5. Nizovi i upravljačke strukture u Javi

5.1. Nizovi

Niz - celina koju čini kolekcija podataka istog tipa.

To je **objekat** sa kojim se operiše kao i sa drugim objektima.

Sastoji se iz elemenata istog tipa (primitivnog ili referentnog).

5.1.1. Deklarisanje nizova (postoje 2 načina):

Primeri:

```
int   pozicija[];
String niska[];
Knjiga naslov[];
// Iste deklaracije na 2. nacin
int [] pozicija;
String [] niska;
Knjiga [] naslov;
```

5.1.2. Kreiranje nizova

Postoje 2 načina: (1) pomoću new-operatora,
(2) nabrajanjem članova

Primeri:

```
Point [] tacke = new Point[20];    // 1.nacin
int [] brojac =  new int[100];      // 1. nacin
// Sledi 2. nacin kreiranja
String [] godDoba;
godDoba ={“prolece”, “leto”, “jesen”, “zima”}
```

5.1.3 Pristup članovima niza

Numeracija članova niza počinje od 0.

Članovima niza pristupamo pomoću: *ImeNiza[indeks]*

Primeri:

```
brojac[50]=2; //  Ne valja: brojac[100] = 3;
x = godDoba[3];
```

5.1.4. Rad sa višedimenzionim nizovima

Višedimenzioni nizovi su nizovi nizova.

Deklarcija se vrši slično kao kod jednodimenzionih nizova:

```
int  dvo [][] ={{1,2,5}, {6,7,8}};
float  tro [][][];
int [][] mat;
```

Kreiranje se vrši pomoću new-operatora ili navođenjem članova niza kao u prvoj naredbi.

```
int mat = new int[10][20];
float tro = new float[10][8][20];
```

Pristup članovima višedimenzionih nizova se vrši kao članovima jednodimenzionih.

```
mat[0][3] = 5; x= mat[7][15];

tro[0][[0][0] =1;
```

5.2. Kontrolne (upravljačke) strukture u Javi

Preuzete iz C/C++

Razlikujemo:

- Blok (sastavnu naredbu)
- Upravljačke strukture grananja
- Upravljačke strukture za opis ciklusa.
- Upravljačke naredbe break i continue.

5.2.1.Blok – niz naredbi ograđen vitičastim zagradama.

Primer:

```
void metod1() {
    int k=30;
    { int  n =10;
      System.out.println("k = "+k);
      System.out.println("n = "+n);
     }
}
```

Promenljiva n je lokalna i nije vidljiva izvan uutra{njeg bloka.

5.2.2. Upravljačke strukture grananja

5.2.2.1. if–naredba

Ima jedan od oblika:

- *if (uslov) naredba ;*
- *if (uslov) naredba; else naredba*;

Primeri:

```
if  (a < b ) System.out.println("a je manje od b");
if (tezina>200) {tezina = 200;
                 ind = true;
                };

else  ind =false;
```

Umesto if-naredbe može se upotebiti operator ? : Ima oblik:

(uslov) ? then–grana : else–grana

Primmer:

```
min = (x<y) ? x : y;
```

5.2.2.2. switch-naredba

Umesto:

```
if (oper = ='+')
        sabrati(arg1, arg2);
else if (oper = = '-')
        oduzeti (arg1, arg2);
else if (oper = = '*')
       pomnozi(arg1,arg2);
  ……….
```

preglednije je:

```
switch(test) {
   case  vrednost1:
            Rezultat1;
         break;
   case  vrednost2:
            Rezultat2;
         break;
   ……
   default: PodrazumevaniRezultat; // opciono
  }
```

Primer: Switch (doba) {

```
case ‘p’: System.out.println(“Setnja”);
          break;
case ‘l’: System.out.println(“Kupanje”);
          break;
case ‘j’: System.out.printl(“Branje grozdja”);
          break;
case ‘z’: System.out.println(“Skijanje”);
          break;
default: System.out.println(“Greska”);
}
```

Primeri:
```
switch (doba) {
    case  “prolece”:
        System.out.println(“Setnja u prirodi”);
      break;
    case “leto”:
        System.out.println(“Kupanje”);
      break;
    case “jesen”:
        System.out.println(Branje voca”);
      break;
    case “zima”:
        System.out.println(“Skijanje”);
    break;
}
```
//
```
switch (broj) {
    case  2:
    case  4:
            f = broj*broj+3;
        break;
    default:   f = broj;
}
```

5.2.3. Cikličke upravljačke strukture

5.2.3.1. for-petlja

Ima oblik:

for (inicijalizacija; provera; izmena) naredba

Bilo koja komponenta for-naredbe može biti izostavljena, ali simbol ; ne mo`e biti izostavljen..

Primeri:

```
for (i=0; i<niz.length; i++) niz[i]=0;
for (int i=0; i<100; i += 2) { a[i]=i+1;
                               b[i]=3;
                             }
```

5.2.3.2. while-petlja (petlja sa preduslovom)

Ima oblik:

while (uslov) naredba

Primer:

```
int  brojac = 0;
while (brojac < 10) {
        f = x-brojac;
        brojac++;
  }
```

5.2.3.3. do-while petlja (petlja sa postuslovom)

Ima oblik:

do {Naredbe} while (uslov);

Primer:

```
int  k = 0;
do  {
      System.out.println (“k = “+k);
      k++;
 } while (k<=10);
```

5.2.4. Upravljačke naredbe break i continue

5.2.4.1 break-naredba **služi za izlaz iz ciklusa (prekid toka)**

Primeri:

```
int brojac =0;
while (brojac <niz.length) {
   if(niz[brojac]= =0) {
      break;
    }
    vektor[brojac] = niz[brojac++];
 }
```

5.2.4.2 continue-naredba - slu`i za nastavljanje ciklusa (toka programa)

```
br1 = 0;
br2 = 0;
while (br1 < niz.length) {
    if  (niz[br1] = = 0)
            continue;
    vektor[br2++]=niz[br1++];
 }
```

Obe naredbe (break i continue) mogu imati labele (identifikator koji mora postojati negde u programu iza kojeg sledi simbol: “:”

```
pocetak:
    for (int i = 0; i < 10; i++) {
        while ( k <  40) {
            if  (i*k= =100)
                    break pocetak;
              ……………
          }
```

Kontrolna pitanja

31. Na koje načine možemo kreirati niz?
32. Ako imamo niz x[20] koje od sledećih naredbi se mogu izvršiti.
33. Navedite vrste kontrolnih struktura u Javi.
34. Opišite if petlju.
35. Opišite switch petlju.
36. Opišite for petlju.

Kontrolna pitanja

37. Opišite while petlju.
38. Napišite oblik do-while petlje.
39. Čemu služe upravljačke naredbe break i continue (navedite primere)?