

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.Int16
System.Int32

System.Int64
System.Single
System.Double

System.Decimal
System.Boolean
System.Char

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

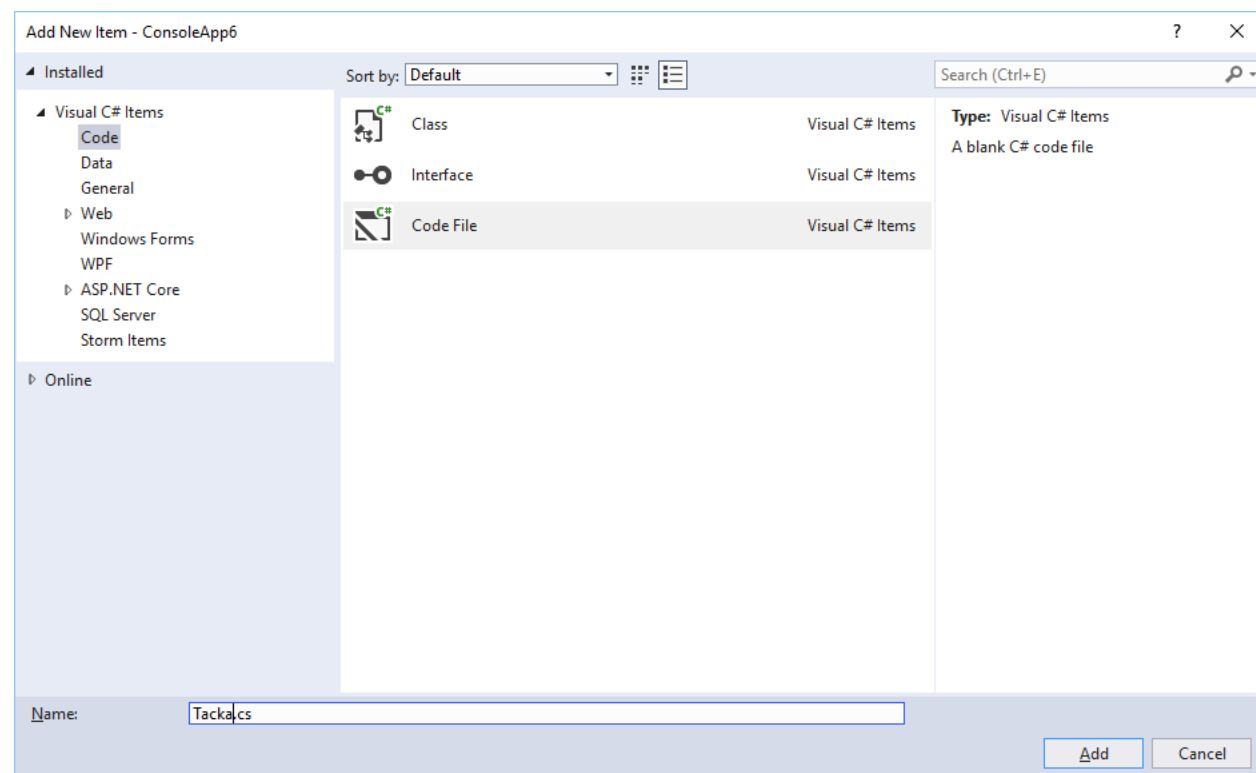
    string s = "1234";
    int i = Int32.Parse(s);
    i++;
    Console.WriteLine(i);

    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 navoditi ključnu reč **new**
- Moguće je deklarirati 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);

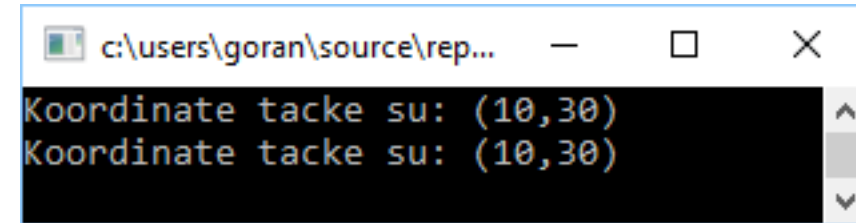
    Tacka t3 = new Tacka(30, 40);
    Stampaj(t3);
    Console.ReadLine();
}
```

- 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();
}
```

A screenshot of a Windows console window. The title bar shows the file path 'c:\users\goran\source\rep...'. The console output consists of two lines: 'Koordinate tacke su: (10,30)' followed by a new line and another 'Koordinate tacke su: (10,30)'. The text is displayed in a monospaced font with a light blue color on a black background.

```
c:\users\goran\source\rep...
Koordinate tacke su: (10,30)
Koordinate tacke su: (10,30)
```

Statički členovi 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 nestatičkim 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

```
internal class Pravougaonik
{
    public double Sirina { get; set; }
    public double Visina { get; set; }

    // Konstruktor
    public Pravougaonik(double sirina, double visina)
    {
        Sirina = sirina;
        Visina = visina;
    }

    // Nestatička metoda
    public double Povrsina()
    {
        return Sirina * Visina;
    }

    // Statička metoda
    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 **System** 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 polja klase
- I statička i nestatička klasa mogu sadržati statički konstruktor
- Statički konstruktor nema modifikator pristupa i nema parametre
- Statički konstruktor se poziva samo jednom, i to pre kreiranja bilo koje instance ili pristupa statičkom članu
- Koristi se za dodelu početnih vrednosti statičkim poljima ili za izvršavanje inicijalnog koda koji se pokreće samo jednom

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();
}
```

Nasleđivanje i polimorfizam

Nasleđivanje

- Nasleđivanje omogućava kreiranje novih tipova podataka na osnovu postojećih tipova
- Nasleđivanje predstavlja vezu između osnovne (bazne) i izvedene klase
- Izvedena klasa nasleđuje sva polja i metode osnovne klase
- Izvedena klasa ima i svoje specifične članove
- Izvedena klasa postaje više specijalizovana

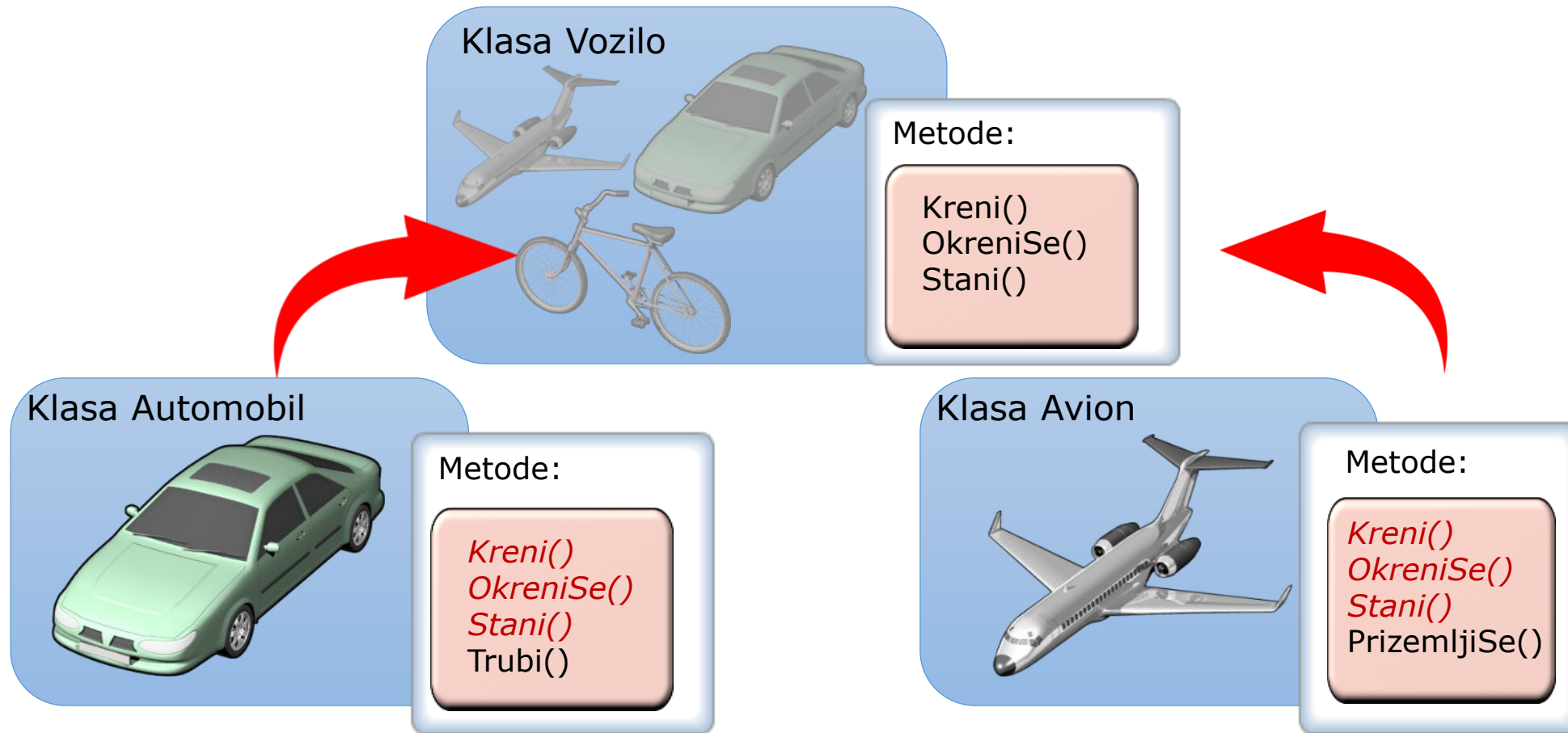
Osnovni pojmovi nasleđivanja

- 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 sealed class NekaKlasa { /* telo klase */ }** – tada se iz ovakve klase ne može vršiti nasleđivanje
- Vrednosni tipovi su *sealed* implicitno tj. ne mogu se nasleđivati

Vidljivost članova klase (modifikatori pristupa)

- Privatni članovi bazne klase nisu vidljivi u izvedenoj klasi, osim ako je izvedena klasa ugnježđena unutar bazne klase
- Protected članovi klase su vidljivi u izvedenoj klasi
- Public članovi klase su vidljivi u izvedenoj klasi
- Ako nema modifikatora, član je po podrazumevanoj vrednosti **private**

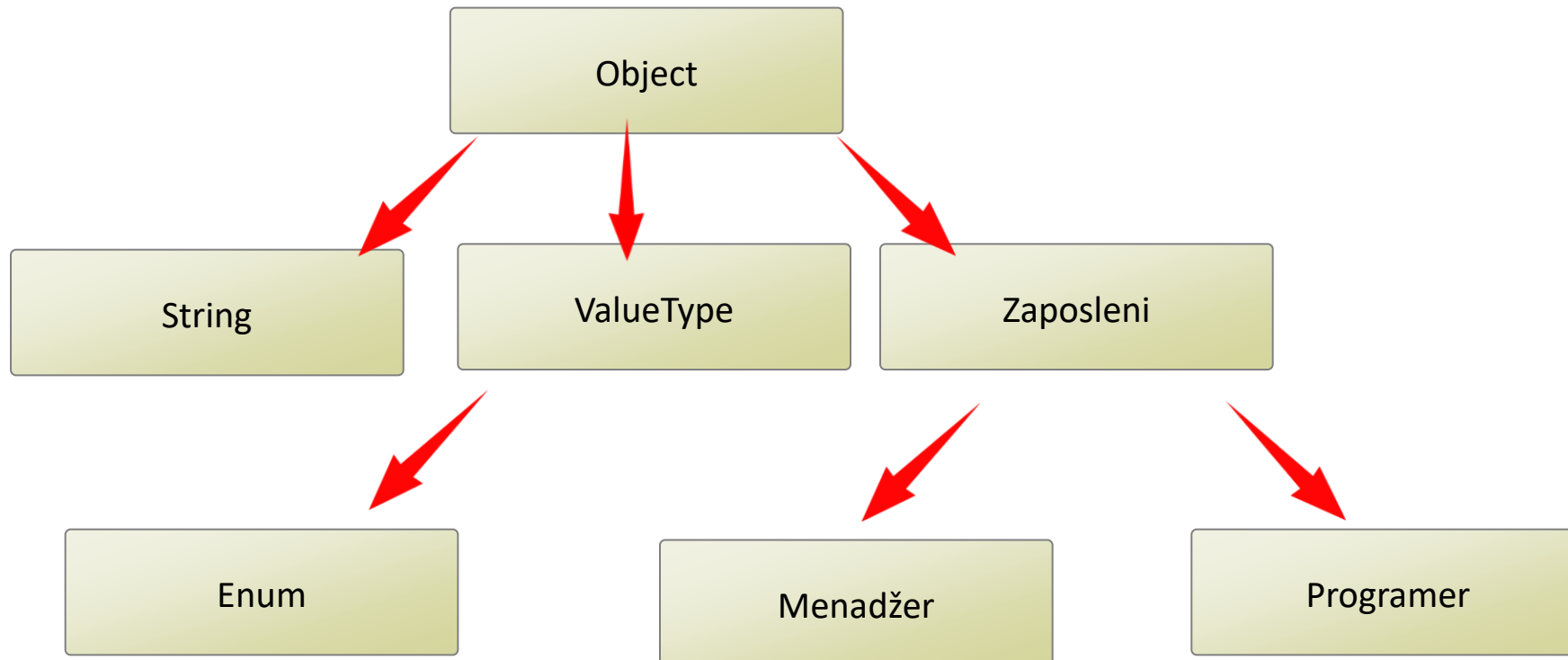
Ilustracija nasleđivanja



Izvedene klase nasleđuju metode osnovne klase i dodaju nove

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

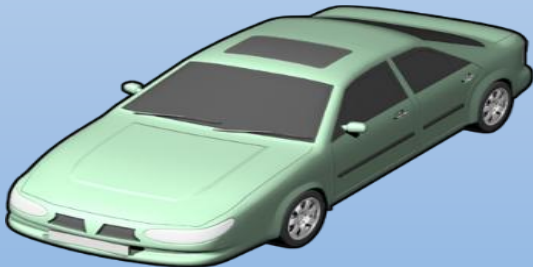
Klasa Vozilo



Metode:

Kreni()
OkreniSe()
Stani()

Klasa Automobil

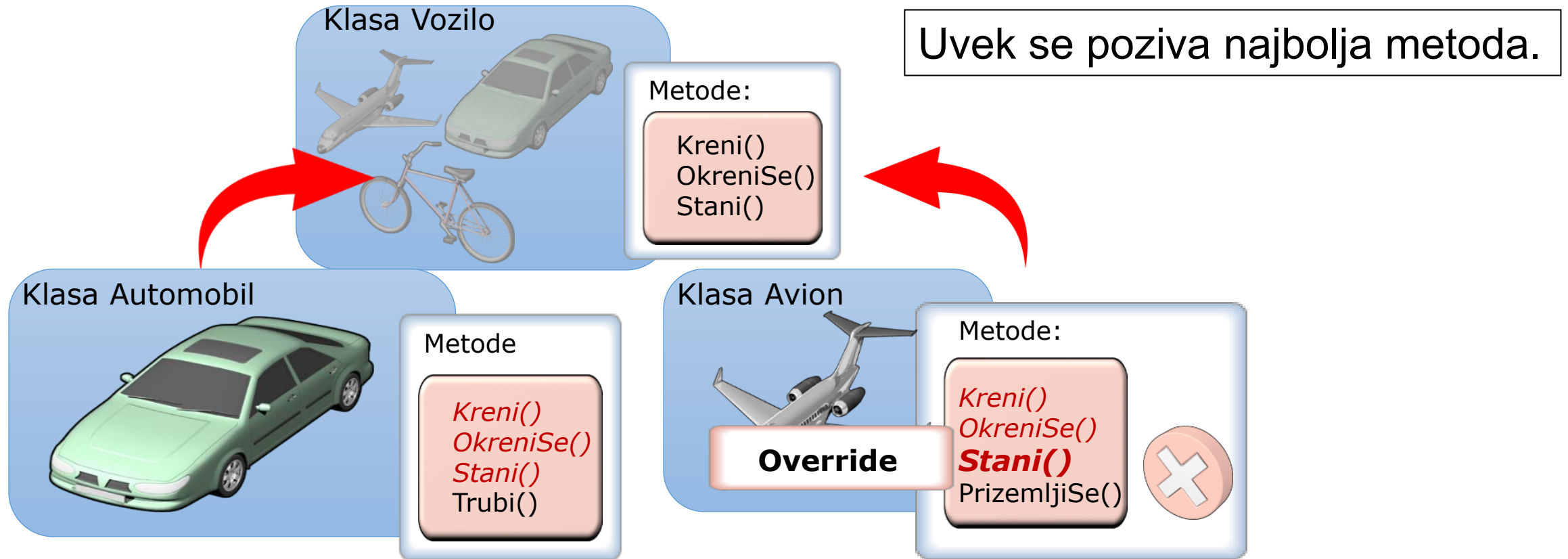


Metode:

Kreni()
OkreniSe()
Stani()
Trubi()

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

Drugo pravilo polimorfizma



Virtuelne metode

- Kada se u baznoj klasi definiše metoda za koju se očekuje da će biti redefinisana u izvedenoj klasi, označava se 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

```
internal class Vozilo
{
    public virtual void Kreni()
    {
        Console.WriteLine("Vozilo kreće");
    }
}
```

Prebrisavanje virtuelne metode u izvedenoj klasi

```
internal class Automobil : Vozilo
{
    public override void Kreni()
    {
        Console.WriteLine("Automobil kreće");
    }
}
```

```
internal class Avion : Vozilo
{
    public override void Kreni()
    {
        Console.WriteLine("Avion ubrzava po pisti i uzleće.");
    }
}
```

Ilustracija 1. i 2. pravila polimorfizma

```
static void Main(string[] args)
{
    Vozilo[] vozila = new Vozilo[3];

    // 1. pravilo polimorfizma
    vozila[0] = new Vozilo();
    vozila[1] = new Automobil();
    vozila[2] = new Avion();

    // 2. pravilo polimorfizma
    foreach (Vozilo v in vozila)
    {
        v.Kreni();
    }

    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);
    }
}
```

Primeri 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

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

Pitanje 3

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 4

Statička klasa može da sadrži jedno nestatičko polje?

- a. Da
- b. Ne

Odgovor: b

Pitanje 5

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

- a. Da
- b. Ne

Odgovor: b

Pitanje 6

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

- a. Da
- b. Ne

Odgovor: b

Pitanje 7

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 8

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

- a. private
- b. noninheritable
- c. sealed

Odgovor c

Pitanje 9

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

Pitanje 10

Koja osobina važi za virtuelne metode?

- a. Mogu se definisati samo u apstraktnim klasama
- b. Mogu se redefinisati (override) u izvedenim klasama
- c. Moraju biti statičke

Odgovor: b

Pitanje 11

Kada se izvršava statički konstruktor klase?

- a. Svaki put kada se kreira nova instanca klase
- b. Izvršava se automatski samo jednom, pre prvog pristupa statičkim članovima klase
- c. Nakon prvog pristupa bilo kom statičkom članu

Odgovor: b