Osnove programiranja 1 v Javi

[Za čiste začetnike]

Podviga sem se lotila, ker sem se sama želela naučiti Javo
preko primerov. Ampak mi je v Eclipsu zmanjkalo prostora za
nove in novejše primere, zato sem se odločila, da bom vse skupaj
spravila na papir in tako tudi pomagala sošolkam in sošolcem.
Zapiski opisujejo vse od osnovnih pojmov do malo bolj zakompliciranih
primerov v Javi. Spoznavanje poteka večinoma preko primerov.
Ob njih so komentarji, ki jih je pametno prebrati. ;-)
Vendar moji zapiski niso dovolj, vsaj prebrati je še potrebno prosojnice
od prof. Mahniča in narediti samostojno nekaj primerov.

[Za dodatno razlago ali kontakt sem na voljo na emailu:
anchyr@gmail.com]

[Zavedam se, da so lahko v mojih zapiskih napake in se
že v naprej opravičujem ter pozivam vse, da mi te
napake sporočijo. Do napake je lahko prišlo zaradi
zatipkanja ali pa zaradi tega, ker ne obvladam Javo 100%]

Anchy R
[23.4.2007]

Kazalo:

Osnovni pojmi

- **int (Integer)** – naravno število (podobno short, byte – manjše, long – večje)

```
int a = 8;
```

```
String niz = "54";
int stevilo = Integer.parseInt(niz); //pretvorimo niz v int
```

- **double** – celo število (podobno je float – vendar manjše)

```
double b = 8.2;
```

```
String niz = "54.872";
double realnoStevilo = Double.parseDouble(niz);//pretvorimo niz v double
```

- **boolean** – tip, ki se ga nastavi na true ali na false

```
boolean jePrav = false;
```

- **char (Character)** – znak (ko primerjamo – znak1 == znak2)

```
char znak = 'a';
```

```
//znak
char c = 'A';

//najpogostejse operacije
Character.isUpperCase(c);  //vrne true (boolean), ce je c velika crka

Character.isLowerCase(c);  //vrne true (boolean), ce je c mala crka

Character.toUpperCase(c);  //pretvori c v veliko crko

Character.toLowerCase(c);  //retvori c v malo crko

Character.isDigit(c);      //vrne true (boolean), ce je c stevilka

Character.isLetter(c);     //vrne true (boolean), ce je c crka

Character.isLetterOrDigit(c);//vrne true (boolean), ce je c crka ali
                              //stevilka

Character.isWhitespace(c); //vrne true (boolean), ce je c whitespace
                           //(presledek, tab, enter,...)
```

- **String** – več znakov (ko primerjamo – niz1.equals(niz2)) – ne moremo spreminjati vsebine (dodajati, brisati znakov,...)

```
String ime = "Maja";
```

```
//niz
String niz = "Bla";

//najpogostejse operacije
niz.charAt(i);                    //vrne znak (char) na i-tem mestu

niz.compareTo(niz2);              //vrne negativno vrednost (int), ce je
                                  //niz prej po abecedi kot niz2, vrne
                                  //pozitivno vrednost (int), ce je niz2
                                  //prej po abecedi kot niz

niz.compareToIgnoreCase(niz2);    //podobno kot zgoraj, samo ignorira
                                  //velike in male crke

niz.startsWith(niz2);             //vrne true (boolean), ce se niz zacne s
                                  //podnizom niz2

niz.endsWith(niz2);               //vrne true (boolean), ce se niz konca s
                                  //podnizom niz2

niz.replace(c, d);                //zamenjamo c (char) z d (char)

niz.equals(niz2);                 //vrne true (boolean), ce sta niza enaka

niz.equalsIgnoreCase(niz2);       //podobno kot zgoraj, samo ignorira
                                  //velike in male crke

niz.indexOf(c);                   //vrne pozicijo (int) znaka c

niz.indexOf(c, 2);                //vrne pozicijo (int) znaka c od 2 znaka
                                  //naprej (pred 2 ne isce)

niz.length();                     //vrne dolzino (int) niza

niz.substring(zac, kon);          //vrne podniz, ki se zacne na zac (int)
                                  //in konca z kon-1 (int)

niz.toLowerCase();                //pretvori vse crke v male

niz.toUpperCase();                //pretvori vse crke v velike

//posebnost
int a = 500;
String niz = Integer.toString(a);//pretvori stevilo iz int v String
```

- **StringBuffer** – lahko spreminjamo vsebino (dodajamo, brišemo,…)

```
//1. nacin inicializacije
StringBuffer niz = new StringBuffer("neki");

//2. nacin inicializacije
String niz1 = "neki";
StringBuffer niz = new StringBuffer(niz1);
```

```
//StringBuffer
StringBuffer niz = new StringBuffer("neki");

//najpogostejse operacije
niz.append(c);                    //dodamo na konec poljuben element
                                  //(char, String,...)

niz.charAt(i);                    //vrne znak (char) na i-tem mestu (int)

niz.delete(zac, kon);             //pobrise podniz, ki se zacne na zac
                                  //(int) in konca na kon-1 (int)

niz.deleteCharAt(i);              //pobrise znak na i-ti poziciji (int)

niz.equals(niz2);                 //glej pri Stringu

niz.indexOf(c);                   //vrne indeks (int), kjer se nahaja znak
                                  //c (char)

niz.indexOf(c, poz);              //vrne indeks (int), kjer se nahaja znak
                                  //c (char) od poz (int) naprej

niz.insert(poz, c);               //vrinemo znak (char), niz (String),… na
                                  //pozicijo poz (int)

niz.length();                     //dolzina niza

niz.replace(zac, kon, niz2);      //brise od zac (int) do kon (int) in
                                  //vstavi niz2 (String)

niz.reverse();                    //obrne znake (zacetni znak je potem na
                                  //koncu, itd)

niz.substring(zac, kon);          //vrne niz (String), ki se zacne na zac
                                  //(int) in konca na kon-1 (int)

niz.setCharAt(poz,c);             //vpise znak c (char) na pozicijo poz
                                  //(int)

niz.toString();                   //pretvorimo v String

//posebnost
int k = 5;
StringBuffer niz2 = new StringBuffer(k);     //prazen niz s k praznimi
prostori
//opomba - ko dodajamo, insertamo,... se kapaciteta povecuje
```

- **inicializacija** – spremenljivko nastavimo na neko začetno vrednost

```
int n;          //brez inicializacije (deklaracija)
int n = 0;      //z inicializacijo
```

- **lokalne spremenljivke** – dostopne znotraj metode (deklarirane v metodi)
- **globalne spremenljivke** – dostopne povsod v razredu (deklarirane takoj na začetku (ne v metodah))
- **referenca this** – poskusi uporabiti v Eclipsu (bo bolj razumljivo)

```
int stevilka = this.matStev;     //glej pri objektu
```

- **argumenti metode main** - <ime razreda> <arg1> <arg2> … <argn> (obravnavamo kot tabelo znakov – dolzina: args[0].length;)
- **if(pogoj) { }**

```
if(a<b)
{
   //kaj se naj izvede, če je pogoj pravilen
}
else if(b<a)
{
   //kaj naj se izvede, če zgornji pogoj ni izpolnjen in je drugi
   //pravilen
}
else
{
   //kaj naj se izvede, če nobeden od pogojev ni pravilen
}
```

- **for(inicializacija; pogoj; povečevanje števca) { }**

```
for(int i=0; i<100; i++)
{
   //kaj se naj ponavlja dokler je pogoj izpolnjen
}
```

- **while(pogoj) { }**

```
while(a<b) //pogoj je na začetku, je možnost, da sploh ne greš v zanko
{
   //kaj se naj ponavlja, dokler je pogoj izpolnjen

   a++; //nekaj iz pogoja moraš spremeniti v vsakem koraku, drugače si v
         //neskončni zanki
}
```

- **do { } while(pogoj)**

```
do
{
   //kaj se naj ponavlja, dokler je pogoj izpolnjen

   a++; //nekaj iz pogoja moraš spremeniti v vsakem koraku, drugače si v
         //neskončni zanki
}while(a<b) //pogoj je na koncu, tako da greš vsaj enkrat v zanko
```

- **break** – uporablja v for, while in do while zankah; prekine zanko

```
for(int i=0; i<100; i++)
{
   if(i<=50)
   {
         //naredi nekaj
   }
   else
   {
         break; //prekini
   }
 }
```

- **continue** – isto kot zgoraj, nadaljuje zanko

```
for(int i=0; i<100; i++)
{
   if(i<=50)
   {
         //naredi nekaj
   }
   else
   {
         continue; //nadaljuj, praktično se to zelo malokrat
                   //napiše, ker program tudi brez tega nadaljuje
   }
 }
```

- **switch(spremenljivka) { … }**

```
switch(ocena)
{
   case 1:
   {
         //naredi nekaj
         break;
   }
   case 2:
   {
         //naredi nekaj
         break;
   }
   default:
   {
         //naredi nekaj, če nič od zgoraj ni prav
         //ne potrebujemo stavka break
   }
}
```

- **(pogoj):(izraz1)?(izraz2)** – če pogoj drži, potem se upošteva izraz1, drugače pa izraz2 – lahko uporabljamo kjerkoli (zanke, izpisi,…)

```
manjsi = (a<b)? a : b;

 //je enako:
if(a<b)
{
   manjsi = a;
}
else
{
   manjsi = b;
}
```

- **(pogoj) || (pogoj)** – ali drži prvi ali drugi pogoj; uporablja v if stavku (ponavadi)

```
if(i<=50 || i>60)
{
   //naredi nekaj
}
```

- **(pogoj) && (pogoj)** – če držita oba pogoja; uporablja v if stavku (ponavadi)

```
if(i<=50 && i>60)
{
   //naredi nekaj
}
```

- **!(...)** – zanikamo

```
a = !b; //v tem primeru mora biti b boolean

if(a!=b)
{
   //naredi nekaj, če a ni enak b-ju
}
```

- **operatorji**

```
<                       //manjse
<=                      //manjse ali enako
>                       //vecje
>=                      //vecje ali enako
```

```
=                       //prirejanje (dolocimo vrednost)
     a=8;
     b=a;
```

```
==                      //primerjanje (gledamo, ce sta enaka)
     a==b               //to se vidi v zankah (spodaj)
     if(a==b)
     {
                        //se zgodi, ce sta a in b enaka
     }
```

```
!=                      //gledamo, ce nista enaka
     a!=b
     if(a!=b)
     {
                        //se zgodi, ce a in b nista enaka
     }
```

- **neizpisljivi znaki (s pomočjo ubežnih sekvenc)**

```
" \t "                  //tab
" \n "                  //nova vrstica (enter)
" \f "                  //form feed (? kvadrat ?)
" \r "                  //carriage return (mehki enter)
" \' "                  //enojni narekovaj
" \" "                  //dvojni narekovaj
```

- **bližnjice v prireditvenih stavkih**

```
s-=i;                   //je isto kot s=s-i;
s*=i;                   //je isto kot s=s*i;
s/=i;                   //je isto kot s=s/i;
i++;                    //je isto kot i+=1; (najprej uporabi, potem poveca)
++i;                    //je isto kot i+=1; (najprej poveca, potem uporabi)
```

tabele

- **1D tabela**

```
//tabeli tipa int, lahko tudi double, boolean, String, char,…
int[] tabela1 = new int[3];   //3 je stevilo elementov (samo rezerviramo
prostor)
int[] tabela2 = {1, 2, 3};    //sam rezervira prostor in vstavi te tri
elemente (tako tudi inicializiramo tabelo (1. nacin))

//2. nacin inicializacije
tabela1[0] = 1;         //prvo mesto v tabeli
tabela1[1] = 2;         //drugo mesto v tabeli
tabela1[2] = 3;         //tretje mesto v tabeli

//3. nacin inicializacije
for(int i=0; i<tabela1.length; i++)            //pomagamo s for zanko (tece
po tabeli)
{
      tabela[i] = i+1;                         //nastavimo vrednost i
      tabela1[i] = (int)(Math.random()*10);    //ko smo na i-tem mestu, se
na random nastavi vrednost
}
```

- **2D tabela**

```
int[][] tabela1 = new int[2][3];          //prvo so vrstice, nato stolpci
int[][] tabela2 = {{1,2,3},{4,5,6}};     //1. nacin

//2. nacin
tabela1[0][0] = 1;
tabela1[0][1] = 2;
tabela1[0][2] = 3;
tabela1[1][0] = 4;
tabela1[1][1] = 5;
tabela1[1][2] = 6;

//3. nacin
for(int i=0; i<tabela1.length; i++)            //gremo cez vrstice
{
   for(int j=0; j<tabela1[i].length; j++)      //gremo cez stolpce
   {
      tabela1[i][j] = j;//na i-ti vrstici in j-tem stolpcu nastavimo
   }                    //vrednost j
}
```

- **3D, 4D,...** (podobno kot zgoraj)
- **tabela objektov**

```
Delavec[] td = new Delavec[100];    //ustvarimo tabelo objektov Delavec
for (int i=0; i<td.length; i++)     //gremo cez 100 delavcev
{
      td[i]=new Delavec(1000+i);    //za vsakega delavca (100 delavcev) se
                                    //ustvari nov objekt
}
```

METODE

- **način dostopa**

```
public (ostali del metode) //do metode lahko dostopa vsak (katerikoli
                           //zunanji razred)
private (ostali del metode) //do metode lahko dostopamo samo znotraj
                           //tega razreda
protected (ostali del metode) //do metode lahko dostopamo znotraj tega
                              //razreda in iz "sina" (dedovanje)
(navadno še en način dostopa) static (ostali del metode) //ta metoda je
                           //shranjena samo enkrat, je ne moremo
                           //spreminjati (recimo main metoda)
final (ostali del metode)  //ne moremo kasneje spreminjati (prepisati
                           //metode) – poleg v metodah se to dolocilo
                           //uporablja za konstante
```

- **metoda, ki ne vrača vrednosti**

```
(način dostopa) void imeMetode (argumenti ali brez)


//primer metode brez argumentov
public (static) void izpisiVse()
{
   //če jo potrebuješ znotraj razreda, mora biti static,
   //drugače pa ni treba
   //nekaj, kar se naj izpise (naredi)
}
//klicanje metode; kličeš jo iz druge metode, navadno iz main
izpisiVse();


//primer metode z argumenti
public (static) void izpisiVse(int stevilo, String niz)
{
   //če jo potrebuješ znotraj razreda, mora biti static,
   //drugače pa ni treba
   //nekaj, kar se naj izpise (naredi), ampak uporabi stevilo in
   //niz iz druge metode
}
//klicanje metode; kličeš jo iz druge metode, navadno iz main
izpisiVse(stevilka,ime);
```

- **metoda, ki vrača vrednost**

```
(način dostopa) int/double/boolean/char/String imeMetode (argumenti) {
return (nekaj);}

//primer metode brez argumentov
public (static) int izpisiVse()
{
   //enako kot zgoraj, ampak mora se nastaviti ena vrednost, v tem
   //primeru int
   return nekoSteviloInt;
}
//klicanje metode; kličeš jo iz druge metode, navadno iz main
izpisiVse();


//primer metode z argumenti
public (static) String izpisiVse(int stevilo, String niz)
{
   //enako kot zgoraj, ampak se mora nastaviti vrednost, v tem
   //primeru String
   return nekString;
}
//klicanje metode; kličeš jo iz druge metode, navadno iz main
izpisiVse(stevilka,ime);
```

- **main metoda**

```
public static void main(String[] args)
{
   //glava mora biti taka kot je, nič drugače
}
```

- **rekurzija** – metoda kliče samo sebe (klic metode je v njej sami)

Vnaprej deklarirani razredi

- **ti razredi so shranjeni v obliki paketov**

```
import java.lang.*;  //primer vstavljanja paketa
```

- **matematična knjižnica (Math)** – to ni potrebno importat

```
Math.random();        //klic metode random v paketu Math
```

- **Character**

```
Character.isDigit(c);       //klic metode isDigit(znak) v paketu Character
```

- **več v knjižnici od Jave**

BRANJE PODATKOV

- **throws Exception** – to mora imeti metoda v glavi

```
public static void main(String[] args) throws Exception
```

- **metoda System.in.read()** – tako preberemo znak (String sestavimo iz znakov, prav tako števila - sestavljamo)

```
char vtipkanZnak;
vtipkanZnak=(char)System.in.read();  //ker vrne int, je treba pretvoriti
```

RAZREDI IN OBJEKTI

- **osnovni koncept OOP:** razred, objekt, atribut, metoda
- **programiranje objektov**
  - deklariranje atributov
  - deklariranje metod
  - deklariranje konstruktorjev
- **kreiranje objektov** – podobno kot tabele, z operatorjem new

```
Objekt neki = new Objekt();
```

- **pisanje konstruktorjev** – cečkratno definirane metode
- **vsak objekt je konkreten primerek nekega razreda** (razred Živali – objekt Pes)
- **vsak objekt vsebuje atribute in metode** (atributi – lastnosti, metode – operacije, delovanje)
- **vsi objekti nekega razreda imajo iste atribute, ampak različne vrednosti**
- **enkapsulacija** – atributi in metode vgrajeni tako, da se obnašajo kot »črna škatla« - vidni so samo navznotraj, za uporabnika niso pomembni (za tankanje bencina ni potrebno vedeti kje je rezervoar, ampak samo kje je odprtina) – dostop private (glej spodaj)
- **razred ter ostalo**

```
//razred Delavec
public class Delavec
{
     //atributi
     //sklicevanje na atribute:
     //imeRazreda.matStev;
     //imeRazreda.priimek;
     private int matStev;
     private String priimek;

     //konstruktor
     //ko ustvarimo nov objekt: Delavec imeRazreda = new Delavec()
     //, se zgodi, kar je tukaj nastavljeno
     //ce hocemo dostopati do metod, moramo napisati:
     // imeRazreda.<metoda>
     Delavec()//tukaj notri lahko dodamo atribute in se potem nastavijo
     {        //ko kreiramo nov objekt

              //ta del potrebujemo, ce delamo s spodnjim testnim
              //razredom
              //Delavec(int ms, String p)
              //{
              //    matStev=ms;
              //    priimek=p;
              //}
           //v tem primeru ne potrebujemo setterjev, ker ze sami
           //nastavimo vrednosti matStev in priimek
           System.out.print("Maticna stevilka:");
           matStev=BranjePodatkov.preberiInt();
```

```
            System.out.print("Priimek:");
            priimek=BranjePodatkov.preberiString();

            //lahko poklicemo se kaksno metodo
            izpisiVse();
      }

      // seterji (metode, ki nastavijo vrednosti)
      //primer klicanja: imeRazreda.vpisiMatSt("12345");
      public void vpisiMatSt(int st)
      {
            this.matStev=st;
      }
      public void vpisiPriimek(String p)
      {
            this.priimek=p;
      }

      //geterji (metode, ki vracajo vrednosti - get (dobimo vrednost)
      //primer: imeRazreda.vrniMatSt();
      public int vrniMatSt()
      {
            return matStev;
      }
      public String vrniPriimek()
      {
            return priimek;
      }

      //metoda, ki izpise vse
      public void izpisiVse()
      {
            System.out.println("Maticna stevilka: "+ this.matStev);
            System.out.println("Priimek: "+ this.priimek);
      }
}
```

```
//razred, ki s katerim delamo (testiramo)
public class TestDelavec
{
      public static void main(String[] args)
      {
              // kreiranje objekta
              Delavec d=new Delavec();

              // vpis vrednosti atributov
              d.vpisiMatSt(234);
              d.vpisiPriimek("Novak");

              // izpis podatkov s pomocjo posameznih metod
              System.out.println("Maticna stevilka: "+d.vrniMatSt());
              System.out.println("Priimek: "+d.vrniPriimek

              // izpis podatkov z metodo izpisiVse
              d.izpisiVse();
      }
}
```

dedovanje

- **koncept dedovanja** - podrazred podeduje atribute in metode nadrazreda
- **redefinicija metod** - v podrazredu lahko ponovno deklariramo podedovano metodo – pomagamo z metodo nadrazreda (super); metode, ki jih ni moč redefinirati: private, static, final, metode znotraj razreda final
- **uporaba konstruktorjev** – konstruktor podrazreda mora poskrbeti za parametre, ki jih zahteva konstruktor nadrazreda
- **mehanizem, ki omogoča, da nek razred podeduje atribute in metode nekega drugega razreda**
  - osnovni razred (base class) – razred, ki služi kot osnova za dedovanje (nadrazred, starš)
  - izpeljan razred (derived class) – razred, ki je bil izpeljan iz osnovnega razreda (torej je razred, ki deduje, podrazred, otrok)
- **primer**

```
//nadrazred
public class Student
{
   //atributi
   private int vpisSt;
   private String priimek;
   private String ime;

   //konstruktor
   Student(int vs, String p, String i)
   {
         vpisSt=vs;
         priimek=p;
         ime=i;
   }

   //seterji
   public void vpisiVpisSt(int st)
   {
         vpisSt=st;
   }
   public void vpisiPriimek(String p)
   {
         priimek=p;
   }
   public void vpisiIme(String i)
   {
         ime=i;
   }

   //geterji
   public int vrniVpisSt()
   {
         return vpisSt;
   }
   public String vrniPriimek()
   {
```

```
        return priimek;
    }
    public String vrniIme()
    {
        return ime;
    }
}

//podrazred
public class IzredniStudent extends Student //podedujemo oz. extendamo
{                                            //Student
    private int znesekSolnine;

    IzredniStudent(int vs,String p,String i,int zn)
    {
        super(vs,p,i);//podedujemo elemente iz nadrazreda
        znesekSolnine=zn; //to nadrazred ne vsebuje, zato jo moramo
                          //sami nastaviti
    }

    //seter
    //potrebujemo samo tistega, ki ga ne vsebuje nadrazred
    public void vpisiZnesekSolnine(int solnina)
    {
        znesekSolnine=solnina;
    }

    //geter
    //potrebujemo samo tistega, ki ga ne vsebuje nadrazred
    public int vrniZnesekSolnine()
    {
        return znesekSolnine;
    }
}
```

```
//redefinicija metode izpisiVse() - ce bi obstajala
public void izpisiVse()
{
    super.izpisiVse();
    System.out.println(znesekSolnine);
}
```

- **abstraktni razred in abstraktne metode** – v abstraktnem razredu specificiramo metodo, ki je še ne moremo sprogramirati; vsak podrazred mora to metodo redefinirat

```
//abstraktni nadrazred
public abstract class Zival
{
	// atribut
	private String ime;

	// konstruktor
	Zival(String imeZivali)
	{
		ime=imeZivali;
	}

	// metode
	public String vrniIme()
	{
		return ime;
	}

	public abstract void oglasanje();  // abstraktna metoda
}

//abstraktni podrazred
public class Pes extends Zival
{

	// konstruktor
	Pes(String imePsa)
	{
		super(imePsa);
	}

	// redefinicija abstraktne metode
	public void oglasanje()
	{
		System.out.println("Hov,hov");
	}
}
```

- **dinamično povezovanje metod** – med izvanjanjem se izbere metoda, ki pripada dejanskemu tipu objekta
  - vsak objekt podrazreda je istočasno tudi objekt nadrazreda (Pes je Zival, obratno ni možno)
  - posledica: spremenljivki tipa nadrazred lahko priredimo naslov objekta, ki pripada kateremukoli podrazredu

```
//imamo spremenljivko z tipa Zival
Zival z = new Zival();

//imamo 3 objekte, ki pripadajo razredom Pes, Krava in Kaca
Pes p=new Pes("Luks");
Krava kr=new Krava("Liska");
Kaca ka=new Kaca("Klopotaca");

//potem so dovoljeni naslednji prireditveni stavki
z=p;
z=kr;
z=ka;

//izvede se metoda tistega podrazreda, katerega objekt je trenutno
//shranjen v spremenljivki z
//torej dinamicni tip spremenljivke z doloca, katera metoda oglasanje()
//se bo izvedla
z.oglasanje();
```

- **razred Object in njegove metode** – univerzalni nadrazred, iz katerega so izpeljani vsi ostali; v vseh razredih so na voljo metode, deklarirane v razredu Object; njegove metode:
  - toString()
  - equals() (<ime objekta>.equals(<imeDrugegaObjekta>))
  - getClass() – vrne objekt tipa Class, ki vsebuje ime razreda
  - hashCode() – izračuna hash kodo (za shranjevanje objektov v razpršenih tabelah)
  - notify(), notifyAll(), wait() – delo z nitmi
  - clone() - za kopiranje objekta
  - finalize() – metoda, ki se izvede ob uničenju objekta
- **koncept vmesnika kot nadomestek za večkratno dedovanje** – vsak podrazred lahko deduje samo od enega nadrazreda, implementira pa lahko več vmesnikov
  - vmesnik je zelo podoben razredu, le, da morajo biti vse metode abstraktne ter vsi atributi (če jih ima) morajo biti static final
  - z vmestnikom predpišemo metode, ki jih mora implementirati podrazred

```
public class Podrazred extends Nadrazred implements Vmesnik1, Vmesnik2
```

Overloading

- deklariranje več metod z istim imenom
- izvede se tista, kateri parametri se ujemajo
- v istem razredu obstaja več metod z enakim imenom, a različnim seznamom parametrov

```
//imamo dve metodi
public static double obresti(double g, double om)
{
   //neki
}
public static double obresti(double g, int om)
{
   //neki
}

//klic metode
obresti(1000.,0.08);       //izvede se prva metoda
obresti(1000.,8);          //izvede se druga metoda
```

Overriding

- **redefinicija metod** – glej dedovanje
- nadrazred in podrazred imata metodo z enakim imenom in enakim seznamom parametrov

GRafika

- **knjižnici AWT (Abstract Window Toolkit; starejša) in Swing** (novejša)
- **kreiranje okna** – okno kot podrazred razreda Jframe

```
import javax.swing.*;

class Okno extends JFrame
{
   //Eclipse vcasih potrebuje dolocene kode, te pa ne spremenijo nicesar
   private static final long serialVersionUID = 1L;

   //konstruktor
   public Okno()
   {

   }
}

public class TestOkna
{
   //atributi morajo biti static, da lahko do njih dostopamo v main
   //metodi, drugace ni potrebno
   private static String naslov = "Prvo okno";
   private static final int SIRINA = 300;
   private static final int VISINA = 400;

   public static void main(String[] args)
   {
         JFrame okno = new Okno();     //ustvarimo novo okno tipa JFrame

         okno.setTitle(naslov);        //nastavimo naslov
         okno.setSize(SIRINA, VISINA); //nastavimo velikost
         //okno.setLocation(x,y); - lega
         //okno.setBounds(x, y, sirina, visina) - velikost in lega
         okno.setResizable(false);     //ce ga lahko, ko je okno odprto,
                                       //povecamo ali ne; v tem primeru
                                       //ne (false)

         //okno.show();                //okno pokazemo (stara oblika)
         okno.setVisible(true);        //okno pokazemo, ce je true,
                                       //drugace ga skrijemo

//problem nastopi, ko hocemo zapreti okno, se program ne konca – moramo
//upoštevati dogodke (spodaj)
   }
}
```

- **dogodki** – zapiranje okna, razredi dogodkov, vmesniki, adapterji, poslušalci

Tip dogodka	Vmesnik	Odzivne metode
ActionEvent	ActionListener	actionPerformed(ActionEvent e)
ItemEvent	ItemListener	itemStateChanged(ItemEvent e)
TextEvent	TextListener	textValueChanged(ActionEvent e)
AdjustmentEvent	AdjustmentListener	adjustmentValueChanged(AdjustmentEvent e)
ContainerEvent	ContainerListener	componentAdded(ContainerEvent e) componentRemoved(ContainerEvent e)
ComponentEvent	ComponentListener	componentMoved(ComponentEvent e) componentHidden(ComponentEvent e) componentResized(ComponentEvent e) componentShown(ComponentEvent e)
FocusEvent	FocusListener	focusGained(FocusEvent e) focusLost(FocusEvent e)

Tip dogodka	Vmesnik	Odzivne metode
MouseEvent	MouseListener MouseMotionListener	mousePressed(MouseEvent e) mouseReleased(MouseEvent e) mouseEntered(MouseEvent e) mouseExited(MouseEvent e) mouseClicked(MouseEvent e) mouseDragged(MouseEvent e) mouseMoved(MouseEvent e)
KeyEvent	KeyListener	keyPressed(KeyEvent e) keyTyped(KeyEvent e) keyReleased(KeyEvent e)
WindowEvent	WindowListener	windowActivated(WindowEvenet e) windowClosing(WindowEvenet e) windowClosed(WindowEvenet e) windowDeactivated(WindowEvent e) windowDeiconified(WindowEvenet e) windowIconified(WindowEvenet e) windowOpened(WindowEvenet e)

Komponente (izvori dogodkov)	Metode za dodajanje poslušalcev
JButton, JCheckBox, JComboBox, JToolBar, JTextField, JRadioButton	addActionListener()
JScrollBar	addAdjustmentListener()
Vse Swing komponente	addFocusListener(), addKeyListener(), addMouseListener(), addMouseMotionListener()
JButton, JCheckBox, JComboBox, JRadioButton	addItemListener()
JWindow, JFrame	addWindowListener()
JSlider	addChangeEvent()

- deklaracija poslušalca
  - za dogodke tipa WindowEvent se ta razred imenuje WindowAdapter in je implementacija vmesnika WindowListener. Vmesnik WindowListener prepisuje 7 metod (glej prej tabelo)
  - v razredu WindowAdapter so te metode implementirane tako, da ob dogodku ne izvršijo nobene akcije – za to poskrbimo sami

```
import java.awt.event.*;   //moramo inportat, da bodo delovali poslusalci
import javax.swing.*;

class Okno extends JFrame
{
   private static final long serialVersionUID = 1L;

   public Okno()
   {
         WindowListener p = new Poslusalec();      //ustvarimo novega
                                                   //poslusaleca
         addWindowListener(p);                     //dodamo p-ju
                                                   //poslusalca
   }
}

public class TestOkna
{
   private static String naslov = "Prvo okno";
   private static final int SIRINA = 300;
   private static final int VISINA = 400;

   public static void main(String[] args)
   {
         JFrame okno = new Okno();

         okno.setTitle(naslov);
         okno.setSize(SIRINA, VISINA);
         okno.setResizable(false);
         okno.setVisible(true);
   }
}

//razred poslusalec
class Poslusalec extends WindowAdapter
{
   public void windowClosing(WindowEvent e)
   {
         System.exit(0);   //ta ukaz konca program
   }
}
```

```
//ce nocemo uporabiti poslusalca kot razred, spremenimo samo sledece:
//razred poslusalec pobrisemo, prav tako klic in namesto napisemo:
//konstruktor
public Okno()
{
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
```

- **upoštevanje karakteristik uporabnikovega računalnika**
  - lahko ugotovimo velikost zaslona in postavimo okno na sredino

```
import java.awt.*;   //to potrebujemo za upostevanje karakteristik
import javax.swing.*;

//razred, ki prikaze centrirano okno
class Okno extends JFrame
{
   private static final long serialVersionUID = 1L;

   public Okno()
   {
         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //podatki o velikosti zaslona
        Toolkit tk=Toolkit.getDefaultToolkit();
        Dimension d=tk.getScreenSize();

        //inicializiramo sirino in visino
        int sirina=d.width;
        int visina=d.height;

        setTitle("Centrirano okno");
        setSize(sirina/2,visina/2);
        setLocation(sirina/4,visina/4);
   }
}

public class TestOkna
{
     public static void main(String[] args)
     {
           JFrame okno=new Okno();
           okno.setVisible(true);
     }
}
```

- **risanje na panel**

```
import java.awt.*;
import javax.swing.*;

class Okno extends JFrame
{
   private static final long serialVersionUID = 1L;

   public Okno()
   {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Toolkit tk=Toolkit.getDefaultToolkit();
        Dimension d=tk.getScreenSize();

        int sirina=d.width;
        int visina=d.height;

        setTitle("Centrirano okno");
        setSize(sirina/2,visina/2);
        setLocation(sirina/4,visina/4);

        //sledece potrebujemo za risanje (risemo na panel - platno)
        Container vsebina = this.getContentPane(); //potrebujemo vsebino,
        JPanel risalnica = new Platno();           //risalnico
        vsebina.add(risalnica);                    //ter risalnico moramo
                                                   //dodati na vsebino
   }
}

//v tem razredu moramo ponovno deklarirat metodo paintComponent
class Platno extends JPanel
{
   private static final long serialVersionUID = 1L;

   //moramo redefinirati to metodo, ce zelimo, da vse dela tako kot mora
   public void paintComponent(Graphics g)
   {
         //podedujemo kar je v nadrazredu v tej metodi
         super.paintComponent(g);
         //narisemo crto
         g.drawLine(10, 10, 100, 70);
   }
}

public class TestOkna
{
     public static void main(String[] args)
     {
           JFrame okno=new Okno();
           okno.setVisible(true);
     }
}
```

- **pregled metod za risanje**
  - risanje crt, lokov in mnogokotnikov

```
g.drawString(niz, x, y);          //izpiše niz tako, da je prvi znak niza
                                  //odmaknjen od gornjega levega oglisca
                                  //panela za x tock v desno in y tock
                                  //navzdol

Font f = new Font("SansSerif",Font.BOLD,15);//f objekt je tipa Font in
                                            //mora biti prej dolocen
g.setFont(f);                     //nastavi obliko crk v skladu s
                                  //parametrom f

FontMetrics fm = g.getFontMetrics(f); //to moramo prej narediti
int dolzina = fm.stringWidth(niz);    //metoda razreda FontMetrics, ki
                                      //za font f vrne dolzino niza

g.drawLine(x1, y1, x2, y2);       //narise crto med tockama (x1, y1) in
                                  //(x2, y2)

g.drawArc(x,y,sirina,visina,zacKot,kot);//narise lok, ki se nahaja
                                  //znotraj navideznega pravokotnika z
                                  //levim zgornjim ogliscem v tocki (x,y)
                                  //in stranicama a=sirina, b=visina
                                  //lok se pricne pri kotu zacKot in
                                  //oklepa kot kot (se konca pri zacKot
                                  //+kot); koti so podani v stopinjah

Polygon p=new Polygon();          //narise mnogokotnik, katerega stranice
                                  //so dolocene s tockami objekta p
//primer za risanje trikotnika
p.addPoint(10,10);
p.addPoint(10,30);
p.addPoint(20,20);
g.drawPolygon(p);

g.drawPolygon(x,y,n);             //narise mnogokotnik, katerega oglisca
                                  //so podana s koordinatami tock v
                                  //tabelah x in y parameter n doloca
                                  //stevilo oglisc

g.drawPolyline(x,y,n);            //narise lomljeno crto, ki povezuje
                                  //tocke, katerih koordinate so v tabelah
                                  //x in y; n je stevilo tock, ki jih je
                                  //treba povezati ce sta prva in zadnja
                                  //tocka identicni, je crta zakljucena
```

- risanje pravokotnikov, krogov in elips

```
g.drawRect(x, y, sirina, visina);      //narise pravokotnik z z levim
                                       //zgornjim ogliscem v tocki (x,y)
                                       //in stranicama a=sirina, b=visina

g.drawRoundRect(x,y,sirina,visina,rH,rV);//narise pravokotnik z
                                       //zaobljenimi oglisci rH in Rv
                                       //dolocata horizontalni in
                                       //vertikalni polmer loka

g.draw3DRect(x,y,sirina,visina,dvig); //narise pravokotnik, ki daje vtis
                                       //gumba ce je parameter dvig enak
                                       //true, je pravokotnik "dvignjen"
                                       //na povrsino okna, sicer pa
                                       //"vgreznjen" ucinek postane
                                       //viden, ce narisemo vec
                                       //pravokotnikov, katerih stranice
                                       //se povecujejo za eno piko,
                                       //koordinate levega zgornjega
                                       //oglisca pa zmanjsujejo za eno
                                       //piko

g.drawOval(x,y,sirina,visina);         //narise elipso, ki se nahaja
                                       //znotraj navideznega pravokotnika
                                       //z levim zgornjim ogliscem v
                                       //tocki (x,y) in stranicama
                                       //a=sirina, b=visina ce sta
                                       //parametra sirina in visina
                                       //enaka, dobimo krog
```

- dolocanje barv

```
g.setColor(Color.barva);   //dolocimo barvo tistemu elementu, ki sledi
                           //(ce sledi g.drawOval(...) - se bo krog
                           //obarval s tisto barvo)
                           //namesto barve napises recimo red, blue,...
                           //(Eclipse sam predlaga)

setBackground(Color.barva);//nastavimo barvo podlage (metodo moramo
                           //poklicati preden prikazemo okno)

setForeground(Color.barva);//nastavimo barvo, s katero risemo
```

- risanje polnjenih likov

```
//podobno kot zgoraj, samo da jih pobarvamo znotraj
g.fillRect(x,y,sirina,visina);
g.fillRoundRect(x,y,sirina,visina,rH,rV);
g.fill3DRect(x,y,sirina,visina,dvig);
g.fillOval(x,y,sirina,visina);
g.fillPolygon(p);
g.fillPolygon(x,y,n);
g.fillArc(x,y,sirina,visina,zacKot,kot);
```

Apleti

- **koncept apleta**
  - program, ki teče znotraj spletnega brskalnika (npr. Internet Explorer)
  - je sestavni del neke spletne strani
  - pokličemo ga iz dokumenta, napisanega v HTML (lahko ga poženemo tudi iz Eclipsa ;-))
- **postopek izdelave**
  - napišemo podobno kot druge samostojne programe (aplikacije); vsak aplet napišemo kot razširitev osnovnega razreda Japlet (javax.swing.*)
  - ustvarimo HTML dokument, ki mora vsebovati ukaz za klic apleta (če delaš v Eclipsu, to ni potrebno) – bolj natančno v zapiskih od prof. Mahniča
  - poženemo spletni brskalnik in naložimo HTML dokument (če delaš v Eclipsu, to ni potrebno) – bolj natančno v zapiskih od prof. Mahniča
- **metode apleta** – init(), start(), stop(), destroy()

```
import javax.swing.*;
import java.awt.*;

public class Aplet extends JApplet
{
    private static final long serialVersionUID = 1L;//Eclipse
    //aplet se zažene (prva metoda, ki se izvede)
    //inicializacija spremenljivk
    //praviloma lahko samo to napišemo, ostale niso potrebne
    public void init()
    {
        JLabel a = new JLabel ("Nekaj napisi.");
        Container c = getContentPane();
        c.add(a);
    }

    //aplet postane aktiven
    //izvede takoj za metodo init() in vsakokrat, ko postane aplet
    //aktiven (ko se uporabnik vrne na stran z apletom – ce jo je prej
    //zapustil ali minimiral
    public void start()
    {

    }

    //se izvede vsakokrat, ko uporabnik zapusti stran z apletom
    public void stop()
    {

    }

    //se izvede, ko uporabnik zapre brskalnik
    public void destroy()
    {

    }
}
```

- **osnovne komponente grafičnega vmesnika** – labela, gumb, vnosno polje, razporejanje komponent po risalni plošči
  - labela

```
JLabel besedilo = new JLabel("Ime");   //ustvarimo novo labelo (izpis
                                       //besedila) s katero izpisemo:
                                       //'Ime'

… = new JLabel();                      //labela brez teksta in slike

… = new JLabel(text);                  //labela s tekstom

… = new JLabel(text, horizontalAlignment); //labela s
                                       //tekstom, poravnanim v skladu s
                                       //parametrom horizontalAlignment

… = new JLabel(Icon image);            //labela s sliko

… = new JLabel(Icon image, horizontalAlignment); //labela s sliko,
                                       //ki je poravnana v skladu s
                                       //parametrom horizontalAlignment

… = new JLabel(text, Icon image, horizontalAlignment);
                                       //labela s tekstom in sliko ter
                                       //predpisano poravnanostjo

besedilo.setText(niz);                 //vpiše niz (String) v že
                                       //kreirano labelo

besedilo.getText();                    //vrne niz (String), ki ga
                                       //vsebuje labela
```

  - vnosno polje

```
JTextField vp = new JTextField("Neki", 8);//ustvarimo novo vnosno polje
                                       //za 8 znakov in vanj napisemo
                                       //'Neki'

… = new JTextField();            //prazno vnosno polje dolžine 0

… = new JTextField(stevilo);     //prazno vnosno polje dolžine stevilo

… = new JTextField(text);        //vnosno polje z vnaprej vpisanim
                                 //tekstom

… = new JTextField(text, stevilo);//vnosno polje s tekstom text in
                                 //dolžino stevilo


vp.setText(niz);           //vpiše niz v vnosno polje vp

vp.getText();              //vrne niz, ki ga vsebuje vnosno polje vp

vp.setEditable(bool);      //določi, ali je možno vpisovanje (bool ima
                           //vrednost true) ali ne (bool je false)
```

- gumb

```
JButton potrdi = new JButton("Potrdi");//ustvarimo nov gumb z imenom
                                       //'Potrdi'

… = new JButton();                     //gumb brez napisa

… = new JButton(text)                  //gumb z napisom text

… = new JButton(text, Icon icon);      //gumb z napisom in sliko

… = new JButton(Icon icon);            //gumb s sliko

potrdi.setText(niz);       //nastavi napis na gumbu potrdi

potrdi.getText();          //vrne napis, ki ga vsebuje gumb potrdi
```

- risalna plošča

```
Container vsebina = getContentPane(); //ustvarimo risalno plosco vsebino
(nanjo dodajamo elemente)

//dodajanje elementov na vsebino (risalno plosco)
vsebina.add(zi);
vsebina.add(i);
vsebina.add(potrdi);
```

- nastavljanje fokusa

```
vp.requestFocus();   //omogoča, da vnaprej nastavimo kurzor tipkovnice v
                     //izbrano vnosno polje ali izpostavimo določen gumb
```

- razporejanje komponent po risalni plošči

```
vsebina.setLocation(sirina, visina);
vsebina.setSize(sirina, visina);
vsebina.setBounds(arg0, arg1, arg2, arg3);
```

- pregled razporejevalnikov (v nekaterih je potrebno v oklepajih navesti vrednosti – katere in koliko že pove Eclipse ;-))

```
//vsebina
Container vsebina = getContentPane();

//1. nacin:
vsebina.setLayout(null);              //brez razporejevalnika

//2. nacin:
FlowLayout flow = new FlowLayout();
vsebina.setLayout(flow);

//3. nacin:
//najpogostejsi nacini:
vsebina.setLayout(new FlowLayout());  //komponente razporedi po vrsticah
                                      //ko v neki vrstici zmanjka
                                      //prostora, nadaljuje v naslednji
                                      //vrstici

vsebina.setLayout(new BorderLayout());//se uporabi, če ne specificiramo
                                      //nobenega razporejevalnika
                                      //površino kontejnerja razdeli na
                                      //5 con, ki se imenujejo "North",
                                      //"West", "Center", "East" in
                                      //"South". Ob dodajanju vsake
                                      //komponente je treba navesti tudi
                                      //cono, kamor naj se komponenta
                                      //doda

vsebina.setLayout(new GridLayout());  //komponente razporedi v celice,
                                      //ki tvorijo matriko, sestavljeno
                                      //iz m vrstic in n stolpcev
                                      //število vrstic in stolpcev
                                      //določimo ob inicializaciji
                                      //razporejevalnika
                                      //vsaka nova komponenta se doda v
                                      //naslednjo celico
                                      //preskakovanje celic ni možno

vsebina.setLayout(new GridBagLayout());//omogoča dodajanje komponent v
                                      //točno določene celice
                                      //posamezne komponente lahko
                                      //zasedajo več celic

vsebina.setLayout(new CardLayout());  //komponente se nalagajo ena na
                                      //drugo
                                      //primeren, ko želimo, da je
                                      //naenkrat vidna samo ena
                                      //komponenta

vsebina.setLayout(new BoxLayout());   //vse komponente razporedi
                                      //v eno vrstico ali v en stolpec
```

- določanje barve

```
gumb.setForeground(Color.red);        //napis na gumbu je rdece barve
gumb.setBackground(Color.yellow);     //ozadje gumba je oranzne barve
```

- določanje pisave

```
Font pisava1 = new Font("TimesRoman",Font.ITALIC,24);//nastavimo pisavo,
                    //obliko ter velikost (kar sledi, bo taksne barve)
labela.setFont(pisava1);   //pisavo dolocimo za vsako komponento posebej
```

- odstranjevanje komponent

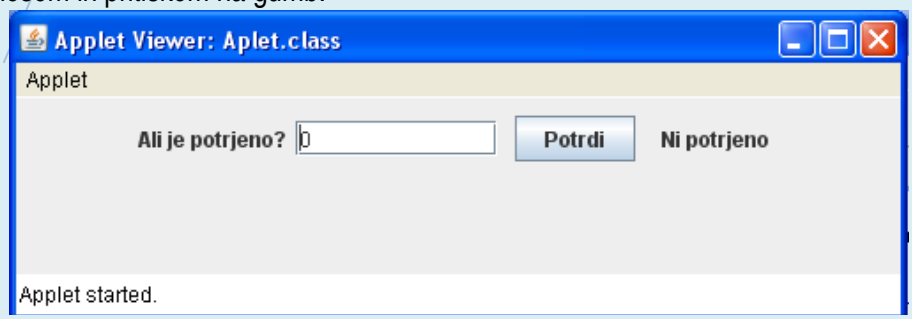
```
remove(gumb);          //odstranimo gumb
repaint();             //moramo repaintat, ce zelimo videti spremembo
```

- dogodki

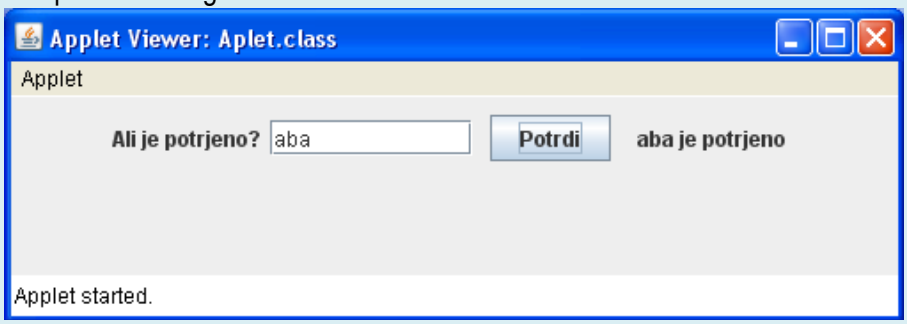
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Aplet extends JApplet
{
    private static final long serialVersionUID = 1L;    //Eclipse
    private final int SIRINA = 500;                     //sirina
    private final int VISINA = 100;                     //visina

    //(1.) deklariramo, inicializiramo labele, vnosna polja, gumbe,...
    private JLabel labela = new JLabel("Ali je potrjeno?");//labela
    private JLabel aliJe = new JLabel("Ni potrjeno");   //labela (izpis)
    private JTextField vnos = new JTextField("0", 10);  //vnosno polje
    private JButton gumb = new JButton("Potrdi");       //gumb

    //(2.) deklariramo, inicializiramo panele
    private JPanel izpis = new JPanel();                //panel
    private JPanel gumbi = new JPanel();                //panel

    //(3.) deklariramo, inicializiramo vsebino (risalno povrsino)
    private Container vsebina = getContentPane();       //vsebina

    //zacetek apleta, inicializacija
    public void init()
    {
         //postavimo sirino in visino
        setSize(SIRINA, VISINA);

         //na panele (2.) dodamo tisto, kar smo prvo deklarirali (1.)
        gumbi.add(gumb);
        izpis.add(aliJe);

        //razporejevalnik tistega, ki sledi
        vsebina.setLayout(new FlowLayout());

        //na vsebino (3.) lahko tudi neposredno dodamo (1.)
        vsebina.add(labela);
        vsebina.add(vnos);

        //na (3.) dodamo tisto, kar smo drugo deklarirali (2.)
        vsebina.add(gumbi);
        vsebina.add(izpis);

        //gumbu dodamo poslusalca
        gumb.addActionListener
        (  //oklepaj!
              new ActionListener()    //nov poslusalec
              {
                    //ko je gumb pritisnjen, se zgodi sledece
                    public void actionPerformed(ActionEvent d)
                    {
                       aliJe.setText(vnos.getText()+" je potrjeno");
                                    //na labelo aliJe (na izpisu)
                                    //napisemo tisto, kar je v
                    }               //vnosnem polju (vnosu) in se
```

```
                                        //besedilo, da je potrjeno
                }
        );//zaklepaj in podpicje!
          //lahko bi ustvarili nov ActionListener, ampak je to hitreje,
         //ce napisemo zgornje v oklepaju

    }
}
```

Pred vnosom in pritiskom na gumb:



Po vnosu in pritiskom na gumb:



Malo spremenljen primer prof. Mahniča (napisala samo to, kar je spremenljeno):

```
public class Aplet extends JApplet implements ActionListener
{                          //implementirat je potrebno ActionListener
    public void init()
    {
        //dodamo poslusalca
        gumb.addActionListener(this);
        //gumb1.addActionListener(this);//ce bi imeli se ta
                                        //gumb in mu dodali poslusalca
    }

    //metoda, ki jo moramo napisati, ker implementiramo
    //ActionListener
    public void actionPerformed(ActionEvent e)
    {
      aliJe.setText(vnos.getText() + " je potrjeno");//ce imamo samo
                                    //en gumb, lahko samo to napisemo
                                    //drugace moramo preveriti ali je
                                    //bil ta pritisnjen

      //ce bi imeli se en gumb, moramo preveriti kateri gumb je bil
      //pritisnjen; to naredimo sledece:
            //Object izvor = e.getSource();
            //if(izvor == gumb1)
            //{
            //    aliJe.setText("Nov gumb");
            //}
    }
}
```