



Distance Learning System

Core Java Programming

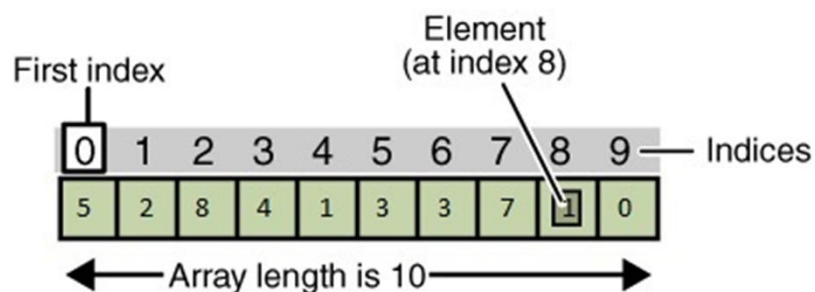
Nizovi i kolekcije

Šta je niz?

- Niz je složeni tip koji podrazumeva smeštanje više vrednosti u jednoj promenljivoj
- Vrednosti koje se smeštaju u niz **moraju** biti istog tipa, i taj tip određujemo prilikom kreiranja niza
- Takođe, broj vrednosti koje možemo smestiti u niz mora se odrediti prilikom kreiranja niza, i nakon toga se ne može više menjati (**imutabilnost**)
- Svaka vrednost unutar niza, identifikuje se pomoću takozvanog indeksa niza – broja koji određuje njenu poziciju

Skladištenje vrednosti u niz

- Svaka vrednost u jednom nizu naziva se **član** niza
- Svaki član ima svoj indeks koji koristimo za pristup tom članu
- Članovi mogu biti bilo kog tipa, ali se njihovi tipovi unutar niza ne smeju (i ne mogu) razlikovati
- Promenljiva kojom je predstavljen niz, sadrži zapravo adresu prvog člana niza



Pozicioniranje članova u nizu

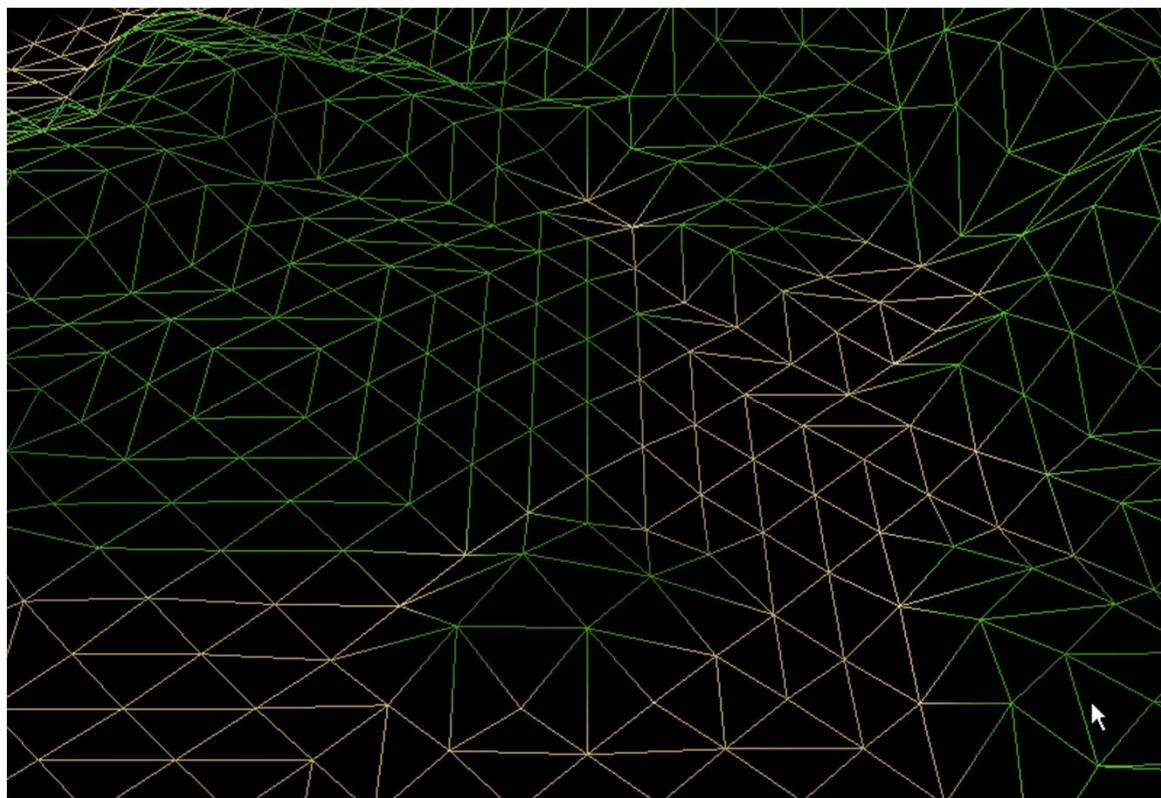
- Ako imamo jedan skup brojeva 3,5,2,6,1,6, primećujemo da se broj 6 ponavlja. To znači da, ako bismo rekli da želimo da iz liste izvučemo broj 6, ne bismo imali preciznu informaciju o kom se od dva moguća broja šest, zapravo, radi. Jedan od načina da se ovo reši, bio bi uvođenje numeracije elemenata ovog skupa:

broj: 3,5,2,6,1,6,
pozicija: 1,2,3,4,5,6

- U ovom slučaju možemo vrlo jednostavno da identifikujemo svaki broj u svakom trenutku. I ako bismo želeli da prikazemo neki od brojeva 6, mogli bismo da prosledimo jednu od dve pozicije (4 ili 6). Po istom principu funkcionišu i nizovi, s tim što kod nizova brojanje pozicija počinje od nule

Pozicija elementa	0	1	2	3	4	5
Element	3	5	2	6	1	6

Gde se sve koriste nizovi?



Nizovi

LINKgroup

Gde se sve koriste nizovi?

sailboat

About 23,700,000 results (0.31 seconds)

Related searches for **sailboat**
Stores: [eBay](#) [West Marine](#) [Amazon](#)
Brands: [Beneteau](#) [Catalina Yachts](#) [MacGregor](#) [Jeanneau](#) [Hobie](#)

Sailboat - Wikipedia, the free encyclopedia
[en.wikipedia.org/wiki/Sailboat](#)
Diagram of a **sailboat**, in this case a typical mono hull sloop with a Bermuda or Marconi rig. A **sailboat** or sailing boat is a boat propelled partly or entirely by sails.
[Parts of a Sailboat - Types - Hulls - Keel](#)

Sailboat Listings - sailboats for sale
[www.sailboatlistings.com/](#)
List your boat for free with free **sailboat** classified ads. **Sailboat** Listings include racers, cruisers, sloops, catamarans, trimarans, daysailers, sailing dinghies, and ...
[Sailboats - Florida](#) - [Search](#) - [California](#)

Beneteau USA
[www.beneteauusa.com/](#)
The American division of the world's largest **sailboat** manufacturer. Located in Marion, South Carolina..

Hunter Sailboats - North America's largest manufacturer of **sailboats** ...
[www.huntermarine.com/](#)
Builds recreational **sailboats** from 9-50 feet. Boat specifications, dealer locator, FAQs, owner's clubs, and logo apparel and accessories are all listed.

[Images for sailboat](#) - Report images

[Sailboats.co.uk](#)

Ads - Why these ads?
VERNICOS YACHTS
[www.vernicos.com](#)
The symbol of yachting in Greece
Brokerage - Yacht Sales - Charter

[Used Sailboats for Sale](#)
[www.tulipyachts.com](#)
Looking for a Sailing Yacht?
Find it in Portugal

Laser Sailboat Parts
[www.backyardboats.com](#)
Backyard Boats' Vanguard, Laser and Sunfish Parts and Accessories Sales

Fareastsails.com
[www.fareastsails.com](#)
Hong Kongs Performance Sailmaker
Wholesale Pricing - Worldwide

Laser Sailboat Parts
[www.lasersailboat.com](#)
Parts, Sails, Accessories
Detailed Pictures, Secure Shopping

[See your ad here >](#)

NIZ

IMDb Find Movies, TV shows, Celebrities and more... All

Register | Login | Help

Your Watchlist

Movies TV News Videos Community IMDbPro Apps

In Theaters
Coming Soon
Top Movies
Showtimes & Tickets
Trailers
Watchlist
MyMovies
New: DVD & Blu-Ray
Top 250
Genres
Editors' Spotlight
Independent Film
Horror
Road to the Oscars
Sundance Film Festival

Sundance Film Festival 2012
Wondering what's going to be happening in Park City the next week? Our **Sundance Film Festival** section is the perfect place to find all the latest from indie film's most prestigious gathering. We've got a [mini-guide to the festival's feature-length films](#), a [photo gallery preview](#), a rundown on [what we're looking forward to seeing](#), and more. We'll also have updates via Facebook, Twitter and our blog -- get everything you need for the **Sundance Film Festival** right here!

NewsDesk
Top News Movie News TV News Celebrity News
Ryan Seacrest Launching TV Network With Mark Cuban, AEG, CAA
15 hours ago | The Hollywood Reporter
Ryan Seacrest is getting his own TV network after all. The multi-hyphenate, along with partners AEG and talent agency CAA, have struck a deal with billionaire entrepreneur Mark Cuban to rebrand his **Hudnet** as AXS TV (pronounced Access) beginning this summer.

Box Office
1. **Constraband** \$24.3M
2. **Beauty and the Beast** \$17.8M
3. **Mission: Impossible - Ghost Protocol** \$11.7M
4. **Joyful Noise** \$11.2M
5. **Sherlock Holmes: A Game of Shadows** \$8.59M
[See more box office results >](#)

Opening This Week
Underworld: Awakening + 42%
Haywire + 305%
Coriolanus + 135%
...

Nizovi

LINKgroup

Kreiranje niza u Javi

- Svi programski jezici znaju za nizove, pa tako i Java
- U kombinaciji sa prethodno naučenim tehnikama, savladavanje nizova učiniće nas dovoljno kompletnim da možemo da rešimo i neke kompleksnije probleme
- Nizovi u Javi (a takođe i u ostalim jezicima) se naročito dobro slažu sa petljama
- Da bi deklarirali jednu promenljivu koja će sadržati niz, treba da tipu (ili nazivu promenljive), dodamo prazne uglaste zagrade:

Obe deklaracije su validne,
Ali je prva rasprostranjenija

```
int[] x;  
int y[];
```

Kreiranje niza u Javi

- Samom deklaracijom niza, ne možemo raditi sa njim ništa pre nego što mu odredimo veličinu i inicijalizujemo ga
- Niz inicijalizujemo ključnom rečju new, čemu sledi naziv tipa, i takođe uglaste zagrade, samo ovaj put, sa brojem članova između

```
int[] array = new int[5];
```

- Prethodna linija znači da je inicijalizovan niz koju i sebi sadrži pet vrednosti tipa int. Ako bi pokušali da prikazemo rezultat te promenljive, dobili bi zanimljiv rezultat na izlazu:

```
int[] array = new int[5];  
System.out.println(array);
```

run: [I@1ea87e7b

- Rezultat je zapravo hash kod objekta, što dokazuje da je niz zapravo ništa drugo do (ne baš standardan) objekat

Pristup članovima niza

- Da bi pristupili određenom članu niza, koristimo uglaste zagrade u koje stavljamo indeks člana koji nas interesuje

```
int[] array = new int[5];  
System.out.println("Value on position 3 is " + (array[3]));
```

- Program na izlazu daje broj 0 za člana 3, jer je 0 podrazumevana vrednost celobrojnog tipa int
- Da bi dodelili neku vrednost određenom članu niza, koristimo takođe uglaste zagrade i indeks, ali ovaj put sa primenom jednakosti:

```
array[3] = 25;
```

Inicijalizacija niza sa vrednostima

- Vodeći se primerom sa prethodnog slajda, jedan niz od pet članova bi mogli inicijalno popuniti na sledeći način:

```
array[0] = 5;  
array[1] = 10;  
array[2] = 15;  
array[3] = 25;  
array[4] = 30;
```

- Ako već unapred znamo koje će vrednosti niz sadržati, tada možemo inicijalizovati niz direktnim unosom vrednosti.

```
int[] array = { 5,10,15,25,30 };
```

- Ovakva sintaksa podrazumeva da se u vitičaste zagrade smeste sve vrednosti niza, poređane prema indeksima

Tipovi podataka u nizu

- Jedan niz može sadržati podatke bilo kog tipa, ali jedan niz takođe nikada ne može sadržati podatke različitog tipa
- Ukoliko hoćemo, možemo „prevariti“ sistem, tako što ćemo izvršiti box-ing, odnosno, staviti u niz podatke maskirane u tip **Object**.

```
int[] array = { 25, "Hello", "Array" }; //I am not working  
Object[] array = { 25, "Hello", "Array" }; //I am ok
```

- Mešanje tipova u nizu može doneti veću štetu nego korist, jer ukoliko ne znamo koje tačno tipove očekujemo na određenoj poziciji, ne možemo uraditi adekvatnu konverziju
- Procedura konverzije tipova u objekte i suprotno, naziva se **boxing / unboxing**

Vežba

- Potrebno je kreirati niz stringova koji sadrži sledeće stringove:
Heart, Cherry, Coin, Melon, Jocker
- U nastavku programa, treba kreirati sistem za generisanje kombinacija (pomoću slučajnog izbora računara)
- Potrebno je da program prikaže dobijenu kombinaciju znakova na izlazu

```
Combination is:  
Jocker Melon Melon Heart Coin
```

Manipulacija nizovima

- Iako znamo da kreiramo niz, uzmemo podatak iz njega ili stavimo neki podatak u njega, ovo najčešće nije dovoljno za efikasan rad sa nizovima, jer ćemo često sa nizovima da radimo još nešto. Na primer, da ih prebrojimo, da ih spojimo, razdvojimo, kopiramo, a naročito, da konvertujemo niz u string i obrnuto.
- Sve ove probleme možemo rešiti ručno, ali takođe, većina njih već postoji ugrađena u Javu

Dobavljanje dužine niza

- U radu sa nizovima, podatak koji nam najčešće treba (a da nije u pitanju sam član niza) jeste njegova dužina
- Najlakši način da dobijemo dužinu niza, jeste korišćenjem njegovog svojstva **length**.

```
int[] array = { 2,3,4,5,6 };  
System.out.println("Niz array ima " + array.length + " članova");
```

Dužina niza array



- Zašto bi dobavljali dužinu niza programabilno, kada već znamo koja mu je dužina?

Kopiranje niza

- Kopiranje je jedan od većih problema u objektnom programiranju. Obzirom da se objekti uvek transportuju po referenci, njihovo istinsko kopiranje praktično nije ni moguće
- Na primer, pokušajte da pogodite šta će biti prikazano na izlazu nakon izvršenja sledećeg koda:

```
int[] array = { 1,2,3,4,5 };  
int[] array1 = array;  
array1[3]=15;  
System.out.println(array[3]);
```

Kloniranje niza

- Mnogo češće od kopiranja niza, biće nam potrebno kloniranje niza. Kloniranje znači kreiranje novog niza, identičnog izvornom.
- Ono se može obaviti na dva načina. Pomoću metode **clone** samog **array** objekta, i pomoću statičke metode **arraycopy**, klase **System**.
- I u prvom (arraycopy) i u drugom slučaju (clone), niz se klonira, ali se njegov sadržaj ne klonira već samo kopira, tako da, ako su u pitanju reference, one će ukazivati na iste objekte i unutar novog niza (***jcex102014 SimpleArray primer clone and arraycopy***)

Klasa Arrays

- Pored pomenute dve metode za kloniranje niza, Java SE takođe kompletnu klasu, čija je namena isključivo pomoć pri rukovanju nizovima. Ova klasa naziva se **Arrays**, a sadrži metode za pretragu nizova, sortiranje, popunjavanje nizova, pretvaranje niza u **listu**.
- Obavezno samostalno detaljno proučite ovu klasu pomoću dokumentacije sa sledeće adrese:

<http://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html>

Vežba

- Postoje dva niza:

```
int[] A = { 1, 2, 3 };  
int[] B = { 4, 5, 6 };
```

- Potrebno je kreirati treći (tročlani) niz (C) koji će sadržati zbir članova nizova A i B koji se nalaze na istoj poziciji (1+4, 2+5...)

Automatski prolazak kroz niz

- Najčešće, kroz niz ćemo prolaziti automatski i obrađivati njegove vrednosti. Ovaj prolazak vrši se obično pomoću petlje
- For petlja je idealna za prolazak kroz niz, jer podrazumeva kontrolnu promenljivu, koju možemo iskoristiti za izlaganje aktuelnog indeksa niza
- Svaki način ima svoje dobre i loše strane

```
String[] cities = { "Paris", "London", "New York" };  
for(int i=0; i<cities.length; i++){  
    System.out.println(cities[i]);  
}
```

```
String[] jedi = { "Qui-Gon Jinn", "Obi-Wan Kenobi", "New York" };  
for(String city : cities){  
    System.out.println(city);  
}
```

```
String[] races = { "Ork", "Undead", "Elf" };  
int counter = 0;  
while(counter<races.length){  
    System.out.println(races[counter++]);  
}
```

Vežba

- Potrebno je kreirati tri niza.
- 1. Niz pojmova (String) u kome će biti bicikl, automobi i mobilni telefon
- 2. Niz naziva (String) u kome će se nalaziti nazivi koji će odgovarati pojmovima iz prethodnog niza
- 3. Niz cena (double) koji će odgovarati cenama pojmova iz prethodnih nizova
- Potrebno je proći kroz nizove i prikazati sve podatke o svakom pojmu (naziv, cenu i tip)

```
BMX is type Bycicle and it costs 150.55
```

```
Renault 4 is type Car and it costs 100.0
```

```
I-Phone is type Mobile Phone and it costs 500.0
```


Vežba

- Dat je niz

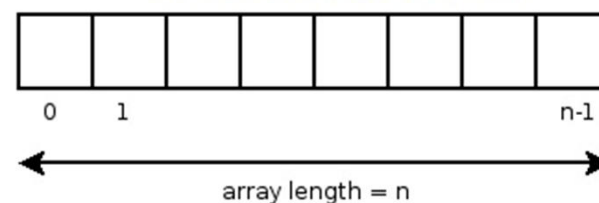
```
int[] array = {2,-5,4,12,54,-2,-50,150};
```

- Potrebno je na osnovu niza array, kreirati dva nova niza, tako da se u jednom nađu svi negativni brojevi iz niza array, a u drugom svi pozitivni brojevi iz niza array (uključujući i nule)

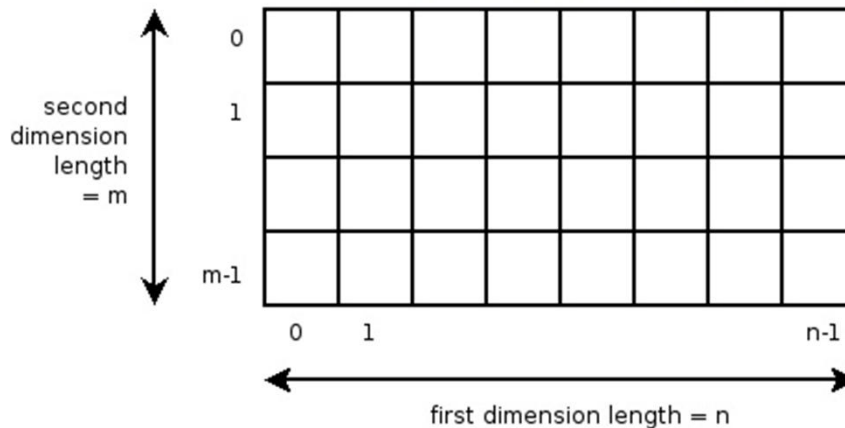
Višedimenzionalni nizovi

- Osim jednodimenzionalnih nizova, Java je u stanju da prepozna i takozvane višedimenzionalne nizove. Ovakvi nizovi korisni su za rad sa podacima koji imaju matričnu ili tabelarnu formu

One-dimensional array



Two-dimensional array



Gde srećemo višedimenzionalne nizove

- Sve transformacije u grafičkim programima realizuju se pomoću matrica, a matrice su ništa drugo do višedimenzionalni nizovi

$$\begin{array}{ccc}
 \text{X-Rotation in 3D} & \text{Z-Rotation in 3D} & \text{Scale in 3D} \\
 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & (4 \times 4) * (4 \times 1) = (4 \times 1)
 \end{array}$$

$$\begin{array}{ccc}
 \text{Y-Rotation in 3D} & \text{Translation in 3D} & \text{Matrix Multiplication} \\
 \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ q \end{bmatrix}
 \end{array}$$

Gde srečemo višedimenzionalne nizove

- Svaka slika na računar, takođe se može reprezentovati kroz višedimenzionalni niz



Višedimenzionalni nizovi

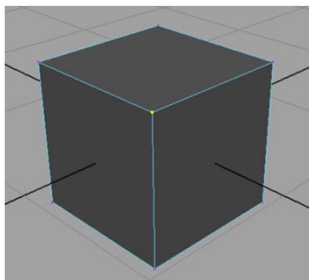
LINKgroup

Gde srećemo višedimenzionalne nizove

- Multidimenzionalni nizovi su odličan način za skladištenje tabelarno struktuiranih podataka

id	firstname	lastname	balance
28512	user_28509	user_28509	500.00
28511	user_28508	user_28508	500.00
28510	user_28507	user_28507	500.00
28509	user_28506	user_28506	500.00
28508	user_28505	user_28505	500.00
28507	user_28504	user_28504	500.00
28506	user_28503	user_28503	500.00

- A takođe i tačaka u prostoru



	S	M	L	XL	XXL
MIN ka svima	100	200	300	500	1.000
MIN u mreži	Neograničeno*	Neograničeno*	Neograničeno*	Neograničeno*	Neograničeno*
SMS	100	200	300	500	1.000
Internet	50 MB	50 MB	50 MB	50 MB	50 MB
Mbps	7,2	7,2	7,2	7,2	7,2
Digitalni servisi	-	-	-	-	-
Din	595	895	1.195	1.595	2.995

Višedimenzionalni nizovi

LINKgroup

Kreiranje višedimenzionalnog niza

(jcex102014 MultiArray)

- Višedimenzionalni niz se takođe može kreirati pomoću inicijalizatora ili dodavanjem članova
- Sledeća linija kreira jedan multidimenzionalni niz celobrojnih vrednosti:

```
int[][] multiArray = new int[3][];
```

Glavni niz je inicijalizovan

Podnizovi nisu inicijalizovani

Može doći do „nazupčenja“

Može i ovako:

```
int[][] multiArray = new int[3][3];
```

Glavni niz je inicijalizovan

Podnizovi JESU inicijalizovani

Kreiranje višedimenzionalnog niza

- Kreiranje višedimenzionalnog niza pomoću inicijalizatora je pregledno i jednostavno:

```
int[][] multiArray = {  
    {2, 5, 8},  
    {1, 4, 3}  
};
```

- Višedimenzionalni niz može imati neograničen broj dimenzija (ne baš bukvalno neograničen, već do 255), ali verovatno gotovo nikada nećemo kreirati više od tri, dok je najčešći scenario upotreba dve dimenzije

```
int[][][][] multiArray = {  
    {{{1, 2, 3}}, {{1, 2, 3}}}  
};
```

Redak scenario

```
int[][][] multiArray = {  
    {{1, 2, 3}, {1, 2, 3}}  
};
```

Češći scenario

Nazupčeni niz

- Podnizovi višedimenzionalnog niza ne moraju biti svi iste veličine
- Višedimenzionalni niz koji ima podnizove različite veličine, naziva se **nazupčeni niz**

```
int[][] multiArray = {  
    {2, 5, 8},  
    {1, 4, 3, 6, 10}  
};
```

Inicijalizacija podnizova bez inicijalizatora

- Inicijalizator je pogodan kada znamo koji će tačno biti sadržaj višedimenzionalnog niza. U suprotnom, nizove ćemo verovatno inicijalizovati programabilno. Na primer, na sledeći način (ili kroz njegove varijacije):

```
int[][] multiArray = new int[3][];  
multiArray[0] = new int[3];  
multiArray[1] = new int[3];  
multiArray[2] = new int[3];
```

Preuzimanje vrednosti članova iz višedimenzionalnog niza

- Vrednosti se iz višedimenzionalnog niza mogu preuzeti na sličan način kao i iz jednodimenzionalnog, pri čemu svake nove uglaste zagrade predstavljaju jednu dimenziju:

```
int[][] multiArray = {  
    {2, 5, 8},  
    {1, 4, 3}  
};  
System.out.println(multiArray[1][2]);
```

3

Md. niz kao tabela

- Višedimenzionalni niz je odličan nosilac tabelarnih podataka
- U sledećem primeru, nalaze se dve tabele:

```
String[][] players = {  
    {"1", "Peter", "Jackson", "150.51"},  
    {"2", "Sally", "Jones", "232.12"},  
};
```

```
String[][] heroes = {  
    {"1", "Diablo", "100"},  
    {"2", "Raynor", "200"},  
};
```

Niz kao nosilac kompleksnih podataka

- Niz može poslužiti za skladištenje kompleksnih podataka. Na primer, jedna tačka u 2d prostoru se može zapamtiti u nizu od dva člana (jedan član za x i drugi za y koordinatu)
- Tako jednu seriju tačaka možemo realizovati kroz višedimenzionalni niz

```
int[][] square = {  
    {10,10},{100,10},{100,100},{10,100}  
};
```


Asocijativni niz

- Postoji posebna vrsta niza u kojoj se pozicija ne označava indeksom, već nekim kodom (stringom, brojem ili čak objektom)
- Ovo nije ni standardan jednodimenzijski indeksirani niz, a ni višedimenzijski. Već se ovakav niz naziva **asocijativnim**
- U Javi ne postoji asocijativni niz kao pojam, ali postoje tehnologije kojima se on jednostavno ostvaruje

Automatski prolazak kroz višedimenzionalni niz

- Za automatski prolazak kroz višedimenzionalni niz koristimo ugnježdene petlje ili eventualno samo jednu petlju, ako imamo fiksni, i dobro poznati sastav podnizova

```
String[][] players = {  
    {"1", "Peter", "Jackson", "150.51"},  
    {"2", "Sally", "Jones", "232.12"},  
};  
  
System.out.println("id\tname\tsurname\tbalance");  
for(int i=0;i<players.length;i++){  
    System.out.print(players[i][0]+"\t");  
    System.out.print(players[i][1]+"\t");  
    System.out.print(players[i][2]+"\t");  
    System.out.print(players[i][3]+"\n");  
}
```

Automatski prolazak kroz višedimenzionalni niz

- Ako ne znamo strukturu podniza, tada kroz oba (ili sve ostale nizove) moramo proći automatski:

```
String[][] person_attributes = {
    {"good", "smart", "pretty", "funny"},
    {"bad", "smart", "pretty"},
    {"ugly"}
};
for(int i=0; i<person_attributes.length; i++){
    System.out.println("***** Person "+i+" *****");
    for(int j=0; j<person_attributes[i].length; j++){
        System.out.print(person_attributes[i][j]+"\\t");
    }
    System.out.println();
}
```

Vežba 1 (jcex102014 MatrixAddition)

- Najbolji način za upoznavanje sa višedimenzionalnim nizovima jeste obrada matrica. Pokušajmo da saberemo dve matrice (A i B) i rezultat smestimo u treću matricu (C) pomoću višedimenzionalnih nizova

- Rešenje:

jcex102014 MatrixAddition

```
int[][] A = {  
    {2, 4, 5},  
    {1, 3, 7},  
    {6, 2, 8}  
};  
int[][] B = {  
    {1, 3, 1},  
    {8, 9, 4},  
    {5, 3, 2}  
};  
int[][] C = new int[3][3];
```

Vežba 2 (jcex102014 DisplayMatrix)

- Date su tri promenljive. Celi brojevi width i height, i niz points. Niz points je dvodimenzionalan i sadrži podnizove fiksnog broja članova koji predstavljaju 2D tačke koordinatnog sistema (x i y)

```
int width = 20,height = 10;  
int[][] points = {{2,4},{1,5},{6,6},{3,2},{0,0}};
```

- Potrebno je na osnovu datih informacija, nacrtati u konzoli prikaznu matricu koja će kao osnovu (jedan piksel) imati znak O
- Za sve koordinate koje su navedene u nizu points, treba iscrtati odgovarajuće tačke na prikaznoj matrici, pomoću karaktera X
- Konačan izlaz programa za podatke iz postavke treba da izgleda kao na slici desno

```
XOOOOOOOOOOOOOOOOOOO  
OOOOOOOOOOOOOOOOOOO  
OOOXOOOOOOOOOOOOOOO  
OOOOOOOOOOOOOOOOOOO  
OOXOOOOOOOOOOOOOOOO  
OXOOOOOOOOOOOOOOOOO  
OOOOOOXOOOOOOOOOOOO  
OOOOOOOOOOOOOOOOOOO  
OOOOOOOOOOOOOOOOOOO  
OOOOOOOOOOOOOOOOOOO
```

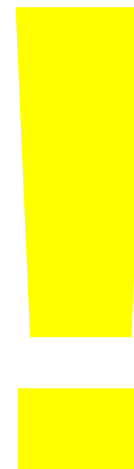
Vežba 3 (jcex102014 WindowsGraphic)

- U prethodnoj vežbi kreirali smo sopstvenu ekransku matricu
- U sledećoj vežbi, možemo „prepakovati“ rešenje tako da koristi gotov Java ekranski sistem (awt biblioteku)

```
public class WindowGraphic extends Panel {  
    int[][] points = {{20,40},{10,50},{60,60},{30,20},{0,0}};  
    @Override  
    public void paint(java.awt.Graphics g) {  
        super.paint(g);  
        Graphics2D g2 = (Graphics2D)g;  
        g2.setColor(Color.red);  
        for(int i=0;i<points.length;i++){  
            g2.fillRect(points[i][0], points[i][1], 10, 10);  
        }  
    }  
    public static void main(String[] args) {  
        Frame f = new Frame();  
        f.add(new WindowGraphic());  
        f.setSize(300,300);  
        f.setVisible(true);  
    }  
}
```

Kolekcije (jcex10 2014 SimpleList)

- Pokušajmo da napravimo sopstveni tip podatka koji će biti sličan nizu, ali će imati mogućnost da prihvati neograničen broj članova
- Ovaj tip koristio bi se za skladištenje isključivo celih brojeva

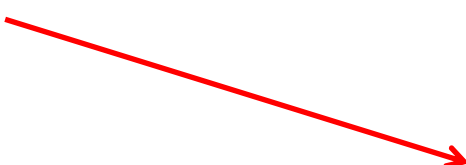


Iterator

- Iterator je sistem koji omogućava sekvencijalno preuzimanje elemenata nekog objekta
- Na primer, ako imamo listu brojeva: 1,2,3,4 logično bi bilo da nam iterator eksponira sekvencijalno brojeve 1,2,3,4, ali taj redosled može biti i drugačiji ili vrednosti kroz koje hoćemo da prođemo, nisu u strukturanoj formi
- U pomenutim situacijama možemo kreirati sopstveni sistem za listanje podataka – iterator

Primer implementacije i korišćenja iteratora

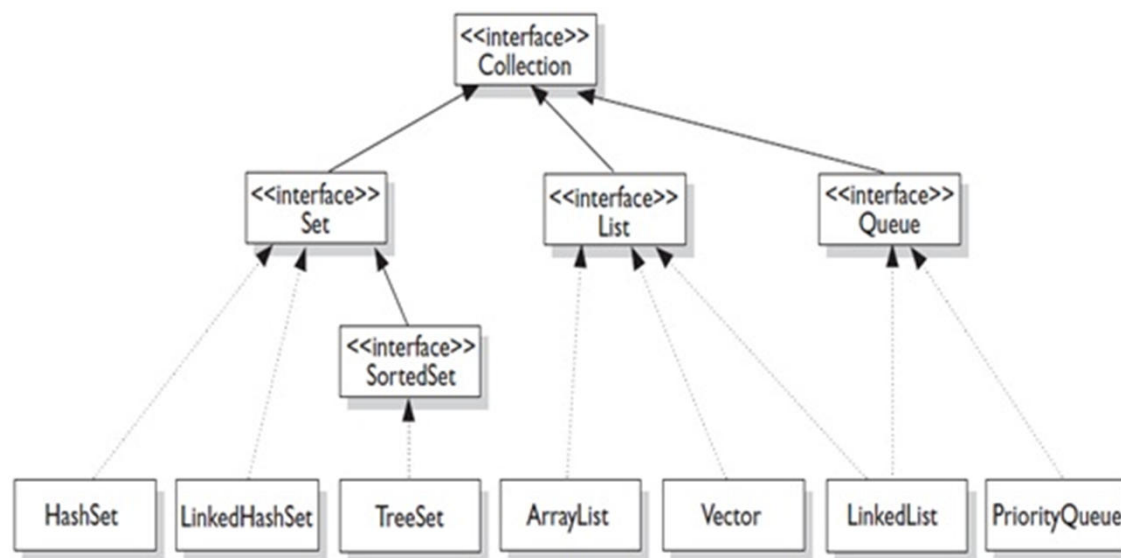
```
public class MyColl implements Iterable<Object>{
    String[] cars = {"bmw", "audi", "mercedes"};
    int current = 0;
    @Override
    public Iterator<Object> iterator() {
        return new Iterator<Object>() {
            @Override
            public boolean hasNext() {
                return current < cars.length;
            }
            @Override
            public Object next() {
                return cars[current++];
            }
        };
    }
}
```



```
public class JavaApplication226 {
    public static void main(String[] args) {
        MyColl col = new MyColl();
        for(Object s : col){
            System.out.println(s);
        }
    }
}
```

Kolekcije

- Ako ste rešili problem sa prethodnog slajda, najverovatnije ste napravili neku vrstu ulancane liste
- Java poseduje ugrađene mehanizme koji ovo rade, i oni svi spadaju u **kolekcije**
- Kolekcije su tipovi koji sadrže više podataka
- Kolekcije mogu sadržati različite tipove
- Kolekcije mogu menjati veličinu nakon kreiranja
- Većina kolekcija nalazi se u paketu **java.util**



Java kolekcije

- Java podrazumevano sadrži nekoliko interfejsa koji definišu ponašanje kolekcija, i nekoliko klasa koje implementiraju te interfejse
 - List
 - ArrayList
 - LinkedList
 - Set
 - HashSet
 - EnumSet
 - LinkedHashSet
 - TreeSet
 - Queue
 - Map
 - HashMap
 - Hashtable
 - TreeMap
 - LinkedHashMap

List

- **List** – predstavlja kolekciju podataka koja najviše podseća na niz. Podaci se mogu unositi linearno, mogu se preuzeti pomoću indeksa, ili staviti na odgovarajući indeks. Ova kolekcija slična je primeru koji smo kreirali u slajdu sa početka lekcije
- List je Interfejs, što znači da ga ne možemo direktno instancirati. Umesto toga, koristimo neku od njegovih implementacija: **ArrayList** ili **LinkedList**.
- Razlika između ove dve implementacije je u tome što ArrayList u pozadini koristi nizove, dok LinkedList koristi ulančanu listu (upravo onakvu kakvu smo napravili u primeru)
- Zbog toga je manipulacija ArrayList-om efikasnija i brža od manipulacije LinkedList-om

Kako se koristi List?

- Instanciranje i dodavanje članova listi, može se izvršiti na sledeći način:

```
List heroes_list = new ArrayList();  
heroes_list.add(10);  
heroes_list.add(30);  
heroes_list.add("Diablo");
```

- Priećujemo da u listu možemo da smestimo podatke bilo kog tipa, što u nizu nismo mogli. Ovo ukazuje da unutar klase dolazi do boxing-a
- Nikada ne pokušavamo da instanciramo List, jer je u pitanju interfejs

```
List heroes_list = new List();
```

Za šta može poslužiti List

- List možemo da iskoristimo za situacije u kojima nismo sigurni sa koliko ćemo podataka raditi
- U najvećem broju slučajeva, List može da nam posluži kao zamena za niz
- Na primer, u chat aplikaciji bi mogli imati određeni broj korisnika koji se konstantno menja. Za skladištenje korisnika u memoriji ArrayList je odlično rešenje

Primer: Pamćenje korisnika u listi

- U sledećem primeru kreirana je lista stringova

```
List cl_users = new ArrayList();
cl_users.add("Peter");
cl_users.add("Sally");
cl_users.add("John");
cl_users.set(1, "Sam");
for(int i=0;i<cl_users.size();i++){
    System.out.println(cl_users.get(i));
}
```

- Primećujemo da se:
 - za dužinu ne koristi svojstvo `length`, već **size**
 - da se za pristup članu ne koriste uglaste zagrade već metod **get**
 - Da se za modifikaciju člana koristi metod **set**

Vežba

- Potrebno je kreirati klasu koja će nositi podatke o jednoj tački (klasa treba da ima polja x i y)
- Potrebno je kreirati listu tačaka i popuniti je tačkama (proizvoljnim)
- Potrebno je kreirati klasu Bounds koja će sadržati podatke o nekoj zoni ($x, y, width$ i $height$), a zatim instancirati ovu klasu
- Potrebno je proći kroz listu tačaka i prikazati samo one koje se trenutno nalaze u zoni

Set

- Kod Set-a jedna vrednost ne može se ponoviti
- Takođe, nije moguć direktan pristup elementima, već se jedino može dobiti iterator, a zatim izvršiti prolazak kroz članove u potrazi za željenim članom

```
System.out.println("***** HashSet *****");
Set users_set = new HashSet();
users_set.add("Peter");
users_set.add("Sally");
users_set.add("Peter");
Iterator iter = users_set.iterator();
while(iter.hasNext()){ System.out.println(iter.next()); }
for(Object user : users_set){ System.out.println(user); }
```

Map

- Mape su kolekcije u kojima se pozicije članova određuju ključevima
- Mape su nešto najbliže asocijativnom nizu u Javi
- Postoji više implementacija Map-a u Javi, i one se većinom razlikuju po performansama i mogućnosti sortiranja članova
- Najbrža Map implementacija je HashMap, ali ona nema mogućnost sortiranja

```
System.out.println("***** HashMap *****");
HashMap users_map = new HashMap();
users_map.put("user1", "Peter");
users_map.put("user2", "Sally");
users_map.put("user3", "John");
System.out.println("user1 is " + users_map.get("user1"));
iter = users_map.entrySet().iterator();
while(iter.hasNext()){
    Map.Entry kvp = (Map.Entry)iter.next();
    System.out.println(kvp.getKey()+":"+kvp.getValue());
}
```

Vežba

- Potrebno je kreirati program koji će prilikom startovanja tražiti od korisnika da unese oznaku jezika. Nakon odabira jezika, program prikazuje pozdravnu poruku na jeziku korisnika.

```
run:  
Choose language: it  
Benvenuti  
BUILD SUCCESSFUL (total time: 2 seconds)
```

Vežba

- Potrebno je kreirati program koji će prilikom startovanja tražiti od korisnika da unese oznaku jezika.
- Nakon odabira jezika, program prikazuje pozdravnu poruku na jeziku korisnika.
- Program zatim traži od korisnika da unese brojeve koje će sabirati (prvo prvi a zatim drugi broj)
- Kada korisnik odabere brojeve, prikazuje se rezultat na izlazu
- SVA PITANJA I PORUKE KOJE PROGRAM PRIKAZUJE KORISNIKU, MOŽAJU BITI NA JEZIKU KOJI JE KORISNIK ODABRAO NA POČETKU PROGRAMA

```
Choose language: sr
Odaberi prvi operand: 2
Odaberi drugi operand: 4
Rezultat je: 6BUILD SUCCESSFUL (total time: 3 seconds)
```

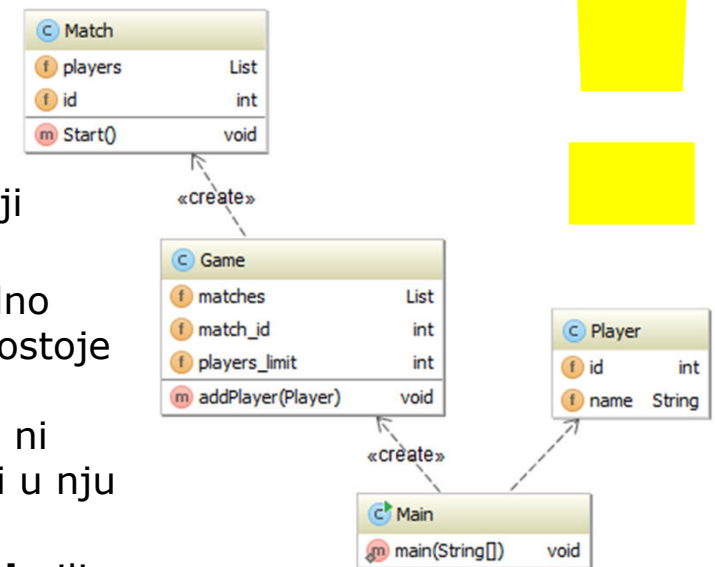
Queue (i Stack)

- Redove koristimo najčešće kada hoćemo da obradimo neke podatke po redosledu po kome su ušli u sistem
- Karakteristika reda je to da se svaki podatak izbacuje iz kolekcije nakon obrade i to po određenom redosledu (FIFO)

```
System.out.println("***** Queue *****");
Queue persons_queue = new LinkedList();
persons_queue.add("Superman");
persons_queue.add("Hulk");
persons_queue.add("Spiderman");
while(!persons_queue.isEmpty()){
    System.out.println(persons_queue.poll());
}
System.out.println("***** Queue *****");
Queue persons_queue = new Queue();
persons_queue.enqueue("Superman");
persons_queue.enqueue("Hulk");
persons_queue.enqueue("Spiderman");
while(!persons_queue.isEmpty()){
    System.out.println(persons_queue.dequeue());
}
```

Vežba 1 (jcex102014 SimpleMatchmaking)

- Multiplayer igra je realizovana pomoću klase **Game**
- Ona ima partije, koje su realizovane kroz klasu **Match**
- Svaka partija ima fiksni broj igrača (za primer, to može biti 1:1 ili 2:2)
- Klasa Game, sadrži sve partije (lista objekata tipa **Match**). Takođe, ona omogućava prihvatanje novih igrača. Metod koji prihvata novog igrača je: **addPlayer(Player)**
- Prilikom aktivacije metode **addPlayer**, proverava se slobodno mesto u postojećim partijama. Ukoliko u nekoj partiji već postoje igrači, dodaje se novi i ukoliko je broj dozvoljenih igrača popunjen, aktivira se metod **Start** za tu igru. Ukoliko nema ni jedne partije sa slobodnim mestima, kreira se nova partija i u nju se smešta novi igrač
- Unos novih igrača može biti vršen putem konzole (**System.in** ili **Scanner**) iz **main** metode glavne klase aplikacije
- Prilikom konstrukcije rešenja poželjno je koristiti dijagram desno



Vežba 2 (jcex102014 MoreSimpleMatchmaking)

- U programu se nalaze dve promenljive: **players_limit (int)** i **games (ArrayList)**.

```
int players_limit = 3;  
List games = new ArrayList();
```

- Potrebno je kreirati program koji će beskonačno prihvatati korisnički unos stringova i na osnovu tog unosa formirati liste igrača za svakog člana games liste, tako da u svakoj igri (članu games liste), bude ukupno po onoliko igrača koliko je određeno promenljivom players_limit.

Vežba 3 (jcex102014 AddNumbers)

- Potrebno je kreirati program koji će prihvatati korisnički unos sa tastature
- Korisnik treba da unosi brojeve jedan za drugim i pritiska taster enter
- Nakon što korisnik unese prazan string (ne unese ništa i pritisne enter), vrši se prikaz svih brojeva koje je uneo, a zatim se prikazuje njihov zbir

```
run:  
2  
3  
4  
  
2 3 4 Result is: 9
```


Vežba 4 (jcex102014 Users)

Data je klasa User

```
public class User {  
    public int Id;  
    public String ime;  
    public User(int id, String ime)  
    {  
        this.Id = id;  
        this.ime = ime;  
    }  
}
```

Takođe, u Main metodi aplikacije postoji sledeći kod:

```
ArrayList usersToAdd = new ArrayList();  
usersToAdd.add(new User(4, "Goran"));  
usersToAdd.add(new User(2, "Ilija"));  
usersToAdd.add(new User(6, "Nikola"));  
  
ArrayList users = new ArrayList ();  
users.add(new User(1, "Petar"));  
users.add(new User(2, "Jovan"));  
users.add(new User(3, "Zoran"));
```

Potrebno je sve korisnike iz liste usersToAdd dodati u listu Users. Ukoliko već postoji korisnik sa istim ID-om, potrebno je zameniti starog korisnika novim korisnikom.

Potrebno je prikazati listu Users.

Vežba 5 (jcex102014 SortUsers)

- Na osnovu postojeće klase User (iz prethodne vežbe) kreirana je sledeća lista.

```
ArrayList users = new ArrayList();  
users.add(new User(4, "Goran"));  
users.add(new User(2, "Ilija"));  
users.add(new User(6, "Nikola"));  
users.add(new User(1, "Petar"));  
users.add(new User(2, "Jovan"));  
users.add(new User(3, "Zoran"));
```

- Potrebno je sortirati korisnike po ID-u i emitovati listu na izlaz.

Vežba 6 (jcex102014 RockPaperScissors)

- Svi znamo za igru: kamen papir makaze
- Kamen je jaci od makaza, makaze od papira, a papir od kamena
- Pokušajte da napravite ovakvu igru u kojoj bi se korisnik nadmetao sa računarom
- Korisnik bi imao mogućnost da odabere jedan od brojeva (0,1 ili 2), a nakon odabira, računar bi odabrao svoj broj (po slučajnom izboru)
- Svaki broj bi predstavljao jedan artefakt (0 kamen, 1 makaze a, 3 papir)
- Kada se završi odabir, računar treba da vidi ko je bolji i prikaže rezultat u memoriji
- Bilo bi sjajno da igrač, umesto da odabere jedan od tri broja, odabere četvrti, kojim bi prikazao statistike dosadašnjih igara (na primer, broj 3)
- Za dobijanje slučajnog broja u Javi možemo koristiti klasu Random:

```
Random rand = new Random();  
rand.nextInt(3); //daje slučajan broj od 0 do 2
```

Zadatak – program Biblioteka

- Program treba da sadrži tri klase:
 - UI – U ovoj klasi će biti preuziman korisnički ulaz i na osnovu njega biti vraćan određeni rezultat. Na primer, klasa može imati metod: **mainMenu()** koji će na izlazu ispisati glavni meni i uzeti od korisnika odabranu stavku
- ```
run:
1 Add book, 2 Show all books, 3 Remove book, 4 Show book details
Choose option: |
```
- Library – klasa koja se bavi knjigama. Treba da sadrži listu knjiga (kolekcija objekata klase Book), kao i elementarne metode za rad sa knjigama
  - Book – Klasa predstavlja knjigu u sistemu. Mora da ima podatke o autoru, naslovu i isbn-u