Kontrola programskog toka

Sekvencija, selekcija, iteracija

Kontrola toka programa

Java naredbe koje služe sa kontrolu toka programa:

- Sekvencija:
  - blok naredba
- Selekcija (grananja):
  - *if (if else), switch*
- Iteracija:
  - *for, while, do while*
- Naredbe prekida programskog toka:
  - *break, continue, return,*
    try – catch – finally, throw

2

Iskazi, blokovi, sekvencija

- Iskaz je svaka instrukcija koja se završava sa ";"
  Primer: `ime = "Pera";`
- Blok je niz iskaza koji se nalaze između vitičastih zagrada:

```
{
    ime1 = "Pera";
    ime2 = "Rale";
}
```

- Blokovi mogu sadržati druge blokove
- **U Javi se sekvencija realizuje pomoću bloka.**

3

Selekcija - If naredba

Sintaksa:

```
if (logički izraz)
    naredba;          // u opštem slučaju je blok naredba
```

Semantika:

- samo ako je istinosna vrednost (tačnost) izraza u zagradi true, izvršava se naredba (ili blok naredba).

```
Npr.   if ( x < 10 ) x = 10;
```

- Ako je vrednost promenjive x manja od deset, tada se promenjivoj dodeljuje vrednost 10
- Ispravan zapis je takođe i:

```
if ( x < 10 )
    x = 10;
```

- ili `if ( x < 10 ) { x = 10;}`

4

Selekcija - If - else naredba

- Sintaksa:

```
if (logički izraz)
  naredba1;
else
  naredba2;
```

Semantika:

- Ispituje se istinosna vrednost logičkog izraza.
- Ako je vrednost izraza `true`, izvršava se naredba1, ako je vrednost `false`, izvršava se naredba2.

5

Selekcija - If - else naredba

Primer:

```
int x, staroX;
...........
if (x != staroX) {
    System.out.print("x je promenjena");
 }
 else {
    System.out.print("x nije promenjena");
 }
```

6

Ugnježdeni `if-else`

```
if (logički izraz)
  naredba1;
else
  naredba2;
```

- Kada se u okviru bloka naredba1 i/ili naredba2 pojavljuje ponovo if – else naredba, govorimo o ugnježdenom if-else

7

Ugnježdeni `if-else`

Primer:

Realnu promenljivu, ako joj je vrednost veća od 100, podeliti sa 100 ako je indikator *false*, a promeniti njenu vrednost u ostatak pri deljenju sa 100, ako je indikator *true*. Ako joj je vrednost manja ili jednaka 100, javiti grešku opsega.

```
boolean indikator;
float vred; boolean indikator = true;
.............
if ( vred > 100 ) {
  if (indikator == true)  {
     vred = vred % 100;   }   // ugnježdeni if-else
  else {
     vred = vred / 100.0; }
} else {
  System.out.print("Vrednost nije u
                    dozvoljenom opsegu"); }
```

8

Ugnježdeni `if-else`

- Često se koristi za izbor između više mogućnosti:

```
if ( n == 1 ) {
  // izvršavanje prvog bloka naredbi
}
else if ( n == 2 ) {
  // izvršavanje drugog bloka naredbi
}
else {
  //ako ni jedan od predhodnih uslova nije
  //zadovoljen izvršava se treći blok naredbi
}
```

9

Primer – ugnježdeni if-else

Primer:
Ispisati opisnu ocenu na osnovu broja bodova ako važi pravilo:
Ocena: preko 90 - odlično
između 60 i 90 – vrlo dobro
do 60 – nedovoljno (padanje)

```
int ocena;
.....
if( ocena > 90 ){
  System.out.println("Odlicno!");
} else if( ocena > 60 ){
  System.out.println("Vrlo dobro!");
} else {
  System.out.println("Zao nam je, pali ste");
}
```

10

Ugnježdeni if-else - česta greška -

Neispravno!

```
if( i == j )
    if ( j == k )
      System.out.print("i je jednako k");
    else
      System.out.print("i nije jednako j");
  // ovde znamo samo da j i i nisu jednaki k!
```

else se uvek odnosi na najbliži if uslov!

Ispravno !

```
if( i == j ) {
  if ( j == k )
  System.out.print("i jednako k");
}
else{
  System.out.print("i nije jednako j");
}
```

11

De Morganovi zakoni (Bulova algebra)

- Negacija disjunkcije jednaka je konjunkciji negacija

$$\neg(x \vee y) = \neg x \wedge \neg y$$

- Negacija konjunkcije jednaka je disjunkciji negacija

$$\neg(x \wedge y) = \neg x \vee \neg y$$

12

De Morganovi zakoni i složeni logički izrazi u if naredbi

- Ako su a i b logičke promenljive ili izrazi, važe sledeće jednakosti:

  ! (a && b) = ! a || ! b

  ! (a || b) = ! a && ! b

- Ove transformacije se koriste kod formulisanja složenih uslova u *if-else* naredbama, odnosno kod negacije složenih logičkih uslova

13

Primeri: Složeni logički izraz u if naredbi

Primer1: Bonus od 2000 se dobija ako se proda više od 10 artikala ili se prodajom ostvari bar 85000

```
double vrednostProdaje = 0.0;
int brojProdaja = 0;
.......
if (vrednostProdaje >= 85000 ||
   brojProdaja > 10){
   System.out.println("Vas bonus je: 2000."); }
```

Primer2: Kandidat se prima ako je mlađi od 19 godina i ima prosek bar 3

```
int brGodina; float ocena;
......
if ( brGodina < 19 && ocena >= 3.0 ){
 System.out.println("Primljeni ste");}
```

14

Primeri: Negacija složenih logičkih izraza u if naredbi

Primer1: Bonus od 2000 se dobija ako se proda više od 10 artikala ili se prodajom ostvari bar 85000. Ispisati kada se ne dobija bonus.

```
double vrednostProdaje = 0.0;
int brojProdaja = 0;
.......
if (vrednostProdaje < 85000  &&
    brojProdaja <= 10){
   System.out.println("Nemate bonus!"); }
```

Primer2: Kandidat se prima ako je mlađi od 19 godina i ima prosek bar 3. Ispisati kada se kandidat ne prima.

```
int brGodina; float ocena;
......
if ( brGodina >= 19 || ocena < 3.0 ){
 System.out.println("Niste primljeni!");}
```

15

switch naredba

- **Sintaksa** switch naredbe:

```
switch(promenjiva){
  case(vrednost1):
      naredba_1;   // blok naredba
      break;
  case(vrednost2):
      naredba_2;
      break;
  ...
  default:
      naredba_n;
      break;
}
```

- upravljačka promenjiva može biti prostog tipa, osim realnog, kao i String objekat

16

`switch` naredba

- **Semantika** `switch` naredbe

```
switch (n) {
  case 1:
   // ako je n = 1 izvršava se prvi blok naredbi
     nar_1;
     break;  // prekida se switch naredba
  case 2:
   // ako je n = 2 izvršava se drugi blok naredbi
     nar_2;
     break; // prekida se switch naredba
  default:
 /* ako ni jedan od predhodnih uslova nije
    zadovoljen, izvršava se default blok naredbi
      nar_def;
      break;   */
}
```

17

Primer `switch`

```
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("Greška u mesecu");break;
      }
   }
}
```

18

Zadaci - switch naredba

- Napisati program koji od korisnika traži unos dva realna broja a zatim obavlja računsku operaciju zbir, razlika, proizvod ili količnik, u zavisnosti od izbora korisnika.

  Pomoć: od korisnika se traži unos broja: 1 za zbir, 2 za razliku, 3 za proizvod, 4 za količnik. Potrebno je koristiti promenjivu tipa *int* koja će primiti odgovarajuću vrednost

- Napisati program koji učitava godinu studija studenta u obliku celog broja i zatim štampa da li je student brucoš, student druge godine, student treće godine ili apsolvent, u zavisnosti da li je unet broj 1, 2, 3 ili 4.

  Dati dva rešenja, uz pomoć **if** i uz pomoć **switch** naredbe.

19

`for` petlja

Sintaksa for petlje:

```
for(inicijalizacija; uslov; inkrement){
    naredba;
}
```

Semantika for petlje:

1. Vrši se inicijalizacija kontrolne promenljive petlje
2. Proverava se uslov (izlazni kriterijum)
3. Ako je uslov *true*:
   1. izvršava se naredba (telo petlje)
   2. povećava se kontrolna promenljiva (inkrement)
   3. ponavlja se od koraka 2.
4. Ako je uslov *false*, prelazi se na sledeću naredbu iza petlje

20

for petlja, ugnježdena for petlja

```
for ( int i = 0; i < n; i++ ) {
   //  i dobija vrednosti od  0 do n-1
   //  telo petlje će se izvršiti n puta
}
```

- Ugnježdena for petlja:

```
for ( int i = 0; i < m; i++ ) {
  for ( int j = 0; j < n; j++ ){
    // telo petlje će se izvršiti  m x n  puta
  }
}
```

Primer:

```
int i;      // može i ovako
for( i = 0; i < 10; i++ ){
    System.out.println(i);
}
```

// ovde **i** nije lokalna promenljiva petlje, već je vidljiva i izvan petlje

Primeri - for petlja

- Napisati program koji ispisuje sve parne brojeve od 1 do 30

```
public class parniDo30{
  public static void main(String[] args){
    System.out.println
         ("Parni brojevi od 1 do 30 su:");
    for (int i = 2; i <= 30; i += 2){
        System.out.println(i);
    }
  }
}
```


while petlja

- Sintaksa **while** petlje

```
while(uslov){
      naredba;  // blok naredba
}
```

- Semantika **while** petlje

1. Proverava se uslov
2. Ako je uslov true, izvršava se naredba (telo petlje)
3. Ako je uslov false, prelazi se na sledeću naredbu posle while

- Moguće je da se ni jednom ne izvrši telo while petlje ako je uslov odmah *false*!

Primer while

```
class WhileDemo {

  public static void main(String[] args){
     int broj = 1;
     while (broj < 11) {
           System.out.print(broj + "," );
           broj++;
     }
  }
}
```

Šta će biti ispisano na kraju programa?

Primer `while`

```
class WhileDemo {

  public static void main(String[] args){
      int broj = 1;
      while (broj < 11) {
            System.out.print(broj + "," );
            broj++;
      }
  }
}
```

```
1,2,3,….,10
```

25

`do while` petlja

- Sintaksa **do while** petlje

```
do {
     naredba;        // blok naredba
   } while(uslov);
```

- Semantika **do while** petlje

1. Izvršava se naredba (telo petlje)
2. Proverava se uslov
3. Ako je uslov true, izvršava se naredba
4. Ako je uslov false, prelazi se na sledeću naredbu posle do while

- Telo `do while` petlje se izvršava bar jednom!

26

Primer: `do while`

```
class DoWhileDemo {

  public static void main(String[] args){
      int br = 1;
      do {
            System.out.print(br + ",");
            br++;
      } while (br <= 11);
  }

}
```

Šta će biti ispisano na kraju programa?

27

Primer: `do while`

```
class DoWhileDemo {

  public static void main(String[] args){
      int br = 1;
      do {
            System.out.print(br + ",");
            br++;
      } while (br <= 11);
  }

}
```

```
1,2,3,….,10,11
```

28

for - primer

Pitanje: Šta radi ovaj program?

```
class ForDemo {
   public static void main(String[] args){
        for(int i=1; i<11; i++){
          System.out.println(i);
        }
   }
}
```

29

for - primer

Pitanje: Šta radi ovaj program?

```
class ForDemo {
   public static void main(String[] args){
        for(int i=1; i<11; i++){
          System.out.println(i);
        }
   }
}
```

Odgovor: Ispisuje 1
2
3
4
5
6
7
8
9
10

30

break, continue

- **break**
  - izaziva bezuslovni prekid `while, do while` ili `for` petlje, kao i `switch` naredbe i
  - skok na prvu sledeću naredbu
- **continue**
  - se koristiti samo u naredbama ciklusa (`while, do while` ili `for`)
  - izaziva bezuslovni prekid tekuće iteracije i prelazak na sledeću iteraciju u petlji

31

break naredba

Primer:

- Deo programa ispisuje redom sve elemente niza celih brojeva a dužine 100, dok ne naiđe na element koji je veći od 80.

  Tada nakon ispisa prekida ovu petlju i prelazi na sledeću naredbu.

```
.....
for (int i = 0; i < 100;  i++) {
    if ( a[ i ] > 80 ) break;
   System.out.println( a[ i ] );
}
....  //  sledeća naredba
```

32

`continue` naredba

Sintaksa:

```
// telo petlje
{ .........
   if (uslov)
     continue;
   .......
 }
```

Semantika:

- continue izaziva bezuslovni prekid tekuće iteracije i prelazak na sledeću iteraciju u petlji.
- može se koristiti samo u naredbama ciklusa (`while`, `do while` ili `for`).

33

`continue` naredba

Primer: Dat je niz **a** koji sadrži 100 celih brojeva. Ispisati sve elemente niza koji su jednaki 0.

// deo programa sa petljom

```
for ( int i = 0; i < 100; i++ ) {
     if ( a[i] != 0 ) continue;
     System.out.println
          ( "a[" + i + "] = " + a[i]);
}
```

Pitanje: Koliko puta se izvršava if naredba, a koliko puta metod println()?

34

`continue` naredba

// deo programa sa petljom

```
for ( int i = 0; i < 100; i++ ) {
     if ( a[i] != 0 ) continue;
     System.out.println
          ( "a[" + i + "] = " + a[i]);
}
```

Pitanje: Koliko puta se izvršava if naredba, a koliko puta println()?

Odgovor: If će se izvršiti 100 puta, a ispis (metod println()) onoliko puta koliko u nizu bude 0 elemenata.

35

Pretraživanje niza – osnovni algoritam

- U datom nizu koji sadrži 100 celih brojeva naći prvu pojavu zadatog broja.

```
.....          // nedostaje deklaracija niza a
int index, clan = 7;
    // clan je promenljiva za broj koji se traži (7)
for (int i = 0; i < 100, i++) {
     if ( a[i] == clan ) {
        index = i;
        break;
     }
}    // program nakon break-a nastavlja u ovoj tački
System.out.print("Traženi elemenat - " + clan +
     " se javlja prvi put na " + index + " . mestu");
```

Pitanje: Čemu služi promenljiva index i šta će po izlasku iz petlje biti u njoj?

36

Pretraživanje niza – osnovni algoritam

Odgovor:

Promenljiva index pamti poziciju elementa niza koji se traži.

37