

Predavanje 4.

OBJEKTNO-ORIJENTISANO PROGRAMIRANJE

2024/25.

Agenda

za danas

Nasleđivanje

Konstrukcija objekata

Nasleđivanje i istoimeni podaci članovi

Nasleđivanje i istoimeni nestatički metodi

Nasleđivanje i istoimeni statički metodi

Final klase, podaci i metodi

Nasleđivanje

AGENDA

Nasleđivanje

Konstrukcija objekata

Nasleđivanje i istoimeni podaci članovi

Nasleđivanje i istoimeni nestatički metodi

Nasleđivanje i istoimeni statički metodi

Final klase, podaci i metodi

Nasleđivanje

Sredstvo za realizaciju *code reuse* ideje.

- Nasleđivanje (*inheritance*)

Mehanizam koji omogućava uvođenje novih tipova/klasa proširivanjem osobina i ponašanja postojećih tipova/klasa.

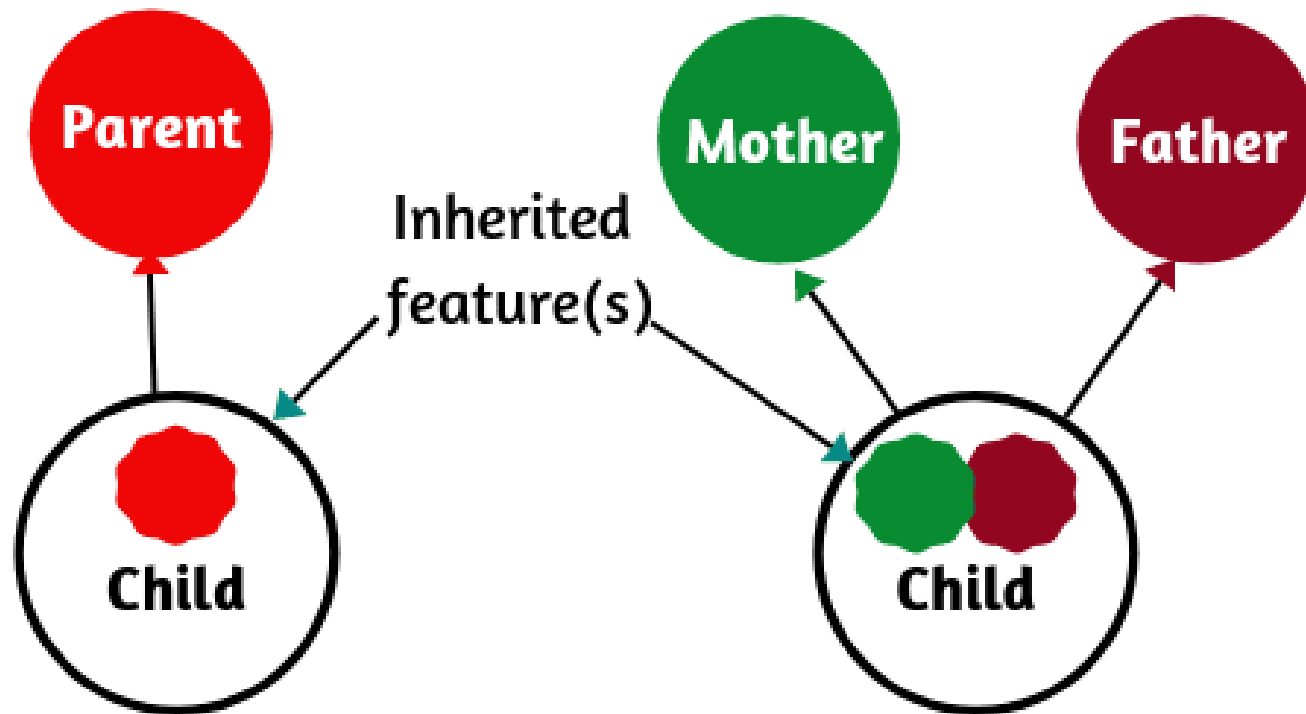
- Klasa koja nasleđuje (proširuje) se naziva

IZVEDENA, POTKLASA ili DETE-KLASA i
predstavlja UŽI tip podatka u odnosu na klasu iz koje je izvedena.

- Klasa koja biva proširena, tj. čije osobine i ponašanja nasleđuje dete-klasa se naziva

SUPERKLASOM, NATKLASOM ili RODITELJSKOM KLASOM i
predstavlja ŠIRI tip podatka u odnosu na klasu iz koje je izvedena.

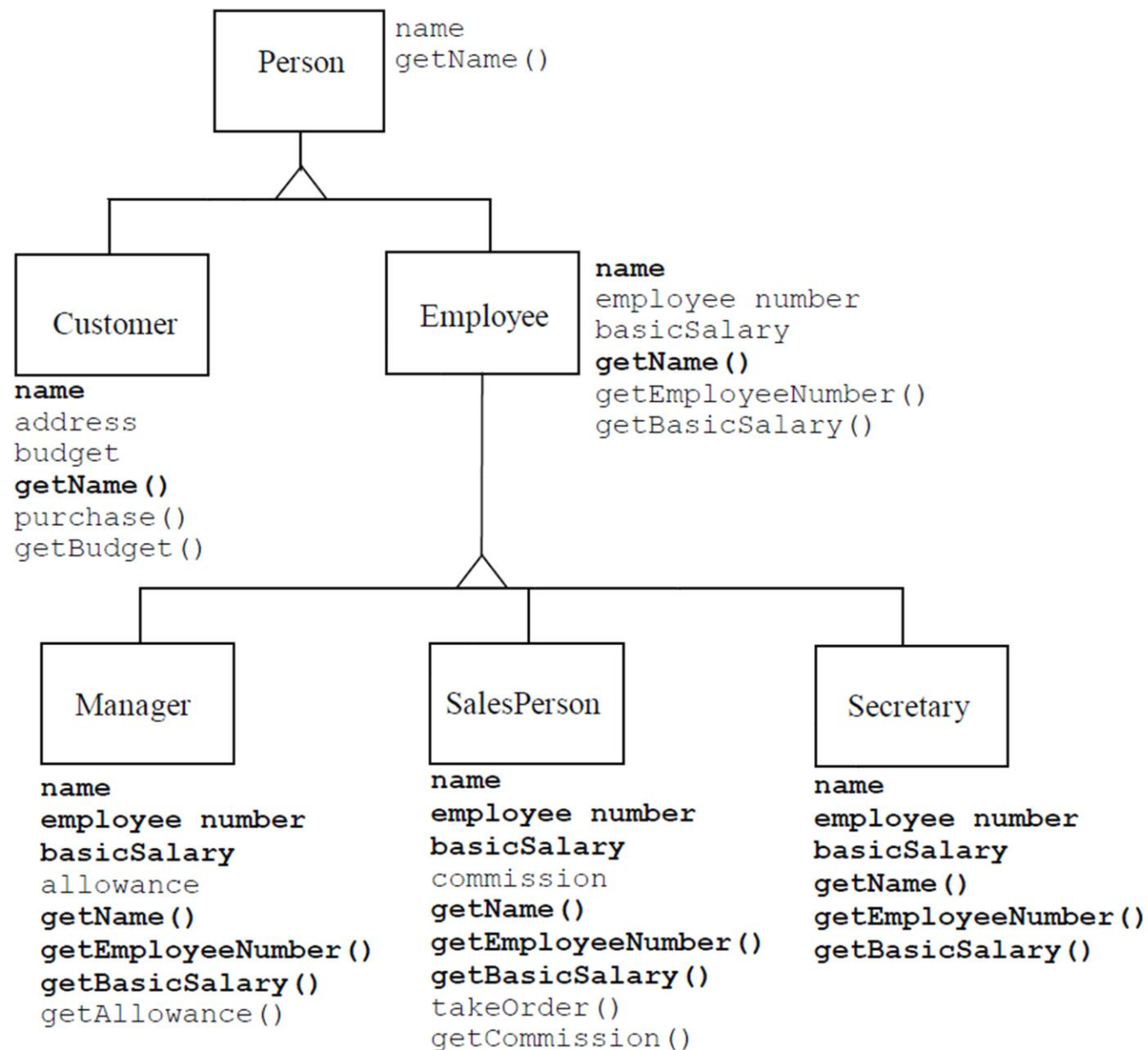
Jednostruko vs Višetruko nasleđivanje



Nasleđivanje u Javi

- Java podržava jednostruko nasleđivanje - svaka klasa može da ima **samo jednu natklasu..**
- Svaka klasa može da ima neograničen broj potklasa.
- Potklase
 - Nasleđuju promenljive, konstruktore* i metode od svoje roditeljske klase.
 - Mogu imati dodatne promenljive i metode, predefinisati nasleđene metode, prekriti nasleđene podatke.

Nasleđivanje u Javi



Nasleđivanje u Javi

- U deklaraciji potklase navode se razlike između nje i njene super klase.

```
class Gradjanin
{
    public String ime;
    public void jaSam() { System.out.println("gradjanin"); }
}
class Student extends Gradjanin {
    public String fakultet;
    public void jaSam() { System.out.println("student"); }
    public void studiram() {
        System.out.println("Studiram " + fakultet);
    }
}
```

Nasleđivanje u Javi

Java:

```
class B extends A {  
    //telo klase B  
}
```

Kažemo da klasa B nasleđuje klasu A, ako:

- su objekti klase B jedna vrsta objekata klase A, odnosno
- objekti klase B imaju sve osobine A (i još neke sebi svojstvene).

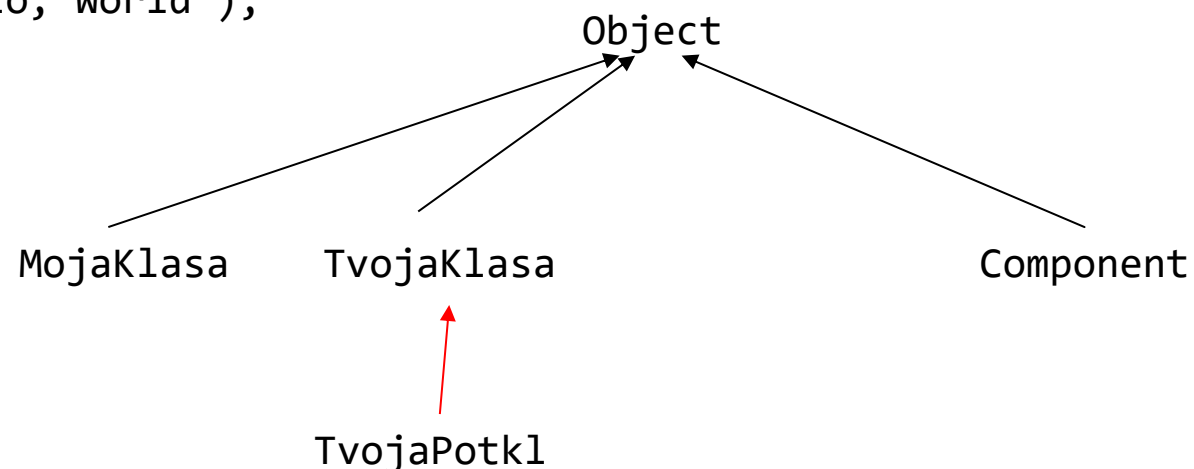
Dakle, objekti klase B su vrsta objekata klase A, a za vezu klase B sa klasom A kažemo da je tipa *a-kind-of*.

Nasleđivanje u Javi

- Sve klase (i systemske i naše) u Javi su direktno ili indirektno izvedene iz klase **Object**. Ako se eksplicitno ne navede natklasa neke klase, onda je ona implicitno izvedena iz **java.lang.Object**.

Npr. `HelloWorld` se, zapravo, prevodi kao:

```
class HelloWorld extends Object {  
    public static void main(String[] args) {  
        System.out.println("Hello, World");  
    }  
}
```



Object klasa

- Svaka klasa definisana u Javi direktno ili indirektno nasleđuje klasu **Object**

- Klase koje ne koriste **extends** klauzulu

Implicitno proširuju klasu **Object**

- klasa **Object** se ne specificira eksplicitno

```
class HelloWorld extends Object {  
    public static void main(String[] args) {  
        System.out.println("Hello, World");  
    }  
}
```

- Metodi klase **Object**

- `protected Object clone()`
- `boolean equals(Object obj)`
- `protected void finalize()`
- `Class<?> getClass()`
- `int hashCode()`
- `void notify()`
- `void notifyAll()`
- `String toString()`
- `void wait()` sa još dve overloaded varijante

Konstrukcija objekata

AGENDA

Nasleđivanje

Konstrukcija objekata

Nasleđivanje i istoimeni nestatički metodi

Nasleđivanje i istoimeni statički metodi

Final klase, podaci i metodi

Konstrukcija objekata

- Kada se kreira objekat, njegovi atributi se postavljaju na podrazumevanu vrednost:
Nula (numerički), false (boolean), null (referenca)
- Nakon inicijalizacije se poziva konstruktor i to tako što se prvo izvršava konstruktor super klase, pa zatim telo konstruktora klase kojoj objekat koji se kreira pripada.
- Izvedena klasa **mora da pozove jedan od konstruktora svoje super klase.**
- Eksplicitan poziv konstruktora super klase:
super(argumenti)
 - ako se konstruktor eksplicitno poziva, taj poziv **mora** biti prva komanda u konstruktoru izvedene klase
 - ako se ne poziva eksplicitno, implicitno će biti pozvan default konstruktor natklase.
 - super() – eksplicitan poziv default konstruktora natklase

super()

```
class A {
    int aData;
    A() { // super();
        System.out.println("Pozvao A()");
    }
}

class B extends A {
    B() { // super();
        System.out.println("Pozvao B()");
    }
}
```

```
class C extends B {
    int cData;
    // C() { super(); }
    C() { // super();
        System.out.println("Pozvao B()");
    }
}

class Test{
    public static void main(String[]
args){
        C c = new C();
        System.out.println(c);
    }
}
```

super(arg)

```
class A {  
    int aData;  
    A(int a) { aData = a;}  
}  
  
class B extends A {  
    B() {  
        super(10);  
        System.out.println("Pozvao B()");  
    }  
}
```

```
class C extends B {  
    int cData;  
    public String toString() { return ""+aData; }  
}  
  
Class Test{  
    public static void main(String[] args){  
        C c = new C();  
        System.out.println(c);  
    }  
}
```

Nasleđivanje i istoimeni podaci članovi

AGENDA

Nasleđivanje

Konstrukcija objekata

Nasleđivanje i istoimeni podaci članovi

Nasleđivanje i istoimeni nestatički metodi

Nasleđivanje i istoimeni statički metodi

Final klase, podaci i metodi

Nasleđivanje i istoimeni podaci članovi

- U dete klasi je moguće definisati podatak član koji ima isto ime i tip kao i podatak nasleđen od roditelja.
- Za pristup originalnom podatku nasleđenom iz osnovne klase se koristi **super**.
- Istoimeni podaci mogu biti i statički i nestatički..

```
class Roditelj {  
    String prezime;  
}
```

```
class Dete {  
    String prezime;  
    void dodajrezime(String novo)  
    {  
        prezime = novo + super.prezime;  
    }  
}
```

```
class Roditelj:  
    prezime
```

```
class Dete:  
    this.prezime  
    super.prezime
```

Svaki objekat deteta sadrži dve promenljive nazvane prezime. Sopstveno (this.prezime) i nasleđeno (super.prezime).

Nasleđivanje i istoimeni nestatički metodi

AGENDA

Nasleđivanje

Konstrukcija objekata

Nasleđivanje i istoimeni podaci članovi

Nasleđivanje i istoimeni nestatički metodi

Nasleđivanje i istoimeni statički metodi

Final klase, podaci i metodi

PREPISIVANJE

U Javi važi sledeće:

Ukoliko se u potklasi definiše **nestatička** metoda sa istim imenom, povratnom vrednošću i argumentima kao i **nestatička** metoda superklase tada se ovom metodom **prepisuje (override)** metoda superklase.

Radnik

```
String ime;  
float koef;  
float getPlata()
```

Sekretarica

```
String[] jezici;
```

Menadzer

```
float bonus;  
float getPlata()
```

PREPISIVANJE

Radnik

```
String ime;  
float koef;  
float getPlata() {return koef*2500;}
```

Objekti tipa Radnik računaju
platu ovako



Menadzer

```
float bonus;  
float getPlata () {return koef*2500 + bonus;}
```

Objekti tipa Menadzer računaju
platu ovako



Pristup originalnim metodama

Za pristup originalnim metodama osnovne klase unutar dete klase se koristi **super**

```
float getPlata(){  
    return super.getPlata()+bonus;  
}
```

getPlata u klasi **Menadzer** – metod u kome se računa plata tako što se osnovna plata, koja se računa kao i za svakog radnika (onako kako je računa metod **getPlata** u klasi **Radnik**), uveća za bonus.

```
class Radnik:  
    getPlata()
```

```
class Menadzer:  
    this.getPlata()  
    super.getPlata()
```

Nasleđivanje i istoimeni statički metodi

AGENDA

Nasleđivanje

Konstrukcija objekata

Nasleđivanje i istoimeni podaci članovi

Nasleđivanje i istoimeni nestatički metodi

Nasleđivanje i istoimeni statički metodi

Final klase, podaci i metodi

PREKRIVANJE METODA

U Javi važi sledeće:

Ukoliko se u potklasi definiše **statička** metoda sa istim imenom, povratnom vrednošću i argumentima kao i **statička** metoda superklase tada se ovom metodom **prekriva** metoda superklase.

```
class Roditelj {  
    public static void staticMethod() {System.out.println("Roditelj:staticMethod()");}  
}  
class Dete extends Roditelj {  
    public static void staticMethod() {System.out.println("Dete:staticMethod()");}  
}  
class Main {  
    public static void main(String[] args) {  
        Roditelj r = new Roditelj();  
        Dete d = new Dete();  
        r.staticMethod(); // ili Roditelj.staticMethod();  
        d.staticMethod(); // ili Dete.staticMethod();  
    }  
}
```

IZLAZ Roditelj:staticMethod()
 Dete:staticMethod()

Koja je razlika između prekrivanja i prepisivanja razumećete kada objasnimo pojam polimorfizma.

Nasleđivanje i istoimeni statički metodi

AGENDA

Nasleđivanje

Konstrukcija objekata

Nasleđivanje i istoimeni podaci članovi

Nasleđivanje i istoimeni nestatički metodi

Nasleđivanje i istoimeni statički metodi

Final klase, podaci i metodi

Primena final modifikatora

final modifikator može biti primenjen na klase, varijable i metode. Uopšteno, povlači za sobom tvrdnju da je definicija elementa na koji se primenjuje konačna.

- **final varijabla** – varijabla predstavlja konstantu i ne menja vrednost
Inicijalizacija ne mora da bude obavljena na mestu gde je konstanta deklarirana, ali je vrednost moguće samo jednom postaviti.
- **final klasa** – definicija tipa je konačna, pa klasa ne može biti proširena, tj. ne može biti nasleđena,
- **final metod** – ne može biti prepisan

Final može da bude i argument metoda i tada se ta varijabla ne može menjati unutar metoda

Javina `String` klasa je final tipa.

Primena final modifikatora

```
class Calculator {  
    final int dime = 10;  
    int count = 0;  
    Calculator (int i) { count = i; }  
    void Inc(final int i) {  
        i++; // illegal  
    }  
}  
class RunCalculator {  
    public static void main(String[] args) {  
        final Calculator calc = new Calculator(1);  
        calc = new Calculator(2); // illegal  
        calc.count = 2;  
        calc.dime = 11; // illegal  
        System.out.println("dime: " + calc.dime);  
    }  
}
```