PT&SCHE

Introduction of part-time and short cycle studies in Serbia

Funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

UNIVERZITET
METROPOLITAN
BEOGRAD

KI103 - JAVA 1: OSNOVE PROGRAMIRANJA U JAVI

Nizovi

Lekcija 07

PRIRUČNIK ZA STUDENTE

KI103 - JAVA 1: OSNOVE PROGRAMIRANJA U JAVI

Lekcija 07

NIZOVI

- Nizovi
- Poglavlje 1: Jednodimenzionalni nizovi
- Poglavlje 2: Učitavanje i obrada jednodimenzionalnog niza
- Poglavlje 3: Domaći zadaci
- Zaključak

UVOD

Jedan niz promenljivi može da sadrži reference ka velikom skupu podataka ili objekata.

U predavanju ćemo se upoznati sa:

- Pojmom niza kao strukture podataka
- Jednodimenzionalnim nizovima
- Klasom **Arrays**
- Naredbom **for-each**
- Nedostacima nizova
- Dvodimenzionalnim nizovima

U vežbi ćemo prikazati:

- Kreiranje programa za učitavanje i obradu jednodimenzionalnog niza
- Kreiranje posebne klase sa statičkim metodama za rad sa jednodimenzionalnim nizovima
- Kreiranje metoda za rad sa jednodimenzionalnim nizovima bez korišćenja petlji
- Kreiranje metoda za rad sa jednodimenzionalnim nizovima sa korišćenjem petlji
- Kreiranje programa za učitavanje i obradu dvodimenzionalnog niza
- Kreiranje metoda za rad sa stringovima bez korišćenja petlji
- Kreiranje metoda za rad sa stringovima sa korišćenjem petlji

Poglavlje 1

Jednodimenzionalni nizovi

SADRŽAJ POGLAVLJA

Kada je niz stvoren, njegova veličina je fiksirana. Niz promenljivih sa referencama se upotrebljava za pristup elementima niza korišćenjem indeksa.

U ovom poglavlju, govorićemo o

- strukturi podataka,
- nizovima čiji su elementi primitivnog tipa
- upotrebi nizova u for petljama, i o
- nizovima čiji su elementu objekti.

1.1 Strukture podataka

ŠTA SU STRUKTURE PODATAKA?

Struktura podataka je skup povezanih elemenata koji su na neki način organizovani

Osnovna jedinica za čuvanje podataka u programu je promenljiva. Međutim, **jedna promenljiva** u svakom trenutku može sadržati samo **jedan podatak**. Ukoliko u programu želimo da više srodnih podataka imamo istovremeno na raspolaganju i da ih posmatramo kao jednu celinu, onda ih možemo organizovati u formi objekta. Istovremeni rad sa velikim brojem podataka zahteva poseban način rada sa njima koji omogućava relativno lako dodavanje, uklanjanje, pretraživanje i slične operacije sa pojedinačnim podacima. Ako se ima u vidu skup, tj. **kolekcija podataka** organizovana na ovakav način, onda se govori o **strukturi podataka**.

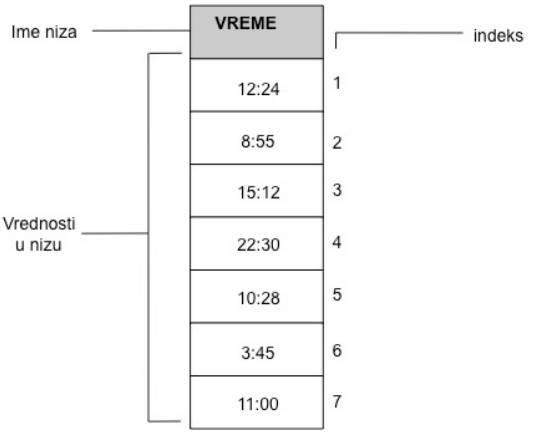
Struktura podatakaje skup povezanih elemenata koji su na neki način organizovani. Strukture podataka nam omogućavaju da sve povezane elemente posmatramo kao jedan entitet. Struktura bi se na primer, mogla sastojati od banana, jabuka i pomorandži. Obratite pažnju da su ova tri elementa organizovana po abecednom redu. Organizacija bi mogla biti i na nekom drugom principu. Najvažnije što se odavde može zaključiti je da (zbog toga što se ovde radi o strukturama podataka) celini ovih elemenata pristupamo zajednički, odnosno to je, u ovom slučaju – voće

1.2 Nizovi primitivnih tipova

PRIMERI JEDNODIMENZIONALNIH NIZOVA

Niz je istorodan skup elemenata, što znači da su svi elementi iste vrste.

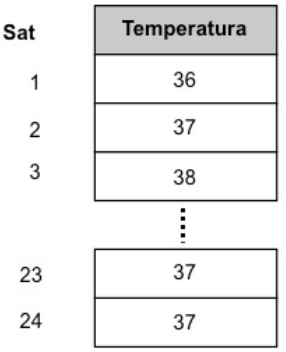
Niz je istorodan skup elemenata, što znači da su svi elementi iste vrste. Najjednostavniji niz je linearan. Takvi nizovi imaju samo jednu dimenziju i nazivaju se *jednodimenzionalnim nizovima*. Na sledećoj slici je dat primer jednog takvog niza**.** Niz **Vreme** bi se mogao koristiti za predstavljanje vremena kada se tokom nedelje desi neki značajan događaj (slika 1). Niz se ponekad naziva indeksiranom promenljivom. Razlozi su što, kao i promenljiva,niz može da sadrži vrednosti, i što se njegovim vrednostima pristupa preko indeksa



Slika 1.1.1 Primer jednodimenzionalnog niza

U matematici su se indeksi pisali potpisano, na primer Vreme1, Vreme2 itd. U većini programskih jezika se indeksi pišu u okviru uglastih zagrada: Vreme[1], Vreme [2] itd. Vreme[1], na prethodnoj slici ukazuje na 12:24. Generalno se za indekse mogu koristiti imena koja imaju svoje značenje u kontekstu aplikacije. Na primer, prethodni iskaz bismo mogli zameniti iskazom: Vreme[ponedeljak]

Na slici 2 je prikazan još jedan jednodimenzionalni niz, koji sadrži temperature pacijenta, koje su zapisivane svakog sata tokom dana. Na primer, Temperatura[3] ukazuje na temperaturu zapisanu trećeg sata



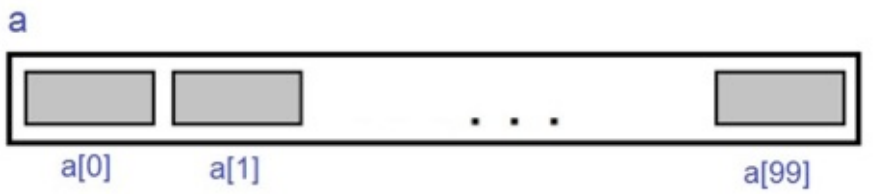
Slika 1.1.2 Još jedan primer jednodimenzionalnog niza

DEFINICIJA NIZA

Niz je struktura podataka koja predstavlja numerisan niz promenljivih istog tipa.

Niz je struktura podataka koja predstavlja numerisan niz promenljivih istog tipa. Pojedinačne promenljive u nizu se nazivaju elementi niza, a njihovredni broj u nizu se naziva indeks. Ukupan broj elemenata niza se naziva dužina niza. U Javi, numeracija elemenata niza počinje od nule. To znači daindeks nekog elementa niza može biti između nule i dužine niza manje jedan. Svi elementi niza moraju biti istog tipa koji se naziva bazni tip niza. Za bazni tip elemenata niza nema ograničenja - to može biti bilo koji tip u Javi, primitivni ili klasni.

Niz promenljivih se kao celina se u programu ukazuje promenljivom specijalnog, nizovnog tipa. Za označavanje svakog elementa niza se koristi zapis koji se sastoji od imena te promenljive i indeksa odgovarajućeg elementa u uglastim (srednjim) zagradama. Na primer, niz dužine 100 na koga ukazuje promenljiva a se sastoji od 100 promenljivih istog tipa čiji je izgled prikazan na slici:



Slika 1.1.3 Ilustracija niza dužine 100

Svaki element niza je obična promenljiva baznog tipa niza i može imati bilo koju vrednost baznog tipa. Na primer, ako je bazni tip niza a na slici deklarisan da bude int, onda je svaka od promenljivih a[0], a[1], ..., a[99] obična celobrojna promenljiva koja se u programu može koristiti na svakom mestu gde su dozvoljene celobrojne promenljive.

Posebnost nizova u Javi se ogleda i u tome što, mada kao objekti moraju pripadati nekoj klasi, njihova klasa ne mora da se definiše u programu. Naime, svakom postojećem tipu T se automatski pridružuje klasa nizova koja se označava T[]. Ova klasa T[] je upravo ona kojoj pripada objekat niza čiji su pojedinačni elementi tipa T. Tako, na primer, tipu int odgovara klasa int[] čiji su objekti svi nizovi baznog tipa int, takođe, ako je u programu definisana klasa Student, onda je automatski raspoloživa i klasa Student[] kojoj pripadaju svi objekti nizova baznog tipa Student.

Za zapis T[] se koriste kraći termini "niz baznog tipa T" ili "niz tipaT".

JEDNODIMENZIONALNI NIZOVI U JAVI

Promenljiva koja ukazuje na neki niz nije obična promenljiva klasnog tipa i njena vrednost može biti samo referenca na neki objekat niza, kao i referenca null

Postojanje klasnog tipa, recimo**, int[]** omogućava nam da deklarišemo neku promenljivu tog klasnog tipa, na primer:

```
int[] a;
```

Kao i svaka promenljiva klasnog tipa, promenljiva a može sadržati referencu na neki objekat klase **int[],** a kao što smo upravo objasnili, objekti klase **int[]** su nizovi baznog tipa***int***.

Objekat niza tipa **int[]** se konstruiše operatorom ***new***, doduše u posebnom obliku, a referenca na taj novi niz se zatim može dodeliti promenljivoj **a:**

```
a = new int[100];
```

gde vrednost 100 u uglastim zagradama određuje dužinu konstruisanog niza. Prethodna dva koraka se, kao što je to uobičajeno, mogu skratiti u jedan:

```
int[] a = new int[100];
```

Ovom deklaracijom se dakle alocira promenljiva a klasnog tipa int[], konstruiše objekat niza od 100 elemenata primitivnog tipa int i referenca na novokonstruisani objekat niza dodeljuje

promenljivoj a. Svaki od 100 elemenata ovog niza je obična promeljiva tipa int čiji sadržaj može biti neka celobrojna vrednost. Efekat prethodne deklaracije je slikovito prikazan na slici:

Slika 1.1.4 Element niza obična promenljiva int

Obratite pažnju na to da se često za promenljivu koja ukazuje na neki niz kraće govori kao da sadrži sam taj niz. Tako u poslednjem primeru kraće kažemo "niz a od 100 elemenata". Međutim, promenljiva nije obična promenljiva klasnog tipa i njena vrednost može biti samo referenca na neki objekat niza, kao i referenca ***null***.

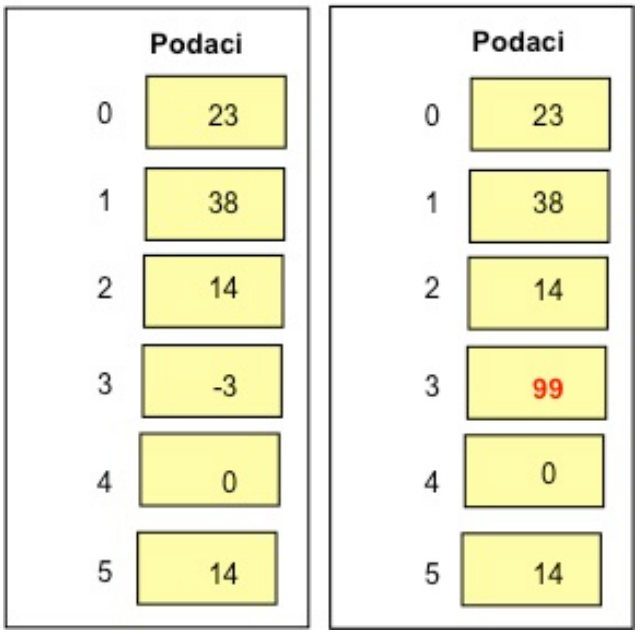
PRISTUP ELEMENTIMA NIZA

Niz je skup elemenata istog tipa.

Sve što je rečeno za nizove uopšte, važi za nizove u programskom jeziku Java. Niz je skup elemenata istog tipa. **Ti** elementi mogu biti primitivnog tipa, kao što je ***int, double*** i sl., ali elementi mogu biti i reference na objekte.

Elementima niza **a[]** na slici 7 se pristupa preko iskaza:

```
a[3]=99;
```



Slika 1.1.5 Primer jednodimenzionog niza

U ovom slučaju je a jedan niz celih brojeva. Prethodnim iskazom smo četvrtom elementu niza dodelili vrednost 99. Obratite pažnju na to da je vrednost indeksa 3, a da se ipak pristupa četvrtom elementu niza. Razlog je što brojanje elemenata u Javi, kao i jezicima C i C++ , počinje od 0

Poošto se izvrši prethodni iskaz, vrednost u četvrtom slotu niza je promenjena i iznosi 99.

Iskaz za pristupanje elementu niza, kao što je a[3], može se koristiti u svim izrazima u kojima se može koristiti promenljiva tipa, koji čine elementi niza. Na primer, ako je vrednost promenljive x 1, onda izraz:

```
(x + a[3]) / 4
```

daje vrednost 25.

DEKLARISANJE I KREIRANJE NIZA

Pošto se niz jednom napravi, njegova veličina, odnosno maksimalan broj elemenata koje može da primi, više se ne može menjati

Nizovi se u Javi deklarišu na sledeći način:

```
tip[] imeNiza;
```

Ovim se kompajleru govori da će **imeNiza** biti upotrebljeno kao ime niza u kojem će se nalaziti elementi tipa tip. Ovo, međutim, predstavlja samo deklaraciju niza. Niz još uvek nije napravljen. Ovim se samo deklariše promenljiva koja će, nekad u budućnosti, ukazivati na niz.

Vrlo često se dešava da se niz u jednom koraku i deklariše i konstruiše i ovo se radi na sledeći način:

```
tip[] imeNiza = new tip[duzina];
```

Ovaj iskaz istovremeno radi dve stvari:

1. Govori kompajleru da će **imeNiza** ukazivati na niz sa elementima određenog tipa (tip)
2. Konstruiše objekat niza, koji će sadržati zadati broj slotova (dužina).

Niz je objekat, kao i bilo koji drugi objekat u Javi. Kao i bilo koji drugi objekat, i on tokom rada programa zauzima svoj deo

glavne memorije. Razlika je samo u sintaksi koja se koristi kod pravljenja novog niza. Konstruktor niza ima sledeću sintaksu:

```
new tip[ duzina ]
```

gde je tip - tip podataka u nizu. Ovim se definiše tip, a takođe i veličina niza. Pošto se niz jednom napravi, njegova veličina, odnosno maksimalan broj elemenata koje može da primi, više se ne može menjati. Na primer iskaz:

```
int[] dt = new int[10];
```

kreira niz **dt**, a istovremeno svaki njegov element dobija vrednost 0.

Dužina nekog niza pokazuje koliko elemenata on može da sadrži. To znači da niz dužine N ima N elemenata, kojima se pristupa preko indeksa od 0 do N-1.

INDEKSI NIZA

Indeksi moraju biti celobrojnog tipa.

Indeksi moraju biti celobrojnog tipa. Sintaktički nije bitno da li između broja i zagrade postoji razmak, tako da je dt[1] i dt[1] potpuno isto. Ne sme se pokušavati pristup elementu koji ne postoji. Ako smo na primer, deklarisali niz:

```
int[] dt = new int[10];
```

onda važe sledeća pravila:

dt[-1] uvek neispravno

dt[10] neispravno (zbog deklaracije)

dt[1.5] uvek neispravno

dt[0] uvek u redu

dt[9] u redu (zbog deklaracije)

Ako u programu postoji izraz koji je uvek neispravan, program se neće iskompajlirati. Sa druge strane, najčešće je veličina niza nepoznata u trenutku kompajliranja. Ona se u tom slučaju, određuje u vreme izvršavanja programa. Pošto se niz pravi u vreme izvršavanja programa, kompajler ne zna njegovu dužinu i ne može da uoči sve greške. Ako u takvim slučajevima, program pristupi elementu koji ne postoji, javlja se greška i program se prekida.

Ako prilikom konstruisanja nema nikakvih drugih informacija o vrednostima elementa niza, onda se svaki elemenat niza

inicijalizuje na podrazumevanu vrednost u skladu sa tipom elementa. Ako su u pitanju celobrojne vrednosti, onda se svi elementi inicijalizuju na 0.

U programu, naravno, možete elementima dodeliti druge vrednosti. Pogledajmo sledeći primer:

```
class NizPrimer1{
  public static void main ( String[] args )  {
    int[] st = new int[5];
    st [0] = 23;
    st [1] = 38;
    st [2] = 7*2;
    System.out.println("st[0] je " + st [0] );
    System.out.println("st [1] je " + st [1] );
    System.out.println("st [2] je " + st [2] );
    System.out.println("st [3] je " + st [3] );
    System.out.println("st [4] je " + st [4] );
  }
}
```

Prethodni program bi trebalo da odštampa sledeće:

st[0] je 23

st [1] je 38

st [2] je 14

st [3] je 0

st [4] je 0

DEKLARISANJE, KONSTRUISANJE I INICIJALIZACIJA NIZA

Indeks niza je uvek ceo broj. Ipak to ne mora biti broj, već može biti i promenljiva tog tipa ili izraz čija je vrednost ceo broj

Indeks niza je uvek ceo broj. Ipak to ne mora biti broj, već može biti i promenljiva tog tipa ili izraz čija je vrednost ceo broj. Sledeći iskazi su, na primer, ispravni:

```
int vrednosti[] = new int[7];
int indeks;
indeks = 0;
```

```
vrednosti [indeks] = 71; // prvi element dobija vrednost 71
indeks = 5;
vrednosti [indeks] = 23;  // šesti element dobija vrednost  23
indeks = 3;
vrednosti[2+2]=vrednosti[indeks -3 ]; //isto kao
vrednosti[4]=vrednosti[0];
```

U istom iskazu se može izvršiti **deklarisanje, konstruisanje i inicijalizacija niza.**

```
int[] dt = {23, 38, 14, -3, 0, 14, 9, 103, 0, -56 };
```

Ovim se deklariše niz celobrojnih vrednosti, po imenu dt. Posle toga se konstruiše niz od 10 elemenata i na kraju se zadate vrednosti dodeljuju tim elementima. Prva vrednost iz liste za inicijalizaciju odgovara indeksu 0, druga indeksu 1 itd. (U ovom primeru dt[0] dobija vrednost 23).

U ovom iskazu ne morate reći koliko elemenata će niz imati. Kompajler će izbrojati vrednosti u listi i napraviti toliko elemenata. Ovakve liste za inicijalizaciju se obično koriste za male nizove.

KOPIRANJE NIZOVA

Niz se kopira u drugi niz, tako što se svaki član niza posebno kopira.

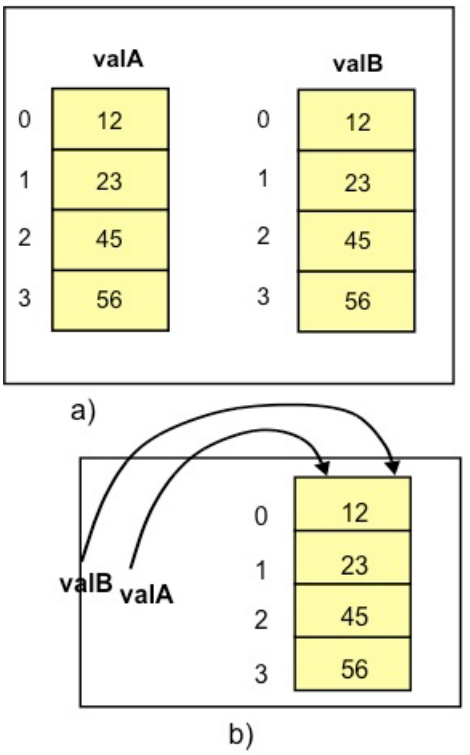
Često se dešava da želite da vrednosti iz jednog niza iskopirate u drugi. Niz se kopira u drugi niz, tako što se svaki član niza posebno kopira. Pogledajmo sledeći primer

```
class Niz4{
  public static void main ( String[] args ){
    int[] valA = { 12, 23, 45, 56 };
    int[] valB = new int[4];
    valB[ 0 ]  = valA[ 0 ] ;
    valB[ 1 ]  = valA[ 1 ] ;
    valB[ 2 ]  = valA[ 2 ] ;
    valB[ 3 ]  = valA[ 3 ] ;
   }
}
```

U ovom primeru smo svaki elemenat niza A kopirali u elemenat niza B. Ovo je isto kao iskaz dodele:

```
pr1 = pr2;
```

gde su obe promenljive tipa int. Nakon što se izvrše četiri iskaza dodele iz prethodnog primera nizovi izgledaju kao na slici 6



Slika 1.1.6 Kopiranje nizova

PRIMER 1

Kreiranje niza, inicijalizacija i ispis

Kreirati niz dužine N, gde se N unosi na zahtev odstrane korisnika.

Potom inicijalizovati niz i ispisati ga korisniku.

OBJAŠNJENJE:

Deklaracija niza int[] a;

U ovom trenutku se još uvek ne zna koliko elemeneta je moguće u niz smestiti.

a = new int[n]; // Pravljenje niza duzine n.

da bismo kreirali niz u memoriji koristimo ključnu reš NEW

NAPOMENA:

Kada želimo da inicijalizujemo SVE elemenete niza ili da ispišemo SVE elemente niza, ili bilo koju drugu operaciju nad svakim elementom - koristimo Petlju (for ili while) da pristupimo svakom elementu.

```
for (int i = 0; i < n; i++) {
            a[i] = sc.nextInt();
        }
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package primer1;

import java.util.Scanner;

/**
 *
 * @author Aleksandra
 */
public class Primer1 {

    public static void main(String[] args) {
        new Primer1();
    }

    public Primer1() {
        int n;
        int[] a; // Deklaracija nizovske promenljive a.
        // Citamo sa standardnog ulaza
        Scanner sc = new Scanner(System.in);
        System.out.println("Unesite dužinu niza:");
        n = sc.nextInt();

        a = new int[n]; // Pravljenje niza duzine n.
        // Svi elementi su inicijalizovani na 0

        //inicijalizacija niza A
        System.out.println("Unesite elemente niza:");
        for (int i = 0; i < n; i++) {
            a[i] = sc.nextInt();
        }

        // ispis pomocu numericke for petlje
        System.out.println("Elementi niza (for petlja):");
        for (int i = 0; i < n; i++) {
            System.out.print(a[i] + " ");
        }

        // ili foreach petlje
        System.out.println("\nElementi niza (foreach petlja):");
        for (int el : a) {
            System.out.print(el + " ");
        }
        sc.close();

    }

}
```

ZADATAK 1

Samostalno vežbanje kreiranja, inicijalizacija i ispis niza

- Kreirajte i učitajte dva celobrojna niza niz1 i niz2, oba dužine 10;
- Učitajte sa tastature članove obe niza;
- Kreirajte niz rezNiz čiji su članovi dobijeni sabiranjem odgovarajućih članova nizova niz1 i niz2;
- Prikažite na konzoli dobijeni niz.

1.3 Nizovi i petlje

PRIMENA NIZA U FOR PETLJI

Indeks niza u Javi uvek počinje od 0 i ide do maksimalnog broja elemenata, umanjenog za 1.

Indeks niza u Javi uvek počinje od 0 i ide do maksimalnog broja elemenata, umanjenog za 1. U skladu sa ovim, prolaz kroz nizove je najlakše uraditi preko petlje, čiji je brojač istovremeno i indeks za pristup članovima niza.

Pogledajmo sledeći primer

```
class BrojacNiza{
   public static void main ( String[] args )  {
    int[] niz = { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9 };
    for ( int indeks= 0 ; indeks < 10 ; indeks ++ ){
        System.out.println(niz[indeks]);
    }
   }
}
```

Bilo bi vrlo dosadno brojiti elemente niza. Možda, prilikom pisanja programa, nećete ni znati koliko elemenata ima u nizu. Često se nizovi kreiraju u vreme izvršavanja programa, i tek tada im se zna dužina. Često i dužina niza zavisi od samih podataka. Moramo biti u mogućnosti da napišemo program koji može da radi sa nizovima čija se dužina ne zna u trenutku pisanja programa.

U Javi postoji način da saznate koliko elemenata ima neki niz. Niz je objekat. Kao i svi objekti, on u sebi sadrži i druge atribute i metode, a ne samo elemente koji su njegovi članovi. U skladu sa tim, objekat niz ima atribut length, koji pokazuje koliko ima elemenata, a opet, u skladu sa ovim, petlja for iz prethodnog primera se može promeniti na sledeći način:

for (int indeks= 0 ; indeks < niz.length; indeks ++)

Ovo ćete koristiti u mnogim programima.

NAREDBA FOR-EACH

for-each petlja korisna u slučajevima kada je potrebno uraditi nešto sa svim elementima nekog niza, jer se ne mora voditi računa o indeksima i granici tog niza.

U verziji Java 5.0 je dodat novi oblik **for** petlje koji omogućuje lakši rad sa svim elementima nekog niza. Ova petlja se popularno naziva for-each petlja, iako se reč each ne pojavljuje u njenom zapisu.

Ako je niz tipa **bazniTip[],** onda for-each petlja za niz ima opšti oblik:

```
for (bazniTip elem : niz){
        ...
        // Obrada aktuelnog elementa elem
        ...
    }
```

Obratite pažnju na to da je znak dve tačke deo sintakse ove petlje. Pored toga, elem je kontrolna promenljiva baznog tipa koja se mora deklarisati unutar petlje. Prilikom izvršavanja for-each petlje, kontrolnoj promenljivoj elem se redom dodeljuje vrednost svakog elementa niza i izvršava se telo petlje za svaku tu vrednost. Preciznije, prethodni oblik for-each petlje je ekvivalentan sledećoj običnoj for petlji:

```
for (int i = 0; i < niz.length; i++) {
        bazniTip elem = niz[i];
        ...
        // Obrada aktuelnog elementa elem
        ...
    }
```

Na primer, sabiranje svih pozitivnih elemenata niza a tipa ***double[]*** može se uraditi for-each petljom na sledeći način:

```
double zbir = 0;
    for(double elem : a){
        if (elem > 0)
            zbir = zbir + elem;
    }
```

Primetimo da je for-each petlja korisna u slučajevima kada je potrebno uraditi nešto sa svim elementima nekog niza, jer se ne mora voditi računa o indeksima i granici tog niza. Međutim, ona nije od velike pomoći ako treba nešto uraditi samo sa nekim elementima niza, a ne sa svim elementima.Obratite pažnju i na to da se u telu for-each petlje zapravo samo čitaju vrednosti elemenata niza (i dodeljuju kontrolnoj promenljivoj), dok upisivanje vrednosti u elemente niza nije moguće. Na primer, ako treba svim elementima prethodno konstruisanog celobrojnog niza a dodeliti neku vrednost, recimo 17, onda bi bilo pogrešno napisati:

```
for (int elem : a){
        elem = 17;
    }
```

PRIMER 2

Pronaći koliko ukupno različitih elemenata ima niz.

Treba napraviti program koji će korisniku dozvoliti da unese do 100 elemenata u niz a potom pronaći broj različitih unetih elemenata.

Pošto za svaki element moramo da proverimo da li se ponovio imamo petlju u petlji pa proveravamo za svaki broj unutar petlje niz da li on postoji u petlji nizPonovoljenih.

Na ovoj način dobijamo novi niz samo sa ponovljenim brojevima

Detaljnije uputstvo je u source code-u.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package primer2;

import java.util.Scanner;

/**
 *
 * @author Aleksandra
 */
public class Primer2 {

    public static void main(String[] args) {
        new Primer2();
    }

    public Primer2() {
        int brojPonovljenih = 0;
        int[] niz = new int[100];
        int[] nizPonovljenih = new int[100];
        Scanner scan = new Scanner(System.in);

        int i = 0;
        //Do god korisnik ne unese vrednsot 1011 pitamo ga za novi broj
        while (scan.hasNext()) {
            int vred = scan.nextInt();
            if (vred == 1011) {
                break;
            }
            niz[i] = vred;
```

```
            i++;
        }
        /*Za svaku vrednost iz niza proveravamo da li ona postoji
         **u nizu ponovoljenih ako postoji ne dodajemo je u niz
         ponvoljenih ako ne postoji dodajemo novi clan*/
        for (int vrednost : niz) {
            boolean ima = false;
            for (int vrednostPonovljeni : nizPonovljenih) {
                if (vrednostPonovljeni == vrednost) {
                    ima = true;
                }
            }
            if (!ima) {
                nizPonovljenih[brojPonovljenih] = vrednost;
                brojPonovljenih++;
            }
        }

        System.out.println("Broj razlici brojeva je " + brojPonovljenih);

    }

}
```

ZADATAK 2

Pronaći koliko jednakih elemenata ima niz.

- Zadatak može biti po analogiji sa prethodnim zadatkom, a možete dati i vlastito rešenje;
- Pronaći koliko jednakih elemenata ima niz;
- naći zbir svih članova niza koji se ponavljaju;
- Štampati rezultate.

1.4 Nizovi objekata

NIZ KAO ARGUMENT

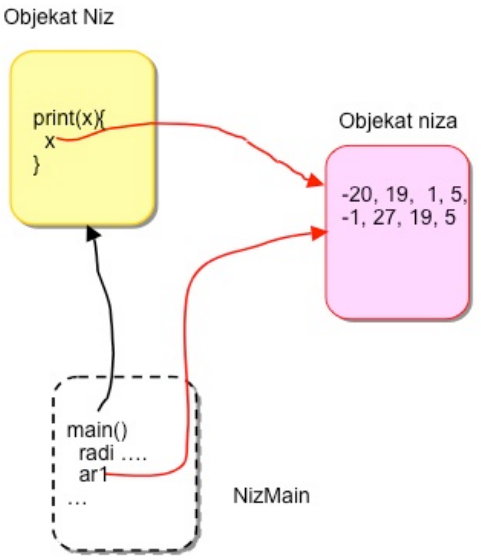
Niz se može proslediti kao argument nekog metoda. Kako je niz objekat, to seprosleđuje referenca niza.

Primeri koje smo do sada videli koriste nizove koji su deklarisani u metodu main(). Vrlo često se dešava da se nizovi deklarišu na jednom mestu, a onda se kao argumenti prosleđuju drugim metodima koji sa njima treba da nešto urade. Pogledajmo sledeći prime

```
class Niz{   // argument x je niz
  void print( int[] x )  {
```

```
    for ( int index=0; index < x.length; index++)
    System.out.print( x[index] + " " );
    System.out.println();
  }
}
class NizMain{
  public static void main ( String[] args ){
    Niz n = new Niz ();
    int[] ar1 = {-20,19,1,5,-1,27,19,5};
    System.out.print  ("\nNiz je: " );
    n.print( ar1 ); // poziva se metod print() klase  Niz
 }
}
```

Formalni argument metoda print u klasi Niz je tipa int[]. Ovo je referenca na objekat. Setićemo se da se objekti u Javi prenose u metode po referenci, odnosno da se u metodu pristupa istom objektu, kao na mestu poziva.



Slika 1.2.1 Niz kao argument

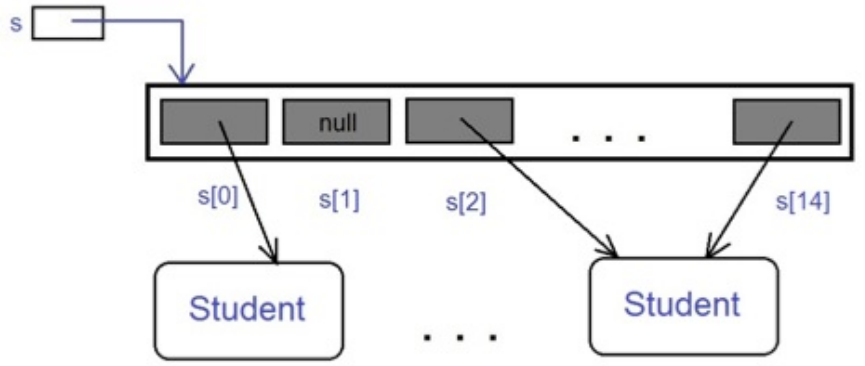
DEKLARACIJA NIZA OBJEKATA

Niz objekata ustvari sadrži samo reference objekata, kao elemente niza.

Deklaracija niza čiji bazni tip predstavlja neki objektni tip može biti ovakva:

```
Student[] s = new Student[15];
```

Ovom deklaracijom se alocira promenljiva s klasnog tipa Student[], konstruiše niz od 15 elemenata klasnog tipa Student i referenca na novi niz dodeljuje promenljivoj s. Svaki od 15 elemenata ovog niza predstavlja promenljivu klasnog tipa Student čiji sadržaj može biti referenca na objekat klaseStudent ili null. Jedan primer sadržaja niza s prikazan je na slici



Slika 1.2.2 Deklaracija niza objekata

U opštem slučaju, ako je brojElemenata celobrojni izraz, onda se izrazom:

```
new bazniTip[brojElemenata]
```

konstruiše niz koji kao objekat pripada klasi **bazniTip[brojElemenata**] i vraća se referenca na taj niz. Vrednost celobrojnog izraza brojElemenata u uglastim zagradama određuje dužinu tog niza, odnosno broj njegovih elemenata.

Nakon konstruisanja nekog niza, broj elemenata tog niza se ne može promeniti. Prema tome, iako možemo pisati, na primer:

```
int k = 5;
    double[] nizX = new double[k*10];
```

to ne znači da je broje elemenata realnog niza nizX promenljiv, već je fiksiran u trenutku njegovog konstruisanja vrednošću celobrojnog izraza 5*10 u uglastim zagradama koja iznosi 50. U ovom primeru dakle, niz nizX ima tačno 50 elemenata.

KREIRANJE NIZA OBJEKATA

Svaki objekat niza automatski sadrži javni celobrojni atribut length u kojem se nalazi dužina niza

Svaki objekat niza automatski sadrži javni celobrojni atribut ***length*** u kojem se nalazi dužina niza. Zato, u prethodnom primeru, program može koristiti atribut nizX.length čija je vrednost 50. Vrednost tog atributa se ne može menjati u programu.

Pošto elementi niza predstavljaju u suštini atribute objekta niza, ti elementi se u trenutku konstruisanja niza inicijalizuju podrazumevanim vrednostima za bazni tip niza - 0 za numerički tip, false za logički tip, '\u0000' za znakovni tip i null za klasni tip. Početne vrednosti elemenata niza se mogu i eksplicitno navesti u deklaraciji niza, unutar vitičastih zagrada i međusobno razdvojene zapetama.

Sledećom deklaracijom:

```
int[] nizStepena = new int[] {1, 2, 4, 8, 16, 32, 64, 128};
```

ili

```
int[] nizStepena = {1, 2, 4, 8, 16, 32, 64, 128};
```

konstruiše se niz **nizStepena** od 8 elementa čije su početne vrednosti 1, 2, 4, 8, 16, 32, 64, 128. Drugim rečima, element nizStepena[0] dobija početnu vrednost 1, element nizStepena[1] dobija početnu vrednost 2 i tako dalje, element nizStepena[7]

dobija početnu vrednost 128. Kada se na ovaj način zadaju početne vrednosti, dužina niza se ne navodi u deklaraciji niza i implicitno se zaključuje na osnovu broja navedenih vrednosti.

Pored toga, te vrednosti ne moraju da budu obični literali, nego mogu biti promenljive ili proizvoljni izrazi pod uslovom da su njihove vrednosti odgovarajućeg baznog tipa niza, na primer:

```
String s1 = new String("Prva reč");
    String[] nizS = new String[] {s1, "Druga reč", "Treća reč"};
```

Ili

```
String s1 = new String("Prva reč");
    String nizS[] = {s1, "Druga reč", "Treća reč"};
```

Opšti oblik operatora new u ovom kontekstu je:

```
new bazniTip[] {listaVrednosti}
```

ili

```
{ listaVrednosti}
```

INDEKSI KAO REFERENCE

Prilikom izvršavanja programa, element niza na koji se odnosi zapis, dobija se izračunavanjem vrednosti celobrojnog izraza u uglastim zagradama

Rezultat ovog izraza je referenca na novokonstruisani niz sa elementima koji su inicijalizovani datim vrednostima. Zbog toga se takav izraz može koristiti u programu na svim mestima gde se očekuje niz tipa bazniTip[]. Na primer, ako je prikažiPredmet() metod čiji jedini parametar jeste niz stringova, onda u programu možemo pisati:

```
prikažiPredmet(new String[] {"CS101", "CS102", "CS103"} );
```

Kao što smo već pomenuli, elementi niza bazniTip[] su obične promenljive baznog tipa niza i mogu se koristiti u programu na svim mestima gde je dozvoljena upotreba promenljive

baznog tipa. Takođe znamo da se elementi niza koriste u programu preko svojih indeksa i imena promenljive koja ukazuje na objekat niza. Na primer, deklaracijom:

```
int[] a = new int[100];
```

dobijamo zapravo 100 celobrojnih promenljivih a[0], a[1], ..., a[99].

Druga odlika nizova koja ih razdvaja među strukturama podataka je ta što se za indekse elemenata nizova u programu mogu koristiti proizvoljni celobrojni izrazi. Ako je i promenljiva tipa int, onda su, na primer a[i] i a[3*i-7] takođe ispravni zapisi elemenata

niza a iz prethodnog primera. Prilikom izvršavanja programa, zavisno od konkretne vrednosti promenljive i, element niza na koji se odnosi zapis, recimo, a[3*i-7] dobija se izračunavanjem vrednosti celobrojnog izraza u uglastim zagradama.

Tako, ako promenljiva i ima vrednost 3, dobija se element a[2]; ako promenljiva i ima vrednost 10, dobija se element a[23] i slično.

Praktičniji primer je sledeća petlja kojom se na ekranu prikazuju vrednosti svih elemenata niza a:

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

U ovoj petlji, početna vrednost brojača i je 0, pa se u prvoj iteraciji prikazuje element a[0]. Zatim se izrazom i++ brojač i uvećava i dobija vrednost 1, pa se u drugoj iteraciji prikazuju element a[1]. Ponavljajući ovaj postupak, broja i se na kraju svake iteracije uvećava za 1 i time se redom prikazuju vrednosti elemenata niza a. Poslednja iteracija koja se izvršava je ona kada je na početku vrednost brojača i jednaka 99 i tada se prikazuje elementa[99].

MOGUĆE GREŠKE KORIŠĆENJA NIZOVA

Dve greške se javljaju: 1) promenljiva tipa niya sadrži null, i 2) promenljiva je van dozvoljenih granica indeksa.

Nakon toga će i dobiti vrednost 100 i zato uslov nastavka petlje i < a.length neće biti tačan pošto **a.length** ima vrednost 100. Time se prekida izvršavanje petlje i završava prikazivanje vrednosti tačno svih elemenata niza a.

Napomena:

U radu sa elementima nekog niza u Javi su moguće dve vrste grešaka koje izazivaju prekid izvršavanja programa. Prvo, pretpostavimo da promenljiva **a** tipa niza sadrži vrednost null. Tada promenljiva a čak ni ne ukazuje na neki niz, pa naravno nema smisla koristiti neki element a[i] nepostojećeg niza. Ova vrste greške je opisana klasom **NullPointerException**.

Drugo, ako promenljiva **a** zaista ukazuje na prethodno konstruisan niz, onda vrednost izraza i u **a[i]** može biti van dozvoljenih granica indeksa niza. To će biti slučaj kada se izračunavanjem

izraza i dobije i < 0 ili i >= a.length. Ova vrste greške je opisana klasom ArrayIndexOutOfBoundsException.

NIZ OBJEKATA JE NIZ REFERENCI OBJEKATA

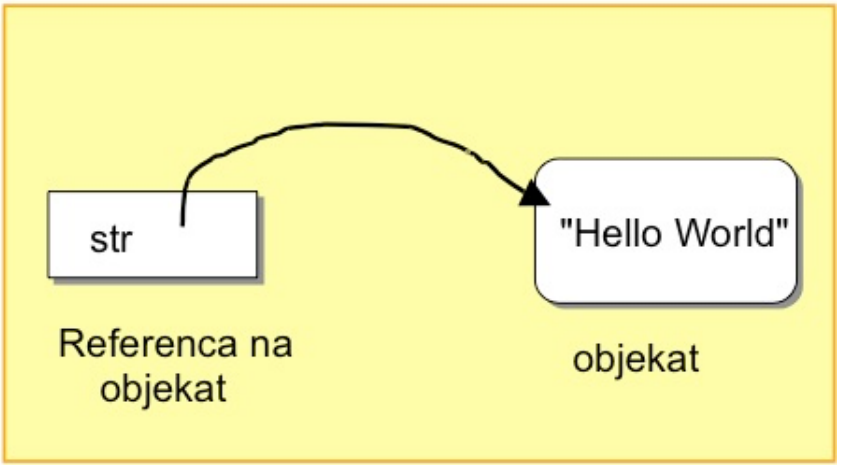
Niz objekata sadrži elemente u vidu referenci objekata.

Elementi niza moraju biti istog tipa. To ne moraju biti primitivni tipovi, već mogu biti i objekti.

Objekat nastaje pozivom njegovog konstruktora, odnosno primenom operatora new. Pošto se objekat napravi, njemu se pristupa preko reference. Referenca se obično čuva u nekoj promenljivoj. Sledeći program će deklarisati promenljivu sa referencom, posle čega se pravi objekat i njegova referenca stavlja u prethodno definisanu promenljivu.

```
String str;         // deklaracija promenljive sa referencom
str = "Hello World" ;  // pravi se objekat i pamti njegova referenca
```

Grafički se to može prikazati kao na slici 3:



Slika 1.2.3 Referenca na obejakat

Naravno da je moguće tu istu promenljivu iskoristiti za referenciranje nekog drugog objekta, kao u sledećem programu:

```
String str;
str = "Zdravo" ;
. . . .
str = "Doviđenja" ;
```

Šta ako je u programu potrebno da pratite reference na hiljadu stringova? Mogli biste da koristite nizove, naravno. Niz referenci na objekte tipa String se deklariše na sledeći način:

```
String[] strNiz;        // 1.
```

Ovim se deklariše promenljiva strNiz, koja će u budućnosti ukazivati na neki niz. Svaki element tog niza bi mogao biti objekat tipa String, ali do sada nemamo niz. Niz sa 8 referenci na objekte tipa String ćemo kreirati na sledeći način

```
strNiz = new String[8] ;  // 2.
```

Sada strNiz ukazuje na niz. Taj niz ima 8 elemenata. Nijedan od ovih elemenata, još uvek ne ukazuje na objekat. Elementi niza su reference na objekte i automatski su inicijalizovani na null vrednost, što je specijalna vrednost koja znači "nema objekta".

Da bismo stvarno napravili objekat tipa string i smestili referencu na njega u niz, treba da uradimo sledeće:

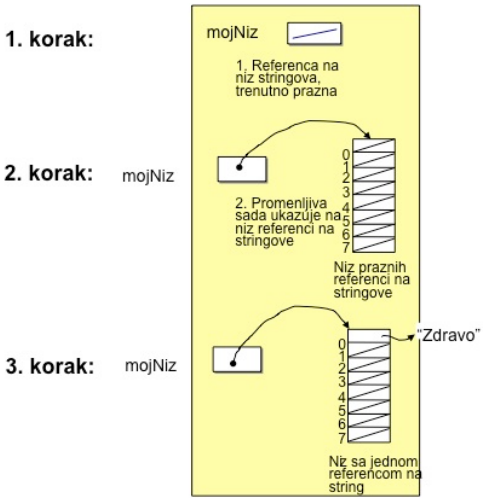
PRIMER INICIJALIZACIJE DEKLARISANOG NIZA

Referenca nekog objekta se smešta u niz objekata, ya svaki objekat na koji niz ukazuje.

Da bismo stvarno napravili objekat tipa string i smestili referencu na njega u niz, treba da uradimo sledeće:

```
strNiz [0] = "Zdravo" ;     // 3.
```

Objekti tipa String su specijalni objekti, kod kojih nije obavezna upotreba operatora ***new*** (on se koristi automatski). Šematski se ovo može prikazati kao na slici 4.



Slika 1.2.4 Primer inicijalizacije deklarisanog niza

Pogledajmo sledeći program: .

```
String[] strNiz = new String[8] ; // kombinovani iskaz
strNiz [0] = "Zdravo" ;
strNiz [1] = "prvi" ;
strNiz [2] = "drugi" ;
strNiz [3] = "treci" ;
```

Ovde smo deklarisali niz od osam članova. Tom prilikom su svi njegovi elementi inicijalizovani na ***null***. Kasnije smo nekim elementima niza (prva 4) dodelili vrednosti. Kolika je sada dužina niza?

Atribut ***length*** objekta niz uvek sadrži deklarisanu veličinu niza (u ovom slučaju 8). To što svi elementi niza nisu dobili referencu, ne utiče na veličinu.

PARAMETRI METODA MAIN()

Metod main() ima niz ulaznih argumentar koje su eference na objekte tipa String. Oni ukazuju na stringove koji sadrže tekst, a koji se na komandnoj liniji unosi prilikom pokretanja programa.

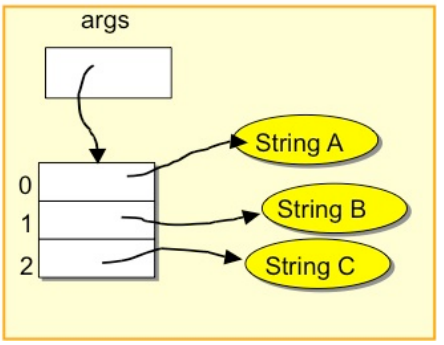
U svakom primeru koji smo do sada radili (a i svaki program pisan u Javi mora ga imati) koristili smo metod main().Prisetimo se njegove signature:

```
public static void main( String[] args )
```

Ovaj potpis metoda main ukazuje na to da metod main() ima ulazni argument koji je niz referenci na objekte tipa String. Ovaj niz pravi Java sistem, pre nego što prepusti kontrolu metodu main(). Elementi ovog niza ukazuju na stringove koji sadrže tekst, a koji se na komandnoj liniji unosi prilikom pokretanja programa. Pretpostavimo da ste program pokrenuli na sledeći način:

```
C:\>java StringDemo stringA stringB stringC
```

Na slici 5 je pokazano kako argument args izgleda u toku rada metoda main(). Reči na komandnoj liniji koje slede posle imena programa su argumenti komandne linije. Onima koji su programirali pod Unixom ovo je poznato, ali je verovatno manje poznato onima koji koriste Windows. Ovo je pogodan način da se u program pošalju neke vrednosti, a da pritom ne mora da se vodi dijalog sa korisnikom. Argumenti su nizovi karaktera. Može biti proizvoljan broj argumenata, a neki specijalni karakteri kao što su < i > nisu dozvoljeni. Svaki argument se od suseda odvaja blanko karakterima



Slika 1.2.5 Parametri metoda main()

Ne podržavaju svi sistemi argumente sa komandne linije. Apple računari uopšte i nemaju komandnu liniju. Integrisana programska okruženja, kao što Cafe ili Jbuilder, takođe imaju ograničenu podršku za argumente sa komandne linije.

Pošto su argumenti sa komandne linije međusobno odvojeni blanko karakterima, postavlja se pitanje kako uneti argument koji sadrži više reči – odvojene blanko karakterima. U tom slučaju se argument stavlja pod dvostruke navodnike.

```
C:\>java SomeProgram "jedan argument" "drugi argument"
```

Java sistem obavezno kreira niz referenci na objekte tipa String, ako oni na komandnoj liniji postoje. Sa druge strane, to ne znači da ih program mora koristiti. Kao što ste videli, mi ih u našim primerima do sada nismo koristili.

PRIMER 3

Kreiranje niza String objekata i njihovo prikazivanje

- Kreira se niz objekata - stringova;
- Program demonstrira kako se kreira niz objekata (linije koda 15 i 16);
- Zatim vrši se unos objekata u niz (linije koda 21 - 24);
- Konačno, prikazuje se učitani niz (linije koda 25 - 28).

```
package nizobjekata;

import java.util.Scanner;

/**
 *
 * @author Vladimir Milicevic
 */
public class NizObjekata {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        String[] imena;
        imena = new String[20];
        Scanner tastatura = new Scanner (System.in);
        System.out.print ("Koliko imena zelite da unesete : ");
        int n = tastatura.nextInt();

        for (int i=0;i<n;i++){
             System.out.print ("Unesite " + (i+1) + ". ime: ");
             imena[i]= tastatura.nextLine();
        }
        for (int i=0;i<n;i++){
             System.out.println ("Niz imena: " + imena[i] + " ");
             }
    }

}
```

PRIMER 4

Referenca na niz

Kreirati niz koji je dužine 5 i odmah inicijalizujte elemente kroz kod.

Ispisati niz a i dužinu niza

Kreirati novi niz b naredbom

int b[] = a; promeniti jedan element niza B i ispisati nizove A i B.Uočiti promene u nizu A.

Objašnjenje:

Duzina niza se dobija preko atributa a.length

int b[] = a; - narredbom dobijamo niz B koji dobija referencu niza A.

Dalje, svaka promena nad nizom B je zapravo promena nad nizom A, jer se u memoriji nikada nije kreirao još jedan novi niz B.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package primer3;

/**
 *
 * @author Aleksandra
 */
public class Primer4 {

    public static void main(String[] args) {
        new Primer4();
    }

    public Primer4() {
        int[] a = {1, 10, 15, 20, 25};

        System.out.println("Niz a ima " + a.length + " elemenata.");

        System.out.println("Elementi niza a su:");
        for (int e : a) {
            System.out.print(e + " ");
        }
        int b[] = a;
        System.out.println("b = a");
        b[2] = 3;
        System.out.println("b[2] = " + b[2] + "\n");

        System.out.println("Ispisujemo ponovo niz a:");
        for (int e : a) {
            System.out.print(e + " ");
        }
```

```
        System.out.println();
    }

}
```

PRIMER 5

Napraviti program koji generiše niz od 20 random elemenata.

Napraviti program koji generiše niz od 20 random elemenata do broja 200. Prikazati kako izgleda niz pre i posle sortiranja.

Svaki element u nizu ima svoj indeks ovo možemo iskoristiti i napraviti niz gde će svaki i indeksa do veličine niza biti neki Random broj do 200.

Za generisanje slučajnih brojeva koristimo objekat klase Random, pristupamo svakom elementu niza i inicijalizujemo ga.
Random r = new Random();
for (int i = 0; i < niz.length; i++) {
niz[i] = r.nextInt(200);
}

r.nextInt(200); - vraća broj od 0 do 200

Koristimo Arrays.sort metodu za sortiranje niza. Arrays.sort(niz);

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package primer4;

import java.util.Arrays;
import java.util.Random;

/**
 *
 * @author Aleksandra
 */
public class Primer5 {

    public static void main(String[] args) {
        new Primer5();
    }

    public Primer5() {
        int[] niz = new int[20];
        Random r = new Random();
        for (int i = 0; i < niz.length; i++) {
            niz[i] = r.nextInt(200);
```

```
        }
        System.out.println("_____________________");
        System.out.println("Niz pre sortiranja");
        System.out.println("------------------");
        for (int i = 0; i < niz.length; i++) {
            System.out.println("Element " + (i + 1) + " je: " + niz[i]);
        }
        Arrays.sort(niz);
        System.out.println("_____________________");
        System.out.println("Niz posle sortiranja");
        System.out.println("--------------------");
        for (int i = 0; i < niz.length; i++) {
            System.out.println("Element " + (i + 1) + " je: " + niz[i]);
        }

    }

}
```

ZADATAK 3

Samostalno vežbanje rada sa nizovima objekata.

- Kreirajte niz objekata klase:

class Studenti {
int ocena;
}

- Učitajte vrednosti za polje ocena za svaki objekata koji je član niza;
- Prikažite sve učitane vrednosti na konzoli.

1.5 Klasa Arrays

VAŽNIJI METODI KLASE ARRAYS

Klasa Arrays sadrži nekoliko statičkih metoda koji obezbeđuju korisne operacije nad nizovima čiji je bazni tip jedan od primitivnih tipova.

Pomoćna klasa **Arrays** iz paketa **java.util** može biti jako korisna u radu sa nizovima. Ona sadrži nekoliko statičkih metoda koji obezbeđuju korisne operacije nad nizovima čiji je bazni tip jedan od primitivnih tipova.

Sledi nepotpun spisak i opis ovih statičkih metoda:

String toString(tip[] a) - vraća reprezentaciju niza a u formi stringa. Pri tom se svi elementi niza a nalaze unutar uglastih zagrada po redu njihovih pozicija u nizu i međusobno su razdvojeni zarezima.

tip[] copyOf (tip[] a, int d) - vraća novu kopiju niza a dužine d. Ako je dužina d veća od a.length, višak elemenata kopije niza se inicijalizuje nulom ili vrednošću false. U suprotnom slučaju, kopira se samo početnih d elemenata niza a.

tip[] copyOfRange (tip[] a, int od, int do) - kopira niz a od indeksa od do indeksa do u novi niz. Element sa indeksom od niza a se uključuje, dok se onaj sa indeksom do ne uključuje u novi niz. Ako je indeks do veći od a.length, višak elemenata kopije niza se inicijalizuje nulom ili vrednošću false.

void sort (tip[] a) - sortira niz a u mestu u rastućem redosledu.

int binarySearch (tip[] a, tip v) - koristi binarnu pretragu za nalaženje vrednosti v u sortiranom nizu a. Ako je data vrednost v nađena u nizu, vraća se indeks odgovarajućeg elementa. U suprotnom slučaju, vraća se negativna vrednost k tako da -k-1 odgovara poziciji gde bi data vrednost trebalo da se nalazi u sortiranom nizu.

void fill(tip[] a, tip v) - dodeljuje svim elementima niza a istu vrednost v.

boolean equals(tip[] a, tip[] b) - vraća vrednost true ukoliko nizovi a i b imaju istu dužinu i jednake odgovarajuće elemente. U suprotnom slučaju, vraća vrednost false.

PRIMER 6

Napraviti program koji množi svaka dva elementa niza i smešta u jedan

Napraviti program koji od korisnika traži da unese paran broj. Nakon unosa parnog broja treba izgenerisati (Random) niz dužine korisničkog unosa. Nakon ovoga treba napraviti niz duplo manje veličine u koje treba smestiti svake dve podeljene vrednosti prvog niza. Ispisati oba niza.

Kod ovakvih zadataka treba dosta paziti oko indeksa.

U ovom primeru treba da imamo dva brojača indeksa jedan za duplo manji niz i jedan za veći niz kako bi bilo lakše raditi operacije.

Detaljniji komenatri su ostavljeni u kodu.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package primer6;

import java.util.Arrays;
import java.util.Random;
```

```
import javax.swing.JOptionPane;

/**
 *
 * @author Aleksandra
 */
public class Primer6 {

    public static void main(String[] args) {
        new Primer6();
    }

    public Primer6() {
        int broj = Integer.parseInt(JOptionPane.showInputDialog("Unesite broj: "));
        //Provera da li je broj paran.
        if (broj % 2 == 0) {
            double[] brojevi = new double[broj];
            Random r = new Random();
            for (int i = 0; i < broj; i++) {
                brojevi[i] = r.nextInt(250);
            }
            //Pravimo niz duplo manje velicine
            double[] deljeniBrojevi = new double[broj / 2];
            int brojPreskoka = 0;
            for (int i = 0; i < deljeniBrojevi.length; i++) {
                //Deljenje svaka dva elementa niza 1 i smestanje vrednost u novi
                //niz sa duplo manje elemenata
                deljeniBrojevi[i] = brojevi[brojPreskoka] / brojevi[brojPreskoka +
1];
                brojPreskoka += 2;
            }
            //Ispisivanje koriscenjem Arrays.toString metode.
            System.out.println(Arrays.toString(brojevi));
            System.out.println(Arrays.toString(deljeniBrojevi));
        } else {
            System.out.println("Niste uneli paran broj");
        }

    }
}
```

PRIMER 7

Napraviti program koji množi elemente niza i pravi novi niz

Napraviti program koji od korisnika traži da unese broj. Zatim treba napraviti niz sa random elementima veličine korisničkog unsoa i tražiti od korisnika da unese elemente tog niza. Nakon unsoa elemenata treba napraviti još dva niza. Jedan niz treba da ima sve iste vrednosti kao prvi samo pomnožene sa 2 a drugi pomnožene sa 3.

Kada korisnik unese vrednosti prvog niza potrebno je imati dve petlje.

Jedna petlja prolazi kroz prvi niz i stavlja vrednosti novog niza na korisnički unos pomnožen sa 2, dok druga treba da radi to isto samo sa 3.

Naprednije rešenje je da se jednom petljom uradi inicijalizacija oba niza:
for (int i = 0; i < kolicinaPodataka; i++) {
nizPutaDva[i] = korisnickiUnos[i] * 2;
nizPutaTri[i] = korisnickiUnos[i] * 3;
}

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package primer7;

import java.util.Arrays;
import javax.swing.JOptionPane;

/**
 *
 * @author Aleksandra
 */
public class Primer7 {

    public static void main(String[] args) {
        new Primer7();
    }

    public Primer7() {
        int kolicinaPodataka =
Integer.parseInt(JOptionPane.showInputDialog("Unesite broj"));
        int[] korisnickiUnos = new int[kolicinaPodataka];
        int[] nizPutaDva = new int[kolicinaPodataka];
        int[] nizPutaTri = new int[kolicinaPodataka];
        for (int i = 0; i < kolicinaPodataka; i++) {
            korisnickiUnos[i] =
Integer.parseInt(JOptionPane.showInputDialog("Unesite " + i + " broj"));
        }
        for (int i = 0; i < kolicinaPodataka; i++) {
            nizPutaDva[i] = korisnickiUnos[i] * 2;
        }
        for (int i = 0; i < kolicinaPodataka; i++) {
            nizPutaTri[i] = korisnickiUnos[i] * 3;
        }

        System.out.println(Arrays.toString(korisnickiUnos));
        System.out.println(Arrays.toString(nizPutaDva));
        System.out.println(Arrays.toString(nizPutaTri));

    }
}
```

PRIMER 8

Napraviti program koji generiše šifre predmeta i smešta u niz

Napraviti program koji od korisnika traži da unese broj predmeta a potom generiše toliko šifara predmeta. Šifra predmeta se sastoji iz dva velika slova i 3 broja. Generisanje predmete smestiti u niz i prikazati niz korišćenjem Arrays.toString metode.

Treba napraviti metodu koja vraća String koji je sastavljen iz 2 karaktera i 3 broja.

Potom treba napraviti niz Stringova za uneti broj korisnik i svakom elementu niza dodeliti rezultat metode.

Nakon ovoga iskorititi Arrays.toString metodu za prikaz elemenata niza.

Metoda getIzmisljeniPredmet izmišnjeni predmet vraća String koji je ime predmeta.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package primer8;

import java.util.Arrays;
import java.util.Random;
import javax.swing.JOptionPane;

/**
 *
 * @author Aleksandra
 */
public class Primer8 {

    public static void main(String[] args) {
        new Primer8();
    }

    public Primer8() {

        int brojPredmetaZaGenerisanje =
Integer.parseInt(JOptionPane.showInputDialog("Unesite broj predmeta za
generisanje"));
        String[] listaImena = new String[brojPredmetaZaGenerisanje];
        for (int i = 0; i < listaImena.length; i++) {
            listaImena[i] = getIzmisljeniPredmet();
        }
        System.out.println(Arrays.toString(listaImena));
    }

    public static String getIzmisljeniPredmet() {
        Random r = new Random();
```

```
        String forReturn = "";
        forReturn += (char) (r.nextInt(25) + 65);
        forReturn += (char) (r.nextInt(25) + 65);
        forReturn += r.nextInt(899) + 100;
        return forReturn;
    }
}
```

PRIMER 9

Napraviti program koji generiše identifikacione brojeve za svaku osobu

Napraviti program koji od korisnika traži da broj osoba koje treba da unese a potom generiše identifikacione brojeve za svakog od korisnika. Niz osoba mora da bude dva puta veći od broja osoba jer u (n) niza treba staviti ID korisnika a u (n+1) niza treba staviti njegovo Ime koje korisnik unosi.

ID korisnika generišemo koristeći Random javinu klasu i metodu nextInt.

Da bi lepo prikazali osobe i njihove ID-ove koristimo for petlju za prolazak kroz niz i ispisivanje informacija

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package primer9;

import java.util.Arrays;
import java.util.Random;
import javax.swing.JOptionPane;

/**
 *
 * @author Aleksandra
 */
public class Primer9 {

    public static void main(String[] args) {
        new Primer9();
    }

    public Primer9() {
        int brojOsoba = Integer.parseInt(JOptionPane.showInputDialog("Unesite broj
osoba"));
        String[] izvestajBrojOsoba = new String[brojOsoba * 2];
        Random r = new Random();
        int brojacPetlje = 0;
        while (brojacPetlje < (brojOsoba * 2)) {
            izvestajBrojOsoba[brojacPetlje] = String.valueOf(r.nextInt(500));
```

```
            izvestajBrojOsoba[brojacPetlje + 1] =
JOptionPane.showInputDialog("Unesite ime");
            brojacPetlje += 2;
        }

        System.out.println(Arrays.toString(izvestajBrojOsoba));
        for (int i = 0; i < izvestajBrojOsoba.length; i += 2) {
            System.out.println("Ime: " + izvestajBrojOsoba[i + 1]);
            System.out.println("Id: " + izvestajBrojOsoba[i]);
        }

    }
}
```

ZADATAK 4

Samostalno vežbanje primene klase Arrays

Zadatak 4 -1

Napraviti program koji generiše niz parnih brojeva. Prikazati izgenerisan niz koristeći Arrays.toString metodu.

Zadatak 4 - 2
Napraviti program koji generiše niz neparnih brojeva a potom taj niz množi sa 2 i prebacuje u novi niz. Prikazati oba niza koristeći Arrays.toString metodu.

1.6 Nedostaci nizova

UVOD

Svaki niz se definiše sa određenim predpostavljenim maksimalnim brojem elemenata. To može biti uzrok određenih problema.

Broj elemenata običnog niza je fiksiran u trenutku konstruisanja niza i ne može se više menjati, tako da dužina niza određuje maksimalan broj podataka koji se pojedinačno čuvaju u elementima niza, ali svi elementi niza ne moraju biti iskorišćeni u svakom trenutku izvršavanja programa. Naime, u mnogim primenama broj podataka koji se čuvaju u elementima niza varira tokom izvršavanja programa. Jedan primer je program za pisanje dokumenata u kojem se pojedinačni redovi teksta dokumenta čuvaju u nizu tipa **String[].** Drugi primer je program za evidentiranje ocena studenata u kojem se koristi dvodimenzionalni niz čiji elementi sadrže ocene evidentiranih studenata. U ovim i sličnim primerima je karakteristično to što postoji niz fiksne dužine, ali je popunjen samo delimično.

Ovaj način korišćenja nizova ima nekoliko nedostatka. Manji problem je to što se u programu mora voditi računa o tome koliko je zaista iskorišćen niz- za tu svrhu se može uvesti dodatna promenljiva čija vrednost ukazuje na indeks prvog slobodnog elementa niza.

Ostali problemi se mogu lakše razumeti na konkretnom primeru programa za evidentiranje broja aktivnih studenata na predmetu CS101, pri čemu se broj aktivnih studenata može uvećati ili smanjivati.

Svaki student može biti predstavljen brojem indeksa, tako da skup aktivnih studenata može biti predstavljen nizom **nizStudenata** tipa ***int[].*** Međutim, s obzirom da se broj aktivnih studenata menja vremenom, potrebno je imati dodatnu promenljivu brojStudenata koja sadrži trenutni broj aktivnih studenata. Pod pretpostavkom da nikad neće biti više od 50 aktivnih studenata istovremeno, relevantne deklaracije u programu su:

```
// Najviše 50 aktivnih studenata
    int[] nizStudenata = new int[50];
    // Na početku, broj aktivnih studenata je 0
    int brojStudenata = 0;
```

Nakon što se unese nekoliko aktivnih studenata (predstavljenih brojevima indeksa), trenutni broj aktivnih studenata će se nalaziti u promenljivoj **brojStudenata**. Pored toga, elementi niza **nizStudenata** na pozicijama od 0 do **brojStudenata** - 1 sadrže konkretne brojeve indeksa za aktivne studente.

Promenljiva **brojStudenata** ima dvostruku ulogu - pored trenutnog broja aktivnih studenata, ova promenljiva sadrži i indeks prvog "slobodnog" elementa niza studenata koji je neiskorišćen.

DA LI JE BITAN REDOSLED UPISIVANJA ELEMENATA NIZA?

Pri dodavanju i uklanjanju elemenata ponekad se mora voditi računa o redosledu studenata.

Dodavanje novog aktivnog studenta u program se postiže dodavanjem tog studenta nizu **nizStudenata** na kraj niza, pri čemu je novi aktivni student predstavljen promenljivom **noviStudent:**

```
// Novi student ide u prvi slobodni element niza
    nizStudenata[brojStudenata] = noviStudent;
    // Trenutni broj aktivnih studenata se uvećava za 1
    brojStudenata++;
```

Ako se, recimo, određeni student ispiše sa fakulteta ili promeni smer, uklanjanje tog studenta se mora izvesti tako da ne ostane "rupa" u nizunizStudenata. Zbog toga, ukoliko treba ukloniti studenta koji se nalazi na k-toj poziciji u nizu nizStudenata, postupak njegovog uklanjanja zavisi od toga da li je redosled studenata bitan ili ne.

Ako redosled studenata nije bitan, jedan način za uklanjanje k-tog studenta je premeštanje poslednjeg studenta na k-to mesto u nizu **nizStudenata** i smanjivanje vrednosti promenljive brojStudenata za jedan:

```
nizStudenata[k] = nizStudenata[brojStudenata - 1];
    brojStudenata--;
```

Student koji se nalazio na k-tom mestu nije više u nizu **nizStudenata**, jer se na to mesto premešta student na poslednjoj poziciji brojStudenata - 1. S druge strane, ova pozicija nije više logički poslednja i postaje slobodna za upisivanje novih studenata, jer se promenljiva **brojStudenata** umanjuje za jedan.

U drugom slučaju kada je bitan redosled studenata, radi uklanjanja k-tog studenta se svi studenti od indeksa k+1 moraju pomeriti ulevo za jedno mesto. Drugim rečima, (k+1)-vi student dolazi na k-to mesto studenta koji se uklanja, zatim (k+2)-gi student dolazi na (k+1)-vo mesto koje je oslobođeno prethodnim pomeranjem, i tako dalje:

```
for (int i = k+1; i < brojStudenata; i++)
    nizStudenata[i - 1] = nizStudenata[i];
    brojStudenata--;
```

KOJI BROJ ELEMENATA NIZA PLANIRATI?

Morase postaviti proizvoljna fiksna granica niza. Šta se dešava ako se javi potreba da se poveća ovaj broj elemenata niza?

Veći problem u radu sa nizovima fiksne dužine je to što se mora postaviti prilično proizvoljna granica za broj elemenata niza. U prethodnom primeru se postavlja pitanje šta uraditi ako broj aktivnih studenata bude veći od 50, kolika je dužina niza **nizStudenata**. Očigledno rešenje je da se taj niz konstruiše sa mnogo većom dužinom. Ali i ta veća dužina ne garantuje da broj studenata neće narasti toliko da opet ni veća dužina niza nije dovoljna za registrovanje svih aktivnih studenata. Naravno, dužina niza se uvek može izabrati da bude vrlo velika i tako da bude dovoljna za sve praktične slučajeve. Ovo rešenje ipak nije zadovoljavajuće, jer se može desiti da veliki deo niza bude neiskorišćen i da zato niz bespotrebno zauzima veliki memorijski prostor, možda sprečavajući izvršavanje drugih programa na računaru.

Najbolje rešenje ovog problema je to da niz početno ima skromnu dužinu, a da se zatim može produžiti (ili smanjiti) po potrebi u programu. Međutim, kako ovo nije moguće, moramo se zadovoljiti približnim rešenjem koje je skoro isto tako dobro. Naime, podsetimo se da promenljiva tipa niza ne sadrži elemente niza, nego samo ukazuje na objekat niza koji zapravo sadrži njegove elemente.

Ovaj niz se ne može produžiti, ali se može konstruisati novi objekat dužeg niza i promeniti vrednost promenljive koja ukazuje na objekat starog niza tako da ukazuje na objekat novog niza. Pored toga, naravno, elementi starog niza se moraju prekopirati u novi niz. Konačan rezultat ovog postupka biće dakle to da ista promenljiva tipa niza ukazuje na novi objekat dužeg niza koji sadrži sve elemente starog niza i ima dodatne "slobodne" elemente. Uz to, objekat starog niza biće automatski uklonjen iz memorije, jer nijedna promenljiva više ne ukazuje na objekat starog niza.

PRIMER 10 - DODAVANJE ELEMENATA NIZA

Za kopiranje nizova koristimo statičku metodu copyOf() klase Arrays

Ako se ovaj pristup primeni u programu za evidentiranje aktivnih studenata, onda se u postupak za dodavanje studenta moraju ugraditi prethodne napomene u slučaju kada je popunjen ceo niz **nizStudenata** početne dužine 50

```
// Dodavanje novog studenta kada je niz popunjen
    if (brojStudenata == nizStudenata.length) {
        int novaDužina = 2 * nizStudenata.length;
        int[] noviNizStudenata = Arrays.copyOf(nizStudenata, novaDužina);
        nizStudenata = noviNizStudenata;
    }
    // Dodavanje novog studenta u uvećani niz nizStudenata
    nizStudenata[brojStudenata] = noviStudent;
    brojStudenata++;
```

Primetimo ovde da promenljiva **nizStudenata** ukazuje na stari delimično popunjen niz ili na novi niz dvostruko duži od starog. Zbog toga, nakon izvršavanja if naredbe, element **nizStudenata[brojStudenata**] je sigurno slobodan i može mu se dodeliti novi student bez bojazni da će se premašiti granicu niza na koji ukazuje promenljiva **nizStudenata**. Za kopiranje nizova koristimo statičku metodu **copyOf()** klase **Arrays.**

PRIMER 11

Napraviti program za određivanje najveće ocene u nizu.

Napraviti program koji će od korisnika tražiti da unese broj studenata. Napraviti niz te veličine i potom pitati korisnika da upiše ocene. Naći najveću ocenu u nizu i ispisati je.

Da bi napravili niz nekog određenog tipa podataka ili klase koristimo uglaste zagrade [].

Primer int[] niz = new int[velicinaniza];Kako bi prošli kroz niz for petljom koristimo dužinu niza (niz.length).

UPUTSTVO ZA TRAŽENJE MAKSIMALNOG ELEMENTANIZA:

1. postaviti element na poziciji 0 na najveći element

2. proći redom i proveriti da li postoji element veći od trenutnoj maksimuma, ako postoji postaviti da maskimum bude taj element i nastaviti dalje.

Kada smo sve elemente niza ispitali, onaj koji se nadje u promenljivoj maksimum je najveći element u nizu.

Za traženje minimuma je sličan postupak.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package prime11;

import java.util.Scanner;

/**
 *
 * @author Aleksandra
 */
public class Primer11 {

    public static void main(String[] args) {
        new Primer11();
    }

    public Primer11() {
        /**
         * Koristimo Scanner za ucitavanje sa konzole
         */
        Scanner scan = new Scanner(System.in);
        /**
         * Citamo unos studentata
         */
        System.out.println("Unesite broj studenata: ");
        int brojStudenata = scan.nextInt();
        // Pravimo niz velicine unetog broja
        int[] studenti = new int[brojStudenata];
        int maxOcena = 0;
        //Prolazimo kroz niz od 0 do velicine niza
        for (int i = 0; i < studenti.length; i++) {
            String value = scan.next();
            studenti[i] = Integer.parseInt(value);
            if (i == 0) {
                maxOcena = studenti[i];
            } else if (studenti[i] > maxOcena) {
                maxOcena = studenti[i];
            }
            System.out.println("Ocena studenta " + i + " je " + value);
        }
        System.out.println("Najveca ocena je: " + maxOcena);
        scan.close();
    }
}
```

ZADATAK 5

Samostalno vežbanje dodavanja članova niza

Kreirajte numerički niz:

- parni članovi niza su dvostruko veći od njihovih neparnih prethodnika;
- Štampati kreirani niz;
- Naći zbir svih članova niza;
- Prikazati zbir na konzoli.

JEDNODIMENZIONALNI NIZOVI (VIDEO)

Video objašnjava primenu dimenzionalnoh nizova.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 2

Učitavanje i obrada jednodimenzionalnog niza

UVODNO RAZMATRANJE

Izneseni pokazni primeri ukazuju na mogućnosti korišćenja jednodimenzionalnih nizova.

U ovom delu lekcije akcenat će biti na analizi i demonstraciji najčešće korišćenih operacija sa nizovima:

- učitavanju nizova;
- obradi nizova i
- ispisivanju nizova.

Sve navedene operacije će biti objašnjene kroz pažljivo birane primere. Na kraju pokušajte samostalno da uradite zadatke koji su u direktnoj vezi sa navedenim operacijama nad nizovima.

PRIMER 12 - UČITAVANJE I OBRADA JEDNODIMENZIONALNOG NIZA

Učitavanje veličine niza n, a zatim sve brojčane elemente niza, proračunava njihovu aritmetičku sredinu

Tekst zadatka: Napisati program koji učitava broj n sa standardnog ulaza, zatim učitava n elemenata niza i ispisuje sumu i aritmetičku sredinu elemenata niza. Program treba da bude deo NetBeans projekta pod nazivom l02.nizovi. Naziv klase treba da bude Primer1A. Za učitavanje podataka od korisnika treba koristiti objekat klase Scanner, a za ispis podataka objekat System.out.

Analiza problema: Korišćenjem objekta ulaz klase Scanner ćemo učitati broj elemenata niza u celobrojnu promenljivu n, a zatim ćemo konstruisati celobrojni niz pod nazivom niz od n elemenata. Celobrojnu promenljivu suma ćemo inicijalno postaviti na vrednost 0, a zatim ćemo je uvećavati za i-ti element niza koristeći petlju for. Na kraju ćemo izračunati i ispisati aritmetičku sredinu i sumu elemenata niza.

Programski kod:

```
package l02.nizovi;
import java.util.Scanner;
public class Primer1A {
    public static void main(String[] args) {
        Scanner ulaz = new Scanner(System.in);
        System.out.println("Unesite broj elemenata niza: ");
        int n = ulaz.nextInt();
        int[] niz = new int[n];
        System.out.println("Unesite elemente niza redom: ");
        int suma = 0;
        for (int i = 0; i < n; i++) {
            niz[i] = ulaz.nextInt();
            suma = suma + niz[i];
        }
        double aritmSredina = (double) suma / n;
        System.out.println("Suma elemenata je: " + suma);
        System.out.printf("Aritmeticka sredina je: %.2f\n", aritmSredina);
    }
}
```

PRIMER 13 – UČITAVANJE I OBRADA JEDNODIMENZIONALNOG NIZA 2. NAČIN

Definisani su metodi ucitajNiz() i sumaNiza()

Tekst zadatka:

Definisati sledeće metode i izmeniti prethodni program tako da poziva ove metode:

public int[] ucitajNiz(int n) - na osnovu parametra n učitati niz od n elemenata i vratiti učitani niz

public int sumaNiza(int[] niz) - na osnovu parametra niz tipa celobrojni niz, vratiti sumu elemenata ovog niza

Program treba da bude deo NetBeans projekta pod nazivom l02.nizovi. Naziv klase treba da bude **Primer1B**. Za učitavanje podataka od korisnika treba koristiti objekat klase **Scanner**, a za ispis podataka objekat **System.out.**

Analiza problema:

Metoda sumaNiza na osnovu parametra niz treba da vrati sumu elemenata ovog niza. Postupak je prilično jasan, međutim treba naznačiti da ako hoćemo da iz statičke metode main() pozivamo ovu metodu, i ova metoda mora biti statička, pošto statička metoda može pozivati samo statičke metode i statičke promenljive (atribute, polja).

O ovim detaljima ćemo detaljnije govoriti u šestom predavanju.

Metoda ucitajNiz na osnovu parametra n treba da učita niz od n elemenata i da vrati učitani niz. Treba naznačiti da ova metoda mora imati pristupobjektu ulaz klase Scanner, tj. deklaraciju ovog objekta treba prebaciti iz metode main() u definiciju promenljivih (atributa)

same klase i postaviti da bude statički (jer statička metoda može pozivati samo statičke promenljive).

PRIMER 13 – 2. NAČIN: PROGRAMSKI KOD

Drugi način rešavanja problema

```
package l02.nizovi;
import java.util.Scanner;
public class Primer1B {
    static Scanner ulaz = new Scanner(System.in);
    public static void main(String[] args) {
        System.out.println("Unesite broj elemenata niza: ");
        int n = ulaz.nextInt();
        int[] nizBrojeva = ucitajNiz(n);
        double aritmSredina = (double) sumaNiza(nizBrojeva) / n;
        System.out.println("Suma elemenata je: " + sumaNiza(nizBrojeva));
        System.out.printf("Aritmeticka sredina je: %.2f\n", aritmSredina);
    }
    public static int[] ucitajNiz(int n){
        int[] niz = new int[n];
        System.out.println("Unesite elemente niza redom: ");
        for (int i = 0; i < n; i++) {
            niz[i] = ulaz.nextInt();
        }
        return niz;
    }
    public static int sumaNiza(int[] niz){
        int suma = 0;
        for (int i = 0; i < niz.length; i++) {
            suma = suma + niz[i];
        }
        return suma;
    }
}
```

PRIMER 14 - UČITAVANJE I OBRADA JEDNODIMENZIONALNOG NIZA–3. NAČIN

Kreirana klasa ArrayUtility koja sadrži sve razvijene metode.

Možemo kreirati i novu klasu **ArrayUtility** koja će sadržati sve pomoćne metode koje se tiču nizova, a definicije metod **sumaNiza** i **ucitajNiz** prebaciti u ovu klasu**.**

Prethodni program ćemo promeniti tako da poziva statičke metode klase **ArrayUtility** po nazivu klase i naziv klase promeniti u **Primer1C**. Prednost ovog pristupa je u tome što više različitih klasa može pozivati statičke metode klase **ArrayUtility.**

PRIMER 14 – 3. NAČIN: PROGRAMSKI KOD

Treći način rešavanja problema

```
package l02.nizovi;
import java.util.Scanner;
public class Primer1C {
    static Scanner ulaz = new Scanner(System.in);
    public static void main(String[] args) {
        System.out.println("Unesite broj elemenata niza: ");
        int n = ulaz.nextInt();
        int[] nizBrojeva = ArrayUtility.ucitajNiz(n);
        double aritmSredina = (double) ArrayUtility.sumaNiza(nizBrojeva) / n;
        System.out.println("Suma elemenata je: " +
ArrayUtility.sumaNiza(nizBrojeva));
        System.out.printf("Aritmeticka sredina je: %.2f\n", aritmSredina);
    }
}
package cs101.L07;
import static cs101.L07.Primer1C.ulaz
public class ArrayUtility {
public static int[] ucitajNiz(int n){
        int[] niz = new int[n];
        System.out.println("Unesite elemente niza redom: ");
        for (int i = 0; i < n; i++) {
            niz[i] = ulaz.nextInt();
        }
        return niz;
    }
    public static int sumaNiza(int[] niz){
        int suma = 0;
        for (int i = 0; i < niz.length; i++) {
            suma = suma + niz[i];
        }
        return suma;
    }
}
```

ZADATAK 6 - RAČUNANJE BROJA NATPROSEČNIH ELEMENATA U NIZU

Ovo je zadatak koji svaki student treba samostalno d auradi.

Napisati program koji učitava broj **n** sa standardnog ulaza, zatim učitava **n** elemenata niza i ispisuje koliko elemenata ovog niza je veće od aritmetičke sredine niza.

Program treba da bude deo NetBeans projekta pod nazivom **l02.nizovi.**

Napraviti program koristeći sva tri prikazana pristupa, tako da naziv klasa bude **Zadatak1A, Zadatak1B i Zadatak1C.**

Za učitavanje podataka od korisnika treba koristiti objekat klase **Scanner**, a za ispis podataka objekat **System.out.**

Poglavlje 3

Domaći zadaci

ZADACI 1-3

Ove zadatke treba samostalno da uradite.

Zadatak 1

Napravi niz brojeva od 100 elemenata. Svaki član se računa kao (2*i)* (2*i). Ispisati sve elemente ovog niza.

Zadatak 2

Napravi niz brojeva od 75 elemenata. Svaki član se računa kao (6*i)/ (2*i). Ispisati sve elemente ovog niza. Kada je i 0 i vrednost treba biti 0.

Zadatak 3

Napraviti program koji generiše niz neparnih brojeva a potom taj niz množi sa 2 i prebacuje u novi niz. Pronaći najmanji element niza.

Zaključak

REZIME

Nizovi su uređeni skupovi podataka, kojim se može manipulisti kao sa objektima.

U predavanju smo se upoznali sa:

- Pojmom niza kao strukture podataka
- Jednodimenzionalnim nizovima
- Klasom Arrays
- Naredbom for-each
- Nedostacima nizova

Kroz konkretne primere je provežbano:

- Kreiranje programa za učitavanje i obradu jednodimenzionalnog niza
- Kreiranje posebne klase sa statičkim metodama za rad sa jednodimenzionalnim nizovima
- Kreiranje metoda za rad sa jednodimenzionalnim nizovima bez korišćenja petlji
- Kreiranje metoda za rad sa jednodimenzionalnim nizovima sa korišćenjem petlji
- Kreiranje programa za učitavanje i obradu dvodimenzionalnog niza
- Kreiranje metoda za rad sa stringovima bez korišćenja petlji
- Kreiranje metoda za rad sa stringovima sa korišćenjem petlji