

Nivoi pristupa, paketi i JavaBeans specifikacija

Nivoi pristupa

Lokalne promenljive i ulazni parametri su, kao što je već navedeno, “vidljivi” u okviru metode u kojoj su deklarirani, dok se atributi i metode klase mogu pozvati bilo gde u okviru klase, ali i izvan nje. Međutim, u nekim situacijama je potrebno ograničiti pristup nekom atributu ili metodi tj. smanjiti im “vidljivost” van klase. U Javi je ovaj problem rešen uvođenjem **nivoa pristupa** i odgovarajućih ključnih reči koje predstavljaju **modifikatore pristupa** (“access modifiers”). Sledi tabela nivoa i modifikatora pristupa sa njihovim opisima.

Nivo pristupa	Modifikator pristupa (ključna reč)	Opis
Javni	public	Klasa, atribut ili metoda koji su označeni na ovaj način su vidljivi svuda i javno su dostupni.
Privatni	private	Klasa, atribut ili metoda koji su označeni na ovaj način su vidljivi samo u okviru “.java” fajla u kojem su napisani.
Podrazumevani (paketski)	(ne piše se ništa)	Klasa, atribut ili metoda koji su označeni na ovaj način su vidljivi samo u okviru paketa u kojem se nalaze.
Zaštićeni	protected	Isto kao i paketski nivo pristupa, samo što se vidljivost proširuje i na sve klase iz drugih paketa koje nasleđuju klasu koja je zaštićena ili sadrži zaštićeni element.

Modifikatori pristupa se mogu navesti na početku deklaracije klase, atributa ili metode i onda se odnose na tu klasu, atribut tj. metodu.

```
modifikator_pristupa1 class NazivKlase {  
  
    modifikator_pristupa2 tip_atributa nazivAtributa;  
  
    modifikator_pristupa3 tip_vrednosti nazivMetode(...) {  
        ...  
    }  
}
```

Koji su praktični aspekti nivoa pristupa i kada se oni zaista koriste? Nivoi pristupa omogućavaju sprovođenje principa **učaurenja podataka (enkapsulacija, “encapsulation”)** u Java programima. Ovo je princip koji se najčešće povezuje sa objektno orijentisanim programiranjem i podrazumeva da se **podacima ne može pristupiti direktno već indirektno preko poziva neke metode**. Implementacija učenja se najčešće izvodi tako što se **atributima klase dodeli privatni nivo pristupa pa im se može pristupiti isključivo preko odgovarajućih javnih metoda**. Ove javne metode najčešće sadrže i kod za logičku kontrolu, pa bilo kakva izmena vrednosti atributa mora da prođe određene provere tj. ne mogu se uneti nedozvoljene vrednosti.

Primer 1

Napraviti javnu klasu **AutomatNovca** koja ima :

- Privatni atribut stanje koji predstavlja količinu novca koja se nalazi u automatu. Početno stanje je 5000.0 din.
- Javnu metodu ulozNovac koja kao ulazni parametar ima iznos novca koji se želi uložiti. Ova metoda vrši ulaganje tj. dodavanje unetog iznosa na trenutno stanje samo ako je uneti iznos veći od nule. U suprotnom, ispisuje se poruka o grešci.
- Javnu metodu podigniNovac koja kao ulazni parametar ima iznos novca koji se želi podići. Ova metoda vrši podizanje željenog iznosa tj, smanjivanje stanja samo ako je uneti iznos veći od nule i ako u automatu ima dovoljno novca. U suprotnom, ispisuje se poruka o grešci.
- Javnu metodu ispisiStanje koja ispisuje koliko još novca ima u automatu.

Napraviti javnu klasu **TestAutomatNovca** koja pravi objekat klase AutomatNovca i poziva njegove metode.

```
public class AutomatNovca {

    private double stanje = 5000;

    public void ulozNovac(double iznos){
        if (iznos <= 0) System.out.println("Greska");
        else stanje=stanje + iznos;
    }

    public void podigniNovac(double iznos){
        if (iznos <= 0) System.out.println("Greska");
        else {
            if (stanje >= iznos) stanje=stanje - iznos;
            else System.out.println(
                "Zao mi je, nema dovoljno novca");
        }
    }

    public void ispisiStanje(){
        System.out.println("U automatu trenutno ima: "+
            stanje+" dinara");
    }
}

public class TestAutomatNovca {

    public static void main(String[] args) {

        AutomatNovca a = new AutomatNovca();

        //Ne moze se direktno pristupiti atributu stanje
        //jer je privatan. Zbog toga sledeca komanda nije
        //dovoljena.

        //a.stanje = 5000;

        //Atributu se jedino moze pristupiti preko metoda
        //"uloziNovac" i "podigniNovac" koje vrse logicku
        //kontrolu pre nego sto promene atribut stanje.
```

```

        a.uloziNovac(430);
        a.podigniNovac(222.34);

        a.ispisiStanje();

    }

}

```

U zadatku se tražilo da klasa AutomatNovca bude javna, pa je ispred deklaracije ove klase dodat modifikator “public”. Atribut “stanje” je privatan (“private”), a metode “uloziNovac”, “podigniNovac” i “ispisiStanje” su javne. Klasa AutomatNovca poštuje princip učenja jer se atributu “stanje” ne može pristupiti direktno već isključivo preko metoda. To se vidi u kodu “main” metode klase “TestAutomatNovca” gde se neuspešno pokušava sa direktnim pozivom. Time se onemogućava direktno unošenje nedozvoljenih vrednosti za ovaj atribut (npr. -1000 dinara). Pre nego što se stanje zaista promeni, metode “uloziNovac” i “podigniNovac” vrše logičke provere (da li je iznos pozitivan i da li ima dovoljno novca) čime se faktički onemogućava da atribut “stanje” čak i indirektno dobije neku nedozvoljenu vrednost.

JavaBeans specifikacija

Sa obzirom na to da učenje podrazumeva sakrivanje podataka i to tako da im se može pristupiti jedino preko poziva neke metode, pojavljuje se novi problem. Kako dati naziv metodi koja npr. vraća vrednost nekog atributa? Neko bi predložio da se metoda nazove “vрати####”, drugi bi je nazvali “uzmi####” dok bi poneko predložio naziv “daj####”. Problem se sastoji u tome što ne postoji jedinstvo pri davanju naziva, pa bi se vrlo verovatno dogodilo da postoji više naziva za metode koje rade isto. Rešenje predstavlja poštovanje **JavaBeans specifikacije**. Ova specifikacija odgovara na mnoga pitanja, ali reguliše i pitanje naziva pojedinih metoda.

Ako neka metoda samo vraća vrednost određenog atributa, ona bi, prema JavaBeans specifikaciji, **trebalo da se naziva “get####” pri čemu ostatak naziva čini naziv atributa**. Metoda koja, na primer, vraća vrednost atributa “program” bi trebalo da se naziva “getProgram”. Jedini **izuzetak su “get” metode za “boolean” attribute koje bi trebalo da se nazivaju “is####”**.

Ako neka metoda samo postavlja novu vrednost određenog atributa, ona bi, prema JavaBeans specifikaciji, **trebalo da se naziva “set####” pri čemu ostatak naziva čini naziv atributa**. Metoda koja, na primer, postavlja novu vrednost atributa “program” bi trebalo da se naziva “setProgram”.

JavaBeans specifikacija poštuje princip učenja, pa važi to da bi svi atributi trebalo da budu privatni, a sve “get” i “set” metode javne.

Primer 2

Napraviti javnu klasu **Osoba** koja ima :

- Privatni atribut ime.
- Privatni atribut prezime.
- Privatni atribut bracoStanje koji može imati vrednost TRUE ako je osoba oženjena ili udata a FALSE ako nije.
- Odgovarajuće javne get i set metode za ove attribute.

```

public class Osoba {

    private String ime;
    private String prezime;
    private boolean bracnoStanje;

    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 boolean isBracnoStanje() {
        return bracnoStanje;
    }
    public void setBracnoStanje(boolean bracnoStanje) {
        this.bracnoStanje = bracnoStanje;
    }

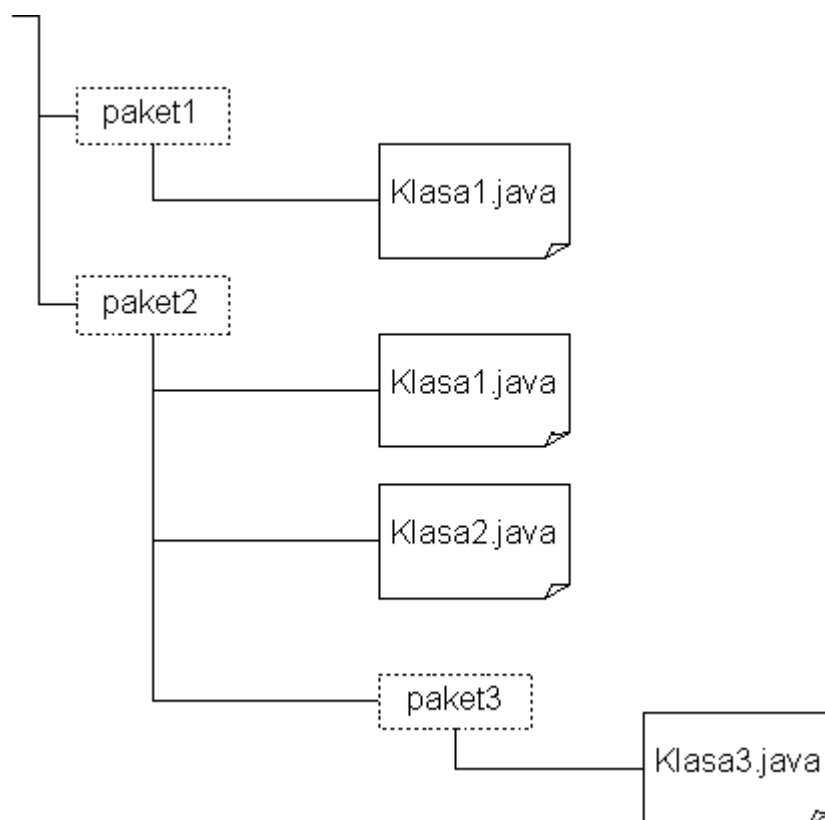
}

```

Paketi

Već je objašnjeno da se u jednoj klasi mogu pojaviti dve metode sa istim nazivom, ali da se moraju razlikovati po ulaznim parametrima. Takođe, pri redefinisaniu metode se piše nova metoda sa istim zaglavljem, ali drugačijom implementacijom. Međutim, da li je moguće u okviru istog programa napisati dve klase sa istim nazivom?

Odgovor na ovo pitanje je potvrđan. U Javi postoji koncept **paketa** koji je umnogome sličan konceptu direktorijuma (“folder”) na hard disku računara (paketi se, u suštini, i implemetiraju kao direktorijumi). Svaki paket predstavlja **celinu koja može da sadrži odgovarajuće fajlove (“.java”, “.class” ali i druge) ali i druge pakete (podpakete)**. Prema tome, paketi **mogu da se organizuju u hijerarhiju**. U Java programu se mogu pojaviti dve klase sa istim nazivom ali moraju da budu u različitim paketima (Slika 1).



Slika 1: Hijerarhija paketa

Ovo, naravno, nije jedini motiv za uvođenje paketa. Paketi su **veoma pogodni za grupisanje klasa** prema sličnosti ili funkciji koju obavljaju. Na ovaj način se programi koji sadrže veliki broj klasa organizuju u više manjih, homogenih celina. Štaviše, paketi mogu da se koriste u kombinaciji sa modifikatorima pristupa da bi se ograničio pristup nekim klasama i njihovim elementima.

Prema konvenciji, **nazivi paketa se pišu isključivo malim slovima**. Razdvajanje između paketa i njegovih podpaketa se vrši znakom tačke (".") pa bi se paket "ljudi" unutar paketa "antropologija" označavao kao "antropologija.ljudi".

U dosadašnjim primerima i zadacima, klase nisu pripadale nijednom konkretnom paketu, pa ih je Java programski jezik automatski smeštao u podrazumevani paket ("default package"). Ako klasa pripada nekom konkretnom paketu, onda se to mora naznačiti na prvoj liniji koda u ".java" fajlu u kojem se nalazi klasa i to pre deklaracije same klase. U tom slučaju se piše ključna reč **"package"** posle koje sledi naziv paketa.

```
package nazivpaketa;
```

```
class NazivKlase {
```

```
    ...
```

```
}
```

Ako se u programu koristi jedna ili više klasa koje pripadaju eksplicitno definisanim paketima, onda je pre njihovog korišćenja potrebno tzv. **uvoženje ("import")**. U tu svrhu se koristi "import" naredba. Ova naredba se piše pre deklaracije klase ali posle deklaracije paketa (ako postoji). Posle ključne reči "import" se navodi **pun naziv klase** koja se želi uvesti. Pun naziv klase se sastoji iz naziva paketa kojem klasa pripada i naziva klase. U jednom ".java" fajlu **može biti nula, jedna ili**

više “import naredbi”.

```
package nazivpaketa1;

import nazivpaketa2.NazivKlase2;
...
import nazivpaketaN.NazivKlaseN;

class Nazivklase1 {

    ...

}
```

Ako je istovremeno potrebno uvesti sve klase koje pripadaju jednom paketu u program, ne moraju se napisati posebne “import” naredbe, već se može iskoristiti znak zvezde (“*”).

```
package nazivpaketa1;

import nazivpaketa2.*;

class Nazivklase1 {

    ...

}
```

Primer 3

Napraviti javnu apstraktnu klasu **Instrument** u paketu **instrumenti** koja ima :

- Javnu apstraktnu metodu *ispisiVrstu* koja nema ulaznih parametara i ne vraća ništa.

Napraviti javnu klasu **Flauta** u paketu **instrumenti.duvackiinstrumenti** koja nasleđuje apstraktnu klasu *Instrument* i ima :

- Implementiranu javnu metodu *ispisiVrstu* koja na ekranu ispisuje da je u pitanju duvački instrument.

Napraviti javnu klasu **Violina** u paketu **instrumenti.gudackiinstrumenti** koja nasleđuje apstraktnu klasu *Instrument* i ima :

- Implementiranu javnu metodu *ispisiVrstu* koja na ekranu ispisuje da je u pitanju gudački instrument.

Napraviti javnu klasu **TestInstrumenti** u paketu **test** koja kreira po jedan objekat klase *Flauta* i *Violina* i poziva metode za ispis vrste instrumenta.

```
package instrumenti;

public abstract class Instrument {

    public abstract void ispisiVrstu();

}
```

```

package instrumenti.duvackiinstrumenti;

import instrumenti.Instrument;

public class Flauta extends Instrument {

    public void ispisiVrstu() {
        System.out.println("U pitanju je duvacki instrument");
    }

}

package instrumenti.gudackiinstrumenti;

import instrumenti.Instrument;

public class Violina extends Instrument {

    public void ispisiVrstu() {
        System.out.println("U pitanju je gudacki instrument");
    }

}

package test;

import instrumenti.duvackiinstrumenti.Flauta;
import instrumenti.gudackiinstrumenti.Violina;

public class TestInstrumenti {

    public static void main(String[] args) {

        Flauta f = new Flauta();
        Violina v = new Violina();

        f.ispisiVrstu();
        v.ispisiVrstu();

    }

}

```

Klasa Instrument se nalazi u paketu "instrumenti" što je i naznačeno preko komande "package". Sa obzirom na to da klase Flauta i Violina nasleđuju ovu klasu a nalaze se u drugim paketima u odnosu na nju, obe moraju prvo da uvezu ovu klasu ("import instrumenti.Instrument;"). Klasa Flauta je u paketu "duvackiinstrumenti" koji se nalazi u okviru paketa "instrumenti" pa je ceo naziv paketa "instrumenti.duvackiinstrumenti". Slično važi i za klasu Violina. Klasa TestInstrumenti koristi klase Flauta i Violina a ne nalazi se u njihovom paketu (već u paketu "test"), pa su navedene dve "import" naredbe na početku - za svaku klasu po jedna.

Sad se može objasniti značenje i poslednja dva nivoa pristupa: podrazumevanog (paketskog) i zaštićenog.

Podrazumevani nivo pristupa (paketski) definiše da se klasa ili neki njen element koji su ovako označeni, mogu koristiti samo u okviru paketa u kojem se nalaze. Oni nisu vidljivi van tog paketa. Ovaj nivo pristupa se dobija kada se ispred deklaracije klase, metode ili atributa **ne napiše nijedan modifikator pristupa**.

Zaštićeni nivo pristupa ("protected") podrazumeva da se klasa ili neki njen element koji su ovako označeni, mogu koristiti samo u okviru paketa u kojem se nalaze, ali su vidljivi i za one klase van paketa koje nasleđuju označenu klasu.

Zadaci

Zadatak 1

Napraviti javnu klasu **KonvertorValute** u paketu **valute**. Ova klasa bi trebalo da ima:

- Privatni atribut kursEvra koji je realan broj i označava vrednost jednog evra u dinarima.
- Privatni atribut kursDolara koji je realan broj i označava vrednost jednog dolara u dinarima.
- Javne set metode za ova dva atributa. Kursevi moraju da budu veći od nule, a u slučaju prekoračenja ovog ograničenja se ispisuje na ekranu poruka o grešci.
- Javnu metodu konvertujDinareUEvre koja, na osnovu unetog dinarskog iznosa izračunava i vraća iznos u evrima po trenutnom kursu.
- Javnu metodu konvertujDinareUDolare koja, na osnovu unetog dinarskog iznosa izračunava i vraća iznos u dolarima po trenutnom kursu.
- Metodu ispisKursnuListu koja ima podrazumevani pristup i ispisuje na ekranu trenutne kurseve za dolar i evro.

Napraviti klasu **TestKonvertorValute** u okviru paketa **valute** koja kreira jedan objekat klase Konvertor valute i postavlja trenutni kurs dolara na 60.34 dinara i evra na 76.89 dinara. Iskoristiti taj objekat za izračunavanje: dolarske protivvrednosti 600 dinara i protivvrednosti 4500 dinara u evrima. Ispisati rezultate računanja, pa zatim ispisati kursnu listu na ekranu.

Napraviti klasu **TestKonvertorValute** (ima isto ime kao i prethodna klasa) u paketu **provera** koji se nalazi u okviru paketa **valute** koja kreira jedan objekat klase Konvertor valute i postavlja trenutni kurs dolara na 55.0 dinara i evra na 80.0 dinara. Iskoristiti taj objekat za izračunavanje: dolarske protivvrednosti 550 dinara i protivvrednosti 16000 dinara u evrima. Ispisati rezultate računanja, pa zatim probati sa ispisom kursne listu na ekranu.

Rešenje:

```
//Klasa pripada paketu valute pa to mora eksplicitno
//da se napise:
package valute;

public class KonvertorValute {

    private double kursEvra;
    private double kursDolara;

    public void setKursDolara(double kursDolara) {
        if (kursDolara > 0) this.kursDolara = kursDolara;
        else System.out.println("Greska");
    }

    public void setKursEvra(double kursEvra) {
        if (kursEvra > 0) this.kursEvra = kursEvra;
        else System.out.println("Greska");
    }

    public double konvertujDinareUEvre (double iznos){
        return iznos / kursEvra;
    }
}
```



```

    public double konvertujDinareUDolare (double iznos){
        return iznos / kursDolara;
    }

    void ispisiKursnuListu(){
        System.out.println("Kurs evra: 1 evro = "+kursEvra+" dinara");
        System.out.println("Kurs dolara: 1 dolar = "+kursDolara+" dinara");
    }
}

//Klasa pripada paketu valute pa to mora eksplicitno
//da se napise:
package valute;

public class TestKonvertorValute {

    public static void main(String[] args) {

        KonvertorValute k = new KonvertorValute();

        //Kurs ne moze da se podesi na ovaj nacin:
        //
        //k.kursDolara = 60.34;
        //k.kursEvra = 76.89;
        //
        //Razlog: atributi kursDolara i kursEvra
        //su privatni pa su dostupni samo u okviru
        //klase KonvertorValute. Kurs se moze podesiti
        //samo na ovaj nacin:
        k.setKursDolara(60.34);
        k.setKursEvra(76.89);

        System.out.println("600 din = "+
            k.konvertujDinareUDolare(600)+" dolara");
        System.out.println("4500 din = "+
            k.konvertujDinareUEvre(4500)+" evra");

        //Ova metoda ima podrazumevani pristup pa joj
        //se moze pristupiti jer klasa TestKonvertorValute
        //pripada istom paketu kao KonvertorValute.
        k.ispisiKursnuListu();
    }
}

//Klasa pripada paketu valute.provera pa to mora eksplicitno
//da se napise:
package valute.provera;

//Klasa KonvertorValute pripada drugom paketu pa mora da se uveze
//koriscenjem import naredbe.
import valute.KonvertorValute;

//Moguće je imati dve istoimene klase (TestKonvertorValute) u programu
//ali moraju da se nalaze u razlicitim paketima.
public class TestKonvertorValute {

    public static void main(String[] args) {

        KonvertorValute k = new KonvertorValute();

        //Kurs ne moze da se podesi na ovaj nacin:

```

```

//
//k.kursDolara = 60.34;
//k.kursEvra = 76.89;
//
//Razlog: atributi kursDolara i kursEvra
//su privatni pa su dostupni samo u okviru
//klase KonvertorValute. Kurs se moze podesiti
//samo na ovaj nacin:
k.setKursDolara(55.0);
k.setKursEvra(80.0);

System.out.println("550 din = "+
    k.konvertujDinareUDolare(550)+" dolara");
System.out.println("16000 din = "+
    k.konvertujDinareUEvre(16000)+" evra");

//Ova metoda ima podrazumevani pristup pa joj
//se NE moze pristupiti jer ova klasa TestKonvertorValute
//NE pripada istom paketu kao KonvertorValute.
//k.ispisiKursnuListu();
}

}

```

Zadatak 2

Napisati javnu klasu **Osoba** u okviru paketa **ljudi** koja ima:

- Privatni atribut ime. Početna vrednost ovog atributa je "N".
- Privatni atribut prezime. Početna vrednost ovog atributa je "N".
- Privatni atribut JMBG koji predstavlja matični broj datu u vidu String-a.
- Privatnu metodu proverijMBG koja kao ulazni argument dobija String (koji predstavlja JMBG) i proverava da li taj String ima tačno 13 znakova (matični broj uvek ima tačno 13 cifara). Ako je String različit od null i ima tačno 13 znakova, metoda vraća TRUE a u suprotnom FALSE.
- Javne get i set metode za sva polja. Nedoovoljene vrednosti za ime i prezime su null String-ovi. Metoda setJMBG koristi metodu proverijMBG da proveriti da li je uneti JMBG u redu. Ako je bilo koje od ovih ograničenja prekoračeno, ispisuje se poruka o grešci na ekranu.
- Zaštićenu (protected) metodu ispisi koja ispisuje podatke o osobi.

Napisati javnu klasu **Zaposleni** u okviru paketa **zaposleni** koja nasleđuje klasu Osoba i ima:

- Privatni atribut plata (realan broj).
- Javne get i set metode za ovaj atribut. Plata mora da bude veća od nule.
- Javnu metodu ispisiZaposlenog koja ispisuje sve podatke o zaposlenom. Metoda poziva metodu ispisi klase Osoba.

Napraviti javnu klasu **TestOsoba** u paketu **provera** koja kreira objekat klase Osoba i objekat klase Zaposleni. Dodeliti osobi ime "Pera Perić" i JMBG "3112979710089", a zaposlenom "Mika Lazic", "3110970710111" i platu od 40000 dinara. Ispisati podatke o obe osobe.

Rešenje:

```

package ljudi;

public class Osoba {

    private String ime = "N";
    private String prezime = "N";
    private String JMBG;

    //Metoda proverijMBG je privatna pa se moze pozivati
    //samo u okviru klase Osoba.
    private boolean proverijMBG (String JMBG){
        if (JMBG == null || JMBG.length() != 13) return false;
        else return true;
    }
}

```

```

    }

    public String getIme() {
        return ime;
    }

    public void setIme(String ime) {
        if (ime != null) this.ime = ime;
        else System.out.println("Greska");
    }

    public String getJMBG() {
        return JMBG;
    }

    public void setJMBG(String jmbg) {
        if (proveriJMBG(jmbg)) JMBG = jmbg;
        else System.out.println("Greska");
    }

    public String getPrezime() {
        return prezime;
    }

    public void setPrezime(String prezime) {
        if (prezime != null) this.prezime = prezime;
        else System.out.println("Greska");
    }

    protected void ispisi(){
        System.out.println("Ime: "+ime);
        System.out.println("Prezime: "+prezime);
        System.out.println("JMBG: "+JMBG);
    }
}

package zaposleni;

import ljudi.Osoba;

public class Zaposleni extends Osoba {

    private double plata;

    public double getPlata() {
        return plata;
    }

    public void setPlata(double plata) {
        if (plata > 0) this.plata = plata;
        else System.out.println("Greska");
    }

    public void ispisiZaposlenog(){
        //Metoda ispisi klase Osoba se poziva koriscenjem reci
        //super. Onda je dostupna u klasi Zaposleni jer ima zasticeni
        //(protected) pristup.
        super.ispisi();
        System.out.println("Plata: "+plata);
    }
}

```

```
package provera;

import ljudi.Osoba;
import zaposleni.Zaposleni;

public class TestOsoba {

    public static void main(String[] args) {

        Osoba o = new Osoba();

        o.setIme("Pera");
        o.setPrezime("Peric");
        o.setJMBG("3112979710089");

        Zaposleni z = new Zaposleni();

        z.setIme("Mika");
        z.setPrezime("Lazic");
        z.setJMBG("3110970710111");
        z.setPlata(40000);

        //Ispisivanje podataka o Osobi ne moze ovako:
        //
        //o.ispisi();
        //
        //Razlog - metoda ispisi je zasticena. Klasa TestOsoba
        //pripada drugom paketu u odnosu na klasu Osoba a ne
        //nasledjuje klasu Osoba, pa pristup nije dozvoljen.
        //Umesto toga, ispis podataka o osobi se mora
        //uraditi ovako:
        System.out.println("Ime: "+o.getIme());
        System.out.println("Prezme: "+o.getPrezime());
        System.out.println("JMBG: "+o.getJMBG());

        z.ispisiZaposlenog();

    }

}
```