Ciklusi i nizovi

Autori:
Goran Savić
Milan Segedinac

1. Ciklusi

Ciklusi (petlje) omogućuju izvršavanje istog koda više puta. Na ovaj način izbegava se dupliranje koda. Takođe, omogućuje se da broj ponovnih izvršavanja koda bude dinamički određen na osnovu stanja podataka u programu. Ponovno izvršavanje određenog koda nazivamo iteriranje kroz kod. U tom smislu, jedna iteracija predstavlja jedno izvršavanje među više ponovljenih izvršavanja.

While ciklus

While ciklus omogućuje ponovno izvršavanje bloka koda dok god je ispunjen određeni uslov. While ciklus se može posmatrati kao generalizacija if naredbe. Dok if naredba jednom izvršava blok koda ako je uslov ispunjen, while ciklus blok koda izvršava iznova dok god je uslov ispunjen. Primer korišćenja while ciklusa je dat u primeru.

```
int broj = 20;
while (broj > 1) {
      System.out.println("Broj je: " + broj);
      broj /= 2;
}
```

Dok god je uslov u zaglavlju while ciklusa ispunjen, izvršavaće se definisano telo ciklusa. Konkretno, program u primeru će ispisati vrednosti 20, 10, 5, 2. Na početku svake iteracije proverava se ispunjenost uslova. Kao i kod if naredbe, u zaglavlju može da stoji bilo kakav uslovni izraz (važno je samo da vraća logičku vrednost kao rezultat).

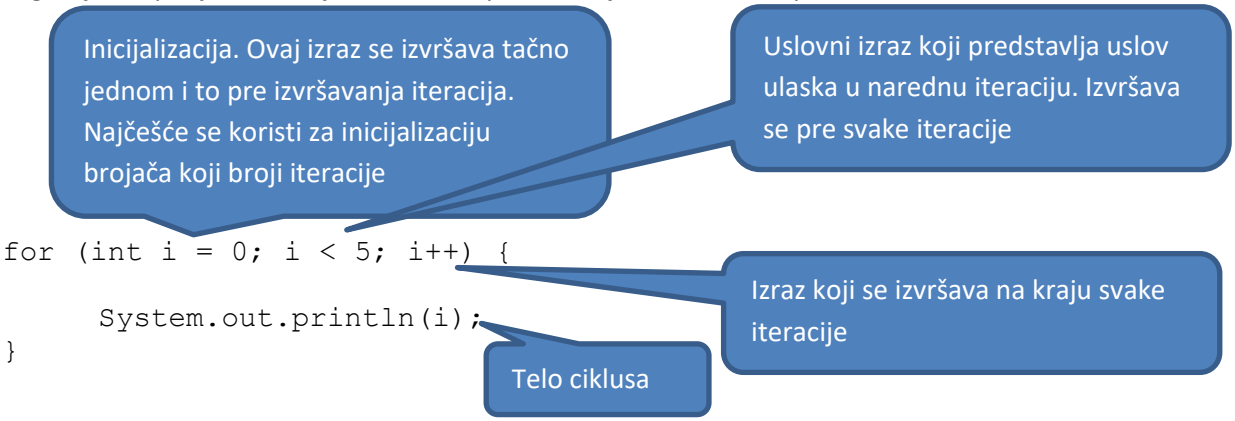
Do-while ciklus

Ako uslov već na početku nije ispunjen, while ciklus se neće nijednom izvršiti. Ako je potrebno implementirati funkcionalnost kakvu obezbeđuje while ciklus, a da se telo ciklusa izvrši bar jednom, koristi se do-while ciklus. Ovaj ciklus najpre izvrši telo ciklusa, pa onda proverava da li da pređe na narednu iteraciju. U nastavku je implementiran prethodni primer korišćenjem do-while ciklusa.

```
int broj = 20;
do {
      System.out.println("Broj je: " + broj);
      broj /= 2;
} while (broj > 1);
```

For ciklus

Za razliku od while i do-while ciklusa kod kojih broj iteracija zavisi od ponašanja programa, for ciklus se koristi kada unapred znamo koliki broj iteracija program treba da izvrši. Kao i prethodno objašnjeni tipovi ciklusa, for ciklus ima telo i zaglavlje. Telo je blok koda koji se iterativno izvršava. Zaglavlje for petlje se sastoji od tri dela i prikazano je kroz naredni primer.



Ciklus iz prethodnog primera će izvršiti 5 iteracija. Brojač i inicijalno ima vrednost 0, a na kraju svake iteracije se uveća za 1. Uslov ulaska u narednu iteraciju je da je brojač manji od 5.

Break i continue naredbe

Kada je reč o ciklusima, moguće je napustiti telo ciklusa naredbom break. Takođe, moguće je napustiti trenutnu iteraciju i preći na sledeću iteraciju naredbom continue. Korišćenje ovih naredbi je ilustrovano sledećim primerom.

```
for (int i = 0; i < 10; i++) {
      if (i == 7) {
            break;
      }
      else if (i == 2) {
            continue;
      }
      System.out.println(i);
}
```

Program iz primera ispisuje vrednosti 0, 1, 3, 4, 5, 6. Kada brojač dođe do 7, petlja se prekida, a kada brojač ima vrednost 2, prelazi se na sledeću iteraciju i ova vrednost se ne ispisuje.

2. Nizovi

Radi lakšeg upravljanja podacima, podatke možemo grupisati u različite strukture podataka. Struktura podataka predstavlja način organizacije podataka. Jedna od osnovnih struktura podataka je niz. Niz predstavlja uređenu kolekciju elemenata. Elementi formiraju sekvencijalnu strukturu, što znači da između elemenata postoji redosled. U nastavku je primer definisanja niza celobrojnih elemenata u programskom jeziku Java.

```
int[] a = {6, 3, 7, 6};
```

Možemo primetiti da simbol [ ] u definisanju tipa podatka ukazuje na to da je reč o nizu. Ključna reč int označava tip podataka koji se skladište u nizu, a to su u ovom slučaju celi brojevi. U Javi, svi elementi u nizu moraju biti istog tipa. Zato kažemo da je niz u Javi homogena kolekcija. U vitičastim zagradama je naveden sadržaj niza.

Operator [ ] se koristi za pristup elementu niza na zadatoj poziciji. Ova operacija se naziva indeksiranje elementa. Operatoru se prosleđuje indeks (pozicija) elementa kojem pristupamo. Važno je naglasiti da indeksi kreću od nula (prvi element ima indeks 0) i da svaki sledeći element ima indeks uvećan za 1. Dat je primer preuzimanja elementa niza.

```
System.out.println("Element sa indeksom 2 je: " + a[2]);
```

Na sličan način se vrši i postavljanje vrednosti u element niza. Ovo je prikazano u narednom primeru.

```
a[2] = 9;
```

Moguće je indeksirati samo element sa neke od postojećih pozicija u nizu. Tako bi, na primer pokušaj indeksiranja elementa na poziciji 12 u nizu iz prethodnog primera izazvao grešku pri izvršavanju programa.

For petlja se često koristi za iteriranje kroz elemente niza. Tako na primer ispis svih elemenata niza se može obaviti korišćenjem for petlje kao u narednom primeru.

```
int[] a = {2, 7, 4, 8, -3};
for (int i = 0; i < 5; i++) {
      System.out.println(a[i]);
}
```

3. Višedimenzionalni nizovi

U prethodnim primerima prikazani su jednodimenzionalni nizovi. Java omogućuje i kreiranje višedimenzonalnih nizova. Dvodimenzionalni niz je niz čiji je svaki element niz. Slično, trodimenzionalni niz je niz čiji je svaki element niz, čiji je svaki element niz. I tako redom možemo dobiti N-dimenzionalne nizove. Dat je primer definisanja dvodimenzionalnog niza celih brojeva.

```
int[][] brojevi = {
      {1, 3, 5},
      {2, 7, 9},
      {6, 6, 4},
      {8, 1, 3}
};
```

Kod ovako definisanog niza, indeksiranje se vrši po dve dimenzije. Tako sledeći primer ispisuje vrednost 9. Izraz brojevi[1] indeksira element na indeksu 1 u nizu brojevi, a to je niz {2, 7, 9}. Izraz brojevi[1][2] u nizu {2, 7, 9} indeksira element na indeksu 2, što je element sa vrednošću 9.

```
System.out.println(brojevi[1][2]);
```

Slično možemo u element niza postaviti vrednost. Dat je primer postavljanja vrednosti 17 u element iz prethodnog primera.

```
brojevi[1][2] = 17;
```

I kroz višedimenzionalne nizove možemo prolaziti korišćenjem for petlje. Ovog puta je potrebno imati više ugnježdenih for petlji. Dat je primer ispisa svih elemenata niza iz prethodnog primera.

```
for (int i = 0; i < 4; i++) {
      for (int j = 0; j < 3; j++) {
            System.out.println(brojevi[i][j]);
      }
}
```