КОНЦЕПТ ООП

Када смо обрађивали наслеђивање, рекли смо да изведена класа наслеђује све чланове родитељске класе осим конструктора, деструктора и приватних чланова. У следећем примеру у изведеној класи постоји идентична метода, **Poruka1()**, као у родитељској класи.

```
using System;

class Roditelj {
    public void Poruka1() { Console.WriteLine("R1"); }
}

class Dete : Roditelj {
    public void Poruka1() { Console.WriteLine("D1"); }
}

class Program
{
    static void Main(string[] args)
    {
        Dete x = new Dete();
        x.Poruka1();
        Console.ReadKey();
    }
}
```

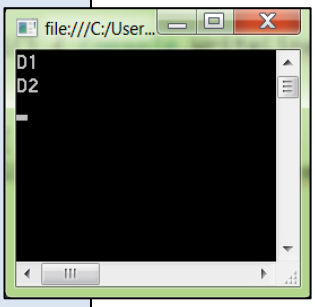
Као последица наведеног појавиће се **УПОЗОРЕЊЕ** о преклапању у називима методе.

1	'Dete.Poruka1()' hides inherited member 'Roditelj.Poruka1()'. Use the new keyword if hiding was intended.	Program.cs	8	18	ConsoleApplication2

Ова ситуација се може решити на два начина:

1. Навођењем кључне речи **new** испред методе у изведеној класи (наслеђена метода се сакрива новом методом у изведеној класи)
2. Навођењем кључне речи **virtual** испред методе испред родитељске методе и кључне речи **override** у изведеној класи (наследђена метода се реимплементира у изведеној класи).

```
using System;

class Roditelj {
    public virtual void Poruka1() { Console.WriteLine("R1"); }
    public void Poruka2() { Console.WriteLine("R2"); }
}

class Dete : Roditelj {
    public override void Poruka1() { Console.WriteLine("D1"); }
    new public void Poruka2() { Console.WriteLine("D2"); }
}

class Program
{
    static void Main(string[] args)
    {
        Dete x = new Dete();
        x.Poruka1();
        x.Poruka2();
        Console.ReadKey();
    }
}
```

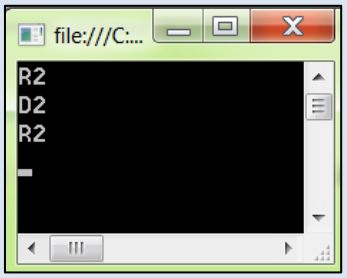


На овај начин решили смо конфликтну ситуацију када имамо методе са истим потписом у родитељској и изведеној класи.

САКРИВАЊЕ МЕТОДЕ

На основу **типа променљиве** одређујемо која ће се позвати метода у родитељској или изведеној класи.

```
using System;

class Roditelj {
    public void Poruka2() { Console.WriteLine("R2"); }
}

class Dete : Roditelj {
    new public void Poruka2() { Console.WriteLine("D2"); }
}

class Program
{
    static void Main(string[] args)
    {
        Roditelj x = new Roditelj();        // x je tipa Roditelj
        x.Poruka2();                        // Poziva se metoda iz roditeljske klase

        Dete y = new Dete();                // y je tipa Dete
        y.Poruka2();                        // Poziva se metoda iz izvedene klase

        Roditelj z = new Dete();            // z je tipa Roditelj
        z.Poruka2();                        // Poziva se metoda iz roditeljske klase
       Console.ReadKey();
    }
}
```



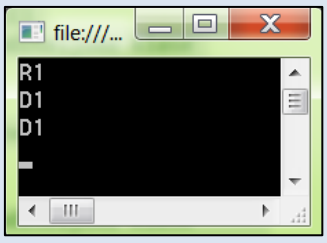
ПОЛИМОРФИЗАМ

Изведена класа наслеђује методе базне тј. родитељске класе. Постоје ситуације у којима желимо да се у изведеној класи пребрише метода родитељске класе и уместо ње дефинише метода са истим потписом али различитим телом методе. Када се креира метода родитељске класе за који се очекује да ће бити пребрисана у изведеној класи метода се означава као виртуелна коришћењем кључне речи **virtual**. Метод у изведеној класи који има исто име као и виртуелни метод у родитељској класи врши "пребрисавање" (**override**) метода из родитељске класе. Дакле, метода изведене класе редефинише виртуелну методу применом модификатора **override** – реимплементација.

Полиморфизам се заснива на идеји да метода која је декларисана у родитељској класи може да се имплементира на више различитих начина у различитим изведеним класама.

На основу **типа инстанце** одређујемо која ће се позвати метода у родитељској или изведеној класи.

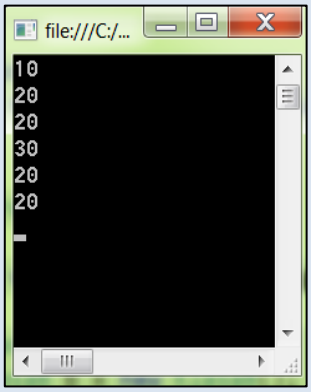
```
using System;

class Roditelj {
    public void Poruka2() { Console.WriteLine("R2"); }
}

class Dete : Roditelj {
    new public void Poruka2() { Console.WriteLine("D2"); }
}

class Program
{
    static void Main(string[] args)
    {
        Roditelj x = new Roditelj();        // x je tipa Roditelj
        x.Poruka2();                        // Poziva se metoda iz roditeljske klase
```



```
        Dete y = new Dete();                // y je tipa Dete
        y.Poruka2();                        // Poziva se metoda iz izvedene klase

        Roditelj z = new Dete();            // z je tipa Roditelj
        z.Poruka2();                        // Poziva se metoda iz roditeljske klase
        Console.ReadKey();
    }
}
```

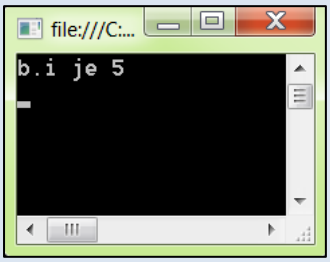
1. Задатак: Анализирати програмски код и проценити шта ће бити приказано?

```
using System;

public class KlasaA {
    public virtual int Metod() { return 10; }
}

public class KlasaB : KlasaA {
    public override int Metod() { return 20; }
}

public class KlasaC : KlasaB {
    public new int Metod() { return 30; }
}

class Program
{
    static void Main(string[] args)
    {
        KlasaA a = new KlasaA();    Console.WriteLine(a.Metod());       // 10
        KlasaB b = new KlasaB();    Console.WriteLine(b.Metod());       // 20
        KlasaA bb = new KlasaB();   Console.WriteLine(bb.Metod());      // 20
        KlasaC c = new KlasaC();    Console.WriteLine(c.Metod());       // 30
        KlasaB cc = new KlasaC();   Console.WriteLine(cc.Metod());      // 20
        KlasaA ccc = new KlasaC();  Console.WriteLine(ccc.Metod());     // 20
        Console.ReadKey();
    }
}
```



2. Задатак: Анализирати и проценити тачност програмског кода?

```
using System;
class Program
{
    static void Main(string[] args)
    {
        B b = new B();
        b.Metod(5);
        Console.WriteLine("b.i je " + b.CitajI());
        Console.ReadKey();

    }
}
class A {
    int i;
    public int CitajI() { return i; }
    public void Metod(int i) { this.i = i; }
}
class B : A {
    public void Metod(string s) {
        Console.WriteLine(s);
    }
}
```



У наведеном примеру позива се метод **Metod(int i)** из класе A. Затим се позива метод **CitajI()** исто из класе A. Програм нема грешку и исписује се као што је приказано на слици.

Метод Metod(string s) и Metod(int i) нису заклоњени зато што немају исти потпис.

3. Задатак: Анализирати и проценити тачност програмског кода?

```
using System;

class A {
    public virtual int Metod() { return 10; }
}
class B : A {
    public new int Metod() { return base.Metod() + 20; }
}
class C : B {
    public override int Metod() { return base.Metod() + 30; }
}

class Program
{
    static void Main(string[] args)
    {
        A a = new A();  Console.WriteLine(a.Metod());
        B b = new B();  Console.WriteLine(b.Metod());
        C c = new C();  Console.WriteLine(c.Metod());
        Console.ReadKey();
    }
}
```

Компајлер пријављује грешку јер метод Metod() у класи C не може бити редефинсан. Проблем можемо решити на два начина:

- кључну реч new у класи B заменити са override. РЕШЕЊЕ: 10 30 60
- класу C наследети из класе A, уместо из класе B. РЕШЕЊЕ: 10 30 40

ЗА НАВЕДЕНЕ ПРИМЕРЕ НАПИСАТИ ТАЧАН ОДГОВОР

231. На програмском језику C# дефинисане су две класе:

```
publicclassRoditelj {
     publicvirtualvoid Poruka1(){ Console.WriteLine("R1"); }
publicvoid Poruka2(){  Console.WriteLine("R2");  }
}
publicclassDete: Roditelj {
publicoverridevoid Poruka1(){ Console.WriteLine("D1"); }
publicnewvoid Poruka2(){ Console.WriteLine("D2"); }
}
```

Унутар функције Main, креирана су два објекта ових класа на следећи начин:

```
Dete x = newDete();
Roditelj y = newDete();
```

Проценити ефекат извршења наведених позива, и на предвиђене линије уписати шта ће се видети на стандардном излазу извршењем позваних метода:

```
x.Poruka1();  _____
x.Poruka2();  _____
y.Poruka1();  _____
y.Poruka2();  _____
```

3

232. На програмском језику C# дефинисане су две класе:

```
publicclassKlasaA {
publicvirtualint Metod() { return 10; } }
publicclassKlasaB : KlasaA {
publicoverrideint Metod() { return 20; } }
publicclassKlasaC : KlasaB {
publicnewint Metod() { return 30; } }
```

Креирани су објекти ових класа и из њих позвана метода ***Metod().*** На предвиђене линије уписати шта метод ***Metod()*** враћа при позиву из наведених објеката:

3

`KlasaA a = newKlasaA(); a.Metod()` враћа вредност _____

`KlasaB b = newKlasaB(); b.Metod()` враћа вредност _____

`KlasaA bb = newKlasaB(); bb.Metod()` враћа вредност _____

`KlasaC c = newKlasaC(); c.Metod()` враћа вредност _____

`KlasaB cc = newKlasaC(); cc.Metod()` враћа вредност _____

`KlasaA ccc = newKlasaC(); ccc.Metod()` враћа вредност _____

206. Дат је код програма у програмском језику C# у ком су дефинисане три класе: classProgram која садржи Main(string[] args) методу, classA и classB. Анализирати дати код и одредити да ли је код исправно написан. Заокружити број испред исказа који даје информацију о тачности кода:

```
classProgram {
publicstaticvoid Main(string[] args) {
B b = newB();
    b.Metod(5);
Console.WriteLine("b.i je " + b.CitajI());
  }
}
classA {
int i;
publicint CitajI(){return i;}
publicvoidMetod(int i) { this.i = i; }
}
classB : A {
publicvoidMetod(string s){
Console.WriteLine(s);
  }
}
```

2

1. Програм има грешку, јер је метод **Metod(inti)** надјачан (предефинисан) са различитим потписом у класи B.
2. Програм има грешку, јер се **b.Метод(5)** не може позвати пошто је метод **Metod(inti)** заклоњен у класи B.
3. Програм има грешку због **b.i**, јер је поље **i** неприступачно из класе B.
4. Програм нема грешке, јер наслеђени метод класе A,**Metod(inti)** није надјачан у класи B, већ је дефинисан преоптерећен метод **Metod(string s).**

186. На програмском језику C# дефинисане су класе:

```
publicclassA {
publicvirtualint Metod() { return 10; }
}
publicclassB : A {
publicnewint Metod() { returnbase.Metod() +20; }
}
publicclassC : B {
publicoverrideint Metod() { returnbase.Metod() +30; }
}
```

Унутар функције Main, креирана су три објекта ових класа и из сваког од њих извршен позив методе ***Metod()***

2

Заокружити број испред понуђеног одговора који представља сценарио који ће се десити при покретању програма:

1. Метод позван из класе A враћа вредност 10, из класе B 10, а из класе C 30
2. Метод позван из класе A враћа вредност 10, из класе B 30, а из класе C 60
3. Програм се покреће, али баца изузетак при позиву методе из класе B jer new не иде у комбинацији са virtual
4. Компилатор јавља грешку јер у класи C метод не може бити редефинисан

225. На програмском језику C# дефинисане су класе:

```
1.publicclassA {
2.    publicvirtualint Metod() { return 10; }
3. }
4.publicclassB : A {
5. publicnewint Metod() { return 20; }
6. }
7.publicclassC : B {
8. publicoverrideint Metod() { return 30; }
9. }
```

3

Компилатор јавља грешку при превођењу овог кода коју је могуће решити на више начина у зависности од очекиваног ефекта.

Која ће решења отклонити грешку у коду:

1. У 5. линији кода метод у класи B прогласити за **abstract** уместо **new**
2. У 5. линији кода кључну реч **new** заменити са **override**
3. У 7. линији кода класу C наследити из класе A, уместо из класе B
4. У 8. линији кода, иза декларације методе у класи C, позвати основни метод **:base()**
5. У 8. линији кода обрисати кључну реч **override** и заменити је са **sealed**

211. Дат је код на C#-у којим су креиране три класе у ланцу наслеђивања. Имајући у виду класификаторе приступа пољима класа, заокружити бројеве испред поља која ће бити видљива унутар класе Sin:

```
public class Deda {
   private double penzija;
   protected string adresa;
   public string ime;
}
public class Otac: Deda {
   private double plata;
   protected string struka;
}
public class Sin: Deda {
   public int razred;
}
```

1. penzija
2. adresa
3. ime
4. plata
5. struka
6. razred

1,5

Изведена класа **Sin** види сва поља родитељске класе **Deda** са модификатором приступа public и protected и сва поља унутар своје класе а то су : **razred**, **adresa** и **ime**.

Дат је код на C#-у којим су креиране три класе у ланцу наслеђивања. Имајући у виду класификаторе приступа пољима класа, заокружити бројеве испред поља која ће бити видљива унутар класе Sin:

```
public class Deda {
   private double penzija;
   protected string adresa;
   private int godinaRodjenja;
   public string ime;
}
public class Otac: Deda {
   private double plata;
   protected string struka;
   private string brojRacuna;
}
public class Sin: Otac {
   public int razred;
}
```

1. penzija
2. adresa
3. ime
4. plata
5. struka
6. razred
7. godinaRodjenja
8. brojRacuna

2

212. Дат је код на C#-у којим су креиране три класе у ланцу наслеђивања. Унутар сваке класе декларисан је по један private, public и protected атрибут. У методи ***Main()*** класе ***Program*** креиран је објекат s класе ***Sin*** (`Sin s = new Sin();`) Заокружити бројеве испред поља која ће бити видљива у креираном објекту s класе Sin:

```
public class Deda {
   private double penzija;
   protected string adresa;
   public string ime;
}
public class Otac: Deda {
   private double plata;
    protected string  firma;
    public string struka;
}
public class Sin: Otac {
    private double prosek;
    protected int razred;
    public string skola;
}
```

1. penzija
2. adresa
3. ime
4. plata
5. struka
6. firma
7. prosek
8. razred
9. skola

1,5

Креирањем објекта s у класи **Program** атрибуту који се виде су само са модификатором **public**.