



Funded by the
Erasmus+ Programme
of the European Union



This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



KI103 - JAVA 1: OSNOVE PROGRAMIRANJA U JAVI

Operatori i grananja

Lekcija 03

PRIRUČNIK ZA STUDENTE

KI103 - JAVA 1: OSNOVE PROGRAMIRANJA U JAVI

Lekcija 03

OPERATORI I GRANANJA

- ✓ Operatori i grananja
- ✓ Poglavlje 1: Relacioni i logički operatori
- ✓ Poglavlje 2: Iskaz if
- ✓ Poglavlje 3: Iskaz if - else
- ✓ Poglavlje 4: Iskaz else-if
- ✓ Poglavlje 5: Naredba switch
- ✓ Poglavlje 6: Uslovni operator (?)
- ✓ Poglavlje 7: Domaći zadaci
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

▼ Uvod

UVOD

Kontrolna stuktura programa upravlja putevima izvršenja programskih instrukcija. Različite vrste grananja puteva izvršenja programa određuju se u tačkama odlučivanja u programima.

Cilj ove lekcije je studenti ovladaju kontrolnom strukturom programa koja određuje redosled izvršavanje programskih naredbi (iskaza). Izvršenje programskih inskaza ne mora uvek da ide redosledom njihovog navođenja u programu. Često u programima imamo potrebu da uvedemo tzv. granjanja koja usmeravaju izvršenje programskih iskaza u dva ili više različitih programskih puteva (niti). Mesto u kome nastaje granjanje programa je mesto odluke, i ono obično zavisi od definisanih uslova . Svaka grana programa yahteva da se prethodno ispune određeni uslovi da bi se program mogao da izvršava određenom granom u programu.

U ovoj lekciji ćemo pokazati kako se definišu uslovi grannjanja u programima, i kakvih sve vrsta granjnanja se mogu koristiti. Mi ćemo navesti primere u Javi, ali vrlo slični iskazi se koriste i kod drugih programskih jezika (npr. C++ ili C#).

▼ Poglavlje 1

Relacioni i logički operatori

RELACIONI OPERATORI

Relacioni operator upoređuje dve vrednosti.

Relacioni operator upoređuje dve vrednosti. Promenljiva čije se vrednost upoređuje, naziva se operand, a **relacioni operator** izvršava operaciju upoređivanja vrednosti koje imaju dva operanda.

Upoređenje vrednosti mogu biti različita (veće, manje, jednako i dr.), te postoje različiti operatori.

Operator Upotreba Vraća **true**(tačno) ako je

> op1 > op2 op1 veće op2

>= op1 >= op2 op1 veće ili jednako od op2

< op1 < op2 op1 manje op2

<= op1 <= op2 op1 manje ili jednako op2

== op1 == op2 op1 i op2 su jednaki

!= op1 != op2 op1 i op2 su različiti

Operandi op1 i op2 čije se vrednosti upoređuju, mogu biti bilo kog numeričkog tipa, ali i tipa **char**. Za znakove tipa char, operatori < i > su definisani prema numeričkim Unicode vrednostima znakova.

Tako, 'a' < 'z' kao rezultat daje true, jer je brojni kod znaka 'a' manji od brojnog koda znaka 'z' u Unicode šemi. Obratite ipak pažnju na to da redosled slova u ovoj šemi nije isto što i njihov alfabetski redosled, jer se velika slova nalaze ispred svih malih slova

Uzmimo dve promenljive x i y da budu operndi (op1 i op2). Vrednosti **x i y** mogu biti čak i logičke vrednosti u slučaju operatora "jednako" (==) i "nije jednako" (!=).

Izraz **x > y** je relacioni izraz. Relacioni izraz je izraz koji vraća boolean (logičku) vrednost, a koji za izračunavanje vrednosti koristi relacioni operator. Ovi izrazi daju neku **logičku vrednost** istinito-**true** ili neistinito-**false**.

Relacioni operatori se uglavnom koriste u okviru uslovnih iskaza, o kojima će kasnije biti više reči.

Relacioni operatori se često koriste zajedno sa logičkim operatorima, čime se dobijaju složeni izrazi.

PRIMER 1: PRIMENA RELACIONIH OPERATORA

Za definisane vrednosti operanada, relacioni operator daje jednu od dve logičke vrednosti: true (istinito) ili false (neistinito)

U sledećem primeru je pokazano kako bi mogli da se koriste relacioni operatori.

Izvršenje prikazanog programa daje sledeći rezultat:

Veće od...

$i > j = \text{false}$

Veće ili jednako...

$i \geq j = \text{false}$

Manje od...

$i < j = \text{true}$

Manje ili jednako...

$i \leq j = \text{true}$

Jednako...

$i == j = \text{false}$

Različito...

$i != j = \text{true}$

```
public class RelacioniOperatori {
    public static void main(String[] args) {
        //Nekoliko brojeva
        int i = 37;
        int j = 42;
        int k = 42;
        //vece od
        System.out.println("i > j = " + (i > j)); //false
        //vece ili jednako
        System.out.println("Veće ili jednako...");
        System.out.println("    i >= j = " + (i >= j)); //false
        //manje od
        System.out.println("Manje od...");
        System.out.println("    i < j = " + (i < j)); //true
        //manje ili jednako
        System.out.println("Manje ili jednako...");
        System.out.println("    i <= j = " + (i <= j)); //true
        //jednako
        System.out.println("Jednako...");
        System.out.println("    i == j = " + (i == j)); //false
        //razlicito
```

```
        System.out.println("Različito...");  
        System.out.println("    i != j = " + (i != j));    //true  
    }  
}
```

ZADATAK 1

Samostalno vežbanje primene relacionih operatora

Zadatak 1-1:

Pretpostavljajući da je x ima vrednost 1, pokaži rezultate sledećih logičkih naredbi:

(true) && (3 > 4)

!(x > 0) && (x > 0)

(x > 0) || (x < 0)

(x != 0) || (x == 0)

(x >= 0) || (x < 0)

(x != 1) == !(x == 1)

Zadatak 1-2:

Da li su sledeći iskazi jednaki?

a. $x \% 2 == 0 \ \&\& \ x \% 3 == 0$

b. $x \% 6 == 0$

Zadatak 1-3:

Koja je vrednost iskaza: $x \geq 50 \ \&\& \ x \leq 100$ ako je x 45, 67 i 101?

RELACIONI OPERATORI (VIDEO)

Video objašnjava primenu relacionih opratora u Javi.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

UPOREĐIVANJE OBJEKATA

Dve objektne reference (promenljive ili konstante u programu) su jednake ako ukazuju na isti objekt. To znači da su jednake ako ukazuju na istu memorijsku lokaciju.

Dve objektne reference (promenljive ili konstante u programu) su jednake ako ukazuju na isti objekt. To znači da su jednake ako ukazuju na istu memorijsku lokaciju.

```
Integer n = new Integer(10);
Integer m = n; // m se poziva na istu memoriju kao i n
if ( m == n ) /* Tačno - memorijske lokacije su iste */ ;
m = new Integer(10); // sada m predstavlja novi Integer objekt
if ( m == n ) /* Pogrešno - m predstavlja drugi objekt*/;
String s = "HELLO"; // kreira String objekt sa vrednošću "HELLO"
String t = "HEL"; // kreira String objekt sa vrednošću "HEL"
t = t + "LO"; // kreira String object sa novom vrednošću
if ( s == t ) /* Pogrešno -- s i t su različiti objekti*/;
```

```
// upređenje String objekata upotrebom ==
String s = "HELLO";
String t = new String("HELLO");
String u = "HEL" + "LO";
if ( s == t ) /* Pogrešno -- s i t nisu isti objekti*/ ;
if ( s == u ) /* Pogrešno -- i ovi objekti nisu isti */;
if ( s == "HELLO" ) /* Tačno -- Java stavlja String konstante u zajedničku
memoriju, te s pokazuje na istu konstantu */;
if ( t == "HELLO" ) /* Pogrešno -- t (objekt) ne koristi istu memoriju kao string
konstanta "HELLO" */ ;
```

Upoređivanje objekata prekrivačkih klasa (wrapper)

```
// upotređenje objekata sa ==

Integer x = new Integer(2);

Integer y = new Integer(2);

Integer z = x;

if ( x == y ) /* Pogrešno -- x i y nisu isti objekti*/ ;

if ( x == z ) /* Tačno-- x i z su isti objekti */;

// kreiranje novog Integer objekta i inicijalizacija sa x
Integer z = new Integer( x );
if ( x == z ) /* Pogrešno -- x i z nisu isti objekti*/ ;
```


Za primitivne tipove podataka, vrednost promenljive ili konstante je stvarna vrednost podatka.

.

```
int x = 2;
char c = 'a'; // Vrednost za x je 2. Vrednost za c je 97 ('a')
```

Vrednost promenljive u memorijskoj lokaciji je objekt

```
String s = "Hello";
// Vrednost za s može biti 0x2F4169 (adresa objekta).
```

METOD EQUALS()

Metod equals() upoređuje vrednosti atributa objekata istog tipa i daje vrednost true ako njihovi atributi imaju iste vrednosti.

Kod objekata (uključujući String objekte) == upoređuje da li dve promenljive ukazuju na isti objekt. Kako možemo da uporedimo ako je vrednost objekata jednaka? Za ovo, klasa objekta mora da obezbedi metod **equals()**

```
String s = new String("HELLO");
String t = "HEL";
t = t + "LO";
if ( s == t ) /* Pogrešno -- s i t su različiti objekti*/ ;
if ( s.equals(t) ) /* Tačno– vrednost za s i t su iste*/;
```

equals() je metod instance definisan u mnogim klasama **equals()** je definisan u klasi **Object**, tako da bilo koji objekt može da nasledi metod **equals()** iz klase **Objekt**.

Ali, (Object).equals() ne radi!. (Object).equals() upotrebljava ==

Svaka klasa bi trebalo da definiše svoje metode **equals()**!

```
Integer n = new Integer( 10 );
Integer m = new Integer( 10 );
if ( n == m ) /* Pogrešno -- m, n su različiti objekti*/ ;
if ( n.equals(m) ) /* Tačno – Klasa Integer definiše equals() kao upoređenje
vrednosti objekta */;
```

Najveći broj bibliotečkih Java klasa imaju svrsishodna equals() metod. Ovo uključuje klase : **String, Byte, Boolean, Character, Integer, Long, Float, Double** (klase prekrivače)

Neki grafički tipovi: **java.awt.Point** (tačke su jednake ako su i x i y koordinate iste) itd.

```
Point p = new Point( 5, 10 );
Point q = new Point( 0, 0 );
if ( p.equals( q ) ) /* Pogrešno – koordinate nisu iste */;
q.move( 5, 10 );
if ( p.equals( q ) ) /* Tačno – koordinate su iste */
```

Pretpostavimo da imamo klasu imenovanu sa Point koja sadrži x,y-koordinate tačke u ravni. Kreiramo i upotrebljavamo objekte Point na sledeći način:

```
Point p = new Point( 5, 10 ); // tačka sa koordinatama (5,10)
Point q = new Point( 0, 0 );
q.move( 3, 4 ); // pomera u (3,4)
...itd...
```

UPOTREBA METODA EQUALS()

Metod equals() se definiše za svaku klasu, u skladu sa njenim specifičnostima (atributima), tako da omogući upoređenje vrednosti atributa objekata iste klase.

Karakteristike metoda **equals()** su:

- to je javna (public) metoda definisana u vašoj klasi
- signatura (način pisanja):

public boolean equals(Object drugo)

- upoređuje neke *atribute* (*promenljive objekta*)

Vi odlučujete šta za objekte znači da budu "jednaki"!

Ovde se daje jedan metod **equals()** za upoređenje Point objekata

```
class Point {
    private double x, y; // atributi: koordinate tačke
    // "drugo" trebalo bi da budu tipa Object, ali pravite to jednostavno...
    public boolean equals( Point drugo ) {
        if (drugo== null ) return false; /* ovo je važno! */
        return this.x ==drugo.x &&&&& this.y ==drugo.y;
    }
}
```

equals() bi trebalo da bude simetričan. To znači da bi:

a.**equals(b)** trebalo da vrati istu vrednost kao i **b.equals(a)**

Ako su atributi neke klase objekti ili neki složeni tipovi (String, vektor, etc.) vaš **equals()** metod mora da upoređuje svaki atribut upotrebom nekog odgovarajućeg upoređivača.

```
public Osoba {
    private String ime;
    private String prezime;
    private String id;
    private String [ ] email; // niz e-mail adresa
    private char pol;
    ...itd..
}
```

```
public class Osoba {
    ...itd..
    public boolean equals ( Object drugi) {
        if (drugi == null ) return false;
        if ( this == drugi) return true;
        // Da li obe promenljive predstavljaju objekte istog tipa?
        if ( this.getClass( ) !=drugi.getClass( ) ) return false;
        // Drugi objekt nije istog tipa, tj. nije klase Osoba.
        // Konvertujemo drugu promenljivu u naš tip, tj. u klasu Osoba:  Osoba who =
        (Osoba) drugi;
        return przime.equalsIgnoreCase( who.prezime )
            &&&& ime.equalsIgnoreCase( who.ime)
            &&&& id.equals( who.id );
    }
}
```

KORIŠĆENJE VITIČASTIH ZAGRADA

Ako se if deo if-else naredbe sastoji od jedne naredbe ne mora da ude unutar para vitičastih zagrada, ali može!

U sledećem programskom fragmentu se računa obim kvadrata za datu veličinu stranice. Pri tome se koristi **if-else** naredba radi provere ispravnosti vrednosti veličine stranice koju unosi korisnik:

```
if (x > y) {
    int temp; // Pomoćna promenljiva
    temp = x; // Vrednost z je stara vrednost x
    x = y; // Nova vrednost x je stara vrednost y
    y = temp; // Nova vrednost y je stara vrednost x
}
```

U ovom primeru se **if** deo **if-else** naredbe sastoji od jedne naredbe zato ne mora da bude unutar para vitičastih zagrada. Sa druge strane, **else** deo **if-else** naredbe u primeru se sastoji od tri naredbe i zato se mora nalaziti unutar para vitičastih zagrada u formi bloka.

Napomena:

Ako se **if** deo **if-else** naredbe sastoji od jedne naredbe ne mora da ude unutar para vitičastih zagrada, ali može!. Neki programeri uvek koriste blokove za sastavne delove **if** i **if-else** naredbi, čak i kada se oni sastoje od samo jedne naredbe, između ostalog zato što se na ovaj način lako dodaju dodatne naredbe u blok, što nije slučaj ako nemamo par vitičastih zagrada.. Razvojno okruženje NetBeans prilikom formatiranja koda uvek dodaje vitičaste zagrade, tj. i jednu naredbu tretira kao blok naredbi.

UPOREĐIVANJE STRINGOVA

Metod `equals()` upoređuje dva stringa. Metod `compareTo()` daje redosled stringova. Operator `==` ispituje da li dva objekta isti, a `equals()` - da li oni imaju iste vrednosti svih atributa.

Pri upoređenju tekstova (stringova), može vam pomoći sledeća tabela:

| | |
|--|---|
| <pre>String str = "Hello"; str.equals("hello") str.equalsIgnoreCase("hello")</pre> | <p>Upoređuje dva stringa. <code>equals</code> upoređuje sa "case", <code>equalsIgnoreCase</code> uzima isto i velika i mala slova.</p> |
| <pre>str.startsWith("he") str.endsWith("lo") str.contains("el")</pre> | <p>Testira da li <code>str</code> počinje sa..., završava se sa..., ili sadrži specificiran string.</p> |
| <pre>String str2 = "Hello George"; str.compareTo(str2) str.compareToIgnoreCase(str2)</pre> | <p>< 0 ako <code>str</code> ide pre <code>str2</code> = 0 ako <code>str</code> je "isto" kao <code>str2</code> > 0 ako <code>str</code> ide posle <code>str2</code></p> |

Slika 1.1 Metodi klase `String` za upoređenje stringova

<p></p>

Rezime:

Za promenljive koji su tipa objekt (objektne reference):

- `(a == b)` testira da li promenljive ukazuju na isti objekt.
- `a.equals(b)` upoređuje vrednost dva objekta

Ako su objekti klase koja primenjuje **`compareTo`**:

- **`a.compareTo(b)`** vraća: < 0 ako je `a` "pre" `b` (u nekom leksičkom redosledu)
 = 0 ako `a` ima isti redosled kao `b`
 > 0 ako je `a` "posle" `b`

UPOREĐENJE STRINGOVA (VIDEO)

Video daje primere u kojima se upoređuju stringovi u njihovoj upravljačkoj strukturi.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMER 2

Poređenje objekata po == i po equals

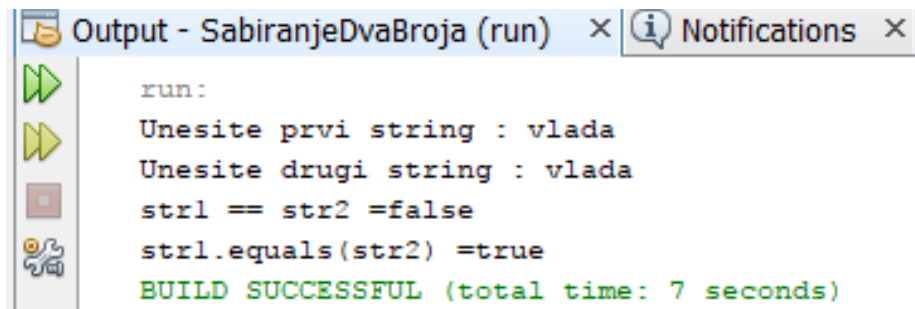
Kreirajte program koji sa tastature unosi dva različita stringa koja čuva pomoću promenljivih str1 i str2.

Testirajte šta će se desiti ako unesete za tastature iste vrednosti:

- u slučaju iskaza str1==str2;
- u slučaju iskaza str1.equals(str2).

Rešenje:

Budući da su str1 i str2 različiti objekti, u prvom slučaju će rezultat uvek biti false. U drugom slučaju, ako unesete identične stringove sa tastature, rezultat će biti true, u suprotnom false.



Slika 1.2 Rezultat izvršavanja programa

```
package l02;

import java.util.Scanner;

/**
 * *
 * @author Vladimir Milićević
 */
public class PoredjenjeStringova {
```

```
public static void main(String[] args) {  
    /*kreiranje Sting referenci kojima su pridružena dva stinga*/  
    String str1;  
    String str2;  
    Scanner tastatura = new Scanner (System.in);  
    //Prva grupa rezultata  
    System.out.print("Unesite prvi string : ");  
    str1 = tastatura.nextLine();  
    System.out.print("Unesite drugi string : ");  
    str2 = tastatura.nextLine();  
    /* radi se o različitim objektima i vratiće false*/  
    System.out.println("str1 == str2 =" + (str1 == str2));  
    /*rezultat zavisi od stringova koji se čuvaju u str1 i str2*/  
    System.out.println("str1.equals(str2) =" + str1.equals(str2));  
  
    }  
}
```

ZADATAK 2

Vežbanje poredjenja stringova

- Otvorite zadatak iz primera Primer2;
- kao 28. liniju koda dodajte `str1 = str2`;
- nakon toga ponovite linije koda 25 i 27;
- Kakav će biti rezultat? Zašto?

METOD EQUALS() (VIDEO)

Video preko primera objašnjava primenu metoda `equal()`.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

LOGIČKI OPERATORI

Logički operatori tipa I, ILI i NE

Kombinovanje relacionih izraza u veće relacione izraze se vrši pomoću **logičkih operatora**. Logički operatoru za logičke izraze ono što su aritmetički operatori za numeričke izraze. Logički operatori se primenjuju na logičke vrednosti i kao rezultat daju isto tako logičku vrednost.

Logički operatori deluju na operande koji su logičkog (**boolean**) tipa. Najvažniji logički operatori su

- logičko I (&&)
- logičko ILI (||)
- logička negacija (!)

Logičko I (&&):

Ovaj operator se u Javi označava sa `&&`. Vrednost izraza sa logičkim `I` je tačna ako su oba operanda tačna. Ako je makar jedan od njih netačan (ili oba), ceo izraz je netačan. To se može predstaviti sledećom tabelom:

| && | tačno | netačno |
|---------|-------|---------|
| tačno | true | false |
| netačno | false | false |

Slika 1.3 Tabela za operciju "I"

Logičko ILI (||):

Ovaj operator se u Javi označava sa `||`. Vrednost izraza sa logičkim `||` je tačna ako je makar jedan od operandata tačan. Izraz je netačan samo ako su oba netačna. To se može predstaviti sledećom tabelom.

| | tačno | netačno |
|---------|-------|---------|
| tačno | true | true |
| netačno | true | false |

Slika 1.4 Tabela za operaciju "ILI"

Logička negacija (!):

Ovaj operator se u Javi označava znakom uzvika (!). Ovaj operator deluje na jedan operand tipa boolean i menja njegovu vrednost. Ako je vrednost operanda tačno, operator negacije je menja na netačno i obrnuto. Ovo se može predstaviti sledećom tabelom:

| ! | tačno | netačno |
|---|-------|---------|
| | false | false |

Slika 1.5 Tabela za operaciju "NE"

PRIMENA LOGIČKIH OPERATORA

Primeri sa logičkim operatorom &&&&&

Primer sa operatorom l (&&):

Pretpostavimo da je profesor rešio da svoje studente koji su polagali test, oceni na sledeći način:

1. Oni koji su osvojili od 50 do 60 poena dobijaju ocenu 6.

2. Oni koji su osvojili od 60 do 70 poena dobijaju ocenu 7.
3. Oni koji su osvojili od 70 do 80 poena dobijaju ocenu 8.
4. Oni koji su osvojili od 80 do 90 poena dobijaju ocenu 9.
5. Oni koji su osvojili od 90 do 100 poena dobijaju ocenu 10.

Ovo ćemo izraziti preko logičkih operatora. Ako se broj poena smesti u promenljivu *ocena*, onda bismo mogli da napišemo:

1. `ocena > 50 && ocena < 60`
2. `ocena > 60 && ocena < 70`
3. `ocena > 70 && ocena < 80`
4. `ocena > 80 && ocena < 90`
5. `ocena > 90 && ocena < 100`

Primer sa operatorom negacije (!):

Operand na koji deluje operator logičke negacije može biti i složen relacioni ili logički izraz. U tom slučaju je potrebno taj izraz staviti u zagrade.

`! (a > b && a < c)`

Kada se ovaj operator koristi kod složenih izraza treba voditi računa o njegovom prioritetu. On ima viši prioritet u odnosu na relacije i aritmetičke operatore, tako da se izvršava pre njih. Pogledajte sledeći izraz u kojem se pita da li računar ima brzinu veću od 2200 MHz i memoriju veću od 512 MB. Negacija izraza bi bila:

`! (brzina > 2000 && memorija > 512)`

-----+-----+-----

`! (T && T)`

-----+-----

`! (T)`

`!T`

F

PRIMER 3

Pisanje programa koji proverava da li je godina prestupna

Napisati program koji proverava da li je uneta godina prestupna.

Prestupna godina je godina koja je deljiva sa brojem 4, ali ne i sa brojem 100, ili godina koja je deljiva sa brojem 400.

Da bi odredili da li je neka godina prestupna, ili ne, možete koristiti sledeći deo programskog koda:


```
// Prestupna godina je deljiva sa 4
boolean isLeapYear = (year % 4 == 0);
// Prestupna godina je deljiva sa 4, ali ne i sa 100
isLeapYear = isLeapYear &&&& (year % 100 != 0);
// Prestupna godina je deljiva sa 4, ali ne i sa 100 ili je deljiva sa 400
isLeapYear = isLeapYear || (year % 400 == 0);
```

Relacioni operator `!=` proverava da li su vrednosti koje se porede različite.

Ceo gornji deo programskog koda možete zameniti samo sa jednom programskom linijom:

```
isLeapYear = (year % 4 == 0 &&&& year % 100 != 0) || (year % 400 == 0);

sa
```

Primer:

- 2017 - Nije prestupna. Nije deljiva sa 4, a ni sa 400.
- 2000 - Prestupna. Deljiva je sa 4, sa 100, ali i sa 400.

Proveriti zašto je 1996. prestupna godina.

Rešenje:

```
package Zadatak2;
import java.util.Scanner;
public class Zadatak2 {

    public static void main(String[] args) {
        // Kreiranje objekta Scanner
        Scanner input = new Scanner(System.in);
        System.out.print("Unesi godinu: ");
        int year = input.nextInt();
        // Provera da li je uneta godina prestupna
        boolean isLeapYear
            = (year % 4 == 0 &&&& year % 100 != 0) || (year % 400
== 0);
        // Prikaz rezultata
        System.out.println(year + " Da li je prestupna? " + isLeapYear);
        input.close();
    }
}
```

Objašnjenje:

Od korisnika se zahteva da preko konzole unese ceo broj koji predstavlja godinu. Unos se vrši pomoću objekta klase *Scanner*. Sledećim logičkim izrazom se proverava da li je godina prestupna:

```
(year % 4 == 0 &&&& year % 100 != 0) || (year % 400 == 0);
```

Vrednost provere se čuva u promenljivoj *isLeapYear* tipa **boolean**, čija vrednost može biti **true** ili **false**. Poslednja linija ispisuje vrednost promenljive *isLeapYear*.

ZADATAK 3

Samostalno vežbanje primene logičkih operatora.

Zadatak 3-1:

Kreirajte program koji proverava da li od stranica a, b i c može da se konstruiše trougao:

- stranice su tipa int i unose se sa tastature;
- kreirajte složeni izraz kojim se proverava da li je dužina neke stranice kraća od zbira dužine druge dve;
- ukoliko ovaj izraz da vrednost true, može da se konstruiše trougao.

Zadatak 3.2:

Napišite logički iskaz koji daje true ako je promenljiva godine veća od 13 a manja od 18.

LOGIČKI OPERATORI (VIDEO)

Video preko primera objašnjava primenu logičkih operatorea (if,...)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

OPERATORI DODELE

Znak = predstavlja zapravo binarni operator dodele u smislu da operacija dodele vrednosti daje rezultat, koji se eventualno može koristiti u sklopu nekog složenijeg izraza.

Do sada smo govorili o **naredbi dodele** kojom se vrednost nekog izraza na desnoj strani znaka = dodeljuje promenljivoj koja se nalazi na levoj strani znaka =. Međutim, znak = predstavlja zapravo **binarni operator dodele** u smislu da operacija dodele vrednosti daje rezultat, koji se eventualno može koristiti u sklopu nekog složenijeg izraza.

Vrednost operatora dodele u opštem obliku

```
ime = izraz;
```

jednaka je vrednosti izraza na desnoj strani znaka jednakosti. Prema tome, efekat operatora dodele je dvojak: nakon izračunavanja izraza se njegova vrednost najpre dodeljuje promenljivoj ime i zatim se daje kao rezultat samog operatora =.

Operator dodele u Javi ima, u stvari, nekoliko varijacija koje se mogu koristiti radi kraćeg pisanja. Na primer:

```
ime += izraz;
```

ekvivalentno je sa:

```
ime = ime + izraz;
```

Svaki binarni operator u Javi se može koristiti na sličan način sa operatorom dodele.

Na primer:

| Operator | Operacija | Primer | Efekat |
|----------|-----------------------|-----------|----------------|
| += | Sabiranje sa dodelom | res += 5; | res = res + 5; |
| -= | Oduzimanje sa dodelom | res -= 5; | res = res - 5; |
| *= | Množenje sa dodelom | res *= 5; | res = res * 5; |
| /= | Deljenje sa dodelom | res /= 5; | res = res / 5; |

Kombinovani operator dodele += se može primeniti i na stringove. Podsetimo se da operator + u kontekstu stringova označava njihovo spa-janje. Pošto je s+=t isto što i s = s + t, ako su s i t stringovi, onda je nova vrednost stringa s jednaka njegovoj staroj vrednosti kojoj su dodati znakovi stringa t. Na primer, ako s ima vrednost "Java", onda se naredbom s += "n"; menja njegova vrednost u "Javan".

OPERATORI DODELE (VIDEO)

Video daje prikaz većeg broja operatora dodele.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

SKRAĆENI OPERATORI DODELE (VIDEO)

Video primenu skraćениh operatola dodele.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIORITETI OPERATORA

Ukoliko se u nekom izrazu pojavljuje više operatora i ako zagradama nije eksplicitno naznačen redosled njihovog izračunavanja, onda se oni primenjuju po unapred definisanom prioritetu.

Ukoliko se u nekom izrazu pojavljuje više operatora i ako zagradama nije eksplicitno naznačen redosled njihovog izračunavanja, onda se oni primenjuju po unapred definisanom **prioritetu**.

U tabeli su dati svi do sada pomenuti operatori u opadajućem redosledu njihovog **prioriteta** - od onih najvišeg prioriteta koji se prvo primenjuju do onih najmanjeg prioriteta koji se poslednji primenjuju. Operatori u jednom redu tabele imaju isti prioritet. Ukoliko se u nekom izrazu nalaze operatori istog prioriteta bez zagrada, onda se unarni operatori i operatori dodele izračunavaju redom sdesna na levo, dok se ostali operatori izračunavaju redom sleva na desno.

Na primer, izraz $x * y / z$ se izračunava postupno kao da stoji $(x * y) / z$, dok se $x = y = z$ izračunava kao da stoji $x = (y = z)$.

Pravila prioriteta operatora ne treba pamti napamet, jer se ona lako zaboravljaju ili pomešaju sa sličnim, ali ne i identičnim, pravilima iz drugih programskih jezika.

Umesto toga, pošto se zagrade mogu slobodno koristiti, uvek kada može doći do zabune treba navesti zagrade da bi se eksplicitno naznačio redosled izračunavanja izraza koji se želi. Time se umnogome povećava čitljivost i smanjuje mogućnost suptilnih grešaka koje je teško otkriti.

| Prioritet | Broj operanada | Operatori |
|-----------|----------------|-----------------------------------|
| 16 | 1 | new |
| 15 | 2 | [] . () |
| 14 | 1 | a++ a-- |
| 13 | 1 | ! ~ ++a --a + - (tip) |
| 12 | 2 | * / % |
| 11 | 2 | + - |
| 10 | 2 | << >> |
| 9 | 2 | < <= > >= instanceof |
| 8 | 2 | == != |
| 7 | 2 | & |
| 6 | 2 | ^ |
| 5 | 2 | |
| 4 | 2 | && |
| 3 | 2 | |
| 2 | 3 | ?: |
| 1 | 2 | = += -= *= /= %= &= ^= = <<= >>= |

Slika 1.6 Prioritet operatora

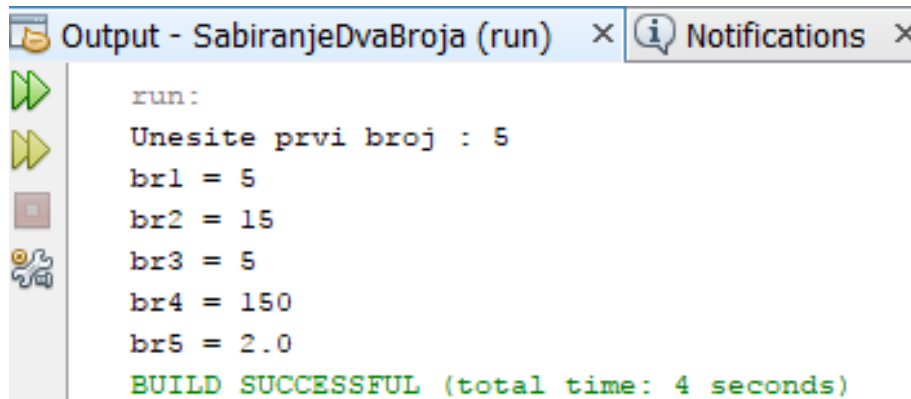
PRIMER 4

Rad sa operatorima dodele vrednosti

- Sa tastature se unosi ceo broj i čuva u promenljivoj br1;

- Deklarišu se celobrojne promenljive br2, br3, br4 i br 5. U početku sve imaju istu vrednost 10.
- Celobrojna promenljiva br2 dobija vrednost $br2 += br1$;
- Celobrojna promenljiva br3 dobija vrednost $br3 -= br1$;
- Celobrojna promenljiva br4 dobija vrednost $br4 *= br2$;
- Celobrojna promenljiva br5 dobija vrednost $br5 /= br3$ - rezultat dati kao realan broj;
- Kreirati program koji prikazuje rezultate navedenih izraza dodele vrednosti.

Izlaz na konzoli bi trebalo ovako da izgleda:



```
run:
Unesite prvi broj : 5
br1 = 5
br2 = 15
br3 = 5
br4 = 150
br5 = 2.0
BUILD SUCCESSFUL (total time: 4 seconds)
```

Slika 1.7 Izlaz iz programa

Sledećim listingom je dato traženo rešenje.

```
package l02;

import java.util.Scanner;

/**
 * *
 * @author Vladimir Milićević
 */
public class DodelaVrednosti {

    public static void main(String[] args) {
        Scanner tastatura = new Scanner (System.in);
        System.out.print("Unesite prvi broj : ");
        int br1 = tastatura.nextInt();
        int br2,br3,br4,br5;
        br2 = br3 = br4 = br5 = 10;
        //Prikaz trazених rezultata
        System.out.println("br1 = " + br1);
        System.out.println("br2 = " + (br2+=br1));
        System.out.println("br3 = " + (br3-=br1));
        System.out.println("br4 = " + (br4*=br2));
        System.out.println("br5 = " + (double)(br5/=br3));
    }
}
```

```
}  
  
}
```

ZADATAK 4

Vežbanje rada sa operatorima dodele vrednosti

- Postoji još jedan operator koji može da se koriti na način prikazan u primeru Primer4;
- Operator % vraća ostatak pri deljenju;
- Na primer $5\%2$ daće rezultat 1;
- U prethodnom primeru dodajte još jednu liniju koja računa $br5\%=br3$;
- Prikažite novu vrednost ua $br5$;
- Pokrenite program.

▼ Poglavlje 2

Iskaz if

DEFINICIJA NAREDBE IF

Naredbe grananja obezbeđuju alternativne puteve toka izvršavanja programa zavisno od vrednosti nekog izraza

Naredbe grananja obezbeđuju alternativne puteve toka izvršavanja programa zavisno od vrednosti nekog izraza. Na primer, ako u nekoj liniji programa korisnik treba da izabere neku opciju, tada zavisno od tog izbora izvršavanje programa treba da se nastavi potpuno različitim putevima. Naredbe grananja omogućavaju da se izabere jedan niz naredbi za izvršavanje, a drugi niz naredbi (ili više njih) da se potpuno preskoči i nikad ne izvrši.

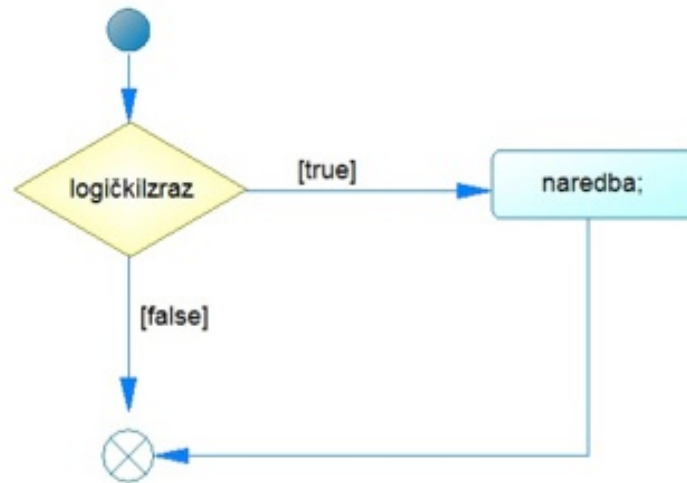
U programu **if** i **if-else** naredba služe za izbor jednog od dva alternativna niza naredbi za izvršavanje zavisno od toga da li je vrednost datog logičkog izraza tačno ili netačno. Treća naredba grananja, switch naredba, služi za izbor jednog od više alternativnih nizova naredbi za izvršavanje zavisno od vrednosti datog celobrojnog ili znakovnog izraza.

Opšti oblik **if naredbe u Javi** je:

```
if (logičkiIzraz)
    naredba;
```

ili

```
if (logičkiIzraz){
    naredba1;
    ...
    naredbaN;
}
```



Slika-1 Izvršenje programa sa naredbom if.

PRIMER 5 - PRIMENE NAREDBE IF

Logički operatori kao operande koriste relacione operatore koji upoređuju veličine pojedinih promenljivih. Rezultat je jedna od dve moguće logičke vrednosti: true ili false.

Primer 5-1:

Ako je promenljiva x veća od 5 ispisati da je promenljiva x veća od 5.

```
if (x > 5)
    System.out.println("X je veće od 5.");
```

Primer 5-2:

Ako za promenljivu x važi: $5 < x < 15$ ispisati da je promenljiva x veća od 5 i manja od 15:

```
if (x > 5 && x < 15)
```

```
System.out.println( "X je veće od 5 a manje od 15");
```

Napomena: u ovom primeru moramo vezati logičkim operatorom I dva relaciona izraza

Primer 5-3: Ako su x, y i z stranice trougla ispisati da li trougao može da se formira:

```
if (x + y > z && y + z > x && + z > y)
```

```
System.ou.println("Trougao može da se formira");
```

Primer 5-4:


```
if (broj % 2 != 0) {    //proverava se da li je broj neparan
    broj = broj + 1; //ako je neparan, neka postane paran
    System.out.println("Broj je pretvoren u paran i njegova vrednost je sada "
+ broj);
}
```

Ako je izraz ocenjen kao tačan, izvršavaju se svi iskazi između vitičastih zagrada, a ako nije, ne izvršava se nijedan.

PRIMER 6

Primena operatora % i if naredbe za proveru deljivosti brojeva.

Napisati program koji proverava da li je uneti broj deljiv sa 2 i 3, sa 2 ili 3, i sa 2 ili 3, ali ne sa oba broja.

Broj a je deljiv sa brojem b ako je ostatak pri deljenju a sa b jednak nuli. Uslov se može zapisati kao sledeći izraz:

```
(a % b == 0)
```

Ako želimo da izvršimo neku naredbu (ili više naredbi) samo ako je ispunjen neki uslov koristimo if naredbu.

```
if(a % b == 0){
    System.out.println(a + " je deljiv sa " + b);
}
```

Štampanje će se izvršiti samo ako je broj a deljiv brojem b, tj. ako vrednost izraza `a % b == 0` bude **true**.

Kombinovanje više uslova:

- Za proveru da li je broj deljiv i sa 2 i sa 3 koristimo logički operator `&&`
- Za proveru da li je broj deljiv sa 2 ili sa 3 koristimo logički operator `||`
- Za proveru da li je broj deljiv sa 2 ili sa 3, ali ne i sa oba broja koristimo logički operator `^`.

Rešenje:

```
package Zadatak1;
import java.util.Scanner;
public class Zadatak1 {

    public static void main(String[] args) {
        // Kreiranje objekta Scanner
        Scanner input = new Scanner(System.in);
        // Unos broja
        System.out.print("Unesi ceo broj: ");
        int number = input.nextInt();
    }
}
```

```
    if (number % 2 == 0 &&& number % 3 == 0) {  
        System.out.println(number + " deljiv je sa 2 i 3.");  
    }  
    if (number % 2 == 0 || number % 3 == 0) {  
        System.out.println(number + " deljiv je sa 2 ili 3.");  
    }  
    if (number % 2 == 0 ^ number % 3 == 0) {  
        System.out.println(number  
            + " deljiv je sa 2 ili 3, ali ne sa oba.");  
    }  
    input.close();  
}  
}
```

Objašnjenje:

Nakon što broj učitamo sa standardnog ulaza, korišćenjem if naredbe proveravamo svaki od slučajeva upotrebom odgovarajućih logičkih operatora. Ako je uslov neke if naredbe ispunjen ispisujemo odgovarajuću poruku.

ZADATAK 5

Vežbanje pisanja jednostavnih grananja u Javi

- Sa tastature se unose dva stringa;
- Napisati program koji proverava da li su dva stringa jednaka;
- Ako jesu spojiti ih i prikazati rezultat spajanja na konzoli.

ISKAZ IF (VIDEO)

Video objašnjava primenu iskaza if i pokazuje njegovu primenu.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 3

Iskaz if - else

DEFINICIJA NAREDBE IF - ELSE

Kada je vrednost logičkog izraza u zagradi jednaka false, izvršava se naredba iza else a preskače naredba iza if.

Često je u programu potrebno uraditi nešto drugo kada je vrednost logičkog izraza jednaka **false**, a ne samo preskočiti naredbu unutar if naredbe. Za takve slučajeve služi if-else naredba.

Opšti oblik if-else naredbe je:

```
if (logičkiIzraz)
    naredba1;
else
    naredba2;
```

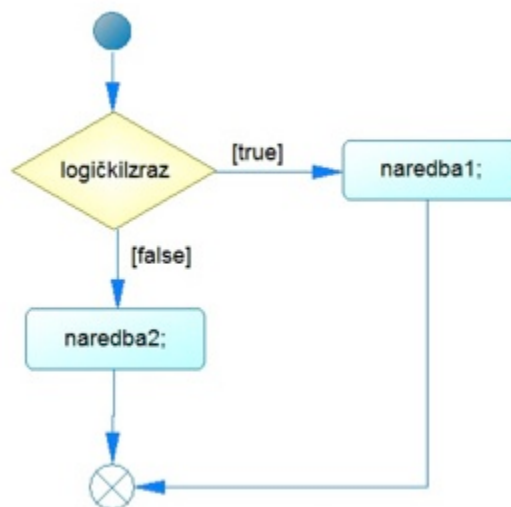
ili:

```
if (logičkiIzraz){
    naredba11;
    ...
    naredba1N;

    else{

        naredba21;
        ...
        naredba2N;
    }
}
```

Izvršavanje **if-else** naredbe je slično izvršavanju if naredbe, osim što u slučaju kada je vrednost logičkog izraza u zagradi jednaka false, izvršava se naredba2 i preskače naredba1. U drugom slučaju, kada je vrednost logičkog izraza jednaka true, izvršava se naredba1 i preskače naredba2. Time se u oba slučaja izvršavanje if-else naredbe završava i program se dalje nastavlja od naredbe koja sledi iza if-else naredbe (slika 2).



Slika- 1 Naredba if - else

NAČIN PRIMENE NAREDBE IF - ELSE

Kada if-else naredba izvršava tačno jednu od dve naredbe unutar if-else naredbe. Te naredbe predstavljaju alternativne tokove izvršavanja koji se biraju zavisno od vrednosti logičkog izraza

Obratite pažnju na to da kada se **if-else** naredbe izvršava tačno jedna od dve naredbe unutar if-else naredbe. Te naredbe predstavljaju alternativne tokove izvršavanja koji se biraju zavisno od vrednosti logičkog izraza.

U sledećem primeru se koristi **if** naredba:

```
if (n % 2 == 0)
    System.out.println("n je paran broj");
    System.out.println("n = " + n);
```

Kada se ovaj programski fragment izvršava, najpre se izračunava logički izraz $n \% 2 == 0$ u zagradi if naredbe. To znači da se izračunava da je ostatak od n podeljeno sa 2 jednako nuli, ili ekvivalentno da li je n paran broj. Ako je to true (tačno), izvršava se naredba:

```
System.out.println("n je paran broj");
```

unutar **if** naredbe, odnosno prikazuje se tekst "n je paran broj" na ekranu. Time se ujedno i završava izvršavanje **if** naredbe.

Ako logički izraz $n \% 2 == 0$ daje **false** (netačno), ništa se dalje ne radi i odmah se završava izvršavanje if naredbe.

Nakon završetka izvršavanja if naredbe u oba slučaja, izvršavanje programa se nastavlja od naredbe koja sledi iza if naredbe, odnosno u ovom primeru je to naredba:

```
System.out.println("n = " + n);
```

kojom se prikazuje vrednost promenljive n na ekranu. Obratite pažnju na to da smo naredbu:

```
System.out.println("n je paran broj");
```

malo uvukli unutar if naredbe čiji je sastavni deo.

Za Java prevodilac je ovo bez značaja, ali smo iskoristili slobodni format jezika Java da bismo naznačili da je ta naredba deo složene naredbe i da se neće uvek izvršiti. Ovo je jedna od odlika dobrog stila programiranja.

U prethodnom primeru if naredbe se ništa ne izvršava kada je n neparan broj. Ukoliko je potrebno nešto uraditi i u tom slučaju, mora se koristiti if-else naredba.

```
if (n % 2 == 0)
    System.out.println("n je paran broj");
else
    System.out.println("n je neparan broj");
System.out.println("n = " + n);
```

PREPORUKE KORIŠĆENJA IF-ELSE NAREDBE

Preskače se else deo složene if-else naredbe ako je uslov zadovoljen, tj. ako istinit (true). Ako nije zadovoljen (false), izvršavaju se naredbe iza else dela.

Za Java prevodilac je ovo bez značaja, ali smo iskoristili slobodni format jezika Java da bismo naznačili da je ta naredba deo složene naredbe i da se neće uvek izvršiti. Ovo je jedna od odlika dobrog stila programiranja. U prethodnom primeru if naredbe se ništa ne izvršava kada je n neparan broj. Ukoliko je potrebno nešto uraditi i u tom slučaju, mora se koristiti if-else naredba.

```
if (n % 2 == 0)
    System.out.println("n je paran broj");
else
    System.out.println("n je neparan broj");
System.out.println("n = " + n);
```

Ovde se opet najpre izračunava logički izraz `n % 2 == 0` u zagradi **if-else** naredbe.

Ako to daje true (tačno), izvršava se naredba:

```
System.out.println("n je paran broj");
```

i preskače se else deo.

Ali ako logički izraz u zagradi daje **false** (netačno), preskače se if deo i izvršava naredba:

```
System.out.println("n je neparan broj");
```

u **else** delu.

Nakon završetka izvršavanja **if-else** naredbe u oba slučaja, izvršavanje programa se nastavlja od naredbe koja sledi iza if-else naredbe, odnosno i u ovom primeru je to naredba:

System.out.println("n = " + n);

kojom se prikazuje vrednost promenljive **n** na ekranu.

U sledećem primeru se koristi oblik **if** naredbe sa blok naredbom da bi se zamenile vrednosti celobrojnih promenljivih **x** i **y**, ali samo ako je početno **x** veće od **y**:

```
if (x > y) {
    int temp; // Pomoćna promenljiva
    temp = x; // Vrednost z je stara vrednost x
    x = y; // Nova vrednost x je stara vrednost y
    y = temp; // Nova vrednost y je stara vrednost x
}
```

Pridržavamo se preporuke dobrog stila Java programiranja da se znak **{** navodi na kraju reda iza kojeg sledi, a odgovarajući znak **}** piše propisno poravnat u novom redu.

PRIMER 7

Paran broj je broj koji je deljiv sa 2. Provera da li je broj deljiv sa 2 se vrši primenom operatora %.

Napraviti program koji proverava da li je uneti ceo broj paran ili neparan. Ako je broj paran, množi se sa 2, a ako je neparan sa 3. Rezultat množenja ispisati na konzolu.

Napomena: Za parnost možete koristiti moduo (%). Ukoliko je **broj%2==0** onda je broj paran.

Objašnjenje:

Korisnik unosi preko konzole jedan ceo broj. Proverava se njegova parnost. Ukoliko je ostatak deljenja sa dva jednak 0, to znači da je broj paran. Ako je broj paran množi se sa 2, a ako je neparan sa 3. Rezultat na kraju ispisujemo na konzolu.

Naredna linija predstavlja operator množenja sa dodelom:

```
broj *= 2;
```

Naredba je ekvivalentna sledećoj:

```
broj = broj * 2;
```

Rešenje:

```
package lo2.grananje;

import java.util.Scanner;
```

```
public class Primer7 {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Unesi ceo broj: ");
        int broj = input.nextInt();
        // Provera da li je broj paran
        if (broj % 2 == 0) {
            broj *= 2;
        } else {
            broj *= 3;
        }
        System.out.println("Rezultat je: " + broj);
        input.close();
    }
}
```

PRIMER 8

Primenom if-else naredbe se proverava da li je težina korisnika veća od idealne za njegovu visinu.

Vreme je da podignemo kvalitet programa na način što će unos i prikazivanje podataka biti obavljen u grafičkom okruženju. O ovome će detaljno biti govora u predmetu koji se bavi kreiranjem grafičkog korisničkog interfejsa. Za sada je tu mala demonstracija na jednostavnim primerima.

Treba napraviti program koji od korisnika zahteva da unese svoju težinu i visinu korišćenjem Java klase JOptionPane. Kada korisnik unese težinu i visinu treba izračunati idealnu težinu koja se računa kao visina-110. Ukoliko je težina veća od idealne preko JOptionPane-a prikazati poruku: "Trebalo bi da smršate", dok ukoliko je manja ili jednaka idealnoj težini poruku: "Vaša težina je potpuno OK."

Novina u ovom zadatku jeste if-else naredba.

```
if (brojKilograma > idealnaTezina) {
    JOptionPane.showMessageDialog(null, "Trebalo bi da smršate");
} else {
    JOptionPane.showMessageDialog(null, "Vaša težina je potpuno OK");
}
```

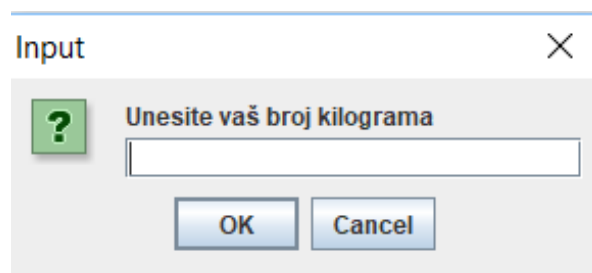
Objašnjenje:

U slučaju da je vrednost izraza *brojKilograma > idealnaTezina* **true**, ispisuje se prva poruka, a ako je vrednost tog izraza **false**, ispisuje se samo druga poruka.

Rešenje:

```
package lo2.grananje;
import javax.swing.JOptionPane;
public class Primer8 {

    public static void main(String[] args) {
        /**
         * Unosimo broj kilograma preko JOptionPane-a i parsiramo ga u Double
         */
        double brojKilograma =
Double.parseDouble(JOptionPane.showInputDialog("Unesite vaš broj kilograma"));
        /**
         * Unosimo visinu preko JOptionPane-a i parsiramo ga u Double
         */
        double visina = Double.parseDouble(JOptionPane.showInputDialog("Unesite
vašu visinu u centimetrima"));
        /**
         * Idealna težina je jednaka visini - 110
         */
        double idealnaTezina = visina - 110;
        /**
         * Ukoliko je broj kilograma veći od idealne težine ispisati : Trebalo
         * bi da smršate Ukoliko je manji ili jednak ispisuje: Vaša težina je ok
         */
        if (brojKilograma > idealnaTezina) {
            JOptionPane.showMessageDialog(null, "Trebalo bi da smršate");
        } else {
            JOptionPane.showMessageDialog(null, "Vaša težina je potpuno OK");
        }
    }
}
```



Slika 3.1 Izgled ekrana kada se pokrene program

PRIMER 9

Primenom if-else naredbe se upoređuje da li je mesečna potrošnja korisnika manja ili veća od raspoloživog porodičnog budžeta.

*Treba napraviti program preko koga korisnik unosi svoj mesečni budžet i nedeljnu potrošnju preko JOptionPane-a. Nakon unosa program treba da izračuna koje ce njegovo stanje biti na kraju meseca kao: budžet - (4*nedeljna potrošnja). Ukoliko je stanje na kraju meseca manje od 0 treba prikazati poruku: "Trebali bi da manje trošite" dok ukoliko je rezultat veći ili jednak*

0 treba ispisati poruku: "Budžet vam je OK". Preostala suma na mesečnom nivou bi bila" (stanje).

Objašnjenje: Na osnovu unetih podataka prvo računamo kakvo će stanje biti na kraju meseca. Zatim primenom if-else naredbe proveravamo da li je stanje negativno, tj. da li će korisnik probiti budžet i ispisujemo odgovarajuće poruke.

U slučaju probijanja budžeta ispisati korisniku za koliko je budžet probijen.

Rešenje:

```
package primer9;
import javax.swing.JOptionPane;
public class Primer9 {

    public static void main(String[] args) {
        double budzet = Double.parseDouble(JOptionPane.showInputDialog("Unesite vaš mesečni budžet"));
        double nedeljnaPotrošnja = Double.parseDouble(JOptionPane.showInputDialog("Unesite vašu nedeljnu potrošnju"));
        double stanjeMesečno = budzet - (4 * nedeljnaPotrošnja);
        if (stanjeMesečno < 0) {
            JOptionPane.showMessageDialog(null, "Treba bi da manje trošite.");
        } else {
            JOptionPane.showMessageDialog(null, "Budžet vam je ok Preostala suma na mesečnom nivou bi bila: " + stanjeMesečno);
        }
    }
}
```

ZADATAK 6

Samostalno vežbanje if else naredbe

- Kreirajte nov Java projekat u paketu l02.grananjeprimer;
- Kreirajte klasu Zadatak6 u ovom paketu;
- Realizujte program koji učitava sa tastature dva stringa;
- Primenom if - else proveriti koji string ima veći broj karaktera;
- Ukoliko prvi ima više (ili imaju jednak broj) štampati poruku "Prvi string je duži";
- U suprotnom štampati poruku "Drugi string je duži".

ISKAZ IF-ELSE (VIDEO)

Video objašnjenja iskaze if i else i daje primere njihovog korišćenja

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 4

Iskaz else-if

UGNJEŽDENE IF-ELSE NAREDBE

Sa gnježdenim if-else naredbama omogućava se višestruko grananje izvršavanja programa.

Bitan detalj u vezi sa opštim oblikom **if** i **if-else** naredbi je da naredba naredba1 ili naredba2 mogu biti bilo koje naredbe jezika Java.

To specifično znači da one mogu biti isto tako **if** ili **if-else** naredbe. Jedan primer ove mogućnosti koja se naziva ugnježđavanje jeste:

```
if (logičkiIzraz1)
    naredba1;
else
    if (logičkiIzraz2)
        naredba2;
    else
        naredba3
```

Međutim, pošto je Java jezik slobodnog formata, ovo se skoro uvek piše u obliku:

```
if (logičkiIzraz1)
    naredba1;
else if (logičkiIzraz2)
    naredba2;
else
    naredba3;
```

Ovaj oblik naredbe grananja omogućuje izbor jedne od tri alternative za izvršavanje: **naredbe1, naredbe2 ili naredbe3.**

Naime, na uobičajeni način, najpre se izračunava **logičkiIzraz1**. Ako je njegova vrednost true, izvršava se naredba1 i preskače sve ostalo.

Ako je njegova vrednost **false**, preskače se naredba1 i izvršava se druga, ugnježđena if-else naredba, tj. izračunava se logičkiIzraz2 i zavisno od toga da li je njegova vrednost true ili false, zatim se izvršava naredba2 ili naredba3.

Pošto nivo ugnježđavanja može biti proizvoljan, ništa nas ne sprečava da nastavimo taj postupak i tako dobijemo naredbu višestrukog grananja u opštem obliku:

```
if (logičkiIzraz1)
    naredba1;
else if (logičkiIzraz2)
    naredba2;
else if (logičkiIzraz3)
    naredba3;
    . . .
else if (logičkiIzrazN)
    naredbaN;
else
    naredbaN+1;
```

ISKAZ ELSE-IF (VIDEO)

Video objašnjava primenu iskaza else - if.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

VIŠESTRUKO GRANANJE KOMANDOM ELSE-IF

Blok naredbi se sastoji od niza naredbi između vitičastih zagrada.

Izvršavanje ove naredbe višestrukog grananja se izvodi tako što se najpre logički izrazi u zagradama izračunavaju redom odozgo na dole dok se ne dobije prvi koji daje vrednost **true**. Zatim se izvršava njegova pridružena naredba i preskače se sve ostalo.

Ako vrednost nijednog logičkog izraza nije **true** (tj. svi logički izrazi daju false), izvršava se naredba u else delu.

U stvari, ovaj **else** deo nije obavezan, pa ako nije naveden i nijedan logički izraz nije true, nijedna naredba se ni ne izvršava.

Naglasimo još da svaka od naredbi unutar ove naredbe višestrukog grananja može biti **blok naredbi koji se sastoji od niza naredbi između vitičastih zagrada**.

To naravno ne menja ništa konceptualno što se tiče njenog izvršavanja, osim što se izvršava blok (niz naredbi) pridružen prvom logičkom izrazu koji ima vrednost **true**.

Jedan primer upotrebe naredbe višestrukog grananja je sledeći programski fragment u kojem se određuje ocena studenta na ispitu na osnovu broja osvojenih poena i standardne skale za ocenjivanje na fakultetu, pri čemu se podrazumeva da je broj poena veći od 0 i manji ili jednak 100:

```
if (logičkiIzraz1)
    if (brojPoena >= 91)
        ocena = 10;
    else if (brojPoena >= 81)
```

```
    ocena = 9;
else if (brojPoena >= 71)
    ocena = 8;
else if (brojPoena >= 61)
    ocena = 7;
else if (brojPoena >= 51)
    ocena = 6;
else
    ocena = 5;
```

VIŠEĆI ELSE DEO

else deo uvek se vezuje za najbliži prethodni if deo koji već nema svoj else deo.

Na kraju, obratimo pažnju na jedan problem kod ugnježđavanja koji se popularno naziva "viseći" else deo.

Posmatrajmo sledeći programski fragment:

:

```
if (x >= 0)
    if (y >= 0)
        System.out.println("Prvi slucaj");
    else
        System.out.println("Drugi slucaj");
```

Ovako kako je napisan, fragment sugerise da imamo jednu if-else naredbu čiji se if deo sastoji od druge if naredbe.

Međutim, kako uvlačenje redova nema značaja za prevodilac i kako važi pravilo da se else deo uvek vezuje za najbliži prethodni if deo koji već nema svoj else deo, efekat datog fragmenta je isti kao da smo napisali

```
if (x >= 0)
    if (y >= 0)
        System.out.println("Prvi slucaj");
    else
        System.out.println("Drugi slucaj");
```

Efekat ovog fragmenta i sugerisana prvobitna interpretacija nisu ekvivalentni.

:

Ako x ima vrednost manju od 0, izvršavanje pod pretpostavkom prvobitne interpretacije bi dovelo do prikazivanja teksta "Drugi slučaj" na ekranu. Ali pravi efekat je zapravo da se preskače ugnježđena if-else naredba, odnosno ništa se ne dobija na ekranu. Ukoliko se zaista želi efekat koji sugerise prvobitna interpretacija, ugnježđena if naredba se može pisati unutar bloka

```
if (x >= 0) {
    if (y >= 0)
        System.out.println("Prvi slučaj");
    }
    else
        System.out.println("Drugi slučaj");
```

ili se ona može pisati u obliku if-else naredba čiji se else deo sastoji od takozvane prazne naredbe označene samo tačkom-zarez (;):

```
if (x >= 0)
    if (y >= 0)
        System.out.println("Prvi slučaj");
    else
        ; // Prazna naredba
else
    System.out.println("Drugi slučaj");
```

ELSE - IF - DOPUNSKO RAZMATRANJE

Ukoliko prilikom donošenja odluke postoji više mogućnosti, može se upotrebiti naredba else if.

Ukoliko prilikom donošenja odluke postoji više mogućnosti, može se upotrebiti klauzula else if.

```
if(izraz1)
    iskaz1;
else if(izraz2)
    iskaz2;
else if(izraz3)
    iskaz3;
else
    iskaz4;
```

Ako je vrednost izraza izraz1 ocenjena kao tačna, izvršava se iskaz1. Ako izraz nema vrednost true, proverava se vrednost izraza izraz2. Ako je njegova vrednost tačna, izvršava se iskaz iskaz2. Posle izvršenja iskaza iskaz2 program se nastavlja kodom koji sledi iza celog iskaza if. Ako vrednost izraza izraz2 nije true, proverava se vrednost izraza izraz3 itd. Ako nijedan od izraza u klauzulama else if nije ocenjen kao tačan (true), izvršava se iskaz (iskazi) iza klauzule else, ako ona postoji.

Da pogledamo kako ovo izgleda na primeru:

```
public class IfElseDemo {
    public static void main(String[] args) {
        int rezultatTesta = 76;
        char ocena;
        if (rezultatTesta >= 90) {
```

```
        ocena = 'A';
    } else if (rezultatTesta >= 80) {
        ocena = 'B';
    } else if (rezultatTesta >= 70) {
        ocena = 'C';
    } else if (rezultatTesta >= 60) {
        ocena = 'D';
    } else {
        ocena = 'F';
    }
    System.out.println("Ocena = " + ocena);
}
}
```

Izlaz iz programa je:

Ocena = C;

ELSE – IF (NASTAVAK)

Iskazi else-if se mogu povezati, te u okviru jednog if iskaza, koristi se i drugi itd.

Kao što vidite, vrednost promenljive rezultatTesta može da zadovolji više od jednog uslova, jer je $76 \geq 70$, a takođe je i $rezultatTesta \geq 60$. Ocena je ipak C, zato što se izvršava samo prvi blok koda koji pripada izrazu $rezultatTesta \geq 70$. Kada se izvrši taj blok, kontrola programa prelazi na kod iza iskaza if, u ovom slučaju na iskaz System.out... Izrazi koji slede iza izraza koji je bio zadovoljen se ne proveravaju, a samim tim se i ne izvršava pripadajući kod.

Iskazi if mogu biti "ugnežđeni" (umreženi). U okviru jednog if iskaza može da se nađe drugi, u okviru tog drugog, treći itd. Evo kako to izgleda

```
if(izraz1){
    if(izraz1-1){
        if(izraz1-1-1)
            skaz1-1-1;
        }else
            iskaz1-1;
    }
}
```

ZADACI SA IF ELSE, ELSE IF, ELSE, POVEZANI IF ISKAZI & ULITAVANJE ZNAKOVA. (VIDEO)

Video daje objašnjenja primene iskaza Else, Else if, Else, i povezane if iskaze

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMER 10

Naredba else if na primeru poređenja stringova

Napisati program koji zahteva od korisnika da unese string koji predstavlja skraćeni naziv smeru. Program ispisuje puni naziv na osnovu skraćenice.

Smerovi:

- IT - Informacione tehnologije
- SI - Softversko inženjerstvo
- IS - Informacioni sistemi
- RI - Računarske igre

Korisnik unosi skraćenicu kao string sa konzole. Na osnovu unete vrednosti se ispisuje odgovarajuća poruka. Za poređenje stringova se koristi metoda equals.

Rešenje:

```
package primer10;

import java.util.Scanner;

public class Primer10 {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Unesite skraceni naziv smeru: ");
        String skracenica = input.next();
        if (skracenica.equals("IT")) {
            System.out.println("Informacione tehnologije");
        } else if (skracenica.equals("SI")) {
            System.out.println("Softversko inženjerstvo");
        } else if (skracenica.equals("IS")) {
            System.out.println("Informacioni sistemi");
        } else if (skracenica.equals("RI")) {
            System.out.println("Računarske igre");
        } else {
            System.out.println("Skraćenica ne odgovara nijednom smeru.");
        }
    }
}
```

ZADATAK 7

Samostalno vežbanje višestrukog grananja

- Kreirajte nov primer pomoću kojeg ćete prikazati koju ocenu je dobio student na osnovu broja osvojenih poena na ispitu;
- Možete iskoristiti kod iz sekcije "Višestruko grananje komandom else-if"

▼ Poglavlje 5

Naredba switch

SWITCH I BREAK KOMANDE

Switch naredba se koristi za uslovno izvršavanje izraza na bazi vrednosti celobrojnog izraza. Break komanda prekida dalje izvršavanje case dela i završava se izvršanje switch naredba.

Treća naredba grananja, switch naredba, koristi se mnogo manje od prethodne dve, ali u nekim slučajevima prirodnije odražava logiku višestrukog grananja.

Najčešći oblik je:

```
switch (izraz) {  
    case literal1:  
        nizNaredbi1  
        break;  
    case literal2:  
        nizNaredbi2  
        break;  
    . . .  
    case literalN :  
        nizNaredbiN  
        break;  
    default:  
        nizNaredbiN  
}
```

Ovom naredbom se najpre izračunava izraz koji može biti:

- celobrojnog tipa
- znakovnog tipa
- string

Zatim se zavisno od dobijene vrednosti izraza izvršava nizNaredbi koji je pridružen jednom od case delova unutar switch naredbe. Pri tome se redom odozgo na dole traži prvi literal uz klauzulu case koja je jednaka vrednosti izračunatog izraza i izvršava se nizNaredbi pridružen pronađenom case delu.

Poslednji slučaj u **switch** naredbi može opciono biti **default** deo koji se izvršava ukoliko izračunata vrednost izraza nije jednaka nijednom literalu **case** delovima. Najzad, ukoliko default deo nije naveden i izračunata vrednost izraza nije jednaka nijednom literalu u case delovima, ništa se dodatno ne izvršava i izvršavanje **switch** naredbe se odmah završava.

Na kraju svakog **case** dela se obično, ali ne uvek, navodi **break** naredba. Efekat **break** naredbe je da se prekine izvršavanje odgovarajućeg case dela, a samim tim i cele **switch** naredbe. Ako se na kraju **case** dela koji se izvršava ne nalazi break naredba, prelazi se na izvršavanje narednog case dela unutar **switch** naredbe. Ovaj postupak se nastavlja sve dok se ne naiđe na **break** naredbu ili se ne dođe do kraja **switch** naredbe. O generalnoj upotrebi **break** naredbe ćemo govoriti u narednom predavanju.

PRIMER SWITCHDEMO

Prilikom donošenja odluke treba imati u vidu da iskaz switch odluku može da donese samo na osnovu celobrojne vrednosti, dok se u iskazu if mogu da koriste različiti uslovi

U primeru koji sledi je upotrebljen iskaz switch koji na osnovu celobrojne vrednosti *mesec* štampa ime meseca, na koji promenljiva ukazuje.

```
public class SwitchDemo {
    public static void main(String[] args) {
        int mesec = 8;
        switch (mesec) {
            case 1: System.out.println("Januar"); break;
            case 2: System.out.println("Februar"); break;
            case 3: System.out.println("Mart"); break;
            case 4: System.out.println("April"); break;
            case 5: System.out.println("Maj"); break;
            case 6: System.out.println("Jun"); break;
            case 7: System.out.println("Jul"); break;
            case 8: System.out.println("Avgust"); break;
            case 9: System.out.println("Septembar"); break;
            case 10: System.out.println("Oktobar"); break;
            case 11: System.out.println("Novembar"); break;
            case 12: System.out.println("Decembar"); break;
        }
    }
}
```

Rezultat ovog programa je Avgust. Isti efekat se mogao postići i pomoću iskaza if: `int mesec = 8;`

```
if (mesec == 1) {
```

```
    System.out.println("Januar");
```

```
} else if (month == 2) {
```

```
    System.out.println("Februar");
```

```
}
```

..... itd.

Stvar ličnog izbora koji ćete od ovih metoda koristiti. Prilikom donošenja odluke treba imati u vidu da iskaz **switch** odluku može da donese samo na osnovu celobrojne vrednosti, dok se u iskazu **if** mogu da koriste različiti uslovi.

Verovatno ste u prethodnom kodu primetili prisustvo iskaza **break**. Ovaj iskaz se najčešće koristi zajedno sa iskazom **switch**, jer se bez njega kontrola toka programa, nakon izvršavanja pravog iskaza, prenosi na sledeći iskaz u **switch** bloku. Ako se doda iskaz **break**, onda se kontrola toka, nakon što se izvrši pravi iskaz, prenosi na kod koji sledi iza iskaza **switch**.

KORIŠĆENJE SWITCH NAREDBE BEZ BREAK NAREDBE

Svi meseci koji imaju 31 dan su bez iskaza break, koji postoji samo kod poslednjeg u nizu. Isto važi i za mesece koji imaju 30 dana (i oni su grupisani i postoji samo jedan iskaz break).

Način kontrole toka programa u iskazu **switch** (bez iskaza break) može ponekad i da bude od koristi. Evo primera u kojem smo to iskoristili:

```
public class SwitchDemo2 {
    public static void main(String[] args) {
        int mesec = 2;
        int godina = 2000;
        int brojDana = 0;
        switch (mesec) {
            case 1:
            case 3:
            case 5:
            case 7:
            case 8:
            case 10:
            case 12:
                brojDana = 31;
                break;
            case 4:
            case 6:
            case 9:
            case 11:
                brojDana = 30;
                break;
            case 2:
                if ( godina % 4 == 0)
                    brojDana = 29;
                else
                    brojDana = 28;
        }
    }
}
```

```
int godina = 2000;
int brojDana = 0;
switch (mesec) {
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
        brojDana = 31;
        break;
    case 4:
    case 6:
    case 9:
    case 11:
        brojDana = 30;
        break;
    case 2:
        if ( godina % 4 == 0)
            brojDana = 29;
        else
            brojDana = 28;
        break;
}
System.out.println("Broj dana je " + brojDana);
}
break;
}
System.out.println("Broj dana je " + brojDana);
}
}
```

Izlaz iz programa je:

Broj dana je 29

Ovaj program izračunava broj dana u mesecu za zadatu godinu i mesec. Kao što vidite, svi meseci koji imaju 31 dan su bez iskaza break, koji postoji samo kod poslednjeg u nizu. Isto važi i za mesece koji imaju 30 dana (i oni su grupisani i postoji samo jedan iskaz break).

U nekom **case** delu **switch** naredbe se može potpuno izostaviti odgovarajući niz naredbi i **break** naredba. Ako iza tog "praznog" case dela dolazi drugi "pravi" case deo, onda niz naredbi ovog drugog dela odgovara dvoma literalima. To prosto znači da će se ovaj niz naredbi izvršiti kada je vrednost izraza jednaka jednom od ta dva literala

NAREDBA DEFAULT

Ako želite da se neki iskazi izvrše u slučaju da nijedan od case iskaza nije zadovoljen, možete da upotrebite naredbu default.

Ako želite da se neki iskazi izvrše u slučaju da nijedan od case iskaza nije zadovoljen, možete da upotrebite naredbu **default**. Iskazi koji slede iza ove naredbe se izvršavaju ako nijedan slučaj (case) nije zadovoljen. Evo kako izgleda prepravljen program iz prvog primera, sada sa dodatom klauzulom default

.

:

```
public class SwitchDemo {  
    public static void main(String[] args) {  
        int mesec = 8;  
        switch (mesec) {  
            case 1: System.out.println("Januar"); break;  
            case 2: System.out.println("Februar"); break;  
            case 3: System.out.println("Mart"); break;  
            case 4: System.out.println("April"); break;  
            case 5: System.out.println("Maj"); break;  
            case 6: System.out.println("Jun"); break;  
            case 7: System.out.println("Jul"); break;  
            case 8: System.out.println("Avgust"); break;  
            case 9: System.out.println("Septembar"); break;  
            case 10: System.out.println("Oktobar"); break;  
            case 11: System.out.println("Novembar"); break;  
            case 12: System.out.println("Decembar"); break;  
            default: System.out.println("Niste uneli ispravan broj za mesec");break;  
        }  
    }  
}
```

ISKAZ SWICH (VIDEO)

Video objašnjava formiranje i korišćenje swicj iskaza, sa i bey breake iskaza.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMER 11 - RAD SA MENIJEM PREKO STANDARDNOG ULAZA

Program prikazuje početni meni sa opcijama na ekranu i od korisnika očekuje izbor jedne od opcija. Zavisno od izabrane opcije, program zatim prikazuje odgovarajuću poruku.

Jedna od čestih primena **switch** naredbe je u programima čiji se rad zasniva na menijima. Program koji sledi prikazuje početni meni sa opcijama na ekranu i od korisnika očekuje izbor jedne od opcija. Zavisno od izabrane opcije, program zatim prikazuje odgovarajuću poruku.

```
import java.util.Scanner;
public class Meni {
    public static void main(String[] args) {
        // Prikazivanje menija na ekranu
        System.out.println("Opcije menija su :");
        System.out.println(" 1. CS101");
        System.out.println(" 2. CS322");
        System.out.println(" 3. CS323");
        System.out.println(" 4. IT250");
        System.out.println(" 5. IT370");
        System.out.println(" 6. CS230");
        System.out.print("Unesite broj opcije koju želite: ");
        // Učitavanje izabrane opcije menija
        Scanner ulaz = new Scanner(System.in);
        int brojOpcije = ulaz.nextInt();
        switch (brojOpcije) {
            case 1:
                System.out.println("Izabrali ste \"Uvod u OOP\".");
                break;
            case 2:
                System.out.println("Izabrali ste \"Programski jezik C#\".");
                break;
            case 3:
                System.out.println("Izabrali ste \"Programski jezik C++\".");
                break;
            case 4:
                System.out.println("Izabrali ste \"Veb sistemi\".");
                break;
            case 5:
                System.out.println("Izabrali ste \"Interakcija
čovek-računar\".");
                break;
            case 6:
                System.out.println("Izabrali ste \"Distribuirani sistemi\".");
                break;
            default:
                System.out.println("Greška: pogrešna opcija!");
        }
    }
}
```

```
}
```

NAPOMENE U VEZI PRIMERA

Java prevodilac prilikom prevođenja programa mora zaključiti da je promenljivoj prethodno dodeljena vrednost kada se ona koristi u programu

Celobrojne promenljive dan, mesec i godina služe za predstavljanje kako ulaznog datog datuma tako i izlaznog sutrašnjeg datuma. U slučaju kada je februar mesec datog datuma, njegov ukupan broj dana moramo odrediti prema tome da li se radi o prestupnoj godini. U svakodnevnoj praksi smo navikli da smatramo da je godina prestupna ako je deljiva sa 4. U Javi se to može izaziti uslovom da je ostatak pri deljenju godine sa 4 jednak 0:

(godina % 4 == 0)

U **default** slučaju **switch** naredbe se koristi metod `System.exit(0)` za prekid rada programa, jer program ne može dalje nastaviti zbog pogrešnog ulaznog podatka.

Najzad, pažljivi studenti su možda primetili naredbu deklaracije:

int maxDanaUMesecu = 0;

na početku dela u kojem se određuje broj dana u aktuelnom mesecu.

Naravno, deklaracija promenljive **maxDanaUMesecu** je na tom mestu neophodna, ali se postavlja pitanje da li je neophodna i njena inicijalizacija (0 je ovde proizvoljno izabrana vrednost) ?

Odgovor je potvrđan, jer se bez njene inicijalizacije dobija poruka o greški kod korišćenja promenljive **maxDanaUMesecu** u **if** naredbi za izračunavanje sutrašnjeg datuma. Razlog ove neočekivane greške je taj što se vrednost promenljive može koristiti samo ako je promenljivoj prethodno nesumnjivo dodeljena neka vrednost. To znači da Java prevodilac prilikom prevođenja programa mora zaključiti da je promenljivoj prethodno dodeljena vrednost kada se ona koristi u programu..

U našem primeru, u **default** slučaju **switch** naredbe se ne dodeljuje nijedna vrednost promenljivoj **maxDanaUMesecu**, pa Java prevodilac pogrešno zaključuje da ona može biti nedefinisana u **if** naredbi iza **switch** naredbe. To naravno nije moguće, jer se u **default** slučaju **switch** naredbe program prekida i zato se nikad ne može izvršiti **if** naredba iza **switch** naredbe sa nedefinisanom vrednošću promenljive **maxDanaUMesecu**.

Najjednostavniji način da se izbegne ovaj problem u ovom primeru je da se promenljivoj **maxDanaUMesecu** dodeli inicijalna vrednost, kako bi se Java prevodilac naterao da zaključi da ova promenljiva ima vrednost u svim slučajevima **switch** naredbie..

PRIMER 12

Slučaj većeg broja grananja

Napisati program koji na osnovu unete ocene prikazuje koliko je potrebno bodova osvojiti da bi se dobila ta ocena.

Primer: Ako korisnik unese ocenu 6, potrebno je ispisati "51 - 60 bodova".

Objašnjenje

Korisnik unosi ocenu kao ceo broj sa konzole. Na osnovu vrednosti unete ocene se vrši grananje naredbom switch. Za svaku ocenu se ispisuje odgovarajuća poruka. Svako dozvoljenoj vrednosti ocene odgovara jedan slučaj, tj. *case*. U slučaju da korisnik ne unese broj koji predstavlja neku ocenu, ispisuje mu se poruka koja govori koji brojevi predstavljaju ocene. Taj slučaj pogrešnog unosa je obrađen *default* naredbom. Naredba *break* je obavezna posle poruke za svaku ocenu. Tako se obezbeđuje da se ispiše samo jedna poruka i da se izvršavanje switch naredbe tu prekine.

Napisati isti program upotrebom if-else naredbi umesto switch naredbe.

Rešenje:

```
package primer11;
import java.util.Scanner;
public class Primer11 {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Unesite ocenu");
        int ocena = input.nextInt();

        switch (ocena){
            case 5:
                System.out.println("0-50 bodova");
                break;
            case 6:
                System.out.println("51-60 bodova");
                break;
            case 7:
                System.out.println("61-70 bodova");
                break;
            case 8:
                System.out.println("71-80 bodova");
                break;
            case 9:
                System.out.println("81-90 bodova");
                break;
            case 10:
                System.out.println("91-100 bodova");
                break;
        }
    }
}
```



```
        default:
            System.out.println("Minimalna ocena je 5, a maksimalna 10.");
    }

    input.close();
}
}
```

PRIMER 13

Vrednost stringa može da služi za grananje u switch naredbi

Zadatak 7 može da se reši upotrebom switch naredbe. Unetu skraćenicu sačuvamo u promenljivoj skracenica i na osnovu vrednosti te promenljive vršimo grananje u switch naredbi. Svaki slučaj predstavlja konkretnu skraćenicu, a default naredba obrađuje slučaj nepostojeće skraćenicе.

```
package primer12;

import java.util.Scanner;

public class Primer12 {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Unesite skraceni naziv smeru: ");
        String skracenica = input.next();

        switch (skracenica) {
            case "IT":
                System.out.println("Informacione tehnologije");
                break;
            case "SI":
                System.out.println("Softversko inženjerstvo");
                break;
            case "IS":
                System.out.println("Informacioni sistemi");
                break;
            case "RI":
                System.out.println("Računarske igre");
                break;
            default:
                System.out.println("Skracénica ne odgovara nijednom smeru.");
        }

        input.close();
    }
}
```

PRIMER 14 - ODREĐIVANJE SUTRAŠNJEG DATUMA

*Koristimo promenljivu **maxDanaUMesecu** koja dobija odgovarajuću vrednost **switch** naredbom na osnovu meseca datog datuma, a onda se određuje sutrašnji datum*

U ovom primeru ćemo pokazati malo složeniji program kojim se određuje sutrašnji datum za neki dati datum.

Ulaz programa su tri cela broja koja određuju ispravan datum u formatu dan, mesec, godina: na primer, "24 11 2013".

Izlaz programa treba da budu tri broja koja predstavljaju sutrašnji datum za dati datum, odnosno u ovom primeru je to "25 11 2013".

Sutrašnji datum je lako odrediti ukoliko je dan datog datuma manji od broja dana u mesecu datog datuma - onda je samo dan sutrašnjeg datuma za jedan veći. Međutim, stvari se komplikuju kada je dan datog datuma jednak broju dana u mesecu datog datuma: na primer, "31 10 2013". Izuzetak od toga je kada je dati datum, recimo, "31 12 2013". Da bismo otkrili o kojem se od ovih slučajeva radi u programu, najpre određujemo ukupan broj dana u mesecu datog datuma. U tu svrhu koristimo promenljivu **maxDanaUMesecu** koja dobija odgovarajuću vrednost **switch** naredbom na osnovu meseca datog datuma.

Nakon što odredimo ukupan broj dana u aktuelnom mesecu, ispitivanjem da li je dan datog datuma manji od ukupnog broja dana u mesecu datog datuma, kao i da li je mesec datog datuma manji od decembra, nije teško odrediti sutrašnji datum.

```
import java.util.Scanner;

public class SutrasnjiDatum {
    public static void main(String[] args) {
        System.out.print("Unesite datum u formatu dan mesec godina: ");
        Scanner ulaz = new Scanner(System.in);
        int dan = ulaz.nextInt();
        int mesec = ulaz.nextInt();
        int godina = ulaz.nextInt();
        // ukupan broj dana u datom mesecu
        int maxDanaUMesecu = 0;
        switch (mesec) {
            case 1:
            case 3:
            case 5:
            case 7:
            case 8:
            case 10:
            case 12:
                maxDanaUMesecu = 31;
                break;
            case 4:
            case 6:
            case 9:
```

```

        case 11:
            maxDanaUMesecu = 30;
            break;
        case 2: {
            if (godina % 4 == 0) {
                maxDanaUMesecu = 29;
            } else {
                maxDanaUMesecu = 28;
            }
            break;
        }
        default:
            System.out.println("Pogrešan mesec!\n");
            System.exit(0);
    }
    // Izračunavanje sutrašnjeg datuma
    if (dan < maxDanaUMesecu)
        dan++;
    else
        dan = 1;
    if (mesec < 12)
        mesec++;
    else {
        mesec = 1;
        godina++;
    }
    System.out.print("Sutrašnji datum je: ");
    System.out.println(dan + " " + mesec + " " + godina);
}
}

```

ZADATAK 8

Samostalno vežbanje primene switch iskaza.

Zadatak 8-1:

Šta je **y** posle izvršenja **switch** komande? Preradite ove iskaze upotrebom **if-else** naredbe.

```

x = 3; y = 3;
switch (x + 3) {
    case 6: y = 1;
    default: y += 1;
}

```

Zadatak 8-2:

Šta je **x** posle izvršenja sledeće **if-else** naredbe? Primenom **switch** naredbe preradite ovaj kod.

```
int x = 1, a = 3;  
if (a == 1)  
    x += 5;  
else if (a == 2)  
    x += 10;  
else if (a == 3)  
    x += 16;  
else if (a == 4)  
    x += 34;
```

▼ Poglavlje 6

Uslovni operator (?)

ŠTA JE USLOVNI OPERATOR ?

Uslovni operator ? operator se može koristiti umesto iskaza if, a umesto iskazy else se koristi operator : .

U programskom jeziku Java postoji i jedan ternarni operator, tj. operator sa tri operanda - **uslovni operator ?:** (ili operator izbora). Pošto se ovaj operator može lako zameniti naredbom grananja, dobar stil programiranja nalaže da operator izbora treba koristiti samo u izuzetnim slučajevima. U svakom slučaju, opšti oblik uslovne operacije se piše na sledeći način:

```
logičkiIzraz ? izraz1 : izraz2
```

Obratite pažnju na to da se znak pitanja piše iza logičkog izraza i da znak dve-tačke razdvaja izraz1 i izraz2. Izvršavanje operacije izbora se odvija u dva koraka: Prvo se izračunava vrednost logičkog izraza, a zatim se u zavisnosti od njegove vrednosti dobija rezultat operacije izbora na sledeći način: ako logički izraz daje tačno, onda je konačni rezultat jednak izračunatoj vrednosti za izraz1 u suprotnom slučaju, ako logički izraz daje netačno, onda je konačni rezultat jednak izračunatoj vrednosti za izraz2. Na primer, naredbom dodele:

```
String parnost = (n % 2 == 0) ? ("Paran broj") : ("Neparan broj");
```

promenljivoj **parnost** skroz levo se dodeljuje string "Paran broj" ako je vrednost promenljive **n** paran broj, tj. ako je izračunata vrednost logičkog izraza **n % 2 == 0** jednaka tačno. Ali, promenljivoj **parnost** se dodeljuje string "Neparan broj" ako je

vrednost promenljive **n** neparan broj, tj. ako je izračunata vrednost logičkog izraza **n % 2 == 0** jednaka netačno. Primetimo da zagrade oko izraza u ovom primeru nisu neophodne, ali je s njima postignuta bolja razumljivost naredbe

Primer: Za dati broj **n** proveriti da li je deljiv sa 3 ili 5 i postaviti odgovarajuću vrednost (tačno ili netačno) logičkoj promenljivoj **deljivSa3Ili5**. Problem možemo rešiti na sledeći način koristeći naredbe **if**, **else-if** i **else**:

```
if (n % 3 == 0)
    deljivSa3Ili5 = true;
else if (n % 5 == 0)
    deljivSa3Ili5 = true;
else
    deljivSa3Ili5 = false;
```

Međutim, ovaj programski kod možemo skratiti vezivanjem uslova relacionim operatorima na sledeći način:

```
if ((n % 3 == 0) || (n % 5 == 0))
    deljivSa3Ili5 = true;
else
    deljivSa3Ili5 = false;
```

Na kraju, ovaj programski kod možemo da svedemo na jednu liniju upotrebom uslovnog operatora:

```
deljivSa3Ili5 = (n % 3 == 0 || n % 5 == 0);
```

OBJAŠNJENJE USLOVNOG OPERATORA

Uslovni operator ? omogućava skraćeno pisanje programa u odnosu na pisanje programa sa if-else naredbom, jer vraća jednu od dve moguće vrednosti.

Pogledajmo iskaz if koji se koristi za izračunavanje apsolutne vrednosti:

```
if ( vrednost < 0 )
    abs = - vrednost;
else
    abs = vrednost;
```

Sledeći iskaz radi isto, ali sa samo jednim iskazom.

```
abs = (value < 0 ) ? -value : value ;
```

U ovom iskazu se koristi uslovni operator. Sa desne strane operatora = je uslovni izraz. Ovaj izraz se ocenjuje preko uslovnog operatora i na kraju dodeljuje promenljivoj **abs**. To se može predstaviti na sledeći način:

```
uslov (tačno ili netačno) ? vrednost-ako-je-tačno : vrednost-ako-je-netačno;
```

Ovo funkcioniše na sledeći način:

1. Uslovni iskaz vraća jednu vrednost.
2. Rezultat može biti jedna od dve vrednosti.
3. Ako je uslov tačan, onda se koristi izraz između ? i :
4. Ako je uslov netačan, koristi se izraz između : i kraja.

Pogledajmo prethodni primer:

```
double vrednost = -34.569;
```

```
double abs;
```

`abs = (vrednost < 0) ? -vrednost : vrednost;`

1. uslov 2. rezultat je

je tačan +34.569

3. +34.569 se dodeljuje promenljivoj abs

PRIMER 15

Primena uslovnog operatora ?: umesto if-else naredbe

Napraviti program koji proverava da li je uneti ceo broj paran ili neparan. Ako je broj paran, množi se sa 2, a ako je neparan sa 3. Rezultat množenja ispisati na konzolu.

Umesto if-else naredbe koristimo uslovni operator ?. Deo koda iz prethodnog rešenja:

```
if(broj%2==0){
    broj *= 2;
}
else{
    broj *= 3;
}
```

Menjamo samo jednom linijom:

```
broj = (broj % 2 == 0) ? (broj*2) : (broj*3);
```

Objašnjenje:

Ako je vrednost logičkog izraza (`broj % 2 == 0`) tačna, onda će vrednost promenljive *broj* biti postavljena na (`broj*2`), a ako je netačna, onda na (`broj*3`).

Rešenje:

```
package primer14;

import java.util.Scanner;

public class Primer14 {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Unesi ceo broj: ");
        int broj = input.nextInt();

        // Provera da li je broj paran
```

```
    broj = (broj % 2 == 0) ? (broj * 2) : (broj * 3);

    System.out.println("Rezultat je: " + broj);

    input.close();
}
}
```

ZADATAK 9

Samostalno vežbanje uslovnog operatora (?)

Preradite donje iskaze upotrebom logičkog operatora.

```
if (ages >= 16)
    ticketPrice = 20;
else
    ticketPrice = 10;
```

ZADATAK 10

Dodatno vežbanje rada sa uslovnim operatorom (?)

Koji rezultat dobijate kada preko tastature uneste sledeće brojeve: 2,3 i 6 u donji program.

```
public class Test {
    public static void main(String[] args) {
        java.util.Scanner input = new java.util.Scanner(System.in);
        double x = input.nextDouble();
        double y = input.nextDouble();
        double z = input.nextDouble();
        System.out.println((x < y &&&& y < z) ? "sorted" : "not
sorted");
    }
}
```

REALCIONI OPERATORI (VIDEO)

Video pokazuje upotrebu relacionih operatora.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 7

Domaći zadaci

ZADACI 1 - 2

Ove zadatke student treba samostalno da uradi

Zadatak 1:

Treba napraviti program koji od korisnika zahteva da unese određenu cenu a zatim i broj meseci (može preko JOptionPane-a). Treba izračunati koliko ukupno korisnik treba da plati za taj broj meseci. ($\text{cena} \times \text{brojmeseci}$). Ukoliko korisnik treba da plati više od 10 000 treba mu ispisati poruku u konzoli: Plaćate previše dok ukoliko unese manje ili jednako 10 000 treba mu ispisati poruku: Trošite taman koliko treba.

Zadatak 2:

Treba napraviti program koji od korisnika zahteva da unese određenu cenu i tekst "pravno" ili "fizičko" (ukoliko unese nešto drugo javiti poruku "Greska u unosu!!!"). Ukoliko korisnik unese ispravno treba mu izračunati i prikazati cenu sa PDV-om (20%) dok ukoliko unese fizičko treba mu izračunati cenu sa porezom od 8%. Rezultat ispisati na konzoli.

ZADACI 3- 5

Zadaci primenjuju if-else i else-if iskaze

Zadatak 3:

Pretpostavimo da je $x = 3$ i $z = 2$. Pokaži rezultat, ako ga ima, sledećeg programa. Koji je rezultat ako je $x = 3$ i $y = 4$? Koji je rezultat ako je $x = 2$ i $z = 2$? Kompletirajte program i pokrenite ga.

```
if (x > 2) {  
    if (y > 2) {  
        z = x + y;  
        System.out.println("z is " + z);  
    }  
} else  
    System.out.println("x is " + x);
```

Zadatak 4:

Pretpostavimo da je $x = 2$ i $y = 3$. Prikaži rezultat sledećeg programa. Koji je rezultat ako je $x = 3$ i $y = 2$? Koji je rezultat ako je $x = 3$ i $y = 3$. Kompletirajte program i pokrenite ga.

```
if (x > 2) {  
    if (y > 2) {  
        int z = x + y;  
        System.out.println("z is " + z);  
    }  
}  
else  
    System.out.println("x is " + x);
```

Zadatak 5:

Šta ne valja u sledećem programu? Uradite ispravno program ukoliko je pogrešno urađen.

```
if (score >= 60.0)  
    System.out.println("D");  
else if (score >= 70.0)  
    System.out.println("C");  
else if (score >= 80.0)  
    System.out.println("B");  
else if (score >= 90.0)  
    System.out.println("A");  
else  
    System.out.println("F");
```

▼ Zaključak

GLAVNI POUKE OVE LEKCIJE

Uslovi grananja u programima definišu se korišćenjem relacionih i logičkih operatatora.

1. Logička promenljiva (**boolean**) može da ima jednu od dve vrednosti: **true** (tačno) i **false** (pogrešno).
2. Relacioni operatori (<, <=, ==, !=, >, >=) proizvode samo logičku vrednost.
3. Operatori grananja se koriste radi programiranja alternativnih puteva izvršenja programskih instrukcija. Postoji nekoliko tipova operatora grananja: jednosmerni **if** naredbe, dvosmerne **if-else** naredbe, povezane if naredbe, višestruke **if-else** naredbe, **switch** naredbe i uslovni isklazi.
4. Sve **if** naredbe služe za doniošenje odluka zavisno od uslova, tj. izraza sa logičkim i relacionim operatorima. Zavisno od obrade dobijene vrednosti true ili false, if naredba usmerava pravac akcija u jednom od dva pravca.
5. Logički operatori **&&**, **||**, **!** i **^** koriste logičke vrednosti i promenljive.
6. Prilikom obrade iskaza **p1 && p2** Java prvo obrađuje p1, a onda p2, ako je p1 **true** (istinit). Ako je p1 **false** (neistinit), onda se ne obrađuje p2. Kada obrađuje iskaz **p1 || p2** Java prvo obrađuje p1, a onda p2, ako je p1 **false** (neistinit). Ako je p1 true (istinit), ne obrađuje se p2.. Prema tome **&&** je logički AND operator, a **||** je logički OR operator.
7. Naredba **switch** pravi odluku upravljanja tokom programa, a koristeći iskaz tipa **char**, **byte**, **short**, **int**, ili **String**.
8. Ključna reč **break** je opcionalna u iskazu sa **switch** naredbom, ali se normalno upravlja na kraju svaku **case** segmenta da bi se preskočila nepotrebna obrada drugih case segmenata, čime se skraćuje vreme izvršenja **switch** naredbe.
9. Redosled izvršenja operacija vrši u skladu sa postavljenim zagradama, prioriteta operatora, i veza operatora.
10. Zagrade se mogu koristiti da bi se uticalo na redosled izvršenja instrukcija.
11. Operatori sa većim prioritetom se prvo izvršavaju. U slučaju operatora sa istim prioritetom, prednost u izvršenju se određuje na osnovu njihovih veza (**associativity**).
12. Svi binarni operatori se izvršavaju s leva u desno, sem operatori dodeljivanja (**assignment**), koji se izvršavaju s desna u levo.