

[Programiranje]



Nastava: prof.dr.sc. Dražena Gašpar

Datum: 23.05.2017.

Nasljeđivanje

A horizontal line with a gradient from light green to light yellow. On the left side, there is a large black left square bracket. On the right side, there is a large yellow right square bracket.

Tri metafore nasljeđivanja:

- ***biološka***: dijete nasljeđuje gene od oba svoja roditelja (dijeli neke crte fizičke/karakterne i sposobnosti roditelja)
- ***biološka***: kreiranje globalne strukture na osnovu relacije “je vrsta od” (taksonomija)
- ***ekonomska***: srodnici nasljeđuju resurse svojih roditelja.

Nasljeđivanje

A decorative graphic consisting of a horizontal line with a gradient from light green to light yellow. On the left side of the line is a large black left square bracket, and on the right side is a large yellow right square bracket.

“nadređeno-podređena relacije između klasa u kojoj podređena (dijete) klasa ima isto ponašanje (odgovornost) kao nadređena (roditelj) klasa plus najmanje još jedno dodatno“

West, D. „Object thinking“, Microsoft Press, USA, 2004

Nasljeđivanje

Nasljeđivanje je mehanizam pomoću kojega se osigurava da jedna klasa (proširena [engl. extended] ili podklasa [engl. subclass]) automatski preuzima (posjeduje) sve osobine i ponašanja, osim onih definiranih kao privatni (engl. private), od druge klase (nadklase [engl. superclass] ili „roditelj“ klase [engl. parent class]).

Nasljeđivanje omogućuje pravljenje opće klase koja definira zajedničke osobine i ponašanja za skup srodnih klasa.

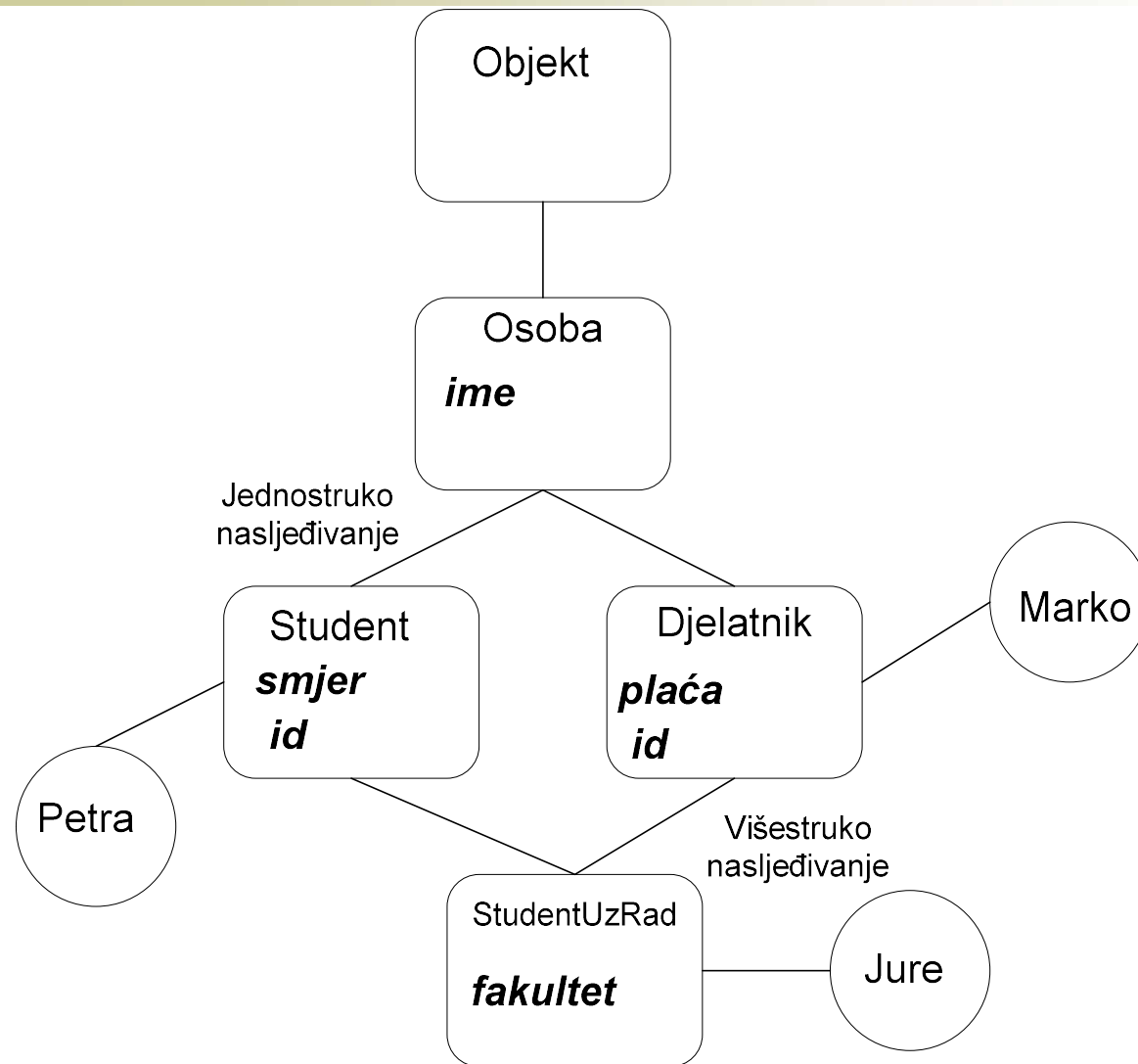
Nasljeđivanje

A horizontal line with a gradient from olive green to light yellow. A large black left square bracket is on the left, and a large yellow right square bracket is on the right.

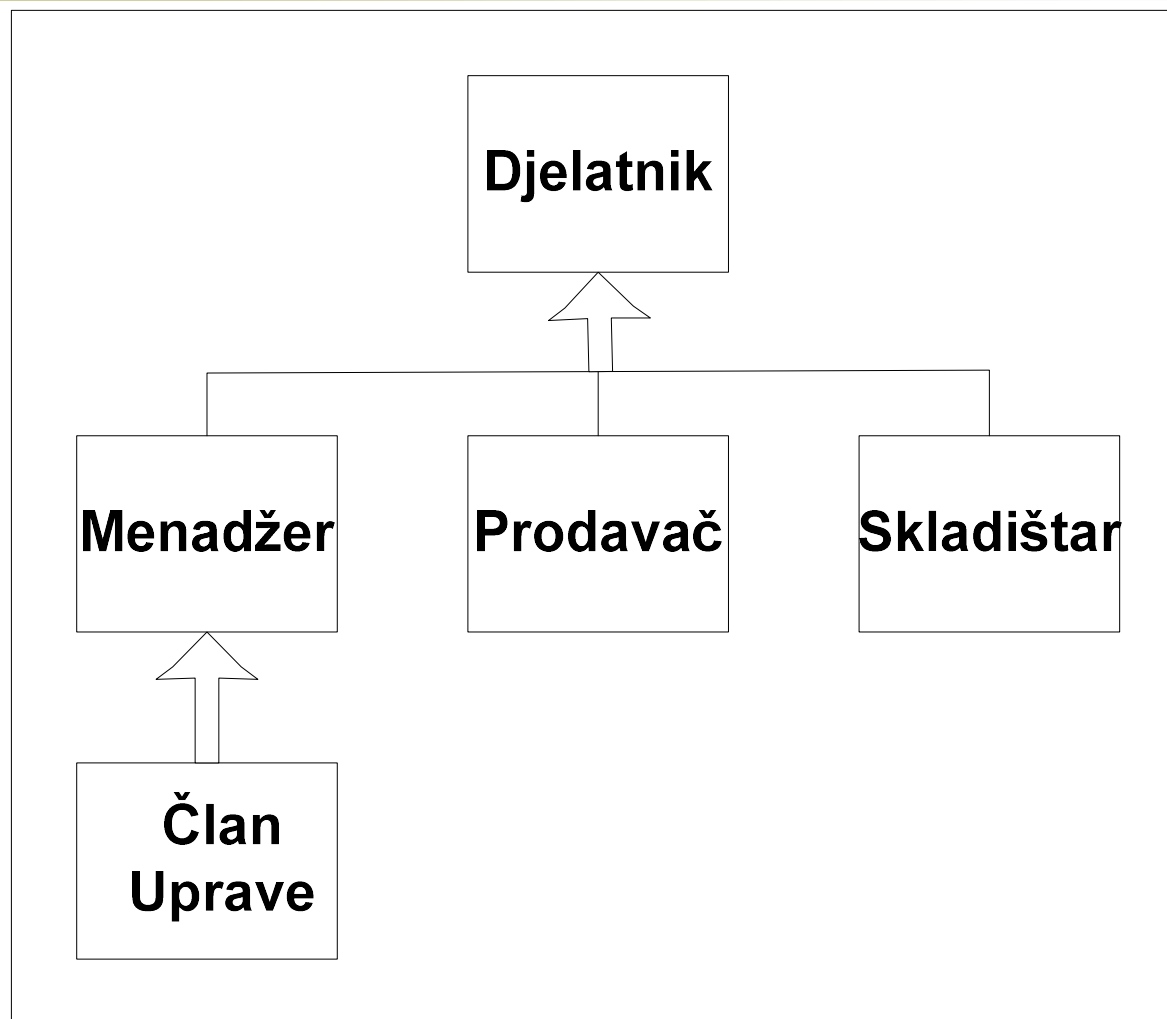
2 tipa nasljeđivanja:

- Jednostruko
- Višestruko

Nasljeđivanje



Nasljeđivanje



Nasljeđivanje

- omogućava izradu hijerarhija klasa
- omogućava stvaranje jedne općenite klase koja definira neka od osnovnih ponašanja objekta, a konkretni objekti mogu nasljeđivati njena svojstva i nadopunjavati ih
- NADKLASA
- PODKLASA

```
class imepodklase extends imenadklase {  
// tijelo klase }
```

Nasljeđivanje

A horizontal line with a gradient from light green to yellow. On the left side, there is a large black left square bracket. On the right side, there is a large yellow right square bracket.

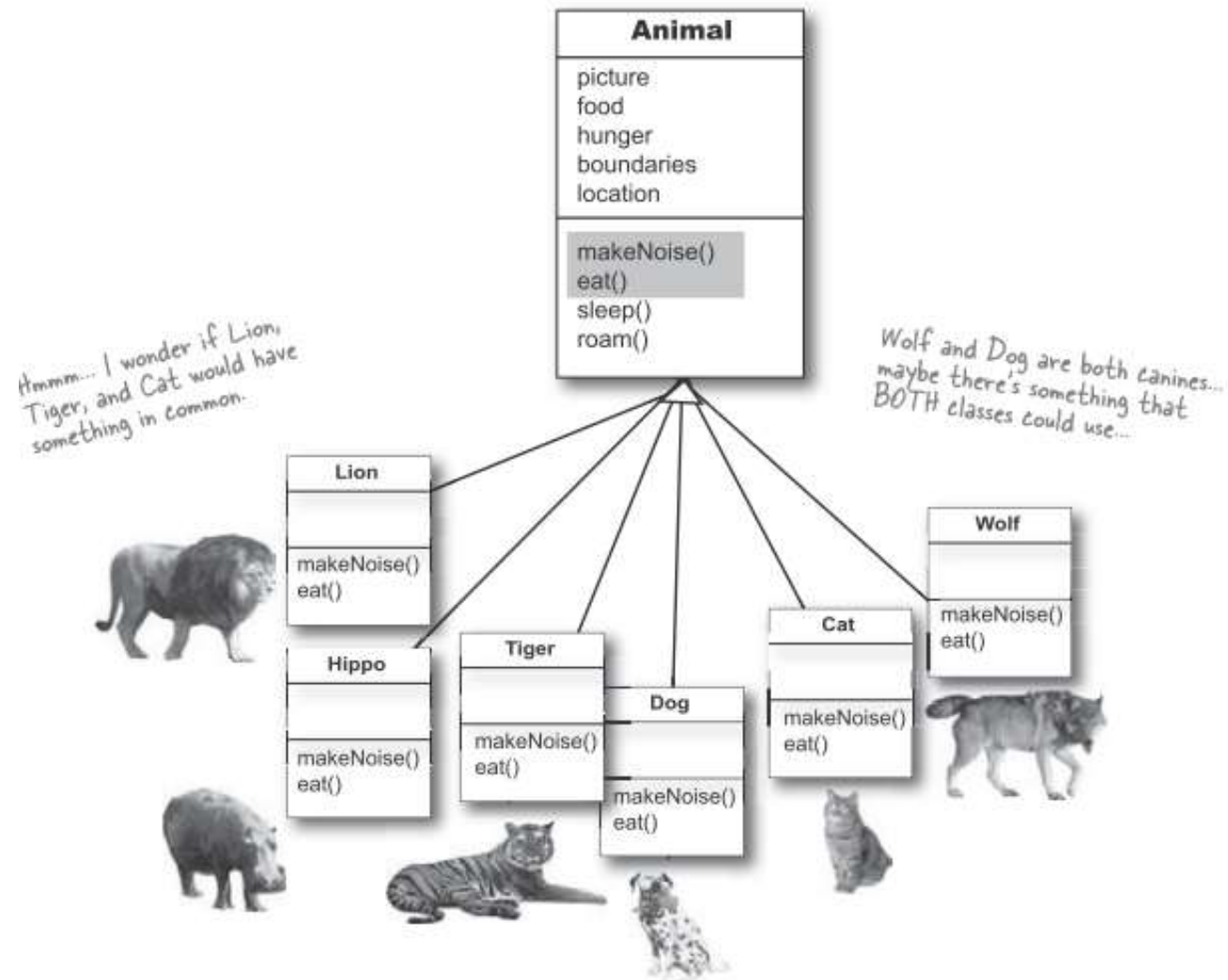
Osnovna pravila:

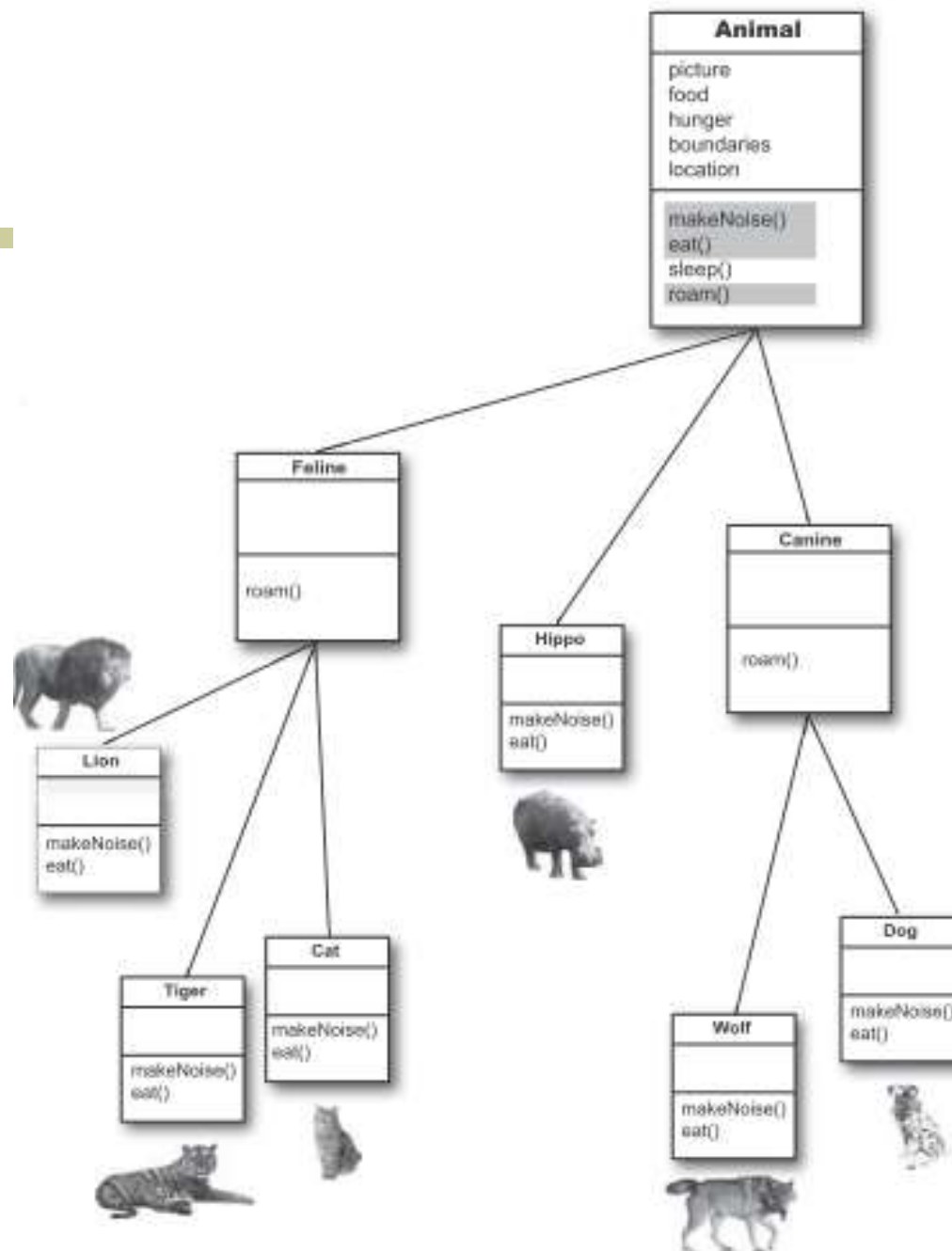
PODKLASA može imati samo jednu NADKLASU !!

NADKLASA može imati više PODKLASA !!

Nijedna klasa ne može biti vlastita NADKLASA.

[Nasljeđivanje]





[Poziv metoda]

make a new Wolf object

```
Wolf w = new Wolf();
```

calls the version in Wolf

```
w.makeNoise();
```

calls the version in Canine

```
w.roam();
```

calls the version in Wolf

```
w.eat();
```

calls the version in Animal

```
w.sleep();
```

Polazi se od najniže
razine u nasljeđivanju



Djelatnik.klasa - Notepad

File Edit Format View Help

```
class Djelatnik
{
    public Djelatnik(String i, double p, int godina, int mjesec, int dan)
    {
        ime = i;
        placa = p;
        GregorianCalendar calendar
        = new GregorianCalendar(godina, mjesec-1, dan);
        // GregorianCalendar koristi 0 za siječanj
        datum_zaposlenja = calendar.getTime();
    }
    public String uzmiime()
    { return ime; }
    public double uzmiplacu()
    { return placa; }
    public Date uzmidatum()
    { return datum_zaposlenja; }

    public void uvecajplacu(double postotak)
    { double uvecanje = placa * postotak/100;
      placa += uvecanje;
    }

    private String ime;
    private double placa;
    private Date datum_zaposlenja;
}
}
```

Start Semestar 2 Microsoft PowerPoint - [...] Djelatnik.klasa - Note... 98% 21:35

Nasljeđivanje

- Menadžeri jesu radnici, ali imaju bonus na plaću

```
class Manager extends Djelatnik  
{ // dodavanje metoda i polja //  
}
```

```
ManagerTest - Notepad
File Edit Format View Help

import java.util.*;

public class ManagerTest
{
    public static void main(String[] args)
    {
        // formira Manager objekt
        Manager sef = new Manager("karlo veliki",80000, 1987, 12, 15);
        sef.postavibonus(5000);

        Djelatnik[] osoblje = new Djelatnik[3];
        // popunjavanje područja "osoblje" s objektima Manager i Djelatnik
        osoblje[0] = sef;
        osoblje[1] = new Djelatnik("Mate Matić", 50000, 1989, 10, 1);
        osoblje[2] = new Djelatnik("Ana Anić", 45000, 1991, 3, 15);
        // prikaz podataka o svim objektima Djelatnik
        for (int i=0; i<osoblje.length; i++)
        {
            Djelatnik d = osoblje[i];
            System.out.println("ime = "+d.uzmiime()+"", plaća = "+d.uzmiplacu());
        }
    }
}

class Djelatnik
{
    public Djelatnik(String i, double p, int godina, int mjesec, int dan)
    {
        ime = i;
        placa = p;
        GregorianCalendar calendar
        = new GregorianCalendar(godina, mjesec-1, dan);
        // GregorianCalendar koristi 0 za siječanj
        datum_zaposlenja = calendar.getTime();
    }
    public String uzmiime()
    { return ime; }
    public double uzmiplacu()
    { return placa; }
    public Date uzmidatum()

```

ManagerTest - Notepad

File Edit Format View Help

```
        { return datum_zaposlenja; }

        public void uvecajplacu(double postotak)
        { double uvecanje = placa * postotak/100;
          placa += uvecanje;
        }

        private String ime;
        private double placa;
        private Date datum_zaposlenja;
    }

    class Manager extends Djelatnik
    {
        public Manager(String i, double p, int godina, int mjesec, int dan)
        {
            super(i, p, godina, mjesec, dan);
            bonus = 0;
        }
        public double uzmiplacu()
        {
            double baznaplaca = super.uzmiplacu();
            return baznaplaca+bonus;
        }

        public void postavibonus(double b)
        { bonus = b; }
        private double bonus;
    }
}
```

Start Semestar 2 Microsoft PowerPoint - [...] ManagerTest - Notep... 98% 22:56

Nasljeđivanje

```
class Manager extends Djelatnik
{ ....
    public void postavibonus(double b)
    { bonus = b;
    }
    private double bonus;
}
```

Primjena:

```
Manager sef = ....;
sef.postavibonus(5000);
```

Nasljeđivanje

Na objekt Djelatnik ne može se primijeniti postavibonus metoda jer nije definirana u klasi Djelatnik.

Na objekt Manager mogu se primijeniti metode uzmiime, uzmidatum iako nisu definirane u klasi Manager, ali one se automatski nasljeđuju od superklase (nadklase).

Polja: ime, plaća i datum_zaposlenja također se nasljeđuju od superklase.

Objekt Manager ima 4 polja: ime, placa, datum_zaposlenja i bonus.

=> Metoda uzmiplacu treba vratiti zbroj place i bonusa, što znači da postojeća metoda iz klase Djelatnik ne odgovara potrebama Managera.

Nadjačavanje (redefiniranje) metoda
engl. override

**Definira se nova metoda koja “nadjačava”
metodu iz nadklase.**

```
class Manager extends Djelatnik  
{ ....  
    public double uzmiplacu()  
    {.... }  
    ....  
}
```

Nadjačavanje (redefiniranje) metoda
engl. override

Implementiranje:

a) Vрати zbroj plaće i bonusa

```
public double uzmiplacu()  
{  
    return placa + bonus; // NE RADI !!??  
}
```

Metoda uzmiplacu iz klase Manager nema izravan pristup privatnim poljima nadklase.

Nadjačavanje (redefiniranje) metoda
engl. override

Implementiranje:

b) Koristi public sučelje tj. public metodu uzmiplacu klase Djelatnik

```
public double uzmiplacu()  
{  
    double baznaplaca=uzmiplacu(); // NE RADI !!??  
    return baznaplaca + bonus;  
}
```

Poziv metode uzmiplacu poziva tu metodu iz klase Manager što za posljedicu ima beskonačan poziv iste metode i dovodi do rušenja programa.

Nadjačavanje (redefiniranje) metoda
engl. override

Implementiranje:

**c) Mora se naglasiti da se zove metoda
uzmiplacu klase Djelatnik**

```
public double uzmiplacu()  
{  
    double baznaplaca=super.uzmiplacu(); RADI  
    !!??  
    return baznaplaca + bonus;  
}
```

Nasljeđivanje i kontrola pristupa

PRAVILO

Član klase koji je deklariran kao privatn, ostati će privatn za svoju klasu, što znači da neće biti dostupan izvan te klase, čak ni za podklasu !!!

Nadjačavanje (redefiniranje) metoda
engl. override

Kada metoda podklase ima isto ime i tip kao neka metoda nadklase, tada metoda podklase ima prednost nad metodom nadklase – tj. nadjačava je.

Kada se unutar podklase pozove tako nadjačana metoda, uvijek će se izvršiti njena verzija koja je definirana u podklasi.

[Nadjačavanje (redefiniranje) metoda]

Ako se želi pristupiti verziji redefinirane metode

u nadklasi koristi se operator super.

Metoda se nadjačava samo ako su imena i tipovi metoda identični.

Operator super

2 načina uporabe:

- A. za pozivanje konstruktora klase**
- B. za pristupanje članu nadklase koji je skriven članom podklase.**

[Pozivanje konstruktora]

```
public Manager  
(String i, double p, int godina, int mjesec, int dan)  
{  
    super(n,s,godina,mjesec,dan);  
    bonus=0;  
}
```

Ovdje je super umjesto “pozovi konstruktor klase Djelatnik s parametrima i, p, godina, mjesec i dan”

Kako Manager konstruktor ne može pristupiti privatnim poljima klase Djelatnik, mora ih inicijalizirati preko konstruktora.

[Pozivanje konstruktora]

U hijerarhiji klasa konstruktori se pozivaju po redoslijedu izvođenja, od nadklase ka podklasi.

[IS – A relacija]

Canine extends Animal

Wolf extends Canine

Wolf extends Animal

Canine IS-A Animal

Wolf IS-A Canine

Wolf IS-A Animal



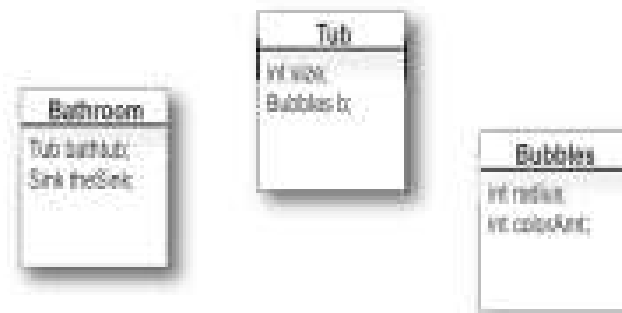
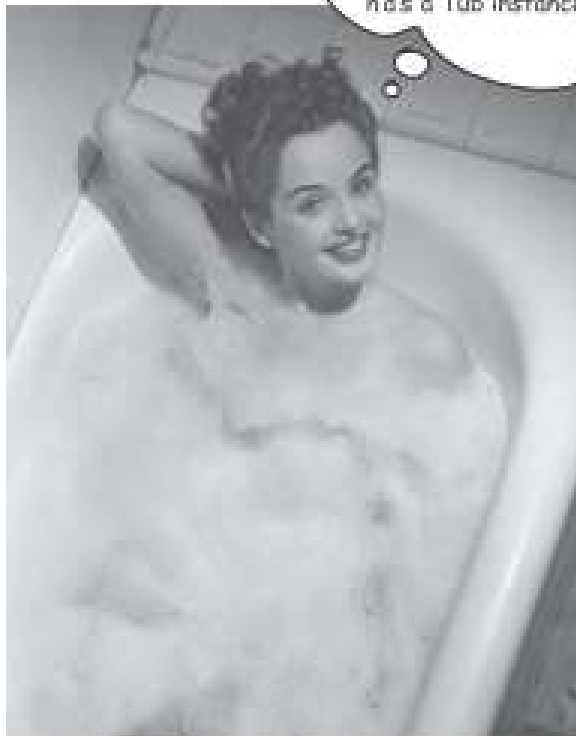
Pas JE životinja

Vuk JE pas

Vuk JE životinja

[HAS – A relacija]

Does it make sense to say a Tub IS-A Bathroom? Or a Bathroom IS-A Tub? Well it doesn't to me. The relationship between my Tub and my Bathroom is HAS-A. Bathroom HAS-A Tub. That means Bathroom has a Tub instance variable.



Bathroom HAS-A Tub and Tub HAS-A Bubbles.
But nobody inherits from (extends) anybody else.

[Polimorfizam]

Poziv

d.uzmiplacu()

Zove “korektnu” metodu

Činjenica da objektna varijabla (d) može referirati na više stvarnih tipova naziva se polimorfizam.

Automatski odabir odgovarajuće metode na runtime-u naziva se dinamičko razrješavanje metoda.

Nasljeđivanje - polimorfizam

Promjenjiva nadklase može referencirati objekt podklase.

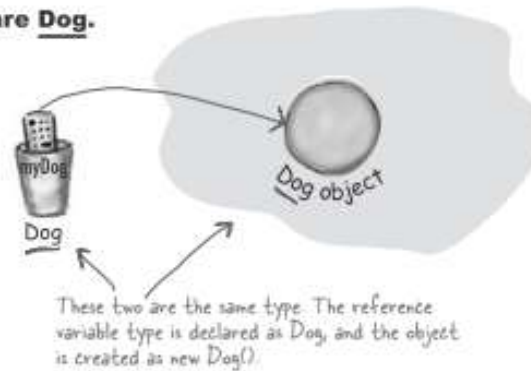
tj.

Referentnoj promjenjivoj nadklase može se dodijeliti referenca na bilo koju podklasu izvedenu iz te nadklase.

[Polimorfizam]

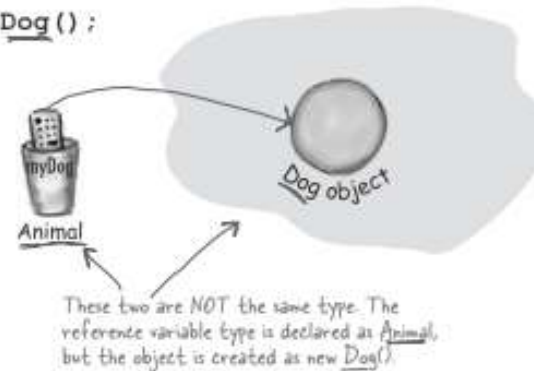
The important point is that the reference type AND the object type are the same.

In this example, both are Dog.



But with polymorphism, the reference and the object can be different.

Animal myDog = new Dog() ;



[Polimorfizam]

```
Animal[] animals = new Animal[5];
```

```
animals [0] = new Dog();
```

```
animals [1] = new Cat();
```

```
animals [2] = new Wolf();
```

```
animals [3] = new Hippo();
```

```
animals [4] = new Lion();
```

```
for (int i = 0; i < animals.length; i++) {
```

```
    animals[i].eat();
```

```
    animals[i].roam();
```

```
}
```

Declare an array of type Animal. In other words, an array that will hold objects of type Animal.

But look what you get to do... you can put ANY subclass of Animal in the Animal array!

And here's the best polymorphic part (the *raison d'être* for the whole example), you get to loop through the array and call one of the Animal-class methods, and every object does the right thing!

When 'i' is 0, a Dog is at index 0 in the array, so you get the Dog's eat() method. When 'i' is 1, you get the Cat's eat() method.

Same with roam().

Nasljeđivanje - polimorfizam

U JAVI – objekt varijable su polimorfne.

Npr. varijabla tipa Djelatnik može referencirati na objekt tipa Djelatnik ili na objekt bilo koje podklase klase Djelatnik (Manager, Tajnica ...)

```
Djelatnik[] osoblje = new Djelatnik[3];
```

```
Manager sef = new Manager(....);
```

```
osoblje[0] = sef;
```

Varijable osoblje[0] i sef odnose se na isti objekt.

osoblje[0] je za kompilator samo objekt Djelatnik

Može se pozvati sef.postavibonus(5000); ali ne i

osoblje[0]. postavibonus(5000); // GREŠKA

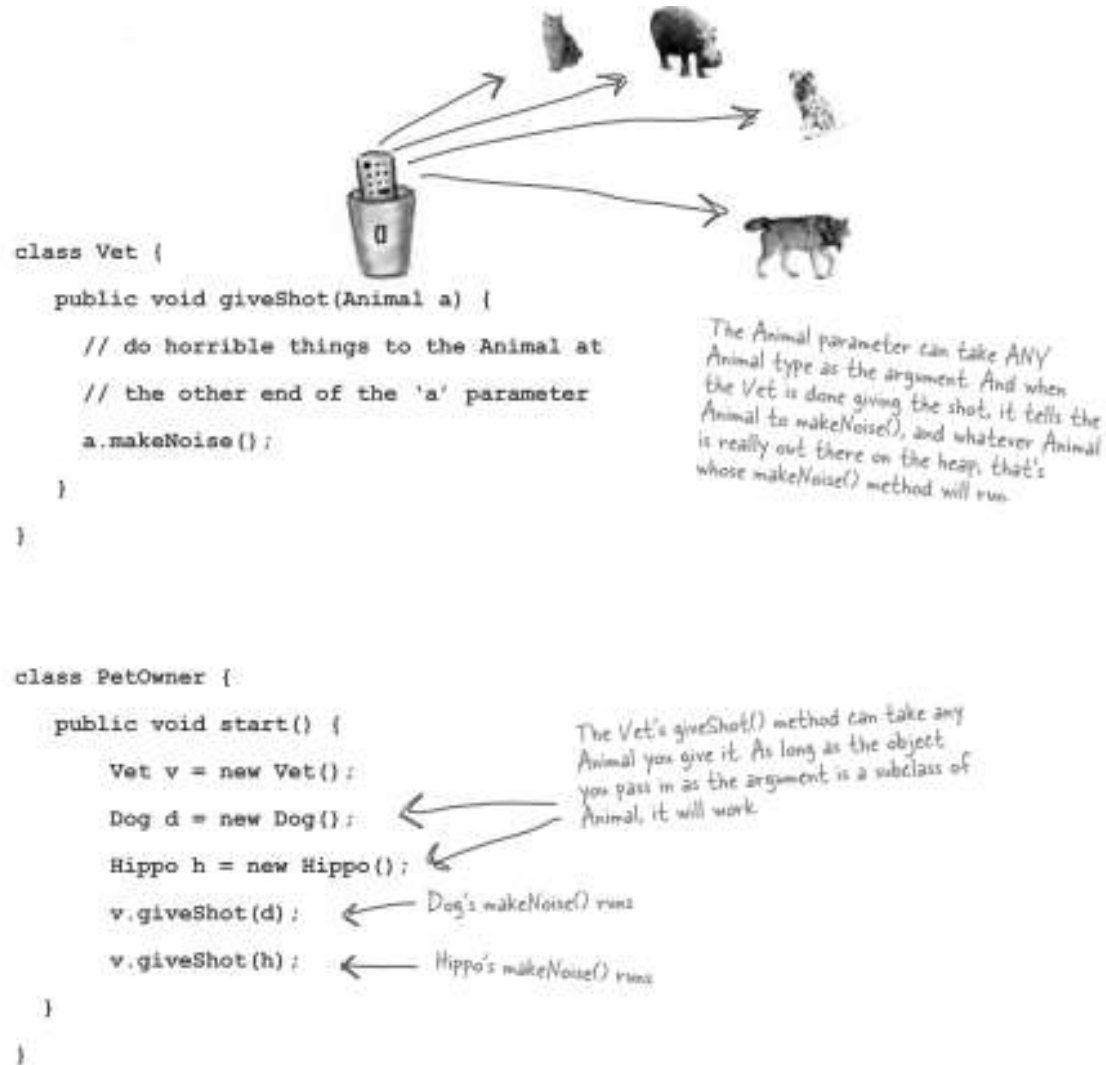
Nasljeđivanje - polimorfizam

Deklarirani tip za osoblje[0] je Djelatnik, a metoda postavibonus nije metoda klase Djelatnik.

Također, ne može se dodijeliti referenca nadklase varijabli podklase:

Manager m = osoblje[i]; // GREŠKA
Nisu svi djelatnici manageri.

[Polimorfizam argumenata]



[Uporaba apstraktnih klasa i metoda]

Klasa:

modifikator abstract class Ime_klase {...}

Metoda:

modifikator abstract tip ime(lista parametara);

[Uporaba apstraktnih klasa]

Ne traži implementiranje metoda.

Klasa može biti definirana kao apstraktna iako nema apstraktnih metoda.

Apstraktne klase nemaju instancu, tj. ako je klasa prijavljena kao apstraktna niti jedan objekt te klase ne može biti kreiran.

Npr. izraz `new Osoba("xx yy")` je greška.

Može se kreirati objektna varijabla apstraktne klase, ali ona mora referirati na neapstraktnu podklasu.

Npr. `Osoba o = new Student("xx yy", "ekonomija");`

OsobaTest - Notepad

File Edit Format Help

```
import java.text.*;
public class OsobaTest
{
    public static void main(String[] args)
    {
        osoba[] ljudi = new osoba[2];
        // popuni podruèje ljudi sa student i djelatnik objektima
        ljudi[0] = new Djelatnik("Mate Haker", 100000);
        ljudi[1] = new Student("Maria Maric", "ekonomija");
        // ispisi ime i opis svih objekata osoba
        for (int i = 0; i < ljudi.length; i++)
        { osoba o = ljudi[i];
          System.out.println(o.uzmiime()+" "+o.uzmiopis());
        }
    }
}
abstract class osoba
{ public osoba(String i)
  { ime = i; }
  public abstract String uzmiopis();
  public String uzmiime()
  { return ime; }
  private String ime;
}
```

Start | [Icons] | Semestar 2 | programi | Command Pr... | OsobaTest ... | Microsoft Po... | [Icons] 13:29

OsobaTest - Notepad

File Edit Format Help

```
class Djelatnik extends osoba
{
    public Djelatnik(String i, double p)
    { // proslijedi ime konstruktoru nadklase
      super(i);
      placa = p;
    }

    public double uzmiplacu()
    { return placa; }
    public String uzmiopis()
    { NumberFormat formatter
      = NumberFormat.getCurrencyInstance();
      return "djelatnik s placom od " + formatter.format(placa);
    }

    public void uvecajplacu(double postotak)
    { double uvecanje = placa * postotak/100;
      placa += uvecanje;
    }

    private double placa;
}

class Student extends osoba
{ /** @param p je ime student
   *   @param m je student gazda
   */
    public Student(String i, String g)
    { // proslijedi i konstruktoru nadklase
      super(i);
      gazda = g;
    }
    public String uzmiopis()
    { return " glavni je student " + gazda;
    }
    private String gazda;
}
```

Start | [Icons] | Semestar 2 | programi | Comman... | OsobaT... | Microsoft... | Norton A... | [Icons] | 13:30

PrimjerAbstract - Notepad

File Edit Search Help

```
// Uporaba apstraktnih metoda i klasa .
abstract class Slika {
    double dim1;
    double dim2;

    Slika(double a, double b) {
        dim1 = a;
        dim2 = b;
    }

    // površina() je sada apstraktna metoda
    abstract double površina();
}

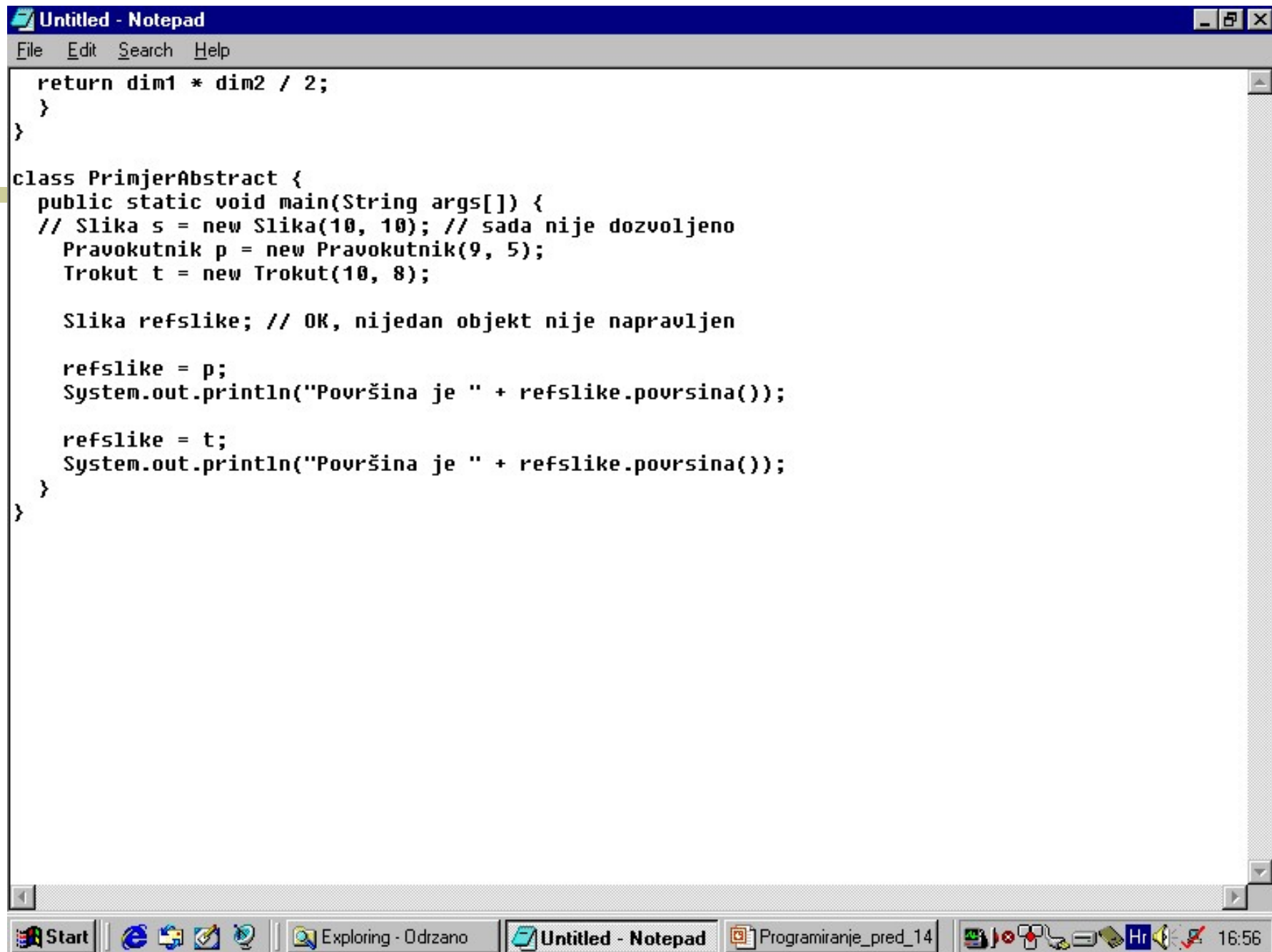
class Pravokutnik extends Slika {
    Pravokutnik(double a, double b) {
        super(a, b);
    }

    // nadjačavanje metode površina() za pravokutnik
    double površina() {
        System.out.println("Površina unutar klase pravokutnik.");
        return dim1 * dim2;
    }
}

class Trokut extends Slika {
    Trokut(double a, double b) {
        super(a, b);
    }

    // nadjačavanje metode površina() za trokut
    double površina() {
        System.out.println("Površina unutar klase Trokut.");
    }
}
```

Start | Exploring - Odrzano | PrimjerAbstract - ... | Programiranje_pred_14 | 16:55



```
return dim1 * dim2 / 2;
}
}

class PrimjerAbstract {
    public static void main(String args[]) {
        // Slika s = new Slika(10, 10); // sada nije dozvoljeno
        Pravokutnik p = new Pravokutnik(9, 5);
        Trokut t = new Trokut(10, 8);

        Slika refslike; // OK, nijedan objekt nije napravljen

        refslike = p;
        System.out.println("Površina je " + refslike.povrsina());

        refslike = t;
        System.out.println("Površina je " + refslike.povrsina());
    }
}
```

Start | e | Exploring - Odrzano | Untitled - Notepad | Programiranje_pred_14 | 16:56

[Vježba]

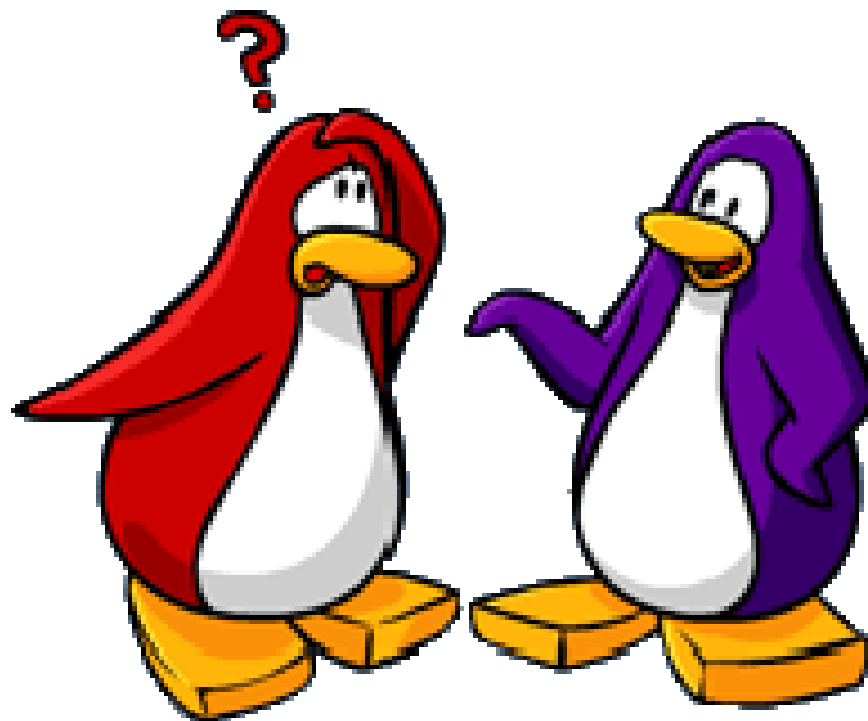
- Formirati polje od 3 elementa
- Formirati klasu Student (ime, prezime, tip, indeks)
- Formirati podklasu Redoviti uz plaćanje (ime, prezime, tip, indeks, školarina)
- Tip: R - redoviti bez plaćanja
P – redoviti uz plaćanje

Student8.java

[Domaća zadaća]

Za primjer s vježbi (prethodni slajd) dodati

- Podklasu Izvanredni (ime, prezime, tip, indeks, školarina)
- Školarina za izvanredne studente se izračunava tako da se školarina redovitih uz plaćanje uveća za 20%.
- Tip: R - redoviti bez plaćanja
- P – redoviti uz plaćanje
- I - izvanredni



Pitanja...