` između 0 i 1
- c. Vraća null vrednost

Odgovor: a

Pitanje 6

Koja je glavna svrha svojstava (properties) u C#?

- a. Omogućavaju kontrolisan pristup privatnim poljima klase
- b. Služe za kreiranje novih objekata
- c. Ubrzavaju izvršavanje programa

Odgovor: a

Pitanje 7

Koji od sledećih primera prikazuje automatsko svojstvo?

- a. `public int Broj { get { return broj; } set { broj = value; } }`
- b. `public int Broj { get; set; }`
- c. `private int broj;`

Odgovor: b