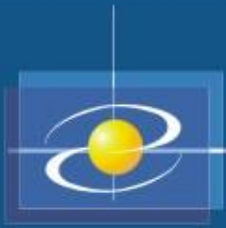


Programski jezik JAVA

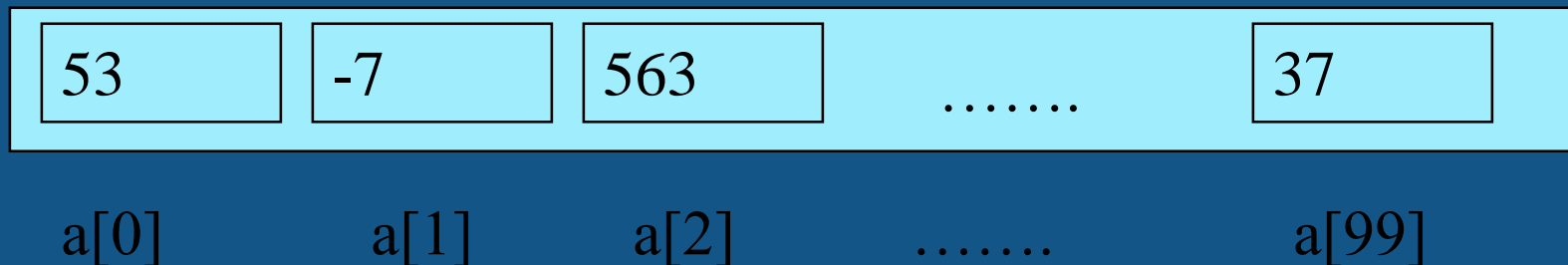
PREDAVANJE 3

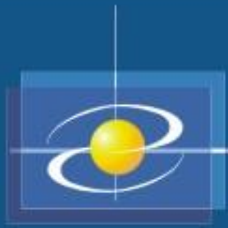
Prezentacija kreirana na osnovu sljedeće literature :
Dejan Živković: Osnove Java programiranja; Bruce Eckel: Misliti na Javi



Nizovi

- Objekti koji predstavljaju nizove promjenljivih istog tipa (primitivnog ili objektnog)
- Pojedinačne promjenljive niza se nazivaju **elementi** niza
- Svaki element niza ima redni broj koji se naziva **indeks** elementa
- Indeksi niza počinju od 0, ne od 1
- Primjer: niz “a” sadrži 100 cjelobrojnih elemenata (promjenljivih):





Nizovi

- **Definisanje nizova**

```
tip[] ime-niza;
```

- **Primjer:**

```
int[] a;
```

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

- **Skraćeni oblik:**

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

```
float[] b = new float[20];
```

```
Auto[] parking = new Auto[5];
```

- **Istovremena inicijalizacija:**

```
int[] dum = {31,28,31,30,31,30,31,31,30,31,30,31};
```

- Nizovi imaju fiksnu dužinu koja se ne može mijenjati



Nizovi

- Korišćenje elemenata nizova:

ime-niza[indeks]

indeks – cjelobrojna promjenljiva ili izraz

- **Primjer:**

```
public static void main(String[] args)
{
    int[] brojač = new int[10];
    String[]
    godDoba={"proljeće","ljeteto","jesen","zima"};
    ...
    brojač[5] = 2*i+1;
    brojač[10] = 17; //greška
    int x = brojač[i+j];
    String s = godDoba[3];
    ...
}
```



Nizovi

- Svaki niz ima definisano konstantno polje **length** koje sadrži dužinu niza (broj elemenata)
- Primjer:

```
int[] brojač = new int[10];  
String[] godDoba=  
    {"proljece", "ljeteto", "jesen", "zima"};  
...  
System.out.println(brojač.length); //10  
System.out.println(godDoba.length); //4
```

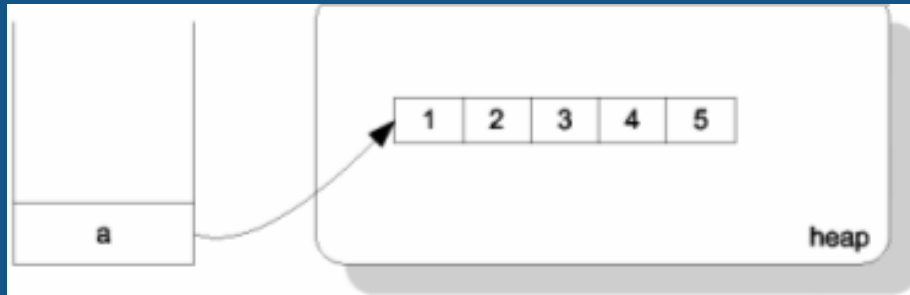
- Kreiranje nizova čiji su elementi primitivni tipovi:

```
int[] a = new int[5];  
for (int i = 0; i < a.length; i++)  
    a[i] = i+1;  
// int[] a = {1,2,3,4,5};  
(System.out.println(Arrays.toString(a)))
```

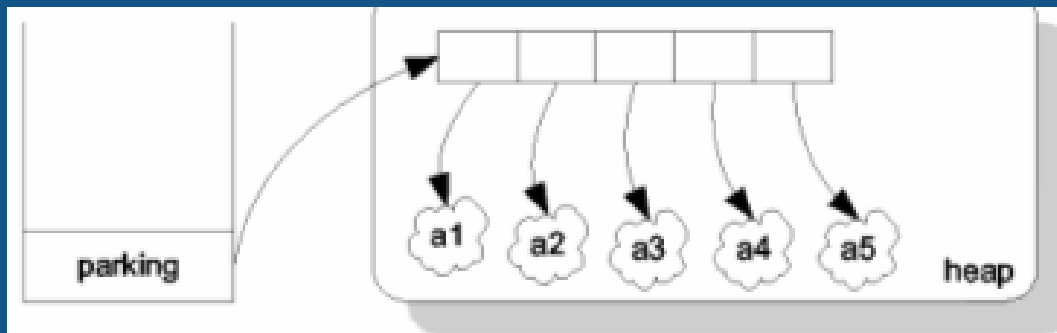


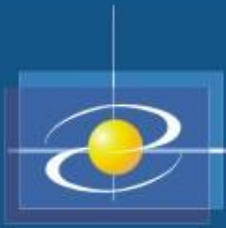
Nizovi

- Stanje u memoriji poslije inicijalizacije:



- Kreiranje nizova čiji su elementi objektnih tipova
`Auto[] parking = new Auto[5];`
To je niz čiji elementi sadrže reference na objekte klase Auto:





Kopiranje nizova

- **Primjer deklarisanja niza:**

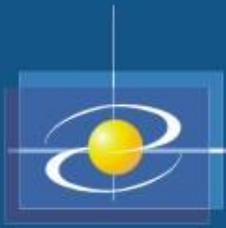
```
double[] x = new double[20];  
double[] y;  
y=x;
```

ovom naredbom samo je kreirana nova referenca na isti niz koji se fizički nalazi na prethodnoj lokaciji.

- **Kopiranje nizova:**

```
public static double[] kopirajNiz(double[] original) {  
    if (original == null)  
        return null;  
    double[] kopija = new double[original.length];  
    for (int i = 0; i < original.length; i++)  
        kopija[i] = original[i];  
    Return kopija;  
}
```

- `y=kopirajNiz(x);`



Nizovi

- Objektnu promjenljivu možemo eksplicitno inicijalizovati sa `null`, što znači da promjenljiva još uvijek ne referencira na neki objekat. Na primjer:

```
int[ ] C = null;
```

- `null` zatim koristimo za ispitivanje da li objektna promjenljiva referencira na neki objekat:

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

- Sve promjenljive članice se automatski inicijalizuju sa `null`.
- Lokalne promjenljive MORAMO sami inicijalizovati.
- Niz ne predstavlja *klasu* i zato nema ugrađenih metoda.
- Na primjer, `length` je promjenljiva, a ne metoda (pišemo `A.length` a ne `A.length()`) i dužinu niza nije moguće dinamički mijenjati.
- Metode za rad sa poljima mogu se naći u klasi: `java.util.Arrays`.
- Sve metode u `Arrays` su statičke, tako da se koriste bez instanciranja objekta tipa `Arrays`.

Naredba for-each

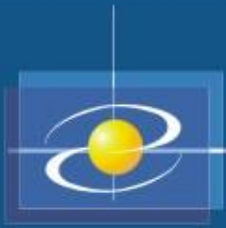


- Umjesto standardne for petlje pri radu sa nizovima često se koristi naredba for-each

```
for (bazni tip element : niz){  
    naredbe ...  
}
```
- **Primjer 1** - (e uzima sekvencijalno svaku vrijednost iz niza a):

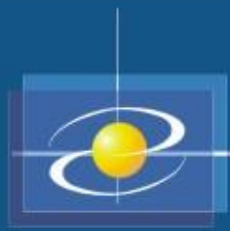
```
for (int e : a) {  
    System.out.println( e );  
}
```
- **Primjer**

```
int zbiri = 0;  
for (int e : a) {  
    if ( e > 0)  
        zbiri = zbiri + e;  
}
```



Klasa Arrays

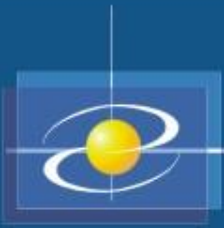
- **String toString(*tip[]* a)** vraća reprezentaciju niza a u formi stringa.
- ***tip[]* copyOf(*tip[]* a, int d)** vraća novu kopiju niza a dužine d
- ***tip[]* copyOfRange(*tip[]* a, int A, int B)** kopira niz a od indeksa A do indeksa B u novi niz.
- **void sort(*tip[]* a)** sortira niz a u rastućem poretku
- **boolean equals(*tip[]* a, *tip[]* b)** – vraća true ako su nizovi iste dužine i imaju iste vrijednosti svih elemenata, u suprotnom vraća false
- **void fill(*tip[]* a, *tip* v)** – dodjeljuje svim elementima niza a vrijednost v.



Dvodimenzioni nizovi

- `int [][] a`

5	-7	1562	371
<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][2]</code>	<code>a[0][63]</code>
-433	344	1	0
<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[1][2]</code>	<code>a[1][63]</code>
3	-71	63	7
<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[2][2]</code>	<code>a[2][63]</code>

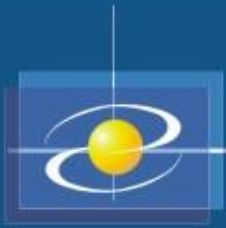


Dvodimenzionni nizovi

- Posmatrajmo tabelu koja u vrstama sadrži podatke za firmu koja ima 10 prodajnih mjesta, a u kolonama su podaci o profitu za svaki mjesec
`double[][] profit = new double[10][12];`

```
public double godišnjiProfit() {  
    double ukupniProfit = 0;  
    for (int i = 0; i < 10; i++)  
        for (int j = 0; j < 12; j++)  
            ukupniProfit += profit[i][j];  
    return ukupniProfit;  
}
```

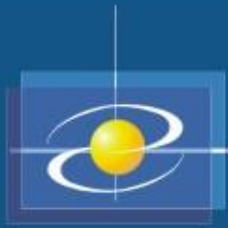
```
private double profitZaMjesec(int m) {  
    double mjesečniProfit = 0;  
    for (int i = 0; i < 10; i++)  
        mjesečniProfit += profit[i][m-1];  
    return mjesečniProfit;  
}
```



Stringovi

- Stringovi predstavljaju nizove znakova između navodnika " "
- Stringovi nijesu primitivne vrijednosti, već objekti klase **String**
- Primjeri:

```
String s = "Ovo je neki tekst";  
String s = new String("Ovo je neki tekst");  
String t = ""; //prazan string  
String t = new String(); //prazan string  
String a = null;  
String[] boje={"crvena","plava","zelena","zuta"};
```



Stringovi

- Operator + omogućava spajanje stringova
- Primjeri:

```
String s = "Ovo je cas " + "iz Jave";
```

```
String s1 = "Ovo je cas ";
```

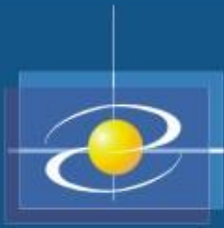
```
String s2 = "iz Jave";
```

```
s1 += s2; //s1 = s1 + s2
```

```
int x = 23;
```

```
System.out.println("Vrijednost x je: " + x);
```

```
//automatski 23 → "23", pa spajanje stringova
```



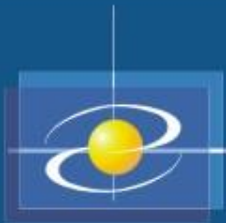
Array, Character, String i StringBuffer klase

- Java 2 platforma enkapsulira funkcionalnost rada sa nizovima karaktera u tri osnovne klase:
- **java.lang.Character**
 - klasa koja enkapsulira rad sa primitivnim tipom podatka char (može sadržati samo jedan karakter)
- **java.lang.String**
 - klasa koja enkapsulira rad sa konstantnim nizom karaktera (dakle, karakteri se ne mijenjaju)
- **java.lang.StringBuffer**
 - kao String, ali uključuje metode za promjene pojedinih karaktera stringa
- Primitivni tip podatka char je Unicode karakter, tj. 2 byte-ni karakter
- ASCII je zastario za današnje potrebe



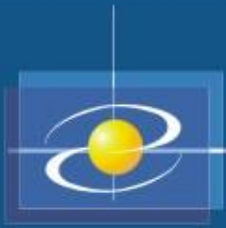
Karaktereri

- Konstruktor klase Character ima sintaksu `Character(char)`
- Neke korisne metode u klasi Character:
- `public char charValue()`
 - vraća vrijednost karaktera kojeg klasa enkapsulira
- `public static int getNumericValue(char ch)`
 - vraća numeričku (Unicode) vrijednost karaktera kao nenegativan cijeli broj; -1 ako karakter ne posjeduje numeričku vrijednost; -2 ako vrijednost nije cijeli nenegativan broj
- `public int compareTo(Character anotherCharacter)`
 - vraća broj <0 ako je karakter kojeg klasa enkapsulira numerički manji od `anotherCharacter`, =0 ako je jednak i >0 ako je veći
- `public String toString()`
 - vraća string koji se sastoji samo od tog karaktera, prerađuje istoimenu metodu klase `Object`
- `public static boolean isUpperCase(char ch)`
 - vraća `true` ili `false` u zavisnosti da li je `ch` veliko ili malo slovo
- `public static boolean isLowerCase(char ch)`
 - vraća `true` ili `false` u zavisnosti da li je `ch` malo ili veliko slovo



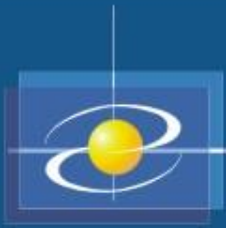
String i StringBuffer

- Postoji nekoliko načina konstrukcije Stringa.
- Npr. želimo napraviti klasu koja enkapsulira string "abc". To možemo napraviti na sledeća tri načina:
 1. `String x="abc";`
 2. `String x=new String("abc");`
 3. `char[] ch={'a', 'b', 'c'};`
`String x= new String(ch);`
- Java ima poseban operator **+** za konkatenciju stringova i konverziju drugih objekata u stringove -to smo već više puta koristili u `System.out.println`.
- Objekti koji reprezentiraju klasu `StringBuffer` moraju se obavezno kreirati sa `new`.



String i StringBuffer

- Neke metode rada sa String i StringBuffer su navedene ovdje:
- **int length()**
 - vraća dužinu (u karakterima) stringa ili string buffera
- **int capacity()**
 - samo za StringBuffer; ukupna količina memorije (izražena u broju karaktera) alocirana za dati buffer
- **char charAt(int index)**
 - karakteri u stringu ili string bufferu su numerisani od 0 do `length() - 1`; metoda vraća karakter na mjestu `index`
- **String substring(int start)**
 - vraća string koji je podstring polaznog stringa koji počinje sa karakterom indeksa `start` i proteže se do kraja
- **String substring(int start, int end)**
 - vraća string koji je podstring polaznog stringa koji počinje sa karakterom indeksa `start` i proteže se do karaktera `end-1`



Stringovi

- Metod `equals` – za poređenje da li dva stringa imaju isti sadržaj:

`s1.equals(s2)`

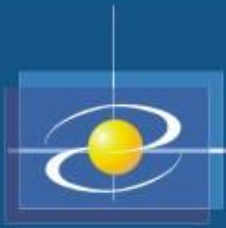
- Primjeri:

```
String s1="Zdravo";  
String s2="Zdravo";
```

```
if(s1 == s2) //true  
if(s1.equals(s2)) //true
```

```
String s1=new String("Zdravo");  
String s2=new String("Zdravo");//novi string
```

```
if(s1 == s2); //false  
if(s1.equals(s2)) //true
```



Stringovi

- Ostali metodi klase **String**

```
String name = "Joe Smith";  
name.toLowerCase();    // "joe smith"  
name.toUpperCase();    // "JOE SMITH"  
" Joe Smith ".trim();  // "Joe Smith"  
"Joe Smith".indexOf('e');    // 2  
"Joe Smith".length();      // 9  
"Joe Smith".charAt(5);      // 'm'  
"Joe Smith".substring(5);   // "mith"  
"Joe Smith".substring(2,5); // "e S"
```