5. ISPITIVANJE UVJETA

Peto poglavlje sastoji se od brojnih primjera korištenja `if` uvjetnog izraza kojima biramo između dvije alternativne akcije.

SADRŽAJ

1. Izrazi za donošenje odluka.
2. `if` naredba s jednom granom.
3. Izbor između različitih alternativa.
4. Tip podataka `boolean`.
5. Usporedba stringova.
6. Blokovi i doseg varijabli
7. Zadaci

---

1. Izrazi za donošenje odluka

U dosadašnjim programima svaki put se prilikom programa izvršavao isti niz izraza u redoslijedu kako su bili napisani. Java poput svih programskih jezika omogućava pisanje programa koji samostalno odlučuju koji će se naredbe izvršiti. Ta vrsta odluke se u Javi najčešće donosi pomoću **if uvjetne naredbe**. Slijedi primjer if uvjetne naredbe:

```
if (odgovor == 6048)
   System.out.println("Točno!");
else
   System.out.println("Netočno!");
```

Ova naredba znači:

> Ako je vrijednost varijable `odgovor` jednaka 6048,
> prikaži poruku "Točno!".
> inače prikaži poruku "Netočno."

Primijetite da je izraz

```
odgovor == 6048
```

upotrijebljen za ispitivanje uvjeta '`odgovor` je jednak 6048'. Simbol `==` koristi se za ispitivanje jednakosti pošto se `=` koristi za dodjeljivanja.

Slijedi program koji koristi `if` naredbu . Program pita koliki je rezultat računske operacije 72 puta 84? Zatim čita korisnikov odgovor i sprema ga u varijablu `odgovor`. Nakon toga slijedi `if` naredba koja provjerava da li korisnik pravilno odgovorio na postavljano pitanje. Ovaj program je poput većine primjera u ovom poglavlju aplikacija, a ne applet.

PRIMJER 1

```
public class Multi1

{  /* Traži od korisnika da unese odgovor na pitanje
      koliko je 72 puta 84, i onda provjeri odgovor. */

   public static void main(String[] args)
   {  ConsoleReader user =
         new ConsoleReader(System.in);

      System.out.println("Koliko je 72 puta 84?");
      int odgovor = user.readInt();
      if (odgovor == 6048)
         System.out.println("Točno!");
      else
         System.out.println("Netočno!");
   }
}
```

Opći oblik `if` naredbe dan je uokviren dolje. Naziv **Logički izraz** zamjenjuje bilo koji izraz čiji je rezultat "točno" ili "netočno" (true or false).

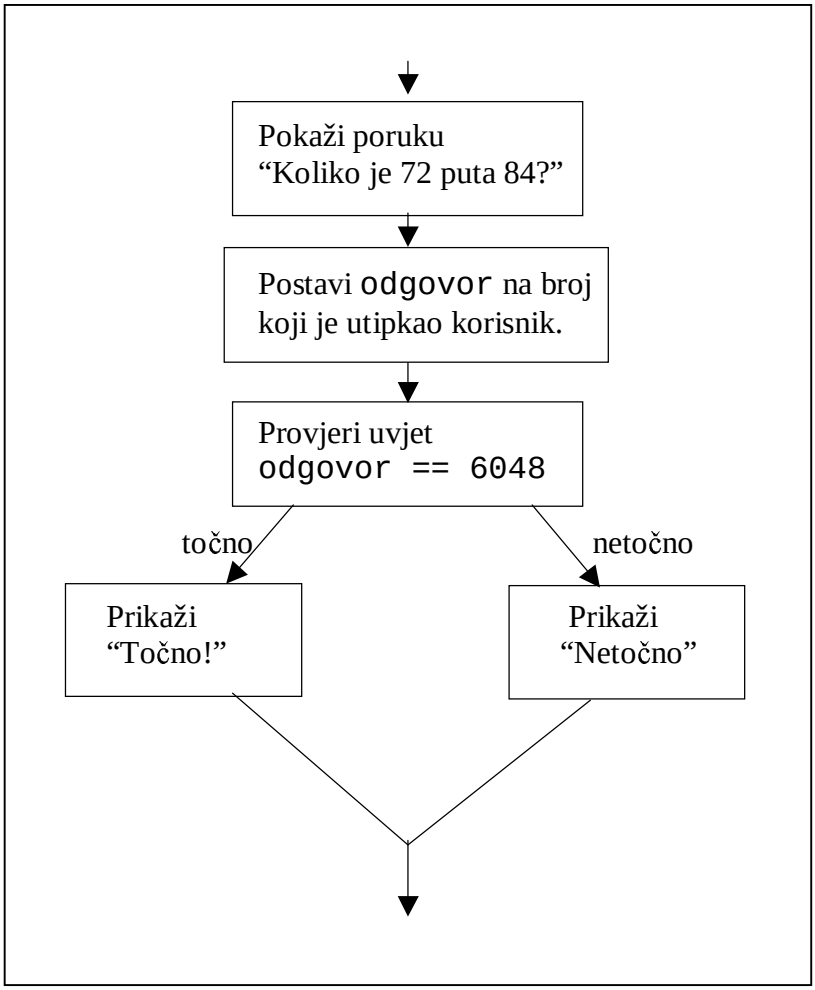
if naredba (s dvije grane)

```
if (LOGIČKI-IZRAZ)
   IZRAZ-1
else
   IZRAZ-2
```

Koda Java izvršava ovu naredbu prvo provjerava da li je rezultat *LOGIČKI-IZRAZ točno ili netočno*. Ako je točno izvršit će IZRAZ-*1*. Ako je netočno izvršit će IZRAZ-*2*.

*IZRAZ-1*se ponekad naziva **prva grana**. *IZRAZ-2*, je **druga grana**. Bilo koji izraz može se upotrijebiti umjesto *IZRAZ-1* i *IZRAZ-2*, čak i druga `if` naredba.

Za vizualizaciju događanja u programu koristit ćemo **dijagram toka.** Dijagram toka prethodnog programa izgleda ovako:



Ponekad je potrebno izvršiti nekoliko naredbi kad je uvjet zadovoljen, a nekoliko kad nije. U tom slučaju možete zamijeniti *UVJET-1* ili *UVJET-2* (ili oboje) s **blokom naredbi**. **Blok** (naredbi) je niz naredbi koje su zatvorene unutar vitičastih zagrada. Čak možete imati i blok bez i jedne naredbe poput `{}`. Blok se može staviti na bilo koje mjesto u programu na kojem inače dolazi naredba.

Slijedi primjer programa s blokom umjesto grane if naredbe (blok je prikazan podebljano).

PRIMJER 2

```
public class Multi2

{  /* Traži od korisnika da unese odgovor na pitanje
      koliko je 72 puta 84, i onda provjeri odgovor. */

   public static void main(String[] args)

   {  ConsoleReader user =
         new ConsoleReader(System.in);

      System.out.println("Koliko je 72 puta 84?");
      int ans = user.readInt();
```

```
      if (ans == 6048)
         System.out.println("Točno!");
      else
      {  System.out.println
            ("Netočno. Još jednom pokušaj !");
         ans = user.readInt();
         if (ans == 6048)
            System.out.println
               ("Napokon točno");
         else
            System.out.println
               ("Opet netočno. Točan odgovor je 6048");
      }
   }
}
```

2. `if` naredba s jednom granom

Postoji kraća forma `if` naredbe u kojoj se izostavlja riječ `else` i druga grana. Slijedi primjer:

```
if (a < b) then
   System.out.println(″Manja vrijednost je ″ + a);
```

Ovo znači: ako je a manje od b tada prikaži poruku koja kaže da je manja vrijednost a inače čini ništa. Ovakva naredba ima isti efekt kao slijedeća if naredba.

```
if (a < b) then
   System.out.println(″Manja vrijednost je ″ + a);
else
   {}
```

Opći oblik `if` naredbe s jednom granom je:

if naredba (s jednom granom)

```
if (LOGIČKI-IZRAZ)
   IZRAZ
```

Ako je *LOGIČKI-IZRAZ* točan tada će se izvršiti *IZRAZ* . Inače je `if` naredba bez ikakvog efekta.

Slijedi još jedan primjer u kojemu tražimo najmanji od četiri broja a,b,c,d i rezultat pohranjujemo u varijablu s.

```
s = a;
if (b < s) s = b;
if (c < s) s = c;
if (d < s) s = d;
```

Za misaonu vježbu pretpostavite slijedeće vrijednosti i odredite tijek izvršavanja:

a = 5, b = 10, c = 3, d = 8.

Slijedi primjer programa s `if` naredbom s jednom granom . Program računa kvadratni korijen unesenog broja. U slučaju da korisnik unese negativan broj program će ispisati poruku o grešci te izvršiti return naredbu.

Ključna riječ `return` znači: završi s izvođenjem metode, u ovom slučaju programa.

PRIMJER 3

```
public class Sqrt

{  /* Pitaj korisnika da unese broj a zatim ispiši
      kvadratni korijen unesenog broja. */

   public static void main(String[] args)
   {  ConsoleReader in =
         new ConsoleReader(System.in);

      System.out.print("Unesi broj: ");
      double x = in.readDouble();
      if (x < 0)
      {  System.out.println
            ("Broj treba biti >= 0.");
         return;
      }
      System.out.println
         ("Kvadratni korijen je " + Math.sqrt(x));
   }
}
```

Primjeri korištenja programa:

```
Unesi broj: 9.2
Kvadratni korijen je 3.03315017762062

Unesi broj: -3
Broj treba biti >=0.
```

Često se s greškama postupa na ovakav način tj. ispitivanjem s `if` naredbom te vraćanjem s naredbom `return.`

Return naredba ima dva osnovna oblika:

`return` naredba
(vraća vrijednost)

`return` *`IZRAZ`*`;`

`return` naredba
(ne vraća vrijednost)

```
return;
```

Prva vrsta return naredbe koristi se unutar metoda koji vraćaju vrijednost. Znači: završi s izvršavanjem metode i vrati vrijednost danu s IZRAZ. Druga se koristi unutar metoda koje ne vraćaju nikakvu vrijednost i znači samo kraj izvršavanja metode

---

3. Izbor između različitih alternativa

Ponekad je potrebno napisati kod koji bira između 3 ili više alternativa. To se može učiniti kombinacijom nekoliko `if` naredbi. Npr. student ostvari određen broj bodova na ispitu od maksimalnih 100. Na osnovu toga treba izračunati ocjenu.

broj bodova 70 - 100	ocjena = 5
broj bodova 60 - 69	ocjena = 4
broj bodova 50 - 59	ocjena = 3
broj bodova 40 - 49	ocjena = 2
broj bodova 0 - 39	ocjena = 1

Pretpostavimo da imamo varijablu `bodova`, koja sadrži broj bodova, a ocjenu želimo pohraniti u varijablu `ocjena`.

Pseudo kod bi izgledao ovako:

```
if (bodova >= 70) ocjena = ″5″;
else Postupaj s bodovima u području od 0 do 69.
```

Da bismo postupali s bodovima u području od 0 do 69, možemo koristiti drugu if naredbu, poput slijedećeg:

```
if (bodova >= 70) ocjena = ″5″;
else if (bodova >= 60) ocjena = ″4″;
else Postupaj s bodovima u području od 0 do 59.
```

Dalje postupamo na sličan način:

```
if (bodova >= 70) ocjena = ″5″;
else if (bodova >= 60) ocjena = ″4″;
else if (bodova >= 50) ocjena = ″3″;
else Postupaj s bodovima u području od 0 do 49
```

Itd. Na kraju slijedi kompletan izraz:

```
if (bodova >= 70) ocjena = "5";
else if (bodova >= 60) ocjena = "4";
```

```
else if (bodova >= 50) ocjena = "3";
else if (bodova >= 40) ocjena = "2";
else ocjena = "1";
```

Primijetite da je ovaj komad koda zapravo `if` naredba !

Slijedi kompletan program za proračun ocjene iz broja bodova:

PRIMJER 4

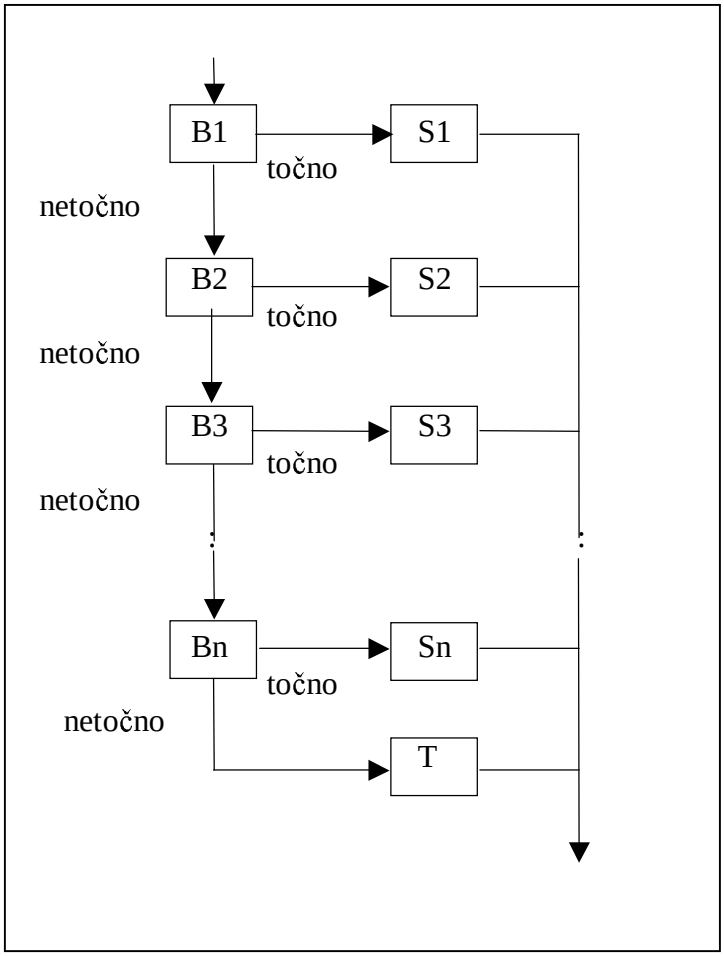
```
public class Ocjena
{  /* Učitaj broj bodova u području od 0 do 100,
      i ispiši odgovarajuću ocjenu.
   */
   public static void main(String[] args)
   {  ConsoleReader in =
         new ConsoleReader(System.in);

      System.out.print("Unesi broj bodova ");
      int bodova = in.readInt()
      String ocjena;
      if (bodova >= 70) ocjena = "5";
      else if (bodova >= 60) ocjena = "4";
      else if (bodova >= 50) ocjena = "3";
      else if (bodova >= 40) ocjena = "2";
      else ocjena = "1";
      System.out.println("Ocjena = " + ocjena);
   }
}
```

Ovom tehnikom ispitujemo niz uvjeta i čim je jedan zadovoljen izvršavamo odgovarajuću akciju.

```
if (B1) S1
else if (B2) S2
else if (B3) S3
  :
  :
else if (Bn) Sn
else T
```

Blok dijagram ovakve if naredbe izgleda ovako:



4. Tip podataka `boolean`

Logički (boolean) izraz poput `odgovor == 6048` izračunava se u jednu od slijedećih vrijednosti `true`(točno) and `false`(netočno). Ove dvije vrijednosti nazivaju se **logičke vrijednosti**. Zajedno čine primitivni tip podataka koji u Javi nazivamo `boolean`.

U mnogim pogledima ove vrijednosti se ponašaju poput drugih tipova vrijednosti kao npr. cjelobrojnih vrijednosti. Npr. metode mogu poput vraćanja cjelobrojnih vrijednosti vraćati logičke vrijednosti. Možete kreirati varijablu tipa `boolean` i dodjeljivati im vrijednosti.

Npr.:

```
boolean ok = (x > 0);
```

kao primjer korištenja logički vrijednosti, proširit ćemo Student klasu iz trećeg poglavlja. Dodat ćemo logički polje u koje ćemo bilježiti je li student upisao predmete tekuće godine. Polje ćemo nazvati `jeUpisan`. Bit će postavljeno na `true` ako je student upisao predmete, a na `false` ako nije. Polje je deklarirano na slijedeći način:

```
private boolean jeUpisan;
```

bit će potrebno učiniti još neke promjene u klasi student. Dodat ćemo slijedeće metode:

`s.upiši()`
Zapiši da je student s upisan.

`s.ispiši()`
Zapiši da Student s nije upisan.

`s.jeliUpisan()`
vrati `true` ako je s upisan, inače vrati `false`.

metode su vrlo jednostavne. Prva postavlja polje `jeUpisan` na `true`, druga postavlja polje `jeUpisan` na `false` i treća vraća vrijednost polja `jeUpisan`.

napravljene su još dvije promjene u klasi Student. U metodi `prikaz` dodano je ispitivanje vrijednosti polja `jeUpisan` na osnovu koje se ispisuje 'Upisan je' ako je njegova vrijednost postavljena na `true`, ili 'Nije upisan' ako je `false`. Konstruktor postavlja vrijednost polja `jeUpisan` na `false`.

Slijedi primjer promijenjene Student klase. Metodi koji nisu mijenjani napisani su u skraćenom obliku.

PRIMJER 5

```
public class Student

{  private String idBroj;
   private String ime;
   private String programStudija;
   private int godina;
   private boolean jeUpisan;

   /* Zabilježi da je student upisan.
   */
   public void upiši()
   {  jeUpisan = true;
   }

   /*  Zabilježi da student nije upisan
   */
   public void ispiši()
   {  jeUpisan = false;
   }

   /* vrati true ako je student upisan
     inače vrati false
   */
   public boolean jeliUpisan()
   {  return jeUpisan;
```

```
    }

    /* Prikaži detalje o studentu
    */
    public void prikaz()
    {  System.out.println
          ("Student ID: " + idBroj);
       System.out.println("Ime: " + ime);
       System.out.println
          ("Program studija: " + programStudija);
       System.out.println("Godina: " + godina);
       if (jeUpisan)
          System.out.println(”Nije upisan.”);
       else
          System.out.println(”Upisan je.”);
    }
```

Definicije klasa setProgramStudija, povecajGodinu i getIme su nepromijenjene

```
     /* Kreiraj novog studenta sa zadanim
        ID brojem, imenom i programom studija
        Polje godina bit će postavljeno na 1.
       Student početno nije upisan.
     */

    public Student(String id, String nm, String prog)
    {  idBroj = id;
       ime = nm;
       programStudija = prog;
       godina = 1;
       jeUpisan = false;
    }
}
```

Kada pišemo logičke izraze možemo koristiti slijedeće relacijske operatore:

==	jednako
!=	nije jednako
<	manje od
<=	manje ili jednako
>	veće
>=	veće ili jednako

Ovi operatori se koriste između bilo koja dva izraza koji označavaju cjelobrojnu ili vrijednost u pokretnom zarezu.
Kompleksnije logičke izraze formiramo korištenjem logičkih operatora.
(B1 i B2 su logički izrazi)

operator	*značenje*
B1 && B2	B1 I B2
B1 || B2	B1 ILI B2
! B1	NEGACIJA B1

Izraz B1 && B2 daje rezultat `true` ako su i B1 i B2 `true`, inače daje `false`. Kada Java izračunava izraz B1 && B2, prvo računa izraz B1 . Ako je B1 `false` izraz B2 se ne izračunava, jer neovisno o njemu cijeli izraz je `false`.

Izraz B1 `||` B2 bit će `true` ako najmanje jedan od B1 i B2 je `true`. Ako su i B1 i B2 vrijednosti `false`, rezultat je `false`. Java također izračunava B1 `||` B2 na 'lijeni način' . Ako je B1 `true` tada B2 se ne računa ! (Zašto?)

Operatori `&&` i `||` su manjeg prioriteta od drugih operatora koje smo dosada sreli. Java će izraz:

```
n >= 0 && n< 10 || n >= 20 && n < 30
```

interpretirati kao:

```
((n>=0) && (n<20)) || ((n>=20) && (n<30))
```

Operator 'not' tj., `!`, je većeg prioriteta nego ostali operatori. Operator = je najmanjeg prioriteta od dosad spomenutih. Stoga možemo izostaviti zagrade u već spomenutom primjeru:

```
boolean ok = (x > 0);
```

U slijedećem dijelu su primjeri upotrebe logičkih operatora.

---

5. Usporedba stringova

Jedan od najčešće upotrebljavanih metoda koja vraća `boolean` vrijednost je metoda `equals`. Svaki string (string je klasa) ima `equals` metodu koja se koristi da li string za kojeg smo pozvali metodu ima istu sekvencu karaktera kao i string koji je proslijeđen kao parametar.

Slijedi primjer u kojemu testiramo da li se string referenciran s st odnosi na isti niz znakova koji su proslijeđeni kao parametar:

```
st.equals("cat")
```

Slijedi primjer jednostavnog programa koji testira sadržaj stringa na navedeni način. program pokušava prepoznati što je korisnik utipkao te daje odgovarajuće odgovore.

PRIMJER 6

```
public class KakoSi
{  /* Pitaj korisnika "Kako je" i daj odgovarajući
        komentar.

   */
```

```
   public static void main(String[] args)
   {  ConsoleReader in =
         new ConsoleReader(System.in);

      System.out.println
         (″Ćao korisniče. kako ide danas ?");
      String odgovor = in.readLine();
      if (odgovor.equals("Dobro"))
         System.out.println
            ("Lijepo je čuti da nekom ide dobro.");
      else if (odgovor.equals("Loše"))
         System.out.println("Žao mi je, valjda će biti bolje");
   }
}
```

Operator == se normalno ne koristi da bi se ispitalo da li su dva stringa ista. Razlog je što Java izraz:

```
S1 == S2
```

gdje S1 i S2 referenciraju stringove interpretira na način da uspoređuje reference, a ne sadržaj stringova. Znači, testira se da li obje reference pokazuju na isti string u memoriji.

Promotrite slijedeći kod:

```
String S1 = “Isti smo”;
String S2 = “Isti smo”;

if(S1==S2){
   System.out.println("Isti")
}
else{
    System.out.println("Različiti")
}
```

Rezultat ispisa bit će "Različiti" jer varijable S1 i S2 referenciraju na **različite stringove** s istim sadržajem.

Korisna alternativa `equals` metodi je slijedeća metoda.

```
st1.equalsIgnoreCase(st2)
```

Ova metoda ne razlikuje velika i mala slova pa ako npr. `st1` je `″the dog″, a st2` je `″The Dog″`, onda će metoda vratiti `true`.

Postoji još mnogo korisnih metoda vezanih za stringove koje možete pogledati u Java dokumentaciji.

Slijedi primjer appleta koji na osnovu korisnikova unosa željenog oblika i boje iscrtava lik na ekranu. Prvo se traži naziv boje te se na osnovu unosa kreira odgovarajući objekt.

Slijedi dio za unos i kreiranje boje:

```
String odgovor1 =
   JOptionPane.showInputDialog
      ("Naziv željene boje?");
if (odgovor1.equalsIgnoreCase("crvena"))
   color = Color.red;
else if (odgovor1.equalsIgnoreCase("žuta"))
   color = Color.yellow;
else if (odgovor1.equalsIgnoreCase("plava"))
   color = Color.blue;
else
   color = null;
```

Program prepoznaje samo nazive ‘crvena’, ‘žuta’ i ‘plava’. Ako korisnik utipka bilo što drugo npr. ‘zelena’ ili ‘%7df&#yx!’, program će spremiti `null` u varijablu `color`.

Slijedeći kod pita korisnika za oblik i rezultat sprema u varijablu `shape`. Program prepoznaje samo dva oblika.

```
String odgovor2 =
   JOptionPane.showInputDialog
      ("Koji oblik želite?");
if (odgovor2.equalsIgnoreCase("kvadrat"))
   shape = new Rectangle2D.Double(50,50,200,200);
else if (odgovor2.equalsIgnoreCase("krug"))
   shape = new Ellipse2D.Double(50,50,200,200);
else
   shape = null;
```

Kad smo postavili vrijednosti u varijable color i shape izvršit ćemo dvije slijedeće naredbe:

```
if (color != null)
   g2.setColor(color);
if (shape != null)
   g2.fill(shape);
```

Ako smo unijeli prepoznatljivu boju Graphics2D objekt bit će postavljen na zadanu boju, a ako nismo , bit će ostavljena njegova prethodna boja.
Ako je program prepoznao lik onda će ga nacrtati, inače ne.

Slijedi kompletan program:

PRIMJER 7

```
// Applet koji vas pita za boju i oblik
// te prikazuje unijeti oblik ispunjen sa zadatom bojom

import java.applet.*;
import java.awt.*;
import java.awt.geom.*;
import javax.swing.*;

public class Oblici extends Applet

{  private Color boja;
   private Shape oblik;

   public void init()
   {
      String odgovor1 =
         JOptionPane.showInputDialog
            ("Naziv željene boje?");
      if (odgovor1.equalsIgnoreCase("crvena"))
         boja = Color.red;
      else if (odgovor1.equalsIgnoreCase("žuta"))
         boja = Color.yellow;
      else if (odgovor1.equalsIgnoreCase("plava"))
         boja = Color.blue;
      else
         boja = null;


      String odgovor2 =
         JOptionPane.showInputDialog
            ("Koji oblik želite?");
      if (odgovor2.equalsIgnoreCase("kvadrat"))
         oblik = new Rectangle2D.Double(50,50,200,200);
      else if (odgovor2.equalsIgnoreCase("krug"))
         oblik = new Ellipse2D.Double(50,50,200,200);
      else
         oblik = null;
   }

   public void paint(Graphics g)
   {  Graphics2D g2 = (Graphics2D) g;

      /* Prikaži obojani lik. */

      if (boja != null)
         g2.setColor(boja);

      if (oblik != null)
         g2.fill(oblik);
   }
}
```

Analizirajmo formu appleta. Postoje dva osnovna zadatka:

Korak 1. Unesi željenu boju i oblik.

Korak 2. Naslikaj željeni lik u željenoj boji.

Drugi korak će se izvršavati svaki put kad bude potrebno osvježiti sadržaj područja prikaza. Stoga je uključen u applet `paint` metodu.
S druge strane prvi korak treba biti izvršen samo jednom pa je uključen u `init` metodu.

Za zadnji primjer u ovom poglavlju slijedi dodavanje još jedne metode u klasu Student. Nazvat ćemo je `edit`. Koristit će `equals` metodu za usporedbu stringova i logičke operatore.

```
s.edit(in)
```

Prikazuje podatke o objektu `s` tipa Student, pita korisnika da li želi napraviti kakve promjene. Ako želi `s` se odgovarajuće ažurira. Korisnikov unos učitavamo pomoću ConsoleReader `objekta`.

Metoda `edit` prikazuje redom vrijednost svakog polja, i čeka korisnikov odgovor. Ako je korisnik zadovoljan s trenutnom vrijednošću onda utipka samo Return (Enter). Ako unese neki string metoda će pretpostaviti da se radi o novoj vrijednosti i spremiti je u odgovarajuće polje.

Primjer mogućeg prikaza na ekranu prilikom izvršavanja metode edit:

```
ID (9912345)
Ime (Jure Antić)
Program studija (Računarstvo) CS
Godina (1) 2
Upisan? (Da) no
```

Svaka se linija sastoji od naziva informacije i njene trenutne vrijednosti te podebljano ispisanog korisnikovog unosa. U slučaju prva dva retka korisnik je samo utipkao Enter i odgovarajuće vrijednosti su ostale iste. Slijedeća tri retka korisnik je unio nove podatke i potrebno ih je pohraniti u odgovarajuća polja.

Slijedi kod koji radi s ID brojem.

```
System.out.print("ID (" + idBroj +") " );
String odgovor = input.readLine();
if (! odgovor.equals(""))
   idBroj = odgovor;
```

Za studentovo ime postupa se skoro na isti način.

Za program studija se postupa na malo drukčiji način.

ispitivanje sadržaja stringa obavlja se na slijedeći način:

```
if (!odgovor.equals(""))
   if (odgovor.equals("Elektronika") ||
        odgovor.equals("Računarstvo") ||
        odgovor.equals("Strojarstvo") ||
        odgovor.equals("Brodogradnja") )
      programStudija = odgovor;
   else
      System.out.println("Nepoznat program studija.");
```

Metoda `edit` postupa s poljem `jeUpisan` opet na različit način. Umjesto ispisa stvarnog sadržaja ispisuje se 'da' ili 'ne' za indikaciju upisa studenta. Korisnik mora odgovoriti s 'da' or 'ne' ako želi napraviti promjenu u tom polju

Slijedi potpuna metoda `edit`:

PRIMJER 8

```
    /* Editiraj studentove podatke
    */

    public void edit(ConsoleReader input)
    {
       System.out.print("ID (" + idBroj +") " );
       String odgovor = input.readLine();
       if (! odgovor.equals(""))
          idBroj = odgovor;

       System.out.print("Ime (" + ime + ") ");
       odgovor = input.readLine();
       if (! odgovor.equals(""))
          ime = odgovor;

       System.out.print
          ("Program studija (" + programStudija + ") ");
       odgovor = input.readLine();
       if (! odgovor.equals(""))
          if (odgovor.equals("Elektronika") ||
             odgovor.equals("Računarstvo") ||
             odgovor.equals("Strojarstvo") ||
             odgovor.equals("Brodogradnja") )
            programStudija = odgovor;
          else
             System.out.println("Nepoznat program studija.");

       System.out.print("Godina (" + godina + ") ");
       odgovor = input.readLine();
       if (! odgovor.equals(""))
          godina = Integer.parseInt(odgovor);

       if (jeUpisan)
```

```
         System.out.print("Upisan? (da) ");
      else
         System.out.print("Upisan? (ne) ");
      odgovor = input.readLine();
      if (! odgovor.equals(""))
         if (odgovor.equals("da"))
            jeUpisan = true;
         else if (odgovor.equals("ne"))
            jeUpisan = false;
         else
            System.out.println("Odgovor nije razumljiv.");
   }

   public class TestStudent

   {  /* Čita podatke o studentu i kreira odgovarajući
         Student objekt.
         Zatim editira objekt Student
          te na kraju prikazuje sadržaj.
      */

      public static void main(String[] args)

      {  /* Kreiraj objekt tipa ConsoleReader za učitavanje
            podataka s tastature */

         ConsoleReader in =
            new ConsoleReader(System.in);

         /* Kreiraj objekt Student. */

         System.out.println
            ("Koji je ID broj studenta?");
         String i = in.readLine();
         System.out.println("Koje je ime studenta?");
         String n = in.readLine();
         System.out.println("Koji je program studija?");
         String d = in.readLine();
         Student st = new Student(i,n,d);

         /* Editiraj pa prikaži podatke studenta. */

         System.out.println();
         System.out.println("Editiraj podatke studenta.");
         st.edit(in);
         System.out.println();
         System.out.println("Student podaci:");
         st.prikaz();
      }
   }
```

Slijedi što bi program mogao ispisivati (unos korisnika podebljano):

```
Koji je ID broj studenta?
9912345
Koje je ime studenta?
Jure Antić
Koji je program studija?
Računarstvo

Editiraj podatke studenta.
ID (9912345) 9912346
Ime (Jure Antić)
Program studija (Računarstvo) Elektronika
Godina (1) 2
Upisan (ne) da

Student podaci:
Student ID: 9912346
Ime: Jure Antić
Program studija: Elektronika
Godina: 2
Upisan.
```

U kodu programa postoji izraz slijedećeg oblika(ispitivanje polja programStudija):

```
if (B1) if (B2) S1 else S2
```

Kojemu `if` pripada `else`.
Ovo je primjer tzv. visećeg else.

Java će to interpretirati na slijedeći način:

1. **`if (B1) { if (B2) S1 else S2 }`**

a ne ovako:

2. `if (B1) { if (B2) S1 } else S2`

Pravilo je da `else` tvori cjelinu s najbližim `if`-om.

U ovim slučajevima preporučljivo je koristiti vitičaste zagrade !

Kod sa vitičastim zagradama:

```
if (! odgovor.equals(""))
{  if (odgovor.equals("da"))
      jeUpisan = true;
   else if (odgovor.equals("ne"))
      jeUpisan = false;
```

```
      else
         System.out.println("Odgovor nije razumljiv.");
   }
```

---

6. Blokovi i doseg varijabli

Zapamtite da je blok sekvenca naredbi zatvorena u vitičaste zagrade. Jedna je bitna stvar u radu s blokovima u Javi. Varijabla koju deklaracijom unutar bloka vrijedi samo unutar bloka. čim Java napusti blok varijable iščezava tj. oslobađa se memorija koja je za nju bila korištena. promotrite slijedeći primjer:

```
if (x == 0)
{  System.out.println(″x is nula.″);
   int y = 1;
} else {
   System.out.println(″x nije nula.″);
   int y = 2;
}
System.out.println(″y je ″ + y);
```

Ovaj program će javiti pogrešku u toku prevođenja. varijabla y ne vrijedi izvan bloka u kojem je deklarirana. Što više programer je kreirao dvije posebne varijable y unutar posebnih blokova.

Dio programa u kojemu možemo koristiti određenu varijablu naziva se **doseg** (scope) te varijable.

prethodni program ima više smisla (kompajlirat će se) ako ga preuredimo na slijedeći način:

```
int y;
if (x == 0)
{  System.out.println(″x je nula.″);
   y = 1;
} else {
   System.out.println(″x nije nula.″);
   y = 2;
}
System.out.println(″y je ″ + y);
```

varijabla deklarirana unutar bloka naziva se **lokalna varijabla**. Tijelo metoda je također blok pa tako svaka varijabla koja je deklarirana unutar metode je lokalna varijabla.
To znači da bi svaka varijabla deklarirana unutar metode trebala prestati s postojanjem nakon izlaska iz metode.

S druge strane, polje objekta postoji sve dok postoji objekt kome pripada. Objekt traje sve dok Java ne zaključi da se više ne upotrebljava. Taj proces nije nimalo jednostavan kako to pokazuje naziv **garbage collection** (odvoz smeća).

Činjenica da lokalne varijable vrijede samo dok se metod izvršava objašnjava nam zašto smo u primjeru 7 upotrijebili dva nova polja Color i Shape. Pošto smo tim referencama u Init metodi dodijelili objekte boje i oblika, reference su trebale ostati sačuvane da bi objektima mogla pristupiti paint metoda.

---

7. Zadaci

1. Napiši program (TriBroja.java) koji čita tri broja u pokretnom zarezu i ispisuje najveći od njih:

   Npr:

   ```
   Unesi tri broja:
   4  9  2.5
   Najveći je broj 9.
   ```

2. Napiši applet (KrugTest.java, Krugtest.txt (HTML))koji crta krug radijusa 100 s centrom u (110,120). Zatim traži od korisnika da unese koordinate neke točke. Ako točka leži unutar Kruga onda ispisuje poruku "Unutar kruga", a ako leži vani poruku "Van kruga".

3. Godina s 366 dana naziva se prestupna godina. Godina je prestupna ako je djeljiva s 4 (npr. 1980), osim ako nije djeljiva s 100 (npr. 1900). Izuzetak su godine djeljive s 400 (npr. 2000) koje su prestupne u svakom slučaju. Nije bilo prestupnih godina prije uvođenja Gregorijanskog kalendara na dan 15.10.1582 (npr., 1500. ipak nije bila prestupna). napiši program koji će korisnika pitati za godinu , a onda ispisati je li ili nije prestupna.

   Provjera 1996 i 2000 su prestupne. 1900 i 1999 nisu.