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

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

CitajPlatu
Equals
GetHashCode
GetType
ime
prezime
ToString

(field) string Radnik.prezime

Debug

Čitanje i setovanje privatnog člana unutar klase

```
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 {0} {1} ima platu od {2} dinara",ime,prezime,plata);
    }
}
```

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)

```
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 {0} {1} ima platu od {2} dinara", ime, prezime, plata);
    }
}
```

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

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

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

    public void Stampaj()
    {
        Console.WriteLine("Radnik {0} {1} ima platu od {2} dinara", ime, prezime, Plata);
    }
}
```

Definisanje klase korišćenjem automatskih svojstava

```
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 {0} {1} ima platu od {2} dinara", Ime, Prezime, Plata);
    }
}
```

Inicijalizator objekta

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

Postavljanje privatnog set aksesora

propg Code Snnipet

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

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

b7ef3276-9cc6-4cef-a462-791965866b17

Pristup svojstvu koje može samo da se čita

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

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.

Pitanje 1

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

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 aksesor
b. privatnog polja i get aksesora
c. get i set aksesora nekog privatnog polja klase

Odgovor: a

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 pridružimo referencu, jednostavno joj pridružimo 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);
    }
}
```

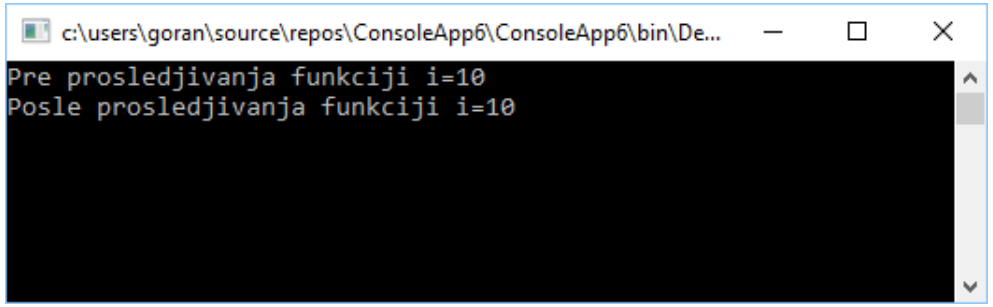
```
static void Main(string[] args)
{
    Osoba os1 = new Osoba {Ime = "Pera", Prezime="Peric" };
    os1.Stampaj();

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

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

Prenos parametara po vrednosti

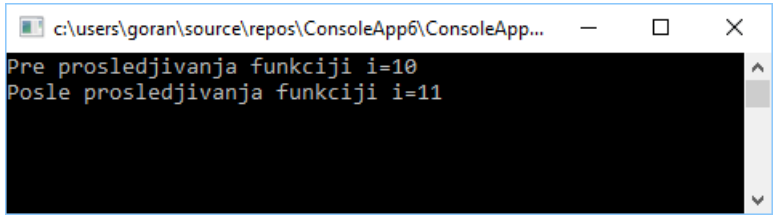
```
public static void Promeni(int a)
{
    a++;
}
```




```
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++;
}
```




```
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();
}
```

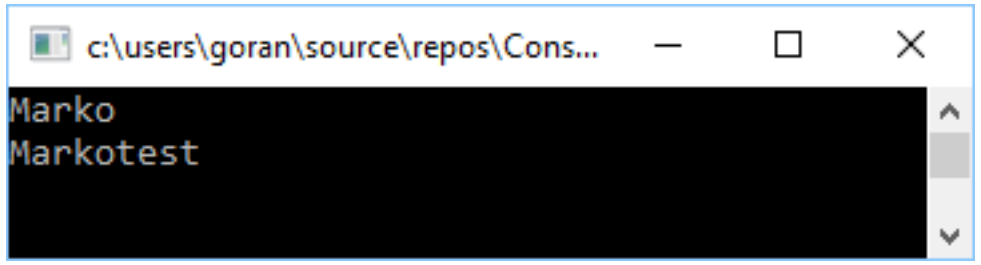
Referenca kao ulazni parameter metode

```
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();
}
```



Izlalzni parametri

- Koristi se ključna reč out da se naglasi da je parametar metode izlazni
- Sličan je ref parametru ali se koristi više za vraćanje parametara 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? x2 = null;
    Console.WriteLine("Unesite broj x1");

    if (int.TryParse(Console.ReadLine(),out int x1  ))
    {
        x2 = x1;
    }

    if (x2.HasValue)
    {
        Console.WriteLine(x2.Value);
    }
    else
    {
        Console.WriteLine("X2 nije definisano");
    }

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