Predavanje 4.

OBJEKTNO-ORIJENTISANO PROGRAMIRANJE

2025/26.

Agenda
za danas

Kompatibilnost tipova

Poziv prepisanih metoda i polimorfizam

Poziv prekrivenih metoda

Kompatibilnost tipova

AGENDA

Kompatibilnost tipova
Poziv prepisanih metoda i polimorfizam
Poziv prekrivenih metoda

Kompatibilnost tipova

- Java je strogo tipiziran jezik

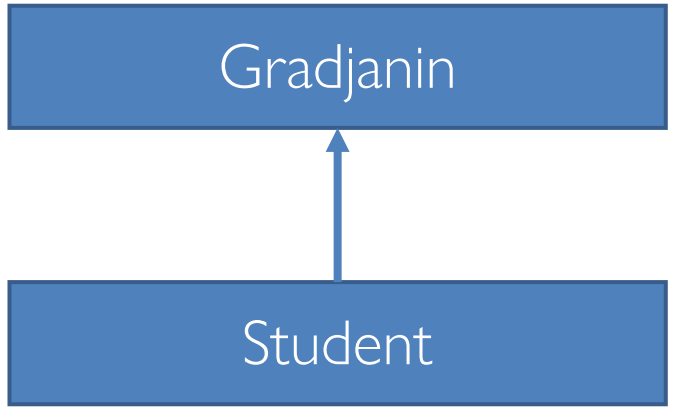
  kompatibilnost tipova se proverava tokom prevodjenja

- Pri operaciji dodele, inicijalizaciji, prenosu parametara i vraćanju rezultata metoda tip izraza mora biti kompatibilan tipu promenljive kojoj se izraz dodeljuje.
- Tip reference koja je rezultat izraza mora biti
  - Tip promenljive kojoj se dodeljuje
  - Njegov podtip
  - ili `null`

Konverzija

- Nadtipovi su manje posebni, smatraju se širim i nalaze se više u hijerahiji tipova
- Konverzija proširivanja (naviše, nagore)
  - Podtip se konvertuje u nadtip
  - Automatski
- Konverzija sužavanja (naniže, nadole)
  - Nadtip se konvertuje u podtip
  - Eksplicitno (cast)

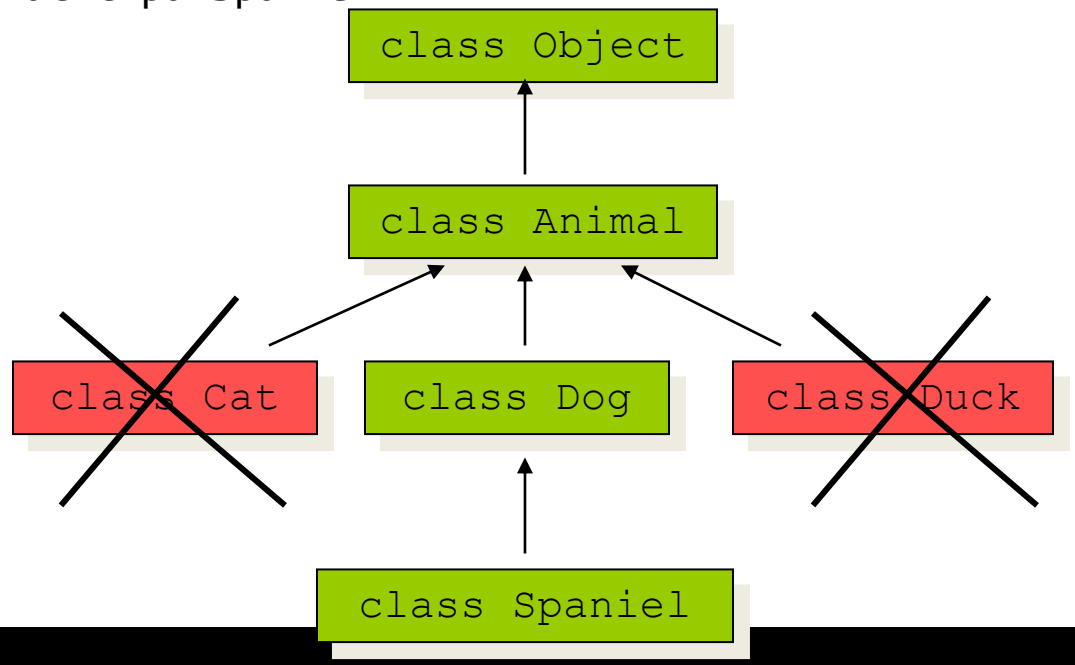
```
class Student extends Gradjanin { … }
…
Sudent s = new Sudent();
Gradjanin g = new Gradjanin();
Gradjanin g2 = s;       // naviše
Sudent s1 = (Sudent) g;  // naniže, obavezan cast
```



Konverzija sužavanja - CAST

- Cast objekata je moguć u slučaju da radimo cast izmedju klasa koje pripadaju istom lancu nasledjivanja (dakle, jedan tip mora da bude podtip drugog u bilo kom redosledu).
- Java zadržava sve informacije o originalnoj klasi kojoj objekat pripada !!!

```
Spaniel aPet = new Spaniel("Fang"); // Kreiraj objekat tipa Spaniel
Animal theAnimal = (Animal)aPet;
Animal theAnimal = aPet;
   // cast nagore - upward cast
Dog aDog = (Dog)theAnimal;
  // cast nadole
  // downward cast
```

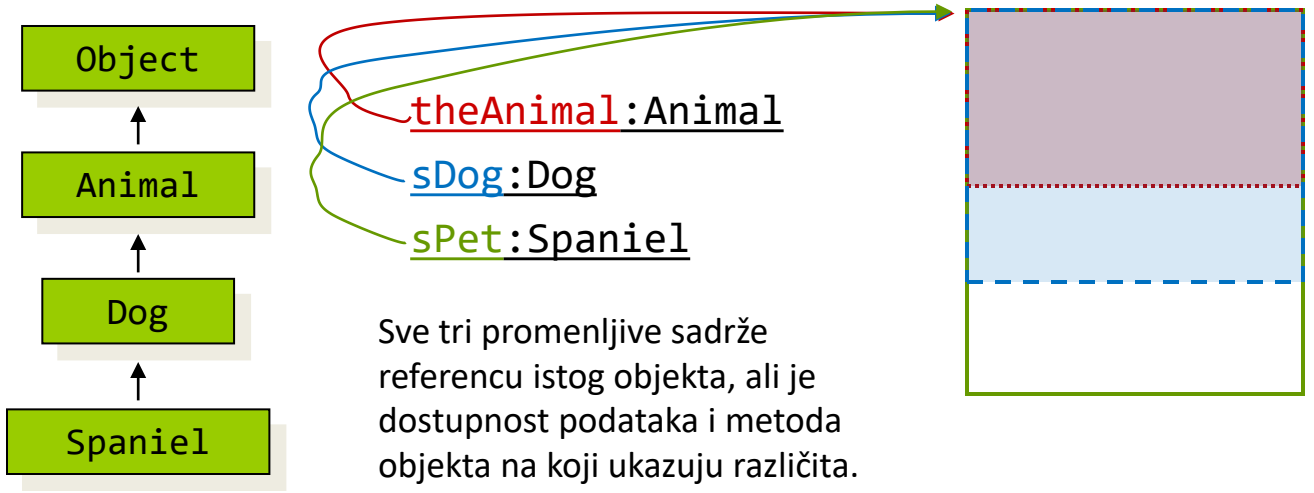


Konverzija sužavanja - CAST

```
Spaniel aPet = new Spaniel("Fang");
Animal theAnimal = aPet;
Dog aDog = (Dog)theAnimal;
```

- aDog se može koristiti **SAMO** za poziv *overrided* metoda iz klase `Spaniel`. Dakle, ako `Spaniel` ima i dodatne metode, oni neće biti dostupni sa aDog refence.
- ako je potrebno pozvati metod specifičan za Spaniel klasu koristeći referencu aDog, NEOPHODAN je cast aDog-a na Spaniel.

```
((Spaniel)aDog).specmetod();
```



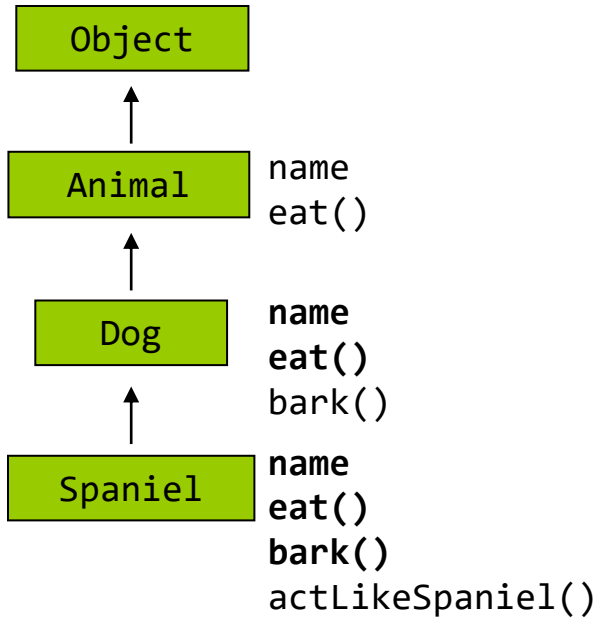
Sve tri promenljive sadrže referencu istog objekta, ali je dostupnost podataka i metoda objekta na koji ukazuju različita.

Kompatibilnost tipova

AGENDA

Kompatibilnost tipova
Poziv prepisanih metoda i polimorfizam
Poziv prekrivenih metoda

poziv prepisanih metoda

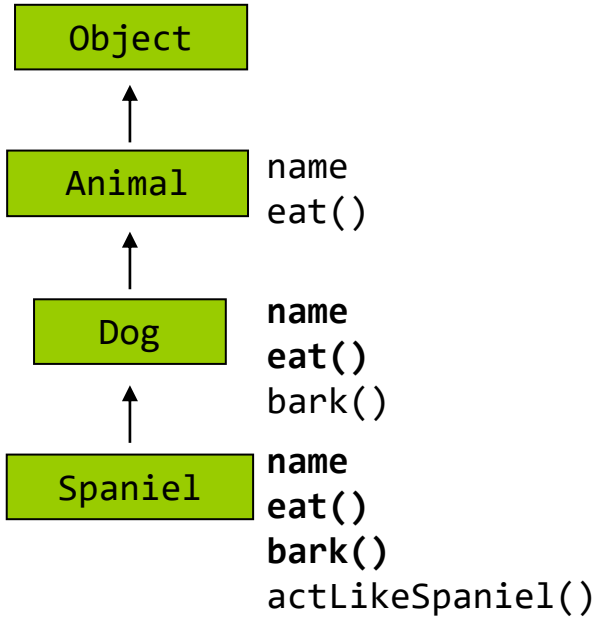



```
Spaniel aPet = new Spaniel("Fang");
Animal theAnimal = aPet;
Dog aDog = (Dog)theAnimal;

// Koji metod će biti pozvan u sledećoj liniji koda?
   Metod definisan u Animal ili Spaniel.
theAnimal.eat();

// Koji metod će biti pozvan u sledećoj liniji koda?
   Metod definisan u Animal, Dog ili Spaniel.
aDog.eat();
```

POZIV PREPISANIH METODA

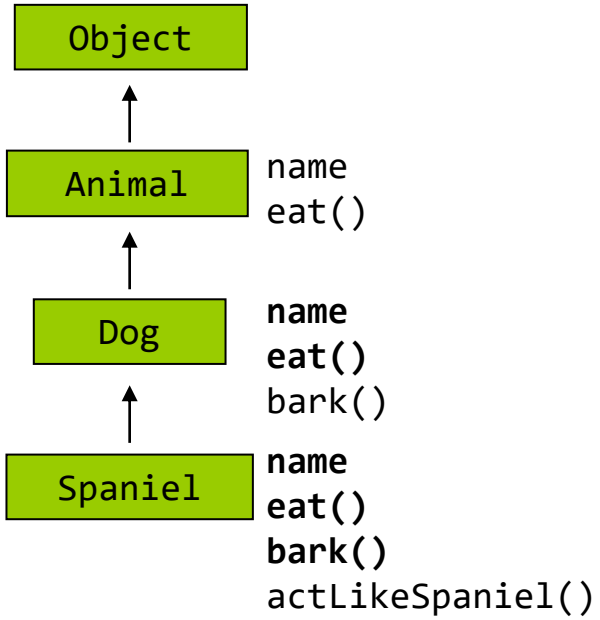



```
Spaniel aPet = new Spaniel("Fang");
Animal theAnimal = aPet;
Dog aDog = (Dog)theAnimal;

// Koji metod će biti pozvan u sledećoj liniji koda?
  Metod definisan u Animal ili Spaniel.
theAnimal.eat(); // Spaniel

// Koji metod će biti pozvan u sledećoj liniji koda?
  Metod definisan u Animal, Dog ili Spaniel.
aDog.eat(); // Spaniel
```

POZIV PREPISANIH METODA



Slanjem poruke

```
ref.eat();
```

biće pozvan metod objekta na koji ukazuje referenca `ref`.

POLIMORFIZAM

- mogućnost da varijablom određenog tipa referenciramo objekte različitih tipova i da automatski pozivamo metode koje su specifične za tip objekta na koji varijabla referencira
- uslovi
  - Poziv metoda podklase kroz varijablu bazne klase
  - Pozvana metoda mora biti i član bazne klase
  - Signatura metode i povratni tip moraju biti isti i u baznoj i u izvedenoj klasi
  - Atribut pristupa ne sme biti restriktivniji u izvedenoj klasi nego što je u baznoj klasi
  - Polimorfizam se odnosi samo na metode – reference baznog tipa mogu se koristiti samo za pristup podacima članovima baznog tipa

Kompatibilnost tipova

AGENDA

Kompatibilnost tipova
Poziv prepisanih metoda i polimorfizam
Poziv prekrivenih metoda

POZIV PREKRIVENIH (STATIČKIH) METODA

```
public class Animal {
    public static void testClassMethod() {
        System.out.println("The class" + " method in Animal.");    }
    public void testInstanceMethod() {
        System.out.println("The instance " + " method in Animal.");     }
}
public class Cat extends Animal {
    public static void testClassMethod() {
        System.out.println("The class method" + " in Cat.");     }
    public void testInstanceMethod() {
        System.out.println("The instance method" + " in Cat.");     }
}
public class Test {
    public static void main(String[] args) {
        Cat myCat = new Cat();
        Animal myAnimal = myCat;
        Animal.testClassMethod();    // stampa: The class method in Animal.
        myAnimal.testInstanceMethod();      // stampa: The instance method in Cat.    }}
```

POZIV PREKRIVENIH (STATIČKIH) METODA

- Dakle, polimorfizma nema kod statičkih metoda. Zašto?

Nadklasa Podklasa	Metod objekta	`static` metod
Metod objekta	prepisuje	compile-time greška
`static` metod	compile-time greška	sakriva