NASLEĐIVANJE

2016/17

PRIRODNO-MATEMATIČKI FAKULTET, UNIVERZITET U KRAGUJEVCU

NASLEĐIVANJE

Sredstvo za realizaciju ***code reuse*** ideje.

- **Nasleđivanje (*inheritance*)**

  mogućnost uvođenja novih tipova/klasa proširivanjem osobina i ponašanja postojećih tipova/klasa.

- Klasa koja nasleđuje (proširuje) se naziva

  IZVEDENA, **PODKLASA** 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, **NADKLASOM** ili **RODITELJSKOM KLASOM** i

  predstavlja širi tip podatka u odnosu na klasu iz koje je izvedena.

NASLEĐIVANJE

- Svaka klasa može da ima neograničen broj podklasa.
- Podklase nisu ograničene na promenljive, konstruktore i metode klase koje nasleđuju od svoje roditeljske klase.
- Podklase mogu dodati i neke druge promenljive i metode ili predefinisati stare metode.
- U deklaraciji podklase navode se razlike između nje i njene superklase.

Java:

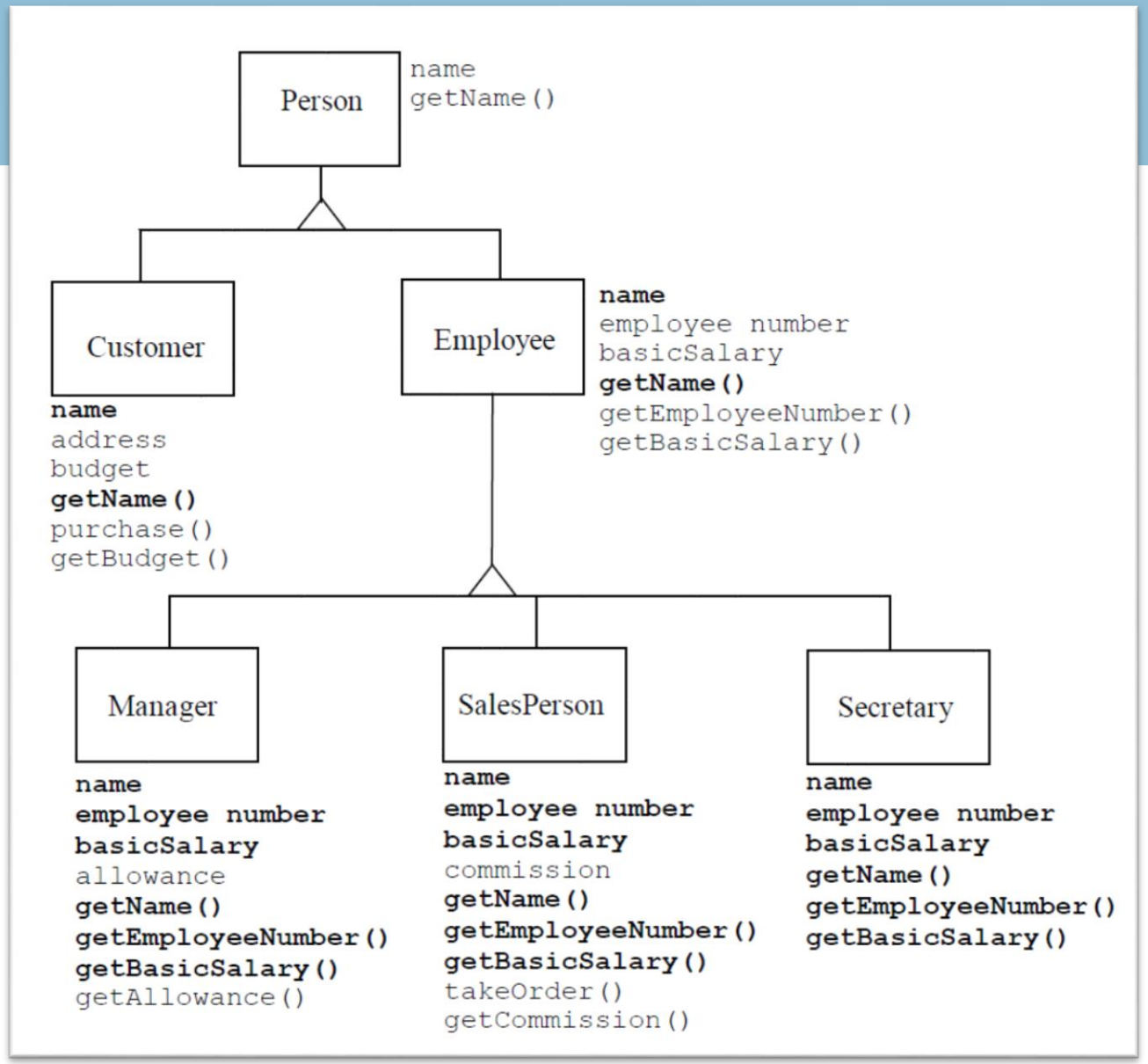
```
class B extends A {
        //telo klase B
      }
```

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*** **.**

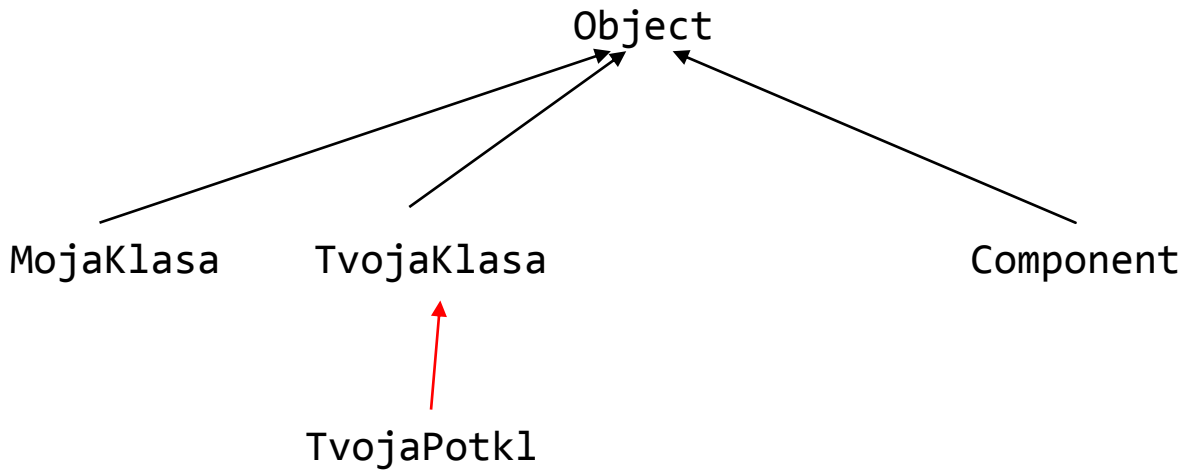
PRIMER HIJERARIJE



JEDNOSTRUKO NASLEĐIVAJNJE

- U Javi klasa može da ima **samo jednu nadklasu**.
- Sve klase (i sistemske i naše) u Javi su direktno ili indirektno izvedene iz klase **Object**. Ako se eksplicitno ne navede nadklasa 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 jos dve oveloaded varijante

KONSTRUKCIJA OBJEKATA

2016/17

PRIRODNO-MATEMATIČKI FAKULTET, UNIVERZITET U KRAGUJEVCU

KONSTRUKCIJA OBJEKATA

- Kada se kreira objekat, njegovi atributi se postavljaju na podrazumevanu vrednost:
  Nula (numerički), false (boolean), null (referenca)

- Nakon inicijalizacije se poziva konstrukor i to tako što se prvo izvršava kostruktor 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 ekplicitno poziva, taj poziv **mora** biti prva komanda u konstruktoru izvedene klase
  - ako se ne poziva eksplicitno, implicitno će biti pozvan default konstruktor nadklase.
  - super() – eksplicitan poziv default konstruktora nadklase

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(); }
   public String toString() {
   return ""+aData;
   }
}
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);
   }
}
```

THIS()

- `this(<lista argumenata>)` poziva odgovarajući vlastiti konstruktor

```
class Robot {
   int rbr;
   Robot(){
     this(1);
   }
   Robot(int rbr){
     this.rbr=rbr;
   }
}
```

- Ako postoji, `this(<lista argumenata>)`
  - Mora biti prva naredba u konstruktoru
  - Tada ne može postojati i eksplicitan poziv `super(...)`

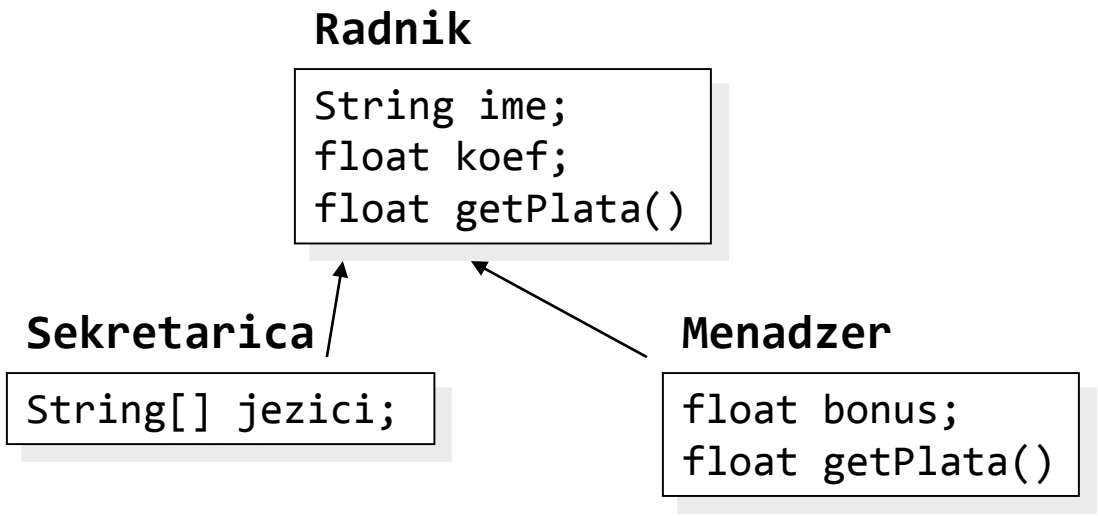
PREPISIVANJE

2016/17

PRIRODNO-MATEMATIČKI FAKULTET, UNIVERZITET U KRAGUJEVCU

OVERRIDING

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



PRISTUP ORIGINALNIM METODAMA

- Za pristup originalnim metodama osnovne klase se koristi **super**

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

Metod **getPlata** u klasi **Menadzer** 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.

ISTOIMENI PODACI ČLANOVI OBJEKTA

```
superclass
double value;
```

```
subclass
double value;
```

```
{
  value=10.25;
  super.value=11.25;
}
```

FINAL PODACI I METODI

2016/17

PRIRODNO-MATEMATIČKI FAKULTET, UNIVERZITET U KRAGUJEVCU

FINAL

- **`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 deklarisana, 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 (overrided)
- Final može da bude i argument metoda i tada se ta varijabla ne može menjati unutar metoda

Javina `String` klasa je final tipa.

FINAL

```
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);}
}
```

PRIMER

2016/17

PRIRODNO-MATEMATIČKI FAKULTET, UNIVERZITET U KRAGUJEVCU

Zaposleni koji sadrži:

- Privatni atribut imeRadnika (String).
- Javni atribut plata koji predstavlja realan broj.
- Privatni atribut bonus koji predstavlja indicator da li zaposleni ima bonus ili ne. Ovaj indicator ima vrednost TRUE ako zaposleni ima bonus a FALSE ako nema.
- Konstruktor koji prima ime radnika i bonus.
- Javnu metodu izracunajPlatu koja ne vraća ništa, ali ima ulazni argument koji predstavlja broj sati koji je zaposleni radio. Vrednost plate se postavlja po formuli broj_sati*osnovni_koeficijent, gde osnovni_koeficijent iznosi 200din.
- getter-e i setter-e za sve promenljive.
- Metod toString koji ispisuje podatke o zaposlenom u obliku imeRadnika:Tip

KancelarijskiRadnik je Zaposleni koji sadrži:

- Privatni atribut straniJezik (String).
- Konstruktor koji prima ime radnika i strani jezik i zna se da kancelarijski radnik nema bonus.
- Javnu metodu izracunajPlatu koja ne vraća ništa, ali ima ulazni argument koji predstavlja broj sati
- koji je zaposleni radio. Vrednost plate se računa tako što se na osnovnu platu koja mu pripada kao
- zaposlenom dodaje još broj_sati*koeficijent_kancelarijskog_radnika, gde
- koeficijent_kancelarijskog_radnika iznosi 120din.
- Metod toString koji ispisuje podatke o kancelarijskom radniku u obliku imeRanika: (plata, nema bonus)

Tip **Menadzer** je Zaposleni koji sadrži:

- Privatni atribut teren koji predstavlja indicator da li menadzer izlazi na teren ili ne. Ovaj indicator ima vrednost TRUE ako izlazi na teren FALSE ako ne izlazi.
- Konstruktor koji prima ime radnika i informaciju o terenu , kao i informacija o bonusu.
- Javnu metodu izracunajPlatu koja ne vraća ništa, ali ima ulazni argument koji predstavlja broj sati koji je zaposleni radio. Vrednost plate se računa tako što se na osnovnu platu koja mu pripada kao zaposlenom dodaje još broj_sati*koeficijent_menadzera, gde koeficijent_menadzera iznosi 150din. Ukoliko menadzer ima bonus na platu se dodaje 1000din a ukoliko ide na teren dodaje se još 500din.
- Metod toString koji ispisuje podatke o kancelarijskom radniku u obliku imeRanika: (plata, informacija o bonusu), gde informacija o bonusu podrazumeva ispisano DA ili NE.

Napraviti **Testnu** klasu i u njoj:

- Kreirati jedan objekat klase KancelarijskiRadnik koji se zove Marko koji zna Nemacki jezik.
- Zatim kreirati drugi objekat klase Menadzer koji se zove Nikola, ne ide na teren i nema bonus.
- Ukoliko je Marko radio 20 sati a Nikola 15, odštamati podatke o onom zaposlenom koji ima veću platu.
- Zatim kreirati niz od 4 zaposlena o Kancelarijski radnik Misa koji zna Francuski o Menadzer Sale koji ne ide na teren ali ima bonus o Kancelarijski radnik Mimi koji zna Engleski o Menadzer Miki koji ide na teren i nema bonus
- Izračunati plate svim zaposlenima ako su svi radili po 30 sati i ispisati podatke o svim zaposlenima.
- Ispisati koliko je novca ukupno isplaćeno za plate zaposlenima.
- Ispisati koliko je novca isplaceno za plate za kancelarijske radnike a koliko za menadzere.