Konstruktori klase

Klasa sa podrazumevanim konstruktorom

```
package konstruktori;

import java.util.Locale;

public class TekuciRacun {

    public String ime;
    public String prezime;
    public double stanje;

    public void uplata(double iznos){

        stanje += iznos;
    }

    public void isplata(double iznos) {

        stanje -= iznos;
    }

    public void stampaj(){

        System.out.format(Locale.ENGLISH,"Korisnik %s %s ima %.2f dinara na racunu%n", ime, prezime, stanje);
    }
```

Definisanje parametarskog konstruktora

```
public TekuciRacun(String ime, String prezime, double stanje) {
        this.ime = ime;
        this.prezime = prezime;
        this.stanje = stanje;
    }
```

Klasa sa parametarskim konstruktorom

```
package konstruktori;

import java.util.Locale;

public class TekuciRacun {

    public String ime;
    public String prezime;
    public double stanje;

    public TekuciRacun(String ime, String prezime, double stanje) {
        this.ime = ime;
        this.prezime = prezime;
        this.stanje = stanje;
    }
    public void uplata(double iznos){

        stanje += iznos;
    }

    public void isplata(double iznos) {

        stanje -= iznos;
    }

    public void stampaj(){

        System.out.format(Locale.ENGLISH,"Korisnik %s %s ima %.2f dinara na racunu%n", ime, prezime, stanje);
    }
}
```

Poziv podrazumevanog konstruktora

```
package konstruktori;

public class Konstruktori {
    Run | Debug
    public static void main(String[] args) {
        TekuciRacun tr2 = new TekuciRacun();
    }
}
```

Poziv parametarskog konstruktora

```
public static void main(String[] args) {
    TekuciRacun tr1 = new TekuciRacun("Marko","Markovic",56789.12);
    tr1.stampaj();
}
```

```
C:\Users\goran\Desktop\java09>cd kon*

C:\Users\goran\Desktop\java09\konstruktori>javac *.java

C:\Users\goran\Desktop\java09\konstruktori>cd..

C:\Users\goran\Desktop\java09>java konstruktori.Konstruktori
Korisnik Marko Markovic ima 56789.12 dinara na racunu

C:\Users\goran\Desktop\java09>
```

Definisanje konstruktora bez parametara

```
public TekuciRacun(){

}
```

Poziv različitih konstruktora

```
public static void main(String[] args) {
    TekuciRacun tr1 = new TekuciRacun("Marko","Markovic",56789.12);
    tr1.stampaj();

    TekuciRacun tr2 = new TekuciRacun();
    tr2.ime = "Petar";
    tr2.prezime = "Petrovic";
    tr2.stanje = 34567.1;
    tr2.stampaj();
}
```

```
C:\Users\goran\Desktop\java09>cd kon*

C:\Users\goran\Desktop\java09\konstruktori>javac *.java

C:\Users\goran\Desktop\java09\konstruktori>cd..

C:\Users\goran\Desktop\java09>java konstruktori.Konstruktori
Korisnik Marko Markovic ima 56789.12 dinara na racunu
Korisnik Petar Petrovic ima 34567.10 dinara na racunu
```

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
package	pristup je ograničen na paket (Public unutar paketa, private izvan paketa)
protected	Pristup je ograničen na članove klase i klase izvedenih iz te klase

Ilustracija private prava pristupa

```
package pravaPristupa;

import java.util.Locale;

public class Radnik {
    public String ime;
    public String prezime;
    private double plata;

    public void stampaj(){

        System.out.format(Locale.ENGLISH,"%s %s Plata: %.2f%n", ime, prezime, plata);
    }
}
```

Ilustracija private prava pristupa

```
package pravaPristupa;

public class PravaPristupa {
    Run | Debug
    public static void main(String[] args) {
        Radnik r = new Radnik();
        r.ime = "Marko";
        r.prezime ="Markovic";
        r.plata = 12345.2;
    }
}
```

Ilustracija private prava pristupa

```
package pravaPristupa;

public class PravaPristupa {
    public static void main(String[] args) {
        Radnik r = new Radnik();
        r.ime = "Marko";
        r.prezime ="Markovic";
        //r.plata = 12345.2;
        r.stampaj();
    }
}
```

```
C:\Users\goran\Desktop\java09\pravaPristupa>cd..

C:\Users\goran\Desktop\java09>java pravaPristupa.PravaPristupa
Marko Markovic Plata: 0.00

C:\Users\goran\Desktop\java09>
```

Čitanje i setovanje privatnog člana unutar klase

```
package pravaPristupa;

import java.util.Locale;

public class Radnik {
    public String ime;
    public String prezime;
    private double plata;

    public double getPlata() {
        return plata;
    }

    public void setPlata(double plata) {
        this.plata = plata;
    }
    public void stampaj(){

        System.out.format(Locale.ENGLISH,"%s %s Plata: %.2f%n", ime, prezime, plata);
    }
}
```

```java
package pravaPristupa;

public class PravaPristupa {
    public static void main(String[] args) {
        Radnik r = new Radnik();
        r.ime = "Marko";
        r.prezime ="Markovic";
        //r.plata = 12345.2;
        r.setPlata(97345.21);
        r.stampaj();
    }
}
```

```
C:\Users\goran\Desktop\java09>java pravaPristupa.PravaPristupa
Marko Markovic Plata: 97345.21
```

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

```
package vrednosti;

public class Vrednosti {
    public static void main(String[] args) {
        //original
        int x = 10;

        //kopija
        int y = x;

        //menjam original
        x++;
        System.out.println("x=" + x);
        System.out.println("y=" + 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

```
package reference;

public class Osoba {
    private String ime;
    private String prezime;
    public String getIme() {
        return ime;
    }

    public void setIme(String ime) {
        this.ime = ime;
    }

    public String getPrezime() {
        return prezime;
    }

    public void setPrezime(String prezime) {
        this.prezime = prezime;
    }

    public void stampaj() {
        System.out.format("Ime: %s Prezime: %s%n", ime,prezime);
    }
}
```

Demonstracija referentnih tipova

```
package reference;
public class Reference {
    public static void main(String[] args) {

        // original
        Osoba os1 = new Osoba();
        os1.setIme("Marko");
        os1.setPrezime("Markovic");

        // kopija
        Osoba os2 = os1;
        // menjam original
        os1.setIme("Mirko");
        // stampam kopiju
        os2.stampaj();
    }
}
```

Prenos parametara po vrednosti

```
package prosledjivanje1;

public class Prosledjivanje1 {

    public static void promeni(int a) {
        a++;
    }

    public static void main(String[] args) {
        int i = 10;
        System.out.println("Pre prosledjivanja funkciji i= " + i);
        promeni(i);
        System.out.println("Posle prosledjivanja funkciji i= " + i);
    }

}
```

Prenos parametara po vrednosti

```
C:\Users\goran\Desktop\java09>java prosledjivanje1.Prosledjivanje1
Pre prosledjivanja funkciji i= 10
Posle prosledjivanja funkciji i= 10

C:\Users\goran\Desktop\java09>
```

Prenos vrednosnih tipova po referenci

```
package prosledjivanje2;

public class Omotac {
    private int i;

    public Omotac(int i){
        this.i = i;
    }
    public void setI(int i) {
        this.i = i;
    }

    public int getI() {
        return i;
    }
}
```

```
package prosledjivanje2;

public class Prosledjivanje2 {
    public static void promeni(Omotac a) {
        int staroI = a.getI();
        a.setI(staroI+1);
    }

    public static void main(String[] args) {
        Omotac o1 = new Omotac(5);
        promeni(o1);
        System.out.println(o1.getI());
    }
}
```

```
C:\Users\goran\Desktop\java09>java prosledjivanje2.Prosledjivanje2
6
```

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

```
package staticki;

public class Osoba {
    private String ime;

    private String prezime;

    public static int brojOsoba = 0;

    public Osoba(){
        brojOsoba++;
    }

    public void stampaj(){
        System.out.println(ime + " " + prezime);
    }


}
```

```
public String getIme() {
        return ime;
    }

    public String getPrezime() {
        return prezime;
    }

    public void setPrezime(String prezime) {
        this.prezime = prezime;
    }

    public void setIme(String ime) {
        this.ime = ime;
    }
```

Upotreba statičkog polja

```
package staticki;

public class Staticki {
    public static void main(String[] args) {
        Osoba.brojOsoba = 10;
        Osoba os1 = new Osoba();
        os1.setIme("Marko");
        os1.setPrezime("Markovic");
        System.out.println("Trenutni broj osoba je: " + Osoba.brojOsoba);
        Osoba os2 = new Osoba();
        Osoba os3 = new Osoba();
        System.out.println("Trenutni broj osoba je: " + Osoba.brojOsoba);
    }
}
```

Definisanje statičke metode

```
package stmetoda;

public 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

```
package stMetoda;

public class StMetoda {
    public static void main(String[] args) {
        // Upotreba nestaticke metode
        Pravougaonik pr1 = new Pravougaonik(12.3, 45.6);
        double p1 = pr1.povrsina();
        System.out.println("p1=" + p1);

        // Upotreba staticke metode
        double p2 = Pravougaonik.racunajPovrsinu(12.3, 45.6);
        System.out.println("p2=" + p2);

    }
}
```

Nasleđivanje i polimorfizam

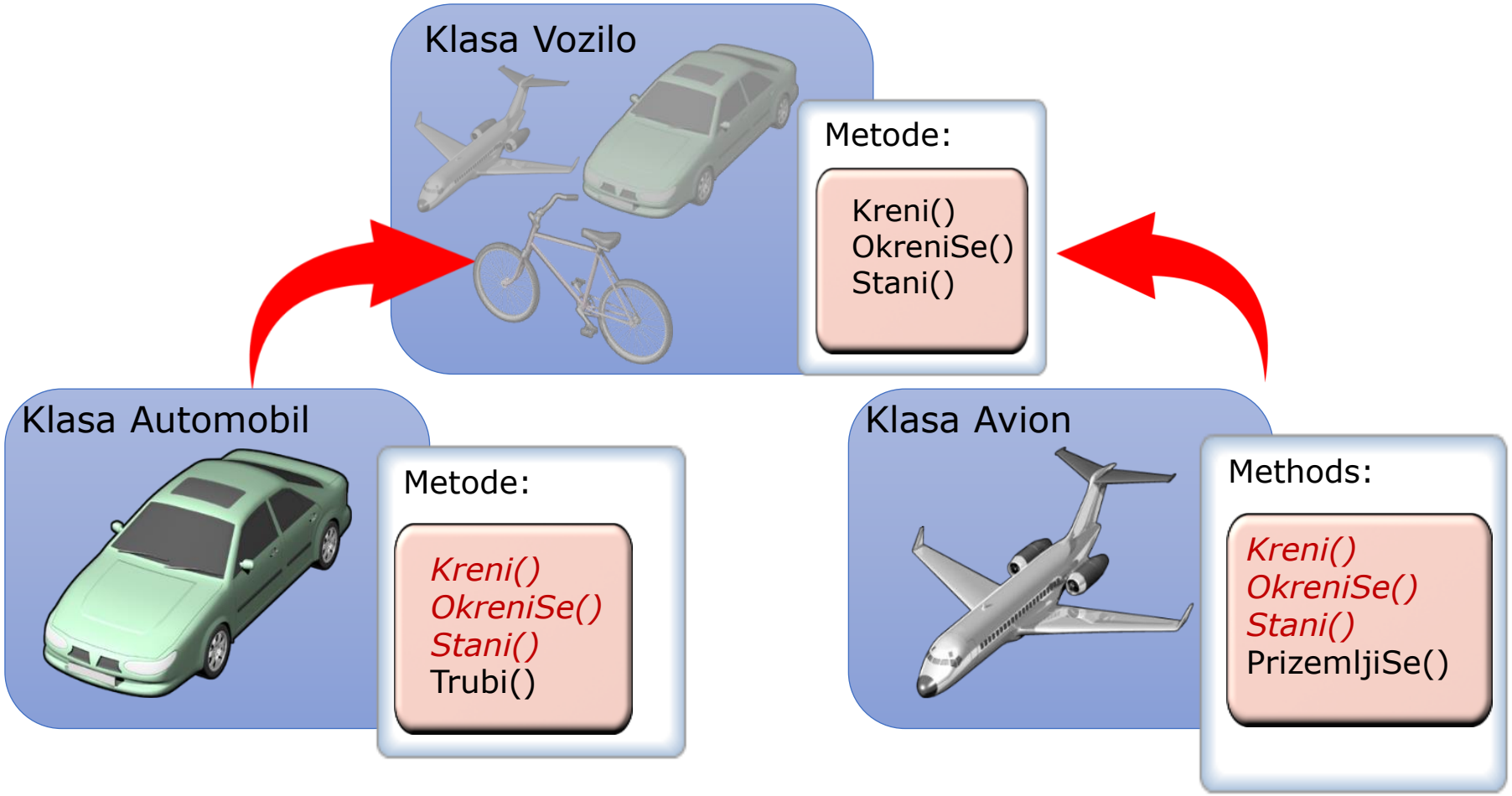
Nasleđivanje

- Nasleđivanje omogućava da se kreiraju novi tipovi podataka na osnovu već postojećih tipova
- Nasleđivanje je vrsta veze 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

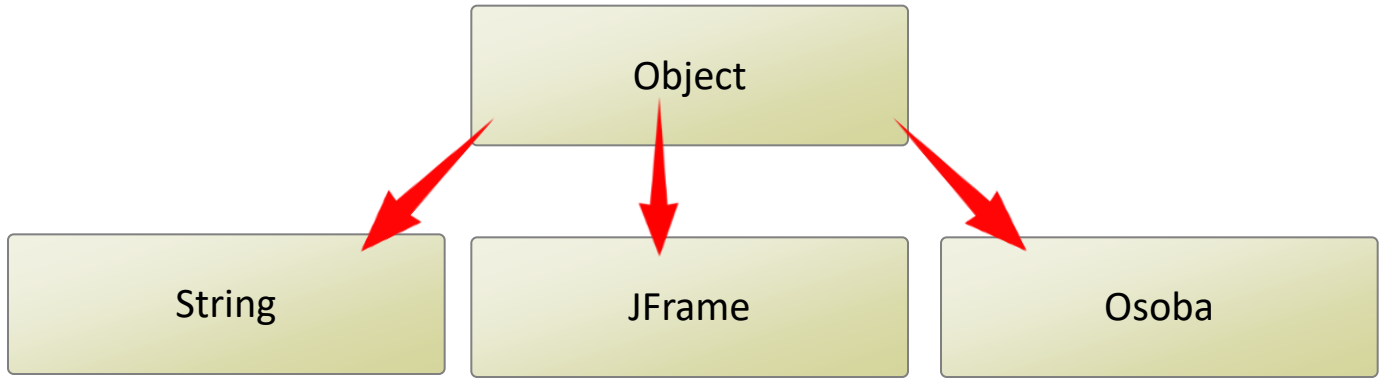
- Osnovna klasa se naziva još i natklasa, bazna klasa, roditeljska klasa (parent class), super klasa
- Izvedena klasa se naziva još klasa potomak (child class) ili podklasa
- Izvedena klasa može da nasledi samo jednu baznu klasu
- Nasleđivanje redukuje ponavljanje koda
- ***public class B extends A*** { telo klase B }
  //(klasa B izvedena iz klase A)
- ***public finall class NekaKlasa {telo klase} –*** tada se iz ovakve klase ne može vršiti nasleđivanje

Ilustracija nasleđivanja



Nasleđivanje u javi

- Klasa Object je bazna klasa za sve klase u javi



Bazna klasa

```
package nasledjivanje1;

public class Osoba {
    private String ime;
    private String prezime;

    public Osoba(){
        System.out.println("Konstruktor bazne klase bez parametara");
    }

    public Osoba(String ime, String prezime){
        this.ime = ime;
        this.prezime = prezime;
        System.out.println("Parametarski konstruktor bazne klase");
    }

    public void stampaj() {
        System.out.println("Ime: " + ime + " Prezime: "+prezime);
    }
}
```

Izvedena klasa

```
package nasledjivanje1;

public class Student extends Osoba {
    private String smer;
    public Student(String smer) {
        System.out.println("Konstruktor izvedene klase");
        this.smer =smer;
    }

    public void stampaj() {
        super.stampaj();
        System.out.println("Smer: "+ smer);
    }
}
```

Konstruktor izvedene klase

```
package nasledjivanje1;

public class Nasledjivanje1 {
    public static void main(String[] args) {
        Student st1 = new Student("Informatika");
        st1.stampaj();
    }
}
```

```
C:\Users\goran\Desktop\java09>java nasledjivanje1.Nasledjivanje1
Konstruktor bazne klase bez parametara
Konstruktor izvedene klase
Ime: null Prezime: null
Smer: Informatika
```

Modifikovana izvedena klasa

```
package nasledjivanje1;

public class Student extends Osoba {
    private String smer;
    public Student(String ime, String prezime, String smer) {
        super(ime,prezime);
        System.out.println("Konstruktor izvedene klase");
        this.smer =smer;
    }

    public void stampaj() {
        super.stampaj();
        System.out.println("Smer: "+ smer);
    }
}
```

Poziv konstruktora izvedene klase

```
package nasledjivanje1;

public class Nasledjivanje1 {
    public static void main(String[] args) {
        //Student st1 = new Student("Informatika");
        Student st1 = new Student("Marko", "Markovic", "Informatika")
        st1.stampaj();
    }
}
```

```
Parametarski konstruktor bazne klase
Konstruktor izvedene klase
Ime: Marko Prezime: Markovic
Smer: Informatika
```

Ilustracija prava pristupa

```java
package ilustracija;

public class A {
    private int a1;
    protected int a2;
    public int a3;

    public A(){
        a1=1;
        a2=1;
        a3=1;
    }

    public int getA1() {
        return a1;
    }

    public void setA1(int a1) {
        this.a1 = a1;
    }

    public void stampaj(){
        System.out.println("private: a1 = " + a1);
        System.out.println("protected: a2 = " + a1);
        System.out.println("public: a3 = " + a1);
    }
}
```

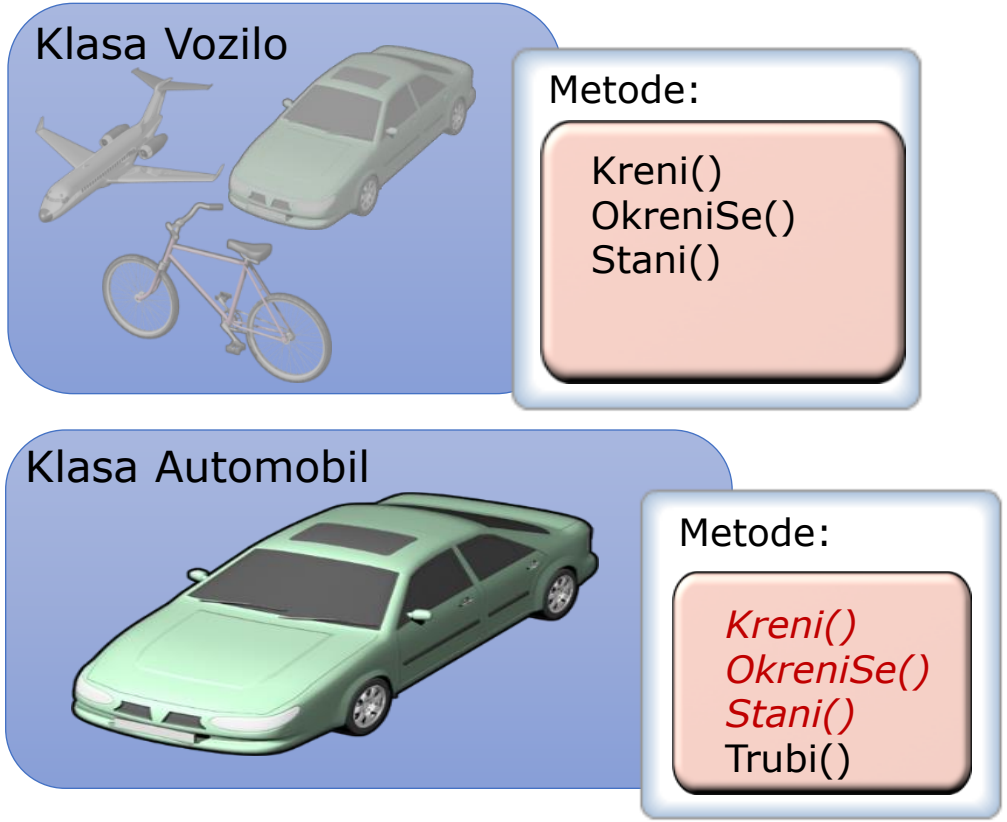
```java
package ilustracija;

public class B extends A{
    private int b;
    public B(){
        b =10;
        System.out.println("Konstruktor izvedene klase");
    }

    public void uvecaj(){
        int a1n = getA1();
        a1n++;
        setA1(a1n);
        a2++;
        a3++;
        b++;
    }

    public void stampaj() {
        super.stampaj();
        System.out.println("b= " + b);
    }
}
```

Ilustracija prava pristupa

```
package ilustracija;

public class Ilustracija {
    public static void main(String[] args) {
        B bo = new B();
        bo.stampaj();
        System.out.println("---------");
        bo.uvecaj();
        bo.stampaj();
        System.out.println("---------");
        System.out.println("a1=" +bo.getA1());
        System.out.println("a2=" +bo.a2);
        System.out.println("a3=" +bo.a3);

    }

}
```

Prvo pravilo polimorfizma

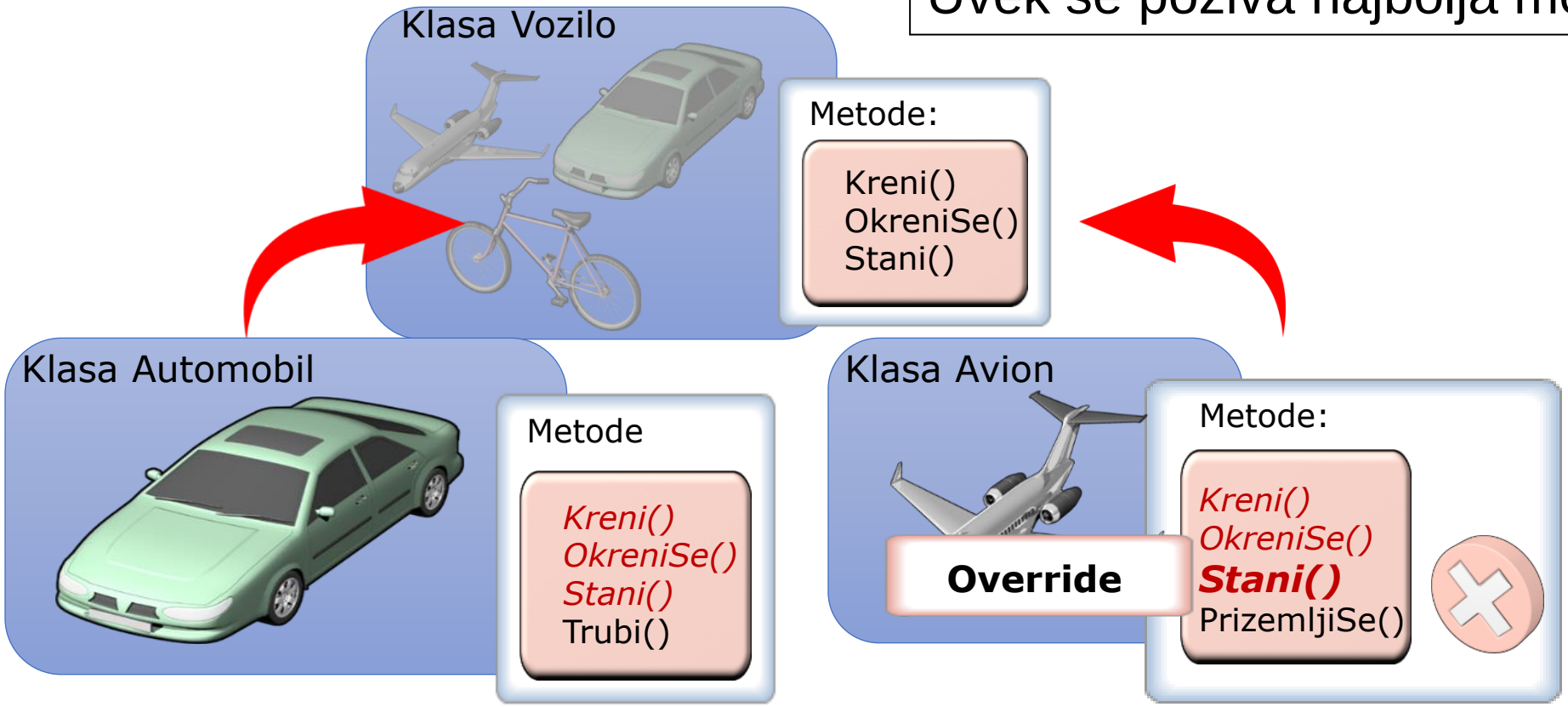
Referenca tipa bazne klase može pokazivati na objekat izvedene klase




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

Drugo pravilo polimorfizma

Uvek se poziva najbolja metoda.



Definisanje bazne klase

```
package nasledjivanje2;

public class Oblik {
    public void ispisi(){
        System.out.println("Ovo je oblik");
    }
}
```

Prebrisavanje metode u izvedenoj klasi

```
package nasledjivanje2;

public class Krug extends Oblik {
    public void ispisi() {
        System.out.println("Ovo je krug");
    }
}
```

```
package nasledjivanje2;

public class Kvadrat extends Oblik {
    public void ispisi(){
        System.out.println("Ovo je kvadrat");
    }
}
```

Ilustracija 1. i 2. pravila polimorfizma

```
package nasledjivanje2;

import java.util.Random;

public class Nasledjivanje2 {
    public static void main(String[] args) {
        Oblik nizOblika[] = new Oblik[10];
        Random rnd = new Random();

        int a = -1;

        for (int i = 0; i < 10; i++) {
            a = rnd.nextInt(2); // 0 ili 1

            if (a == 0) {
                nizOblika[i] = new Krug(); // 1.p . polimorfizma
            } else {
                nizOblika[i] = new Kvadrat();
            }
        }
        for (Oblik oblik : nizOblika) {
            oblik.ispisi(); // 2.p polimorfizma
        }
    }
}
```