

Nizovi

Promenljive predstavljaju lokacije u okviru radne memorije računara u koje je moguće smestiti neku vrednost (slovo, broj itd.). U Javi, one se deklarishu navođenjem tipa promenljive (int, boolean, double...) pa onda i naziva promenljive. Za svaku vrednost koju je potrebno negde uskladištiti, dovoljno je definisati po jednu promenljivu. Ali, šta se dešava kada je potrebno uskladištiti više vrednosti - na primer 50 celih brojeva. Jedno rešenje je da se deklarise 50 celobrojnih promenljivih i da se svakoj od njih dodeli po jedna vrednost. Ovo rešenje, međutim, nije praktično (šta ako se radi o 1000 celih brojeva) i ne može se primeniti ako se unapred ne zna potreban broj promenljivih.

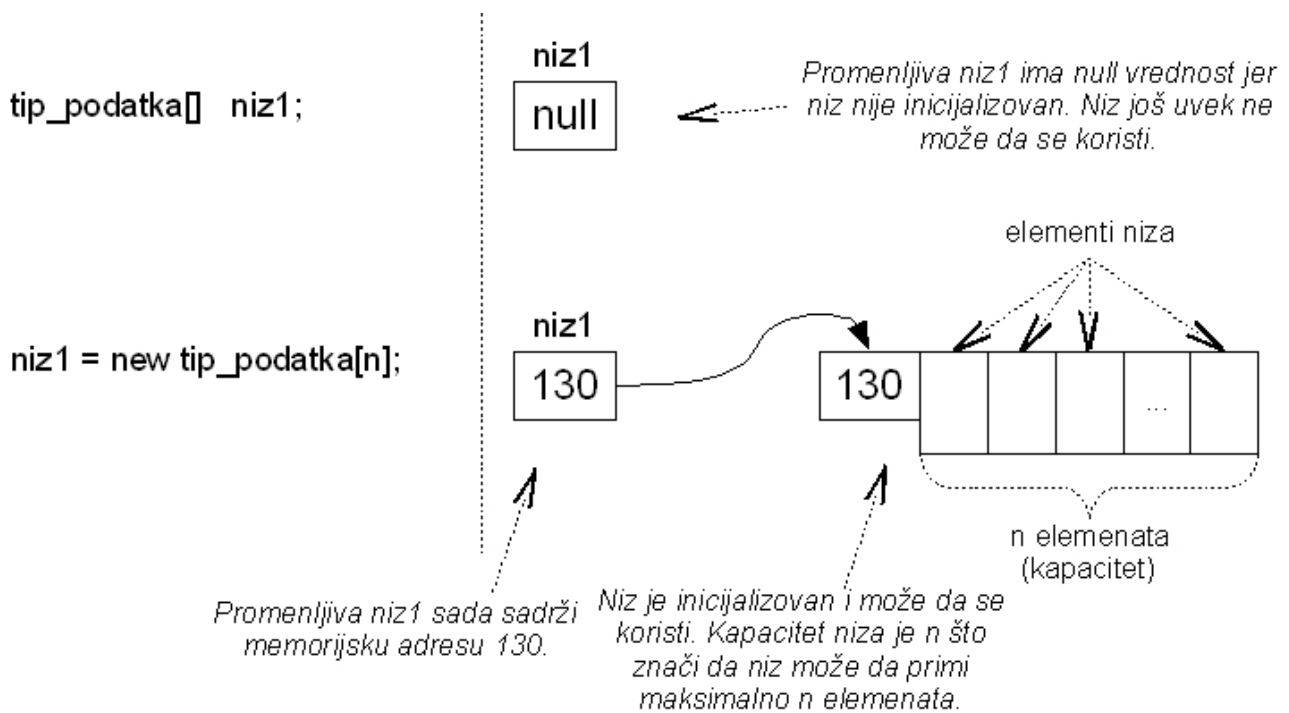
Pravo rešenje za ovakve situacije predstavljaju **nizovi**. Nizovi su promenljive koje **mogu da uskladište više vrednosti odjednom**. Deklaracija niza se vrši na sledeći način:

```
tip_podatka[] nazivPromenljive;
```

Deklaracija je skoro identična kao za običnu promenljivu, jedini dodatak predstavljaju uglaste zagrade koje se navode posle tipa podatka. Kao što je rečeno, nizovi mogu da uskladište više vrednosti odjednom. Ove vrednosti se nazivaju **elementi niza**. Svi elementi jednog niza su **uvek istog tipa** (npr. niz celih brojeva, niz realnih brojeva...). Da bi niz mogao da se koristi on **prvo mora da se inicijalizuje**. Inicijalizacija se vrši na sledeći način:

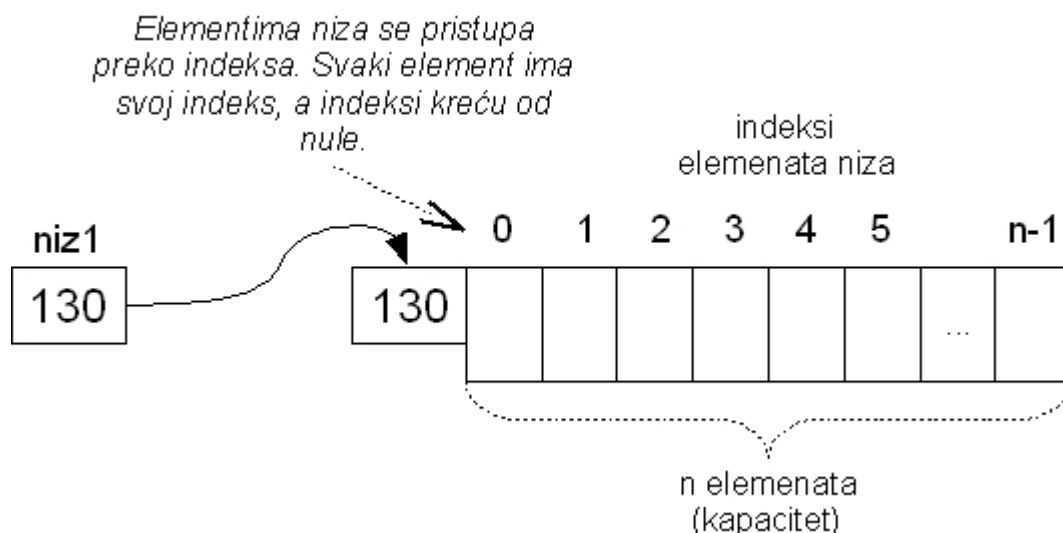
```
nazivPromenljive = new tip_podatka[ceo_broj];
```

Šta zapravo inicijalizacija radi? Promenljiva koja predstavlja niz u stvari nije sam niz već **pokazivač na niz**. Inicijalizacijom se alocira memorija za niz i tek tada se niz može koristiti. Broj koji se nalazi u zagradi mora da bude neki nenegativan ceo broj. On predstavlja **kapacitet niza** tj. koliko će elemenata niz moći maksimalno da sadrži. Na primer, niz celih brojeva kapaciteta 100 može da primi 100 celih brojeva. Jednom kada se odredi, **kapacitet niza ostaje fiksiran** i ne može se smanjiti niti povećati (Slika 1). Niz se može ponovo inicijalizovati (ako je potrebno da se poveća ili smanji kapacitet), ali se tada brišu elementi niza.



Slika 1: Inicijalizacija niza

Promenljiva koja predstavlja niz ima samo jedan naziv, a odnosi se na više elemenata, pa se postavlja pitanje kako se pristupa pojedinačnim elementima niza. Svaki element niza ima svoj **indeks**. Indeks elementa je ceo broj koji predstavlja redni broj elementa u nizu. **Indeksi elemenata počinju od nule** a ne od jedan. Tako je, na primer, indeks prvog elementa niza kapaciteta 10 uvek nula a poslednjeg 9 (Slika 2).



Slika 2: Indeksi elemenata niza

Pozivanje pojedinačnih elemenata niza uz pomoć indeksa se vrši na sledeći način:

`nazivPromenljive[indeks]`

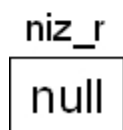
Ako je potrebno **dobiti vrednost kapaciteta niza** (npr. da bi se videlo da li je niz dovoljno dugačak) to se radi na sledeći način:

`nazivPromenljive.length`

Primer 1

Neka je potrebno deklarirati niz realnih brojeva. Odgovarajuća komanda i njen efekat su dati na slici (Slika 3).

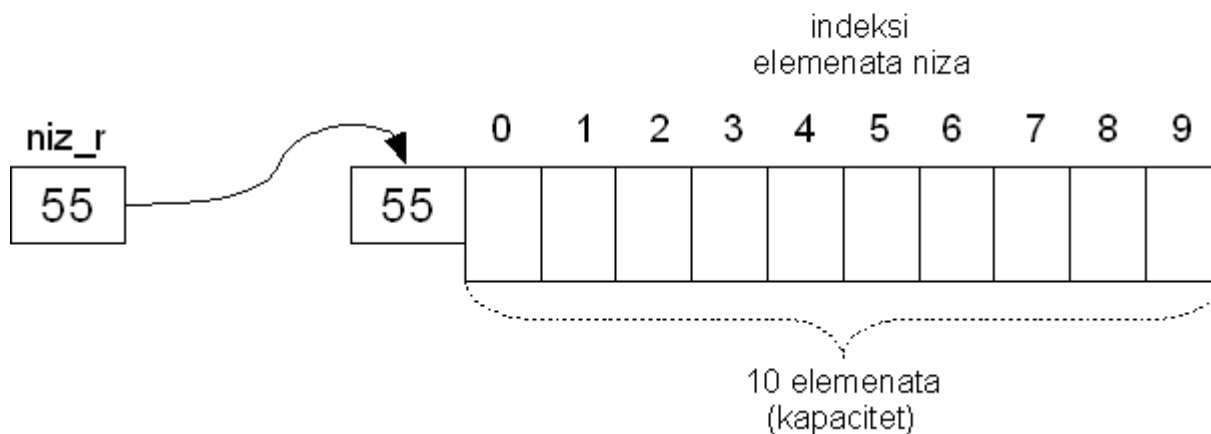
```
double [] niz_r;
```



Slika 3: Deklaracija niza realnih brojeva

Zatim, neka je potrebno inicijalizovati niz tako da mu maksimalni kapacitet bude 10 elemenata (Slika 4).

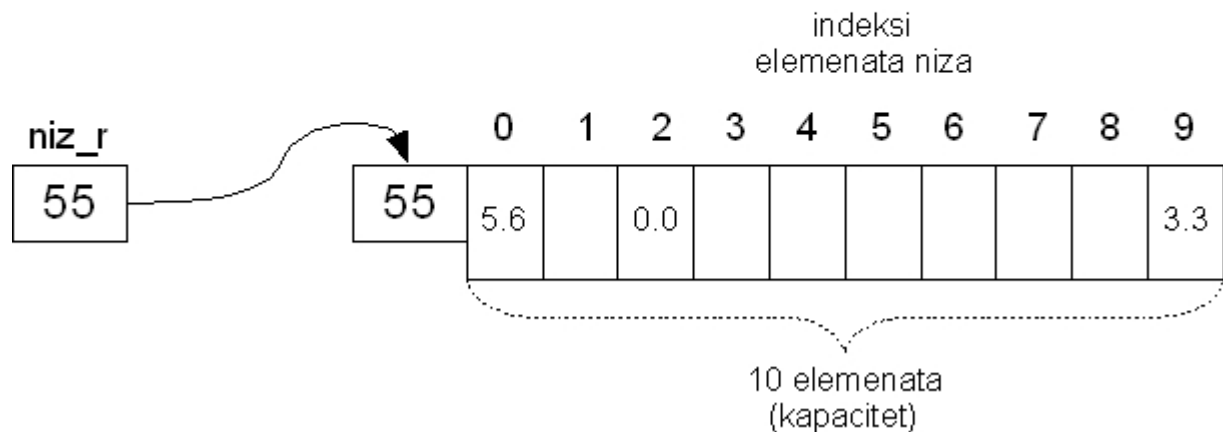
```
niz_r = new double [10];
```



Slika 4: Inicijalizacija niza realnih brojeva na kapacitet 10

Na kraju, neka je potrebno prvom elementu niza dodeliti vrednost 5.6, trećem 0, a poslednjem 3.3 (Slika 5).

```
niz_r [0] = 5.6;  
niz_r [2] = 0;  
niz_r [9] = 3.3;
```



Slika 5: Dodeljivanje vrednosti nekim elementima niza

U praksi, nizovi se često koriste u kombinaciji sa FOR petljom. Najčešće se brojač FOR petlje koristi da simulira indeks elemenata niza. U tom slučaju, postavlja se da brojač petlje ima početnu vrednost nula (jer indeksi niza kreću od nule), a petlja se izvršava sve dok brojač ne stigne do maksimalne vrednosti indeksa.

Primer 2

Napraviti klasu **MesecniProfiti** koja ima:

- Atribut *profiti* koji predstavlja niz od 12 realnih brojeva. Svaki element niza predstavlja profit za određeni mesec (januar, februar, ... , decembar).
- Metodu koja kao ulazne parametre prima realan broj koji predstavlja profit i ceo broj koji predstavlja redni broj meseca na koji se taj profit odnosi (1 - januar, 2 - februar, ... , 12 - decembar). Metoda unosi odgovarajuću vrednost profita za taj mesec u niz.
- Metodu koja na ekranu ispisuje profit za svaki mesec.

Napraviti klasu **TestMesecniProfiti** koja kreira jedan objekat klase **MesecniProfiti**, unosi profit za februar koji iznosi 122.33 i ispisuje sve mesečne profite na ekranu.

```
class MesecniProfiti {  
  
    double[] profiti = new double[12];  
  
    void unesiProfit(double profit, int mesec) {  
        //Ovde stoji [mesec-1] a ne mesec jer  
        //indeksi niza kreću od nule pa je indeks  
        //za januar 0 iako je to prvi mesec (1) a  
        //indeks za decembar 11 iako je to dvanaesti  
        //mesec.  
        profiti[mesec-1] = profit;  
    }  
  
    void ispisi() {
```

```

        for(int i=0; i<profiti.length;i++)
            System.out.println(profiti[i]);
    }

}

class TestMesecniProfiti {

    public static void main(String[] args) {

        MesecniProfiti mp = new MesecniProfiti();

        mp.unesiProfit(122.33, 2);

        mp.ispisi();

    }

}

```

Atribut profiti predstavlja niz realnih brojeva koji se inicijalizuje na odgovarajući kapacitet - 12.

Metoda “unesiProfit” kao ulazni parametar prima iznos profita i redni broj meseca na koji se profit odnosi.

Metoda “ispisi” koristi FOR petlju da ispiše sve elemente niza. Brojač petlje “i” kao početnu vrednost dobija 0 a ne 1 jer indeksi elemenata niza počinju od nule. Brojač će se u svakoj iteraciji uvećavati za jedan, a petlja će se izvršavati sve dok je brojač manji od kapaciteta. To svakako ima smisla jer će u poslednjoj iteraciji vrednost brojača biti “kapacitet-1” (“profiti.length-1”) a to je upravo indeks poslednjeg elementa niza. U svakoj iteraciji petlje se ispisuje po jedan element niza i to onaj element niza koji ima indeks koji je jednak vrednosti brojača petlje za tu iteraciju (“profiti[i]”). Tako, u prvoj iteraciji će “i” imati vrednost 0 i ispisaće se “profiti[0]” a to je upravo prvi element niza. U drugoj iteraciji “i” će imati vrednost 1 pa će se ispisati “profiti[1]” (drugi element niza) i tako dalje. U poslednjoj iteraciji “i” će imati vrednost “kapacitet-1” pa će se ispisati poslednji element niza.

Niz koji je dat u prethodnom primeru se uvek koristi do maksimalnog kapaciteta. Šta to znači? Niz ima dvanaest elemenata i uvek se koriste svi elementi (svaki predstavlja iznos profita za neki mesec u toku godine). Međutim, moguće su i situacije u kojima se ne koristi pun kapacitet niza. Primer ovakve situacije je niz ocena na ispitnom roku iz nekog predmeta - zna se da je ispit prijavio određeni broj studenata ali realno uvek na ispit izađe manje studenata. U tim slučajevima, pogodno je uvesti i još jednu pomoćnu promenljivu koja će da ima funkciju brojača elemenata u nizu. Ovaj brojač će da sadrži podatak o tome koliko stvarno ima elemenata u nizu, a ne koji je maksimalni kapacitet niza. Naravno, da bi brojač bio ažuran, potrebno je uvećati ga za jedan svaki put kada se doda neki element u niz i umanjiti ga za jedan svaki put kada se izbaci neki element iz niza.

Primer 3

*Napraviti klasu **OceneNaIspitnomRoku** koja ima:*

- *Atribut ocene koji predstavlja ocene studenata na ispitnom roku. Zna se da ispit može maksimalno da polaže 100 studenata.*
- *Atribut brojac koji predstavlja trenutni broj elemenata u nizu. Početna vrednost je 0.*
- *Metodu koja kao ulazni argument dobija ocenu na ispitu i unosi je u niz i to na prvo*

slobodno mesto.

- Metodu koja na ekranu ispisuje sve ocene na ispitnom roku.

Napraviti klasu **TestOceneNaIspitnomRoku** koja kreira jedan objekat klase **OceneNaIspitnomRoku**, unosi u njega ocene 5, 10, 10, 7 i 8 i ispisuje sve ocene na ekranu.

```
class OceneNaIspitnomRoku {

    int[] ocene = new int[100];
    int brojac = 0;

    void unesiOcenu(int ocena){
        //Promenljiva brojac predstavlja
        //trenutni broj elemenata u nizu
        //ali i predstavlja indeks prvog
        //slobodnog elementa niza.
        ocene[brojac] = ocena;

        //Novi element je dodan u niz
        //pa je potrebno uvecati brojac
        //za jedan.
        brojac++;
    }

    void ispisi(){
        //Brojac FOR petlje ide od nule
        //do trenutnog broja elemenata niza
        //(brojac) a ne do kapaciteta (length)
        //jer niz nije pun.
        for(int i=0;i<brojac;i++)
            System.out.println(ocene[i]);
    }

}

class TestOceneNaIspitnomRoku {

    public static void main(String[] args) {

        OceneNaIspitnomRoku oir =
            new OceneNaIspitnomRoku();

        oir.unesiOcenu(5);
        oir.unesiOcenu(10);
        oir.unesiOcenu(10);
        oir.unesiOcenu(7);
        oir.unesiOcenu(8);

        oir.ispisi();
    }

}
```

Zadaci

Zadatak 1

Napraviti klasu **NizBrojeva** koja ima:

- Atribut koji predstavlja niz celih brojeva. Ovaj niz uvek ima maksimalno 10 elemenata.
- Atribut koji predstavlja brojač elemenata niza. Brojač na početku ima vrednost nula jer je niz prazan.
- Metodu za dodavanje elemenata u niz. Ova metoda prima kao ulazni argument broj koji je potrebno dodati u niz. Broj se dodaje na prvo slobodno mesto u nizu. Naravno, potrebno je u okviru metode i uvećati vrednost brojača za jedan.
- Metodu koja ispisuje na ekranu prvi element niza.
- Metodu koja ispisuje na ekranu deseti element niza.
- Metodu koja ispisuje na ekranu jedan od elemenata niza čiji se indeks unosi u obliku argumenta metode.
- Metodu za ispisivanje svih elemenata niza.
- Metodu koja računa i vraća zbir elemenata niza.
- Metodu koja računa i vraća proizvod elemenata niza.
- Metodu koja pronalazi i vraća minimalni element niza.
- Metodu koja pronalazi i vraća maksimalni element niza.
- Metodu koja proverava da li se određeni broj nalazi u nizu. Ako se nalazi, metoda vraća TRUE, u suprotnom vraća FALSE. Broj koji se traži se unosi kao ulazni argument.

Napraviti i klasu **ProveraNizaBrojeva** koja kreira dva objekta klase NizBrojeva i u prvi unosi elemente 4 i 7, a u drugi 3, 5 i 10. Ispisati na ekranu zbir elemenata prvog niza i minimalni element drugog niza.

Rešenje:

```
class NizBrojeva {  
  
    int[] niz = new int[10];  
    int brojac = 0;  
  
    void dodajElement(int a) {  
        niz[brojac]=a;  
        brojac++;  
    }  
  
    void ispisiPrvi() {  
        System.out.println (niz[0]);  
    }  
  
    void ispisiDeseti() {  
        System.out.println (niz[9]);  
    }  
  
    void ispisiElement(int indeks) {  
        System.out.println(niz[indeks]);  
    }  
  
    void ispisi() {  
        for (int i=0; i<brojac; i++) System.out.println(niz[i]);  
    }  
  
    int zbirNiza() {  
        int zbir = 0;  
        for(int i=0; i<brojac; i++) zbir=zbir+niz[i];  
        return zbir;  
    }  
  
    int proizvodNiza() {  
        int proizvod = 1;  
        for (int i=0; i< brojac; i++) proizvod = proizvod*niz[i];  
    }  
}
```

```

        return proizvod;
    }

    int minimalniElement() {
        int min=niz[0];
        for (int i=0; i<brojac; i++) if (niz[i]<min) min=niz[i];
        return min;
    }

    int maksimalniElement() {
        int max=niz[0];
        for (int i=0; i<brojac; i++) if (niz[i]>max) max=niz[i];
        return max;
    }

    boolean provera (int x){
        for (int i=0; i<brojac; i++) if (niz[i]==x) return true;

        return false;
    }
}

class ProveraNizaBrojeva {

    public static void main (String[] args){

        NizBrojeva n1 = new NizBrojeva();
        NizBrojeva n2 = new NizBrojeva();

        n1.dodajElement(4);
        n1.dodajElement(7);

        n2.dodajElement(3);
        n2.dodajElement(5);
        n2.dodajElement(10);

        System.out.println("Zbir elemenata prvog niza je "+
            n1.zbirNiza());
        System.out.println("Minimalni element drugog niza je "+
            n2.minimalniElement());
    }
}

```

Zadatak 2

Napraviti klasu **NizCelihBrojeva** koja ima:

- Atribut koji predstavlja niz celih brojeva.
- Atribut koji predstavlja brojač elemenata niza. Brojač na početku ima vrednost nula jer je niz prazan.
- Konstruktor u kome se niz kreira tako da mu maksimalni kapacitet bude jednak vrednosti koja se prosleđuje konstruktoru u obliku ulaznog argumenta.
- Metodu za dodavanje elemenata u niz. Ova metoda prima kao ulazni argument broj koji je potrebno dodati u niz. Pre nego što se izvrši dodavanje, proverava se da li je kapacitet prekoračen (da li je brojač dostigao maksimalni kapacitet niza). Ako je kapacitet prekoračen, ispisuje se poruka o grešci. Ako nije, broj se dodaje na prvo slobodno mesto u nizu. Ako je element dodat, potrebno je u okviru metode uvećati vrednost brojača za jedan.
- Metodu koja sabira vrednosti prvog i poslednjeg elementa niza i vraća rezultat. Ako je niz prazan, ispisuje se poruka o tome i metoda vraća nulu.
- Metodu koja sabira samo pozitivne elemente niza i vraća njihovu vrednost.
- Metodu koja množi samo negativne elemente niza i vraća njihov proizvod.
- Metodu koja vraća broj ponavljanja nekog broja u nizu. Broj se prosleđuje metodi u vidu ulaznog argumenta.

- Metoda koja ispisuje na ekranu članove niza koji su parni brojevi.
- Metoda koja ispisuje članove niza koji su deljivi sa 5.
- Metoda koja ispisuje članove niza u obrnutom redosledu.

Potrebno je napisati i klasu **ProveraNizaCelihBrojeva**. U okviru nje je potrebno kreirati tri objekta klase NizCelihBrojeva: prvi kapaciteta 3 člana, drugi kapaciteta 5 članova, a treći kapaciteta 10 članova. U prvi niz je potrebno ubaciti elemente 1, -1 i 3 u drugi elemente 34 i 45 a u treći elemente 56, 67 i – 89. Ispisati na ekranu zbir prvog i poslednjeg elementa prvog niza. Drugi i treći niz ispisati u obrnutom redosledu.

Rešenje:

```
class NizCelihBrojeva {

    int[] celiBrojevi;
    int brojac;

    public NizCelihBrojeva (int kapacitet){
        celiBrojevi = new int[kapacitet];
        brojac = 0;
    }

    void dodajElement(int a){
        if (brojac < celiBrojevi.length){
            celiBrojevi[brojac] = a;
            brojac++;
        }
        else System.out.println("Niz je vec popunjen do maksimalnog kapaciteta");
    }

    int saberiPrviIPoslednji(){
        if (brojac>0) {
            int rezultat = celiBrojevi[0]+celiBrojevi[brojac-1];
            return rezultat;
        }
        else {
            System.out.println("Niz je prazan");
            return 0;
        }
    }

    int saberiPozitivne (){
        int zbir = 0;
        for (int i=0; i<brojac; i++)
            if (celiBrojevi[i]>0) zbir=zbir+celiBrojevi[i];
        return zbir;
    }

    int pomnoziNegativne (){
        int proizvod = 1;
        for (int i=0; i<brojac; i++)
            if (celiBrojevi[i]<0) proizvod=proizvod*celiBrojevi[i];
        return proizvod;
    }

    int brojPonavljanja (int b){
        int ponavljanje = 0;
        for (int i=0; i<brojac; i++) if (celiBrojevi[i]==b) ponavljanje++;
        return ponavljanje;
    }

    void ispisiParne(){
        for (int i=0; i<brojac; i++)
            if ((celiBrojevi[i]%2)==0)
```

```

        System.out.println(celiBrojevi[i]);
    }

    void ispisiDeljiveSaPet() {
        for (int i=0; i<brojac; i++)
            if ((celiBrojevi[i]%5)==0)
                System.out.println(celiBrojevi[i]);
    }

    void ispisiObrnuto() {
        for(int i=brojac-1; i>=0; i--) System.out.println(celiBrojevi[i]);
    }
}

class ProveraNizaCelihBrojeva {

    public static void main (String[] args){

        NizCelihBrojeva n1 = new NizCelihBrojeva(3);
        NizCelihBrojeva n2 = new NizCelihBrojeva(5);
        NizCelihBrojeva n3 = new NizCelihBrojeva(10);

        n1.dodajElement(1);
        n1.dodajElement(-1);
        n1.dodajElement(3);

        n2.dodajElement(34);
        n2.dodajElement(45);

        n3.dodajElement(56);
        n3.dodajElement(67);
        n3.dodajElement(-89);

        System.out.println("Zbir prvog i poslednjeg elementa prvog niza je "
            +n1.saberiPrviIPoslednji());
        n2.ispisiObrnuto();
        n3.ispisiObrnuto();
    }
}

```

Zadatak 3

Potrebno je napraviti klasu **NizCena** koja predstavlja niz cena raznih proizvoda i ima:

- Atribut koji predstavlja niz cena proizvoda (primer cene: 100.5 dinara). Maksimalni kapacitet niza je uvek 100 elemenata.
- Atribut koji predstavlja brojač elemenata niza. Brojač na početku ima vrednost nula jer je niz prazan.
- Metodu za dodavanje nove cene u niz. Dodavanje se vrši samo ako je nova cena veća od nule i ako u nizu ima mesta (brojač je manji od maksimalnog kapaciteta). U suprotnom, potrebno je ispisati poruku o grešci. Ako se ubacivanje izvrši, potrebno je brojač uvećati za jedan.
- Metodu koja izračunava i vraća prosečnu cenu proizvoda. Ukoliko je niz prazan, ispisuje se poruka da je niz prazan i vraća se nula.
- Metodu koja vraća najnižu cenu proizvoda. Ukoliko je niz prazan, ispisuje se poruka da je niz prazan i vraća se nula.
- Metodu koja vraća najvišu cenu proizvoda. Ukoliko je niz prazan, ispisuje se poruka da je niz prazan i vraća se nula.
- Metodu koja vraća razliku između najniže i najviše cene. Ukoliko je niz prazan, ispisuje se poruka da je niz prazan i vraća se nula.
- Metodu koja ispisuje samo one cene proizvoda koje su veće od neke zadate vrednosti. Ta vrednost se unosi u metodu kao ulazni argument. Ukoliko je niz prazan, ispisuje se poruka o tome.

Napisati klasu **TestNizCena** koja kreira jedan objekat klase NizCena. U ovaj niz cena je potrebno uneti sledeće cene: 123.0 , 234.55, 301.0 i 5000.0. Kada se to uradi, potrebno je na ekranu ispisati one cene koje su veće od 300.0 dinara i razliku između najniže i najviše cene.

Rešenje:

```
class NizCena {

    double[] cene = new double[100];
    int brojCena = 0;

    void dodajCenu(double cena) {
        if ((cena>0) && (brojCena<100)) {
            cene[brojCena]=cena;
            brojCena++;
        }
        else System.out.println("Greska");
    }

    double prosecnaCena() {
        if (brojCena == 0) {
            System.out.println("Niz je prazan");
            return 0;
        }
        else {
            double suma=0;
            double prosek;
            for (int i=0; i<brojCena;i++) suma=suma+cene[i];
            prosek=suma/brojCena;
            return prosek;
        }
    }

    double najnizaCena() {
        if (brojCena == 0) {
            System.out.println("Niz je prazan");
            return 0;
        }
        else {
            double min = cene[0];
            for (int i=0; i<brojCena;i++)
                if (cene[i]<min) min=cene[i];
            return min;
        }
    }

    double najvisaCena() {
        if (brojCena == 0) {
            System.out.println("Niz je prazan");
            return 0;
        }
        else {
            double max = cene[0];
            for (int i=0; i<brojCena;i++)
                if (cene[i]>max) max=cene[i];
            return max;
        }
    }

    double razlikaMaxMin() {
        if (brojCena == 0) {
            System.out.println("Niz je prazan");
            return 0;
        }
    }
}
```

```

        else {
            double razlika = najvisaCena()-najnizaCena();
            return razlika;
        }
    }

    void ispisiCeneVeceOd (double cena){
        if (brojac == 0) System.out.println("Niz je prazan");
        else
            for(int i=0;i<brojac;i++)
                if(cene[i]>cena)
                    System.out.println(cene[i]);
    }
}

class TestNizCena {

    public static void main (String[] args){

        NizCena nc = new NizCena();

        nc.dodajCenu(123.0);
        nc.dodajCenu(234.55);
        nc.dodajCenu(301.0);
        nc.dodajCenu(5000.0);

        nc.ispisiCeneVeceOd(300.0);

        System.out.println("Razlika najvise i najnize cene je "+
            nc.razlikaMaxMin()+" dinara");
    }
}

```

Zadatak 4

Napisati klasu **DNKLANAC** koja predstavlja deo DNK lanca čoveka i ima:

- Atribut koji predstavlja niz karika DNK lanca. Svaka karika DNK lanca može da ima samo jednu od vrednosti: 'A', 'C', 'G' ili 'T'.
- Atribut koji predstavlja trenutni broj karika u lancu.
- Konstruktor u kome se lanac kreira tako da mu maksimalni broj karika bude jednak vrednosti koja se prosleđuje konstruktoru u obliku ulaznog argumenta. Ako se desi da je ulazni argument manji ili jednak nula, maksimalni kapacitet treba podesiti na 256 karika.
- Metodu za dodavanje karika u DNK lanac. Nova karika se daje u vidu ulaznog argumenta. Dodavanje se vrši samo ako nova karika ima vrednost 'A', 'C', 'G' ili 'T' i ako u lancu ima mesta (broj karika je manji od maksimalnog kapaciteta). U suprotnom, potrebno je ispisati poruku o grešci.
- Metodu koja prebrojava i vraća koliko ima 'A' karika u lancu.
- Metodu koja vraća trenutnu dužinu lanca (broj karika).
- Metodu koja vraća maksimalni kapacitet lanca.
- Metodu koja vraća broj nepopunjenih mesta u lancu.
- Metodu koja ispisuje karike DNK lanca u jednom redu.
- Metodu koja ispisuje karike DNK lanca u jednom redu ali u obrnutom redosledu.

Potrebno je napraviti klasu **TestDNKLANAC** koja kreira jedan DNK lanac maksimalne dužine 8 karika i unosi u njega elemente ACCGTTTT. Potrebno je ispisati ovaj DNK lanac u regularnom i obrnutom redosledu.

Rešenje:

```

class DNKLANAC {

    char[] karike;

```

```

int brojKarika;

public DNKLanac (int max_duzina){
    if (max_duzina>0){
        karike = new char[max_duzina];
        brojKarika = 0;
    }
    else {
        karike = new char[256];
        brojKarika = 0;
    }
}

void dodajKariku (char karika){
    if ( ((karika=='A')||(karika=='C')||
        (karika=='G')||(karika=='T')) &&
        (brojKarika<karike.length)) {
        karike[brojKarika]=karika;
        brojKarika++;
    }
    else System.out.println("Greska, karika nije dodata!");
}

int prebrojAKarika (){
    int broj = 0;
    for (int i=0; i<brojKarika; i++)
        if (karike[i]=='A') broj++;
    return broj;
}

int trenutnaDuzinaLanca(){
    return brojKarika;
}

int maksimalniKapacitetLanca(){
    return karike.length;
}

int preostaliKapacitet(){
    return (karike.length-brojKarika);
}

void ispisiDNKLanac (){
    //Kada je potrebno ispisivati vise stvari u jednom redu
    //koristi se System.out.print a ne System.out.println naredba
    for (int i=0;i<brojKarika; i++)
        System.out.print(karike[i]);
}

void ispisiDNKLanacObrnuto (){
    //Kada je potrebno ispisivati vise stvari u jednom redu
    //koristi se System.out.print a ne System.out.println naredba
    for (int i=brojKarika-1;i>=0; i--)
        System.out.print(karike[i]);
}

}

class TestDNKLanac {

    public static void main (String[] args){

        DNKLanac dnk = new DNKLanac(8);

```

```

        dnk.dodajKariku('A');
        dnk.dodajKariku('C');
        dnk.dodajKariku('C');
        dnk.dodajKariku('G');
        dnk.dodajKariku('T');
        dnk.dodajKariku('T');
        dnk.dodajKariku('T');
        dnk.dodajKariku('T');

        dnk.ispisiDNKLanac();

        //Ova komanda je dodata samo zato da bi se obrnuti DNK
        //lanac ispisao u sledecem redu. Probajte da uklonite
        //ovu komandu i startujte program da vidite sta ce da se desi.
        System.out.println();

        dnk.ispisiDNKLanacObrnuto();

    }

}

```

Zadatak 5

Potrebno je napraviti klasu **Autobus** koja ima:

- Atribut koji predstavlja niz sedišta u autobusu. Svako sedišta može da bude slobodno ili zauzeto. Ako je slobodno, vrednost odgovarajućeg elementa niza je TRUE, a ako je zauzeto, onda je FALSE. Autobus ima tačno 50 sedišta.
- Konstruktor koji postavlja vrednost svih sedišta iz niza na slobodna (TRUE).
- Metodu za uvođenje putnika u autobus. Ova metoda prima kao ulazni argument broj sedišta na koje bi trebalo uvesti putnika (brojevi sedišta su od 0 do 49). Ako je sedišta slobodno (TRUE), sedišta postaje zauzeto (FALSE) a ako je već bilo zauzeto, ispisuje se poruka o grešci.
- Metodu koja proverava da li ima slobodnih mesta u autobusu. Metoda vraća TRUE ako ima slobodnih mesta, u suprotnom FALSE.
- Metodu koja vraća broj slobodnih mesta u autobusu.
- Metodu koja vraća broja zauzetih mesta u autobusu.
- Metodu koja ispisuje status svakog sedišta iz autobusa u obliku “Sedište broj ## je slobodno” ili “Sedište broj ## je zauzeto”.

Potrebno je napraviti klasu **TestAutobus** koja kreira jedan objekat klase **Autobus** i uvodi u njega tri putnika: na prvo, dvadeseto i poslednje mesto u autobusu. Posle toga, potrebno je ispisati status svih mesta u autobusu.

Rešenje:

```

class Autobus {

    boolean[] sedista = new boolean [50];

    public Autobus () {
        for (int i=0;i<50;i++) sedista[i]=true;
    }

    void uvediPutnika (int broj_sedista){
        if (sedista[broj_sedista]==true) sedista[broj_sedista]=false;
        else System.out.println("Sediste broj "+broj_sedista+
                                " je vec zauzeto");
    }

    boolean imaSlobodnihMesta() {
        for (int i=0; i<50; i++)
            if (sedista[i]==true) return true;
    }
}

```

```

        return false;
    }

    int brojSlobodnihMesta() {
        int broj_s=0;
        for (int i=0; i<50; i++)
            if (sedista[i]==true) broj_s++;
        return broj_s;
    }

    int brojZauzetihMesta() {
        int broj_z=0;
        for (int i=0; i<50; i++)
            if (sedista[i]==true) broj_z++;
        return broj_z;
    }

    void ispisStatusaAutobusa() {
        for (int i=0; i<50; i++)
            if (sedista[i]==true)
                System.out.println("Sediste broj "+i+" je slobodno");
            else
                System.out.println("Sediste broj "+i+" je zauzeto");
    }
}

class TestAutobus {

    public static void main(String[] args) {

        Autobus a = new Autobus();

        a.uvediPutnika(0);
        a.uvediPutnika(19);
        a.uvediPutnika(49);

        a.ispisStatusaAutobusa();
    }
}

```

Zadatak 6

Potrebno je napraviti klasu **UsluzneFunkcije** koja ima:

- Statičku metodu koja kao ulazni parametar dobija niz celih brojeva i ispisuje njegove elemente na ekranu.
- Statičku metodu koja kao ulazni parametar dobija niz realnih brojeva i ispisuje njegove elemente na ekranu.
- Statičku metodu koja kao ulazni parametar dobija niz celih brojeva i pravi i vraća kopiju ovog niza.
- Statičku metodu koja kao ulazni parametar dobija dva niza celih brojeva i vraća treći niz koji se dobija spajanjem dva uneta niza (nadovezivanjem elemenata drugog niza posle elemenata prvog niza). Na primer, ako se unesu dva niza po 5 elemenata, metoda vraća niz od 10 elemenata koji sadrži sve elemente dva uneta niza.

Potrebno je napraviti klasu **TestUsluzneFunkcije** koja poziva metode klase **UsluzneFunkcije**.

Rešenje:

```

class UsluzneFunkcije {

    static void ispisi(int[] niz) {
        for(int i=0; i<niz.length; i++)
            System.out.println(niz[i]);
    }
}

```

```

static void ispisi(double[] niz){
    for(int i=0; i<niz.length;i++)
        System.out.println(niz[i]);
}

static int[] kopiraj(int[] niz){
    //Kapacitet kopije treba da bude isti
    //kao kapacitet originalnog niza.
    int[] kopija = new int[niz.length];

    //Kopiranje elemenata niza.
    for(int i=0; i<niz.length;i++) kopija[i] = niz[i];

    return kopija;
}

static int[] spoji(int[] niz1, int[] niz2){
    //Kapacitet treceg niza treba da bude jednak
    //zbiru kapaciteta dva ulazna niza.
    int[] niz3 = new int[niz1.length+niz2.length];

    //Kopiranje elemenata prvog niza.
    for(int i=0; i<niz1.length;i++) niz3[i] = niz1[i];

    //Kopiranje elemenata drugog niza, ali od pozicije
    //koja predstavlja kraj elemenata prvog niza
    for(int i=0; i<niz2.length;i++)
        niz3[niz1.length+i] = niz2[i];

    return niz3;
}

}

class TestUsluzneFunkcije {

    public static void main(String[] args) {

        int[] niz1 = new int[4];

        niz1[0] = 0;
        niz1[1] = 1;
        niz1[2] = 2;
        niz1[3] = 3;

        int[] niz2 = new int[2];

        niz2[0] = 4;
        niz2[1] = 5;

        int[] kopija = UsluzneFunkcije.kopiraj(niz1);
        UsluzneFunkcije.ispisi(kopija);

        int[] niz3 = UsluzneFunkcije.spoji(niz1, niz2);
        UsluzneFunkcije.ispisi(niz3);
    }

}

```


Višedimenzionalni nizovi

Višedimenzionalni podaci (npr. matrice) mogu da se predstavje u Javi korišćenjem višedimenzionalnih nizova. Višedimenzionalni niz je veoma sličan jednodimenzionalnom nizu u svakom pogledu, a njegova deklaracija se vrši na sledeći način (stavlja se onoliko uglastih zagrada koliko podaci imaju dimenzija):

```
tip_podatka[][] nazivPromenljive;
```

Ovo je deklaracija dvodimenzionalnog niza. Od svih višedimenzionalnih nizova se, u principu, najčešće koriste dvodimenzionalni nizovi i to u onim slučajevima kada je potrebno predstaviti neku matricu ili tabelu sa podacima.

I ovde je potrebna inicijalizacija niza, sa tim što se pri inicijalizaciji navodi kapacitet niza po svakoj dimenziji:

```
nazivPromenljive = new tip_podatka[ceo_broj_1][ceo_broj_2];
```

Broj u prvoj zagradi ("ceo_broj_1") predstavlja broj redova matrice, a broj u drugoj zagradi ("ceo_broj_2") broj kolona. Pristup elementima dvodimenzionalnog niza se takođe vrši preko indeksa, ali svaka dimenzija ima svoj indeks pa se, u slučaju dvodimenzionalnog niza, obavezno navode oba indeksa (prvi predstavlja red matrice a drugi kolonu):

```
nazivPromenljive[indeks_1][indeks_2]
```

U radu sa dvodimenzionalnim nizovima se najčešće koriste dve ugnježdene FOR petlje pri čemu brojač svake petlje glumi indeks jedne dimenzije niza.

Primer 4

Napraviti klasu **Matrica** koja ima:

- Atribut matrica koji predstavlja dvodimenzionalni niz celih brojeva.
- Konstruktor koji kao ulazni parametar prima broj redova i broj kolona koji matrica treba da ima i inicijalizuje atribut matrica na te kapacitete.
- Metodu koja pretvara matricu u nula matricu - matricu čiji su svi elementi jednaki nuli.
- Metodu koja matricu pretvara u jediničnu matricu tj. podešava vrednosti elemenata matrice tako da elementi koji se nalaze na glavnoj dijagonali imaju vrednost 1 a ostali 0. Metoda prvo proverava da li je matrica kvadratna (da li ima isti broj redova i kolona), pa ako nije ispisuje poruku o grešci na ekranu.
- Metodu koja na ekranu ispisuje vrednosti elemenata matrice i to tako da se u jednom redu na ekranu ispišu svi elementi koji pripadaju istom redu matrice.

Napraviti klasu **TestMatrica** koja kreira jedan objekat klase Matrica dimenzija 5x5 i poziva njegove metode.

```
class Matrica {  
  
    int[][] matrica;  
  
    Matrica(int brojRedova, int brojKolona) {  
        matrica = new int[brojRedova][brojKolona];  
    }  
}
```

```

void nulaMatrica() {
    // Broj redova se dobija pozivanjem komande
    // matrica.length dok se broj kolona dobija
    // kao duzina prvog reda tj. matrica[0].length
    for (int i = 0; i < matrica.length; i++)
        for (int j = 0; j < matrica[0].length; j++)
            matrica[i][j] = 0;
}

void jedinicznaMatrica() {
    if (matrica.length == matrica[0].length) {
        for (int i = 0; i < matrica.length; i++)
            for (int j = 0; j < matrica[0].length; j++)
                if (i == j)
                    matrica[i][j] = 1;
                else
                    matrica[i][j] = 0;
    }

    else
        System.out.println("Greska!");
}

void ispisi() {
    for (int i = 0; i < matrica.length; i++) {
        for (int j = 0; j < matrica[0].length; j++)
            System.out.print(matrica[i][j] + " ");
        System.out.println();
    }
}

}

class TestMatrica {

    public static void main(String[] args) {

        Matrica m = new Matrica(5,5);

        m.nulaMatrica();
        m.ispisi();

        m.jedinicznaMatrica();
        m.ispisi();
    }

}

```