Objektno orjentisano programiranje:

Naredbe za kontrolu izvršenja programa

[Prilagođeno od prof. dr Gordana Đorđević]

Algoritamske strukture

- Svaki niz komandi koji se napiše u okviru tela neke metode predstavlja algoritam odnosno niz koraka koji vodi ka rešenju nekog problema.
- Algoritamske strukture koje se mogu primeniti u Javi su:
  - Linijska algoritamska struktura – svaka komanda se izvršava samo jedanput i to u redosledu u kojem je napisana, bezuslovno i bez ponavljanja
  - Razgranata algoritamska struktura – na početku se proverava određeni uslov, pa ako je uslov ispunjen izvršava se jedna grupa naredbi, a ako nije izvršava se neka druga grupa naredbi
  - Ciklična algoritamska struktura – određena naredba (ili skup naredbi) se izvršava više puta zaredom

Realizacija algoritamskih struktura u Javi

- Realizacija linijske strukture u Javi je veoma jednostavna; dovoljno je napisati naredbe jednu ispod druge i one će se izvršavati sekvencijalno
- Za realizaciju razgranate i ciklične algoritamske strukture koriste se naredbe za kontrolu toka izvršavanja programa:
  - Naredbe za uslovno granjanje (razgranata algoritamska struktura)
    - IF naredba
    - SWITCH naredba
  - Naredbe za ciklično ponavljanje (ciklična algoritamska struktura)
    - FOR naredba
    - WHILE naredba
    - DO – WHILE naredba

IF naredba

- Deklaracija IF naredbe

```
if (...uslov...)    komanda_1;
```

- Deklaracija počinje rezervisanom reči **if** (malim slovima)
- nakon rezervisane reči piše se logički uslov koji se proverava; uslov je logički izraz koji se može proveriti i svesti na tačan ili netačan
- Nakon uslova ispisuje se komanda/e koja se izvršava ako je uslov ispunjen

Operatori za kreiranje logičkog izraza

- Operatori za poređenje vrednosti se mogu koristiti isključivo sa prostim tipovima podataka, jer ne funkcionišu u situacijama kada treba porediti dva objekta (String, Calendar i drugi objekti)

Napomena: operatori "=" i "==" su različiti. "=" je operator za dodeljivanje vrednosti, a "==" je operator za poređenje jednakosti

Operator	Opis
>	Veće od
<	Manje od
>=	Veće ili jednako
<=	Manje ili jednako
==	Jednako
!=	različito

Primer: IF naredba

- Napravi klasu ProveraCelihBrojeva koja ima:
  - Metodu proveriZnak koja proverava da li je broj koji se prosleđuje kao parametar pozitivan, negativan ili 0 i ispisuje poruku o tome na ekranu u formi "Broj____je____".
  - Metodu proveriVeceManjeJednako koja prima dva broja A i B kao parametre i proverava da li je A>B, A<B ili A=B i ispisuje odgovarajuću poruku o tome na ekranu

```
Class ProveraCelihBrojeva {

        void proveriZnak (int a) {
        if (a == 0)
                System.out.println ("Broj "+a+"je jednak nuli");
        if (a > 0)
                 System.out.println ("Broj "+a+"je veći od nule");
        if (a < 0)
                 System.out.println ("Broj "+a+"je manji od nule");
        }

        void proveriVeceManjeJednako (inta, intb) {
         if (a > b)
                System.out.println ("Broj "+a+"je veći od broja"+b);
        if (a == b)
                System.out.println ("Broj "+a+"je jednak broju"+b);
        if (a < b)
                System.out.println ("Broj "+a+"je manji od broja"+b);
        }
}
```

IF naredba

- If naredba može imati i else deo (ali ne mora).
- Posle rezervisane reči else piše se komanda koja se izvršava ukoliko uslov nije ispunjen

```
if (...uslov...)     komanda_1;
                else komanda_2;
```

Primer: IF naredba

- Dopuni klasu ProveraCelihBrojeva tako da ima:
  - Metodu razlicito koja prima dva broja kao parametre i vraća TRUE ako su brojevi različiti, a u suprotnom vraća FALSE
  - Metodu proveriParnost koja prima ceo broj kao parametar i proverava da li je paran ili neparan; metoda vraća TRUE ako je paran, a FALSE ako je neparan

```
boolean razlicito (int a, int b) {
        if (a != b) return true;
             else return false;
      }

      boolean proveriParnost (int a) {
        if (a%2 == 0) return true;
             else return false;
      }
```

Složeni uslovi u IF naredbi

Operator	Opis	primeri
&&	I (AND)	((a>2)&&(a<5))
||	ILI (OR)	((x<0)||(x<25.5))
!	NE (NOT)	(!(a>b))

Primer

- Dopuni klasu ProveraCelihBrojeva tako da ima i:
  - Metodu koja proverava da li je parametar a u rasponu od 100 do 200 uključujući i te vrednosti; ako jeste metoda vraća TRUE, a inače FALSE
  - Metodu koja proverava da li je parametar a manji od 0 ili veći od 50; metoda vraća TRUE ako je ispunjen jedan ili drugi uslov, a inače vraća FALSE

```
boolean proveraRaspona1 (int a) {
  if ((a >= 100)&&(a <= 200)) return true;
      else return false;
}

boolean proveraRaspona2 (int a) {
  if ((a < 0)||(a > 50)) return true;
      else return false;
}
```

Ugnježdene IF naredbe

- U nekim situacijama se traži da se nakon provere jednog uslova provere još neki uslovi, ali samo ako je prvi uslov zadovoljen

```
if (...uslov1...) if (...uslov2...) komanda_11;
                                    else komanda_12;
                  else if (...uslov3...) komanda_21;
                                    else komanda_22;
```

Primer: Ugnježdene IF naredbe

- Napravite klasu VisinskeGrupe tako da ima:
  - Metodu proveriVisinu koja kao parametar prima visinu neke osobe u cm (realan broj) i ispisuje na ekranu da li ta osoba pripada niskim osobama (manje od ili jednako 158 cm), srednje visokim osobama (više od 158 cm, manje od ili jednako 179 cm) ili visokim osobama (više od 179 cm). Ako je uneta visina van granica (120 do 240 cm) na ekranu je potrebno ispisati samo poruku o grešci.

```
Class VisinskeGrupe {
   void proveriVisinu (double visina) {
        if ((visina < 120)|| (visina > 240))
                System.out.println ("Visina je van granica");
        else
                if (visina <= 158)
                        System.out.println ("Osoba je niska");
                if ((visina > 158)&& (visina <= 179))
                        System.out.println ("Osoba je srednje
   visine");
                if (visina > 179)
                        System.out.println ("Osoba je visoka");

        }

}
```

Primer: Ugnježdene IF naredbe

- Napraviti klasu **Automobil**. Ova klasa bi trebalo da ima:
  - Atribut **model.**
  - Atribut **marka.**
  - Atribut **kubikaza** (ceo broj).
  - Atribut **snaga** (ceo broj).

- Napraviti klasu **TroskoviRegistracije** koja ima:
  - Statičku metodu **obracunajOsiguranje** koja kao parametar dobija objekat klase Automobil i vraća iznos koji je potrebno platiti kao obavezno osiguranje pri registraciji. Tarife osiguranja zavise od snage motora i date su u sledećoj tabeli. Pre izračunavanja, potrebno je proveriti da li je snaga u okviru granica (20-1000 KS), pa ako nije, ispisati poruku o grešci na ekranu i vratiti vrednost 0.

Snaga (KS)	Cena (din)
s<55	3.000,00
55<=s<75	4.500,00
75<=s<150	7.300,00
150<=s	9.000,00

Rešenje

```
Class Automobil {
        String marka;
        String model;
        int kubikaza;
        int snaga;
}
static double obracunajOsiguranje (Automobil a) {
        If ((a.snaga < 2) || (a.snaga > 1000)) {
                System.out.println ("Snaga je van granica");
                return 0;
                }
        else {
                 If(a.snaga < 55) return 3000;
                 If ((a.snaga >= 55) && (a.snaga < 75)) return 4500;
                 If ((a.snaga >= 75) && (a.snaga < 150)) return 7300;
                 If(a.snaga >150) return 9000;
                return 0;
        }
}
```

SWITCH naredba

- SWITCH naredba omogućava ispitivanje više uslova odjednom i predstavlja način za izvršavanje *višestrukog grananja* u Javi
- Deklaracija SWITCH naredbe

```
switch (...selektor...) {
                    case vrednost_1: komanda_1;
                                                          break;
                    case vrednost_1: komanda_1;
                                                          break;
:
:
                    default: komanda_d;
}

}
```

- Selektor može biti bilo koja promenljiva koja je celobrojnog ili char tipa (promenljive drugih tipova ne mogu biti selektor). Selektor može biti i neki izraz čiji rezultat je celobrojnog ili char tipa.
- SWITCH naredba ima i telo u kojem su definisane sve moguće grane i komande koje bi trebalo izvršiti u okviru grane
- Grana se definiše rezervisanom rečju case iza koje sledi neka vrednost
- Pri izvršavanju ove naredbe poredi se trenutna vrednost selektora sa vrednostima koje se nalaze iza reči case; kada se nađe grana koja ima vrednost jednaku vrednosti selektora, izvršava se komanda koja je napisana u produžetku te grane. Ako se ne nađe ni jedna grana koja ima vrednost identičnu vrednosti selektora, izvršava se ona grana koja je označena sa default.
- SWITCH naredba ne mora imati default granu
- SWITCH naredba podrazumeva da se bira i izvršava samo jedna grana; break naredba služi da prekine izvršavanje SWITCH naredbe nakon što se pronađe grana koja zadovoljava postavljeni uslov.

Primer: SWITCH naredba

- Napraviti klasu Sifarnik koja ima:
  - statičku metodu koja kao parametar dobija ocenu učenika i na ekranu ispisuje da li je u pritanju ocena: odličan, vrlo dobar, dobar, dovoljan ili nedovoljan. Ako je unet broj koji je veći od 5 ili manji od 1 ispisati poruku o grešci.

```
Class Sifarnik {
    static void ispisiOcenu (int ocena) {
          switch (ocena) {
          case 5:
                    System.out.println ("Odličan");
                    break;
          case 4:
                    System.out.println ("Vrlo dobar");
                    break;
          case 3:
                    System.out.println ("Dobar");
                    break;
          case 2:
                    System.out.println ("Dovoljan");
                    break;
          case 1:
                    System.out.println ("Nedovoljan");
                    break;

          default:
                    System.out.println ("Greška");
          }

}
```

FOR naredba

- Ponekada je za rešavanje nekih konkretnih situacija potrebno jednu ili više naredbi ponoviti veći broj puta. U ovakvim situacijama koriste se naredbe za ciklično ponavljanje poput FOR naredbe
- Deklaracija FOR naredbe:

```
for (komanda_z1; uslov_z; komanda_z2) komanda_p;
```

- Deklaracija počinje rezervisanom reči `for` posle koje sledi zagrada sa dve komande i uslovom
- Prva komanda u zagradi (`komanda_z1`) izvršava se samo jednom i to na početku
- Uslov predstavlja logički izraz koji se proverava pre izvršavanja svake iteracije (kruga). Ako je uslov zadovoljen izvršiće se još jedna iteracija, ako nije, petlja se prekida.
- Poslednja komanda u zagradi (`komanda_z2`) izvršava se na kraju svake iteracije
- Posle zagrade se nalazi komadna (`komanda_p`) koju je potrebno izvršiti više puta

Koraci izvršavanja petlje

- Početak petlje
  - Izvršava se `komanda_z1`
- 1. iteracija
  - Proverava se `uslov_z` (uslov važi)
  - Izvršava se `komanda_p`
  - Izvršava se `komanda_z2`
- 2. iteracija
  - Proverava se `uslov_z` (uslov važi)
  - Izvršava se `komanda_p`
  - Izvršava se `komanda_z2`
- 3. iteracija
  - Proverava se `uslov_z` (uslov važi)
  - Izvršava se `komanda_p`
  - Izvršava se `komanda_z2`
- poslednja iteracija
  - Proverava se `uslov_z` (uslov ne važi – petlja se prekida)
- Kraj petlje

Primer: FOR naredba

- Napraviti klasu Ispisivac koja ima:
  - Statičku metodu ispisiPoruku koja na ekranu pet puta ispisuje poruku “Dobar dan”

```
Class Ispisivac {
  static void ispisiPoruku () {
     for (i=1; i<=5; i++)
          System.out.println (“Dobar dan”);
     }

}
```

Primer: FOR naredba

- Dodaj klasi Ispisivac:
  - Statičku metodu ispisiBroj koja na ekranu ispisuje brojeve od 0 do 30

```
Class Ispisivac {
  static void ispisiBroj () {
     for (i=0; i<=30; i++)
              System.out.println (i);
     }
}
```

Primer: FOR naredba

- Dodaj klasi Ispisivac:
  - Statičku metodu ispisiParneBrojeve koja će na ekranu ispisati sve brojeve u rasponu od 1 do 25 koji su parni

```
Class Ispisivac {
   static void ispisiParneBrojeve () {
      for (i=0; i<=25; i++)
      If (i%2 == 0) System.out.println (i);
      }
}
```

WHILE naredba

- Kada je za rešavanje nekih konkretnih situacija potrebno jednu ili više naredbi ponoviti veći broj puta, a pri tome je broj ponavljanja unapred poznat korisno je upotrebiti FOR naredbu
- U slučaju kada je broj ponavljanja unapred nepoznat koristi se WHILE naredba
- Deklaracija WHILE naredbe:

```
while (...uslov...) komanda_p;
```

- Deklaracija počinje rezervisanom reči `while` posle koje sledi zagrada sa nekim logičkim uslovom
- Posle zagrade piše se komanda ili više komandi koje je potrebno ponoviti više puta
- Za razliku od FOR petlje WHILE petlja nema brojač; izlazak iz petlje ne zavisi od vrednosti brojača, već se dešava kada uslov u zagradi prestane da važi
- Uslov je logički izraz i formira se na isti način kao kod IF naredbe
- Ukoliko se desi na samom početku petlje da uslov nije ispunjen `komanda_p` se neće izvršiti nijednom (neće se izvršiti nijedna iteracija)

Koraci izvršavanja petlje

- 1. iteracija
  - Proverava se `uslov_` (uslov važi)
  - Izvršava se `komanda_p`
- 2. iteracija
  - Proverava se `uslov_` (uslov važi)
  - Izvršava se `komanda_p`
- 3. iteracija
  - Proverava se `uslov_` (uslov važi)
  - Izvršava se `komanda_p`
- poslednja iteracija
  - Proverava se `uslov_`(uslov ne važi – petlja se prekida)
- Kraj petlje

Primer: WHILE naredba

- Napravi klasu Uvecanje koja ima:
  - statičku metodu koja kao parametar dobija neki pozitivan ceo broj A i množi ga samim sobom sve dok ne postane veći od 1000. Ovako umnožen broj je potrebno ispisati na ekranu.

```
Class Uvecanje {
  static void uvecajBroj (int a) {
      int rez = 1
      while (rez < 1000) rez = rez * a;
      System.out.println (rez);
      }
}
```

Primer: WHILE naredba

- Dodaj klasi Ispisivac:
  - Statičku metodu koja ispisuje na ekranju prvih N celih brojeva većih od nule koji su deljivi sa 5 ili sa 6. N je dato kao parametar metode.

```
Class Ispisivac {
   static void ispisiPrvihNBrojeva () {
        int brojdeljivih = 0;
        int i = 1;
        while (brojdeljivih < n){
               if ((i % 5 == 0) || (i % 6 == 0)){
                      System.out.println (i);
                      brojdeljivih ++;
               }
               i++;
        }
   }
}
```

DO-WHILE naredba

- Ukoliko se desi da uslov nije ispunjen na samom početku WHILE petlje, nijedna iteracija se neće izvršiti. Ponekad je potrebno da se obezbedi izvršenje bar jedne iteracije. U tom slučaju se koristi naredba DO-WHILE.
- Deklaracija DO-WHILE naredbe:

```
do
   komanda_p;
while (...uslov...);
```

- 1. iteracija
  - Izvršava se `komanda_p`
  - Proverava se `uslov_` (uslov važi)
- 2. iteracija
  - Izvršava se `komanda_p`
  - Proverava se `uslov_` (uslov važi)
- 3. iteracija
  - Izvršava se `komanda_p`
  - Proverava se `uslov_` (uslov važi)
- poslednja iteracija
  - Izvršava se `komanda_p`
  - Proverava se `uslov_`(uslov ne važi – petlja se prekida)
- Kraj petlje