Strukture

Strukture

- Strukture su vrednosni tipovi podataka
- Podaci iz strukture se čuvaju na steku
- Koriste se za modelovanje stavki koje sadrže manje količine podataka
- Strukture mogu da sadrže polja i metode kao i klase
- Ne mogu se definisati sopstvene hijerarhije nasleđivanja između struktura
- Ne mogu se definisati strukture koje proizilaze iz klasa ili drugih struktura

Sistemske strukture

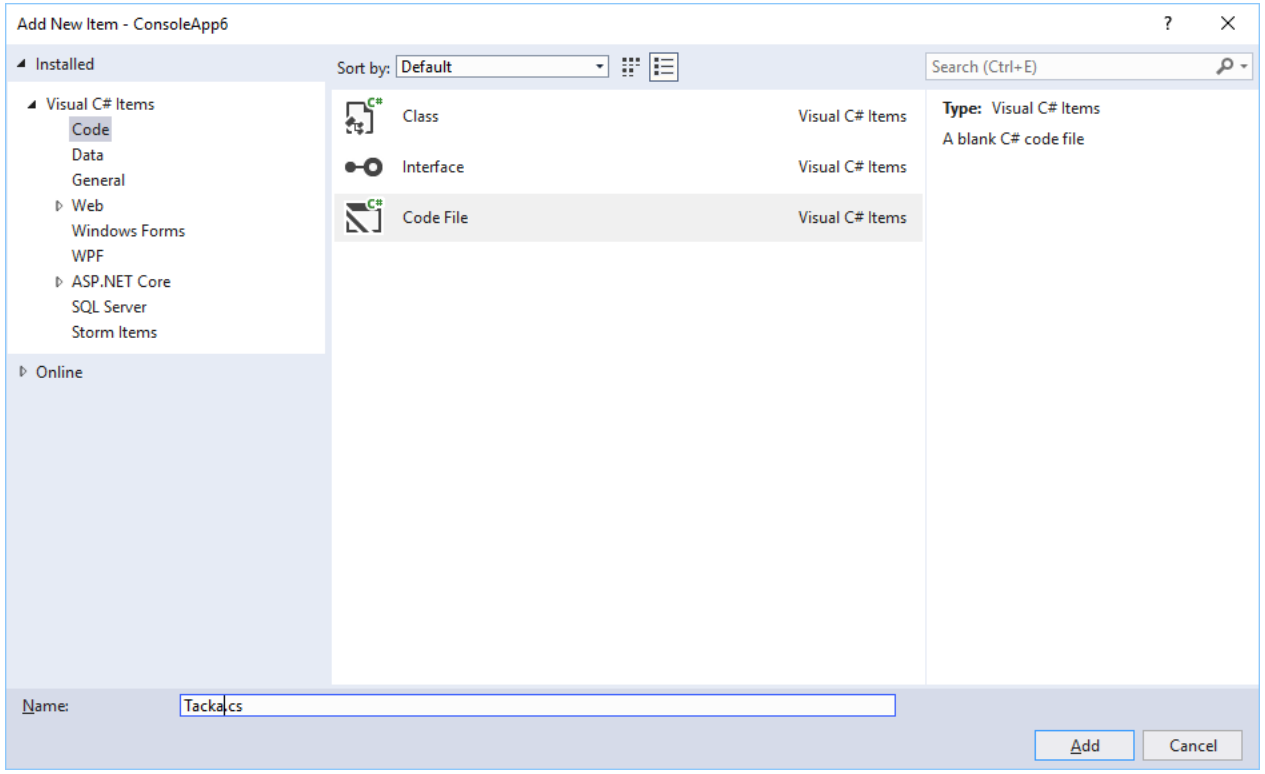
System.Byte	System.Int64	System.Decimal
System.Int16	System.Single	System.Boolean
System.Int32	System.Double	System.Char

```
static void Main(string[] args)
{
    Console.WriteLine("Najveci ceo broj je {0}", Int32.MaxValue);
    Console.WriteLine("Najmani ceo broj je {0}", Int32.MinValue);

    string s = "1234";
    int i = Int32.Parse(s);
    i++;
    Console.WriteLine(i);

    Console.ReadKey();
}
```

Osobine strukture

- Ne može se definisati konstruktor bez parametara za strukturu jer za razliku od klase kompajler generiše sopstveni default konstruktor za strukturu
- Svaki konstruktor mora eksplicitno inicijalizovati svako polje u strukturi
- Nije dozvoljena inicijalizacija polja pri samoj deklaraciji
- Pri kreiranju promenljive tipa strukture nije neophodno je navoditi ključnu reč new
- Moguće je deklarisati nullable promenljivu tipa strukture

Dodavanje cs fajla



Primer strukture

```
struct Tacka
{
    public int x;
    public int y;

    public Tacka(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}
```

Program.cs

```
static void Stampaj(Tacka t)
{
    Console.WriteLine($"Koordinate tacke su: ({t.x},{t.y})");
}
```

Inicijalizacija strukture

```
static void Main(string[] args)
{
    // Deklarisi objekat
    Tacka t1;
    // Inicijalizacija
    t1.x = 10;
    t1.y = 20;

    Stampaj(t1);

    Tacka t2 = new Tacka();
    t2.x = 15;
    t2.y = 20;
    Stampaj(t2);

    Tacka t3 = new Tacka(30, 40);
    Stampaj(t3);
    Console.ReadLine();
}
```

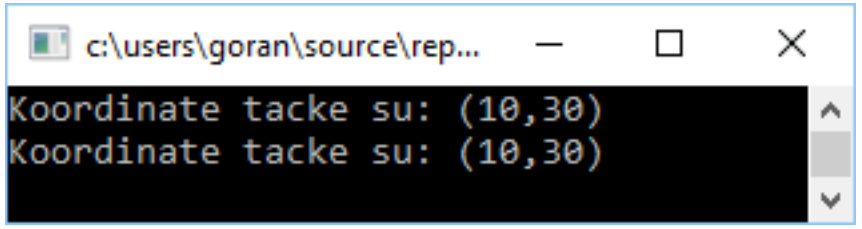
Ne mora se koristiti new operator da bi se kreirala instanca strukture

Operator new samo inicijalizuje polja u strukturi
Uvek koristiti konstruktor da bi se osigurala inicijalizacija polja u strukturi

Prosleđivanje strukturnog tipa metodi

```
public static void PromeniTacku(Tacka t)
{
    t.x += 1;
    t.y += 1;
}
```

```
static void Main(string[] args)
{
    Tacka t1 = new Tacka(10, 30);
    Stampaj(t1);
    PromeniTacku(t1);
    Stampaj(t1);
    Console.ReadLine();
}
```



Pitanje 1

Struktura predstavlja:

a. Vrednosni tip podataka
b. Referentni tip podataka
c. I vrednosni i referentni tip podataka

Odgovor: a

Pitanje 2

Da li je moguće kreirati novi strukturni tip podataka na osnovu prethodno definisane strukture od strane korisnika

a. da
b. ne

Odgovor: b

Statički članovi klase

Statički članovi klase

- Pripadaju klasi a ne instanci klase
- Pristupa im se preko imena klase
- I metode i polja i svojstva klase mogu biti statički
- Pozivaju se bez kreiranja objekata klase
- Statičke metode i svojstva ne mogu pristupati ne - statičkim poljima u klasi u kojoj se definišu

Statički članovi klase

- Statičko polje se često koristi da čuva broj kreiranih objekata klase
- Statičko polje se koristi i za deljenje vrednosti između različitih instanci te klase
- Statički član klase se definiše korišćenjem ključne reči static pre povratnog tipa

Primer definisanja statičkog polja

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

    public static int brojOsoba = 0;

    public Osoba()
    {
        brojOsoba++;
    }
}
```

Upotreba statičkog polja

```
static void Main(string[] args)
{
    Console.WriteLine($"Pocetni broj osoba je: {Osoba.brojOsoba}");
    Osoba os1 = new Osoba();
    Osoba os2 = new Osoba();
    Console.WriteLine($"Trenutni broj osoba je: {Osoba.brojOsoba}");
    Osoba.brojOsoba = 10;
    Console.WriteLine($"Trenutni broj osoba je: {Osoba.brojOsoba}");
    Console.ReadLine();
}
```

Definisanje statičke metode

```
class Pravougaonik
{
    private double sirina;
    private double visina;
    public Pravougaonik(double sirina, double visina)
    {
        this.sirina = sirina;
        this.visina = visina;
    }

    public double Povrsina()
    {
        return sirina * visina;
    }

    public static double RacunajPovrsinu(double a, double b)
    {
        return a * b;
    }
}
```

Poziv statičke metode

```
static void Main(string[] args)
{
    //Upotreba nestaticke metode
    Pravougaonik pr1 = new Pravougaonik(12.3, 45.6);
    double P1 = pr1.Povrsina();
    Console.WriteLine("P1= " + P1);

    //Upotreba staticke metode
    double P2 = Pravougaonik.RacunajPovrsinu(12.3, 45.6);
    Console.WriteLine("P2= " + P2);
    Console.ReadLine();
}
```

Statička klasa

- Ne može se instancirati
- Statičke klase sadrže samo statičke članove
- Koristi se kao kontejner za skup metoda koje rade sa ulaznim parametrima
- Klasa Math iz prostora imena sistem sadrži statičke metode za izvršavanje matematičkih operacija

Statička klasa Math

```
public static class Math
```

```
static void Main(string[] args)
{
    double x = Math.PI;
    double pi = Math.Round(x, 2);
    Console.WriteLine("pi= " + pi);
    Console.ReadLine();
}
```

Statički konstruktor

- Klasa može sadržati statički konstruktor koji se koristi za inicijalizaciju statičkih članova
- I statička i ne statička klasa mogu sadržati statički konstruktor
- Statički konstruktor nema modifikator pristupa i nema parametre
- Statički konstruktor se poziva automatski pre kreiranja instanci i pre pristupa bilo kom statičkom članu
- Statički konstruktor se poziva jednom pre poziva bilo kog instans konstruktora

Statičko svojstvo

```
class Radnik
{
    public static int brojRadnika;
    private static int brojac;

    public string Ime { get; set; }
    public string Prezime { get; set; }

    // staticki konstruktor
    static Radnik()
    {
        brojRadnika = 10;
        brojac = 0;
    }
    // staticko read only svojstvo
    public static int BrojRadnika
    {
        get { return brojac + brojRadnika; }
    }
    public Radnik()
    {

        brojac++;
    }
}
```

Poziv statičkog svojstva

```
static void Main(string[] args)
{
    Radnik r1 = new Radnik {Ime="Marko", Prezime="Markovic" };
    Console.WriteLine(Radnik.BrojRadnika);
    Radnik r2 = new Radnik { Ime = "Milan", Prezime = "Petrovic" };
    Console.WriteLine(Radnik.BrojRadnika);
    Console.ReadLine();
}
```

Pitanje 1

Javnom statičkom polje klase A se u nekoj metodi klase B

a. može pristupiti pre instanciranja klase A

b. ne može pristupiti pre instanciranja klase A

c. može pristupiti samo nakon instanciranja klase B

Odgovor: a

Pitanje 2

Statička klasa može da sadrži jedno nestatičko polje?

a. Da
b. Ne

Odgovor: b

Pitanje 3

Da li je moguće upotrebiti ključnu reč this unutar statičke metode:

a. Da
b. Ne

Odgovor: b

Pitanje 4

Kreirana je klasa Student koja modeluje Studente nekog fakulteta.
Da li polje Ime klase Student može biti statičko:

a. Da
b. Ne

Odgovor: b

Nasleđivanje i polimorfizam

Nasleđivanje

- Nasleđivanje omogućava da se kreiraju novi tipovi podataka na osnovu već postojećih tipova
- Nasleđivanje je vrsta relacije imeđu osnovne(bazne) klase i izvedenih klasa
- Izvedena klasa nasleđuje sva polja i metode osnovne klase
- Izvedena klasa ima članove svojstvene samo izvedenoj klasi
- Izvedena klasa postaje više specijalizovana

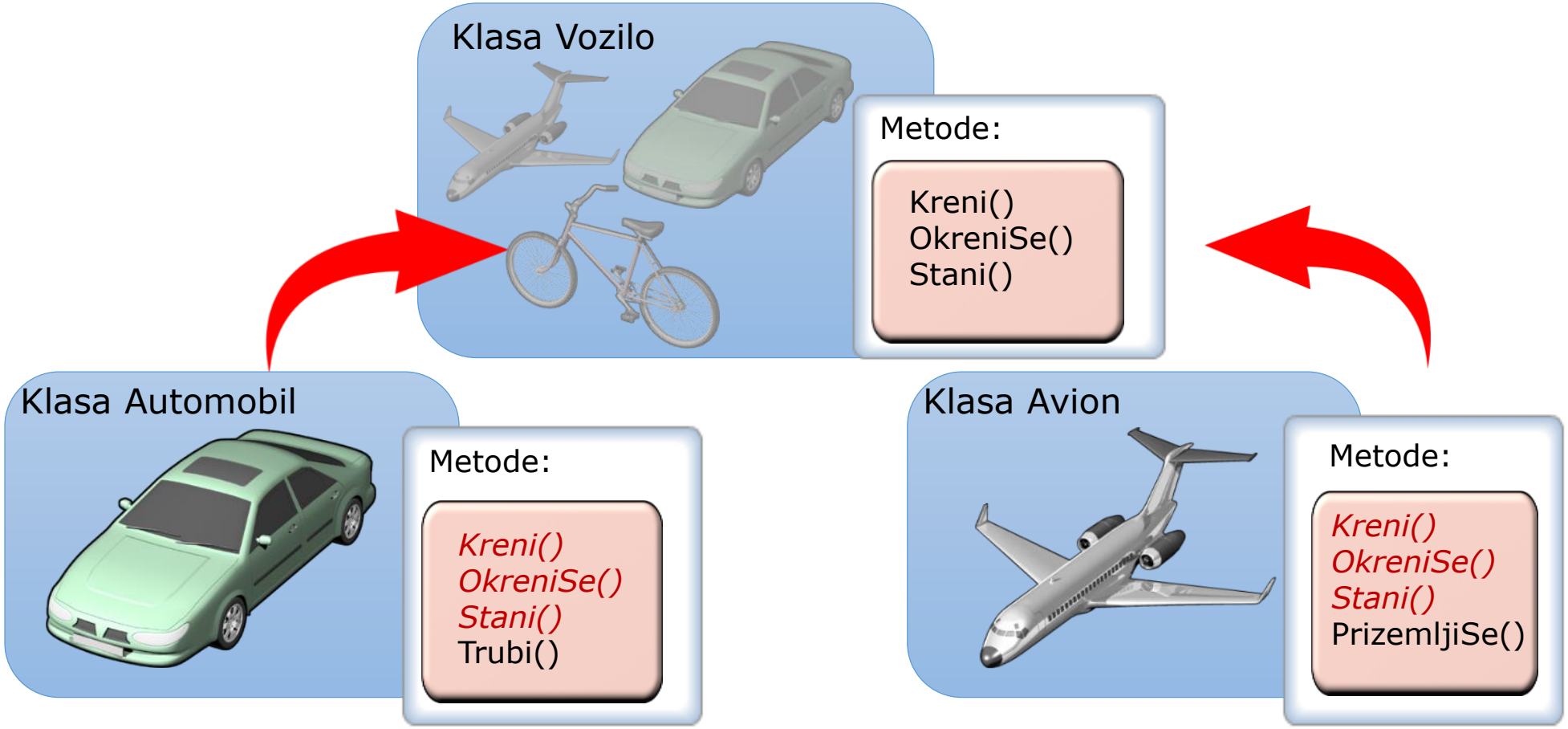
Nasleđivanje-pojmovi

- Osnovna klasa se naziva još i natklasa, bazna klasa, roditeljska klasa (parent class), super klasa
- Izvedena klasa se naziva još klasa potomak (child class) i podklasa
- Izvedena klasa može da nasledi samo jednu baznu klasu
- Nasleđivanje redukuje ponavljanje koda
- public class B : A { telo klase B }
  //(klasa B izvedena iz klase A)
- public seald class NekaKlasa {telo klase} – tada se iz ovakve klase ne može vršiti nasleđivanje
- Vrednosni tipovi su seald implicitno tj. ne mogu se nasleđivati

Vidljivost članova

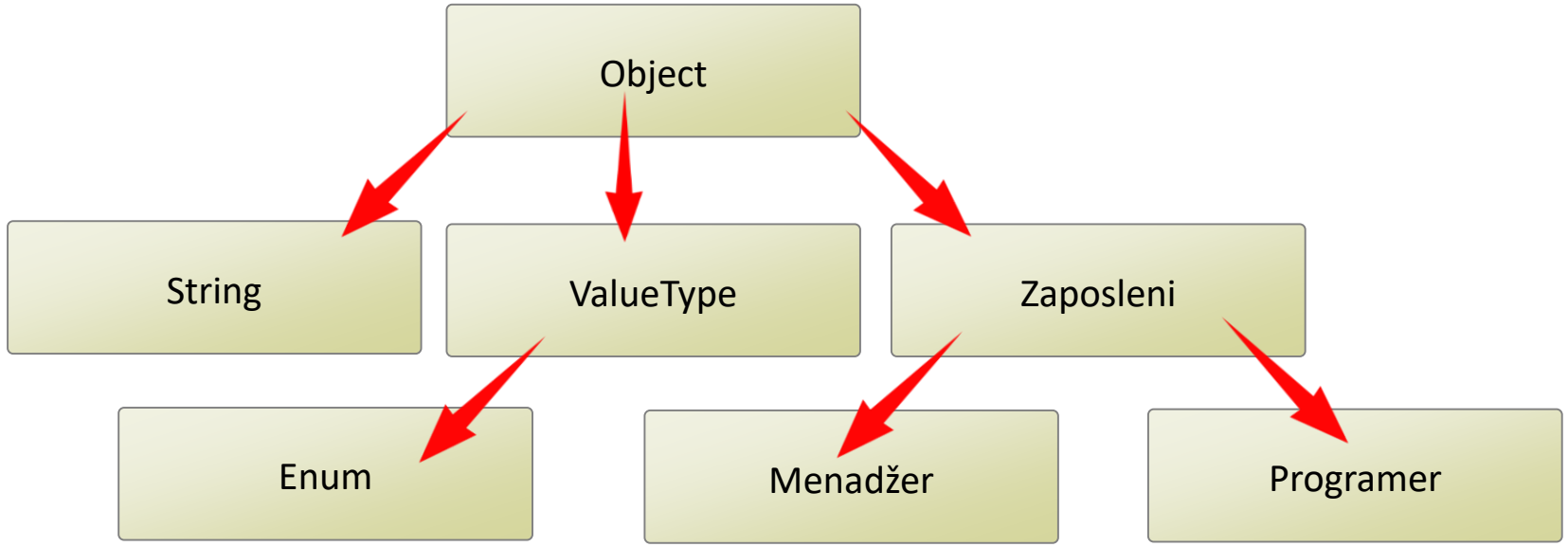
- Privatni članovi bazne klase nisu vidljivi u izvedenoj klasi osim ako izvedena klasa nije ugnježdena u baznu klasu
- Protected članovi klase su vidljivi u izvedenoj klasi
- Public članovi klase su vidljivi u izvedenoj klasi

Ilustracija nasleđivanja



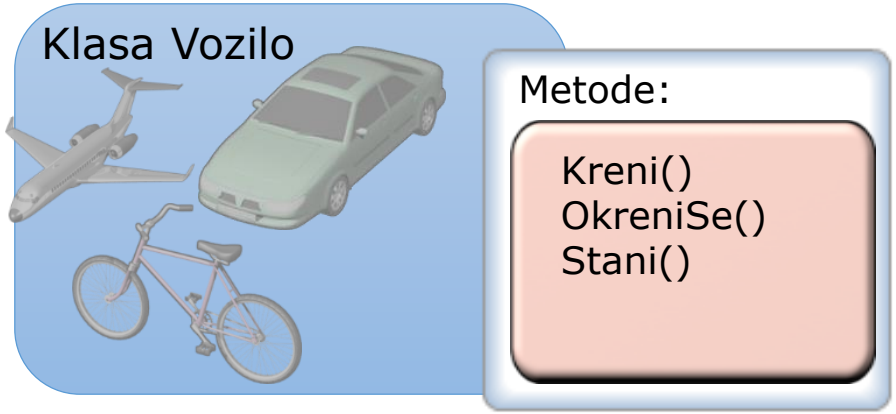
Nasleđivanje u .NET Frameworku

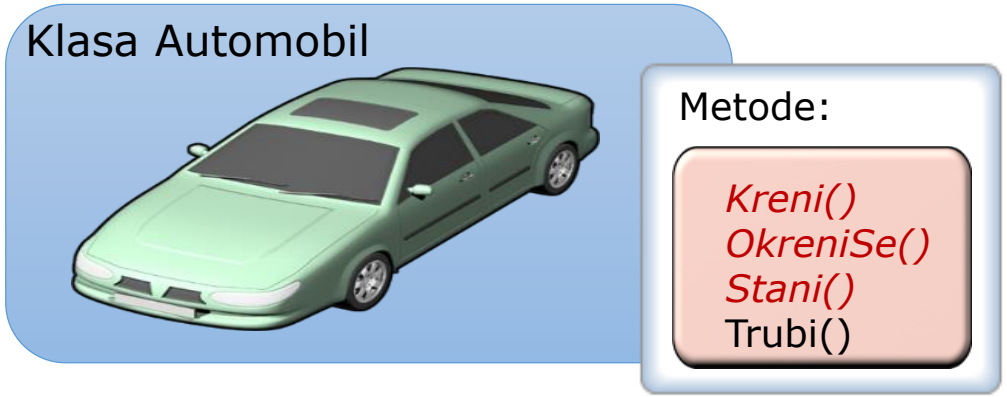
- Svi tipovi u .NET frameworku su nasleđeni direktno ili indirektno iz klase System.Object



Prvo pravilo polimorfizma

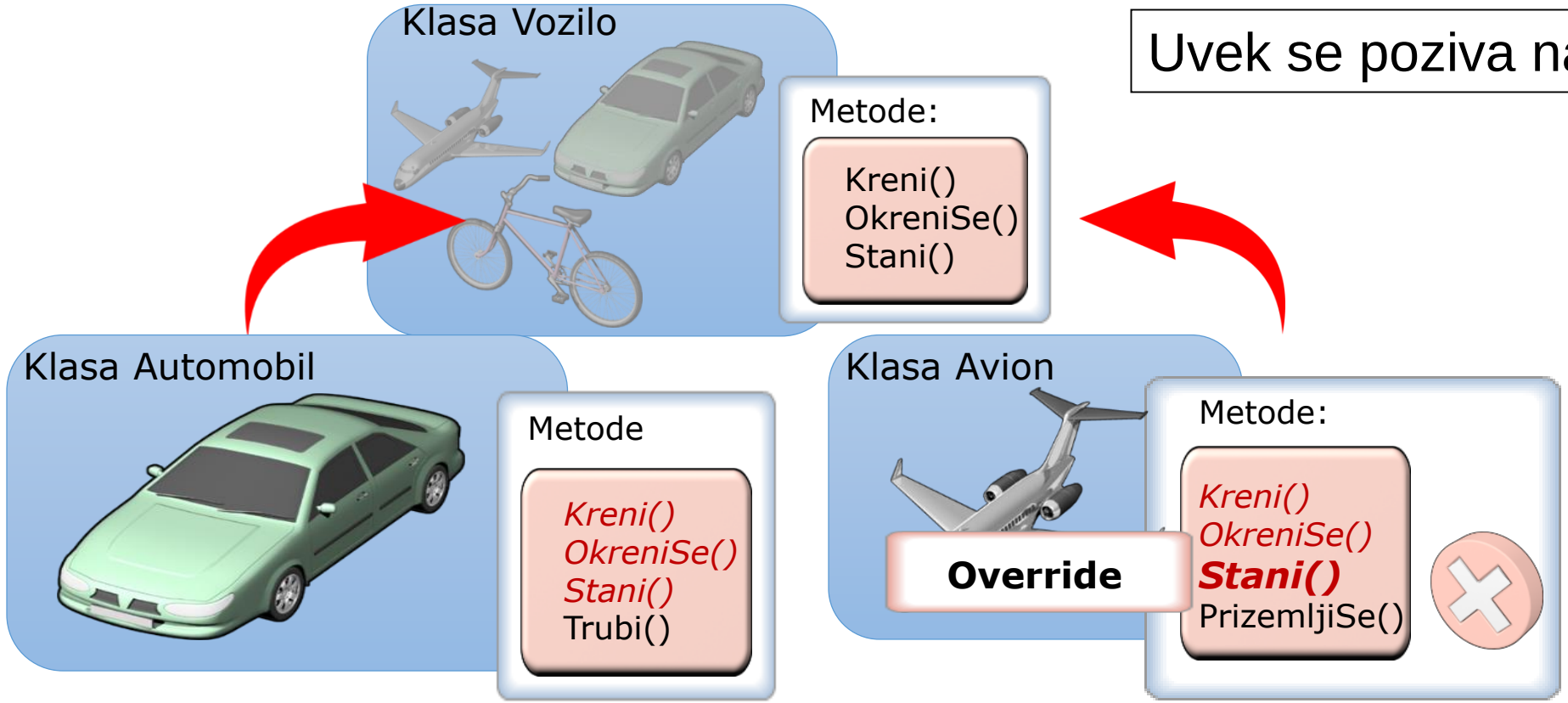
Referenca tipa bazne klase može pokazivati na objekat izvedene klase






```
Vozilo V = new Automobil();
V.Kreni();   //Dozvoljeno
V.Trubi();   //Nije dozvoljeno
```

Drugo pravilo polimorfizma



Virtuelne metode

- Kada se kreira metoda u baznoj klasi za koju se očekuje da će biti promenjena u izvedenoj klasi, metoda se definiše kao virtuelna (***virtual***).
- Metoda u izvedenoj klasi koja ima isto ime kao i virtuelna metoda u baznoj klasi vrši "prebrisavanje" (***override***) metode iz bazne klase

Definisanje virtuelne metode u baznoj klasi

```
class Oblik
{
    public virtual void Crtanje()
    {
        Console.WriteLine("Genericka metoda za crtanje");
    }
}
```

Prebrisavanje virtuelne metode u izvedenoj klasi

```
class Krug : Oblik
{
    public override void Crtanje()
    {
        Console.WriteLine("Crtanje kruga");
    }
}
```

```
class Kvadrat : Oblik
{
    public override void Crtanje()
    {
        Console.WriteLine("Crtanje kvadrata");
    }
}
```

Ilustracija 1. i 2. pravila polimorfizma

```
static void Main(string[] args)
{
    Oblik[] oblici = new Oblik[4];
    // 1. pravilo polimorfizma
    oblici[0] = new Krug();
    oblici[1] = new Krug();
    oblici[2] = new Kvadrat();
    oblici[3] = new Kvadrat();

    foreach (Oblik obl in oblici)
    {
        // 2. pravilo polimorfizma
        obl.Crtanje();
    }
    Console.ReadLine();
}
```

Bazna klasa

```
class Osoba
{
    public string Ime { get; set; }
    public string Prezime { get; set; }
    public Osoba(string Ime, string Prezime)
    {
        this.Ime = Ime;
        this.Prezime = Prezime;
    }

    public virtual void Stampaj()
    {
        Console.WriteLine($"Osoba: {Ime} {Prezime}");
    }
}
```

Izvedena klasa

```
class Student : Osoba
{
    public string Smer { get; set; }
    public Student(string Ime, string Prezime, string Smer):base(Ime,Prezime)
    {
        this.Smer = Smer;
    }

    public override void Stampaj()
    {
        base.Stampaj();
        Console.WriteLine("Smer: " + Smer);
    }

}
```

Primera 1 i 2 pravila polimorfizma

```
static void Main(string[] args)
{
    Student st1 = new Student("Pera", "Peric", "Informatika");
    st1.Stampaj();

    Osoba os1 = new Student("Mika", "Mikic", "Istorija");
    os1.Stampaj();

    Console.ReadLine();
}
```

Pitanje 1

Prvo pravilo polimorfizma glasi:

a. Uvek se poziva najbolja metoda.
b. Objektu bazne klase se uvek može pristupiti posredstvom reference tipa izvedene klase
c. Objektu izvedene klase se uvek može pristupiti posredstvom reference tipa bazne klase

Odgovor: c

Pitanje 2

Ukoliko želimo da zabranimo nasleđivanje naše klase, onda to postižemo korišćenjem ključne reči ispred imena klase

a. private
b. noninheritable
c. seald

Odgovor c

Pitanje br 3

Drugo pravilo polimorfizma glasi:

a. Uvek se poziva najbolja metoda.
b. Objektu bazne klase se uvek može pristupiti posredstvom reference tipa izvedene klase
c. Objektu izvedene klase se uvek može pristupiti posredstvom reference tipa bazne klase
d. Uvek se poziva metoda bazne klase.

Odgovor: a