

\$ OOP 03

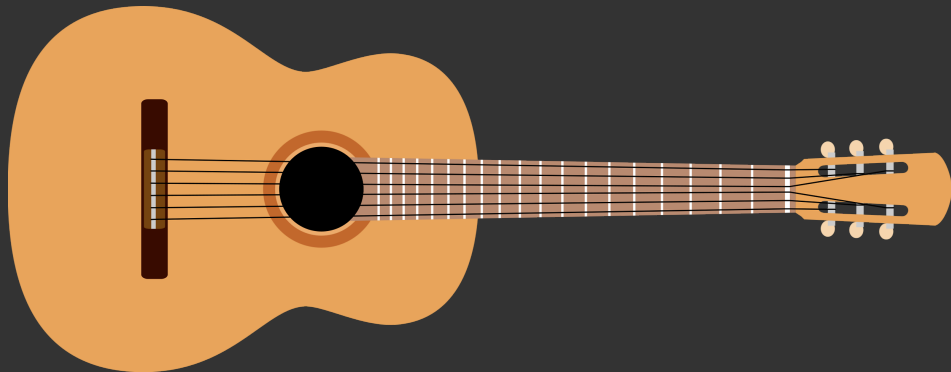
\$ Stringovi

Name: Ognjen Milinković[†]

Date: 8. mart 2021.

[†]ognjen_milinkovic@matf.bg.ac.rs

> Koliko stringova vidite?



§ Sadržaj

1. Stringovi

- String u jeziku C

- Niz karaktera u Javi

- Klasa String

- Imutabilnost klase String

- Klase StringBuilder i StringBuffer

§ Obrada stringova

- > Jedna od najčešćih operacija u programiranju
- > Očekujemo da brzo završimo obradu i nastavimo dalje
- > Tekstuelna reprezentacija podataka je ljudima čitljiva

§ String i C

> Kako je realizovan tip String u jeziku C?

§ String i C

- > Kako je realizovan tip String u jeziku C?
- > nikako

§ String i C

- > Kako je realizovan tip String u jeziku C?
- > nikako
- > String emuliramo koristeći niz karaktera

§ Niz karaktera u Javi

> Niz karaktera možemo konstruisati i u Javi

```
public static void main(String[] args) {  
    char[] t = {'Z', 'd', 'r', 'a', 'v', 'o', '.'};  
    for (char e: t)  
        System.out.print(e);  
    System.out.println();  
}
```

> Primetimo da ne postoji završna nula

> U pitanju je niz tako da nemamo previše funkcija za njegovu obradu

§ Klasa String

- > Java poseduje ugrađenu klasu String
- > Tipove delimo na `primitivne` i `klasne`
- > String je nekako između ta dva
 - * Dozvoljava se sintaksa dodele kao kod primitivnih tipova
 - * `String t = "Zdravo svima :)";`
- > Ipak, String je zvanično `klasni` tip

§ Kako pravimo String

> Dozvoljene su dve sintakse

```
public static void main(String[] args) {  
    // Objektna sintaksa  
    String s1 = new String("Zdravo studenti!");  
  
    // Sintaksa koja je prirodna ljudima  
    String s2 = "Zdravo studenti!";  
  
    System.out.println(s1);  
    System.out.println(s2);  
  
    System.out.println("Duzina s1: " + s1.length());  
    System.out.println("Duzina s2: " + s2.length());  
  
    // Ne dozvoljava se pristup kao kod niza sa s1[3]  
    System.out.println("s1[3] = " + s1.charAt(3));  
}
```

§ Neke metode klase String

Neke od korisnih metoda¹:

- > `charAt(int i)`
 - * vraća karakter na indeksu `i`
- > `length()`
 - * vraća dužinu stringa
- > `compareTo(String s)`
 - * leksikografski poredi sa stringom `s`
- > `equals(String s)`
 - * poredi string sa stringom `s`
- > `toArray()`
 - * vraća niz karaktera

¹Tipovi su negde uklonjeni radi ilustracije, pogledati dokumentaciju detaljnije

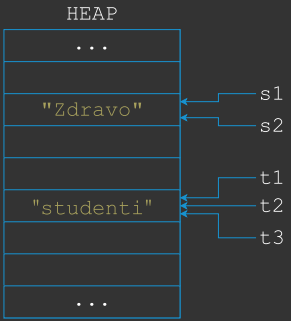
§ Imutabilnost stringa

- > Stringovi u Javi su `imutabilni`²
- > Alociraju se na hip memoriji
- > Odnosno, `string` u Javi predstavlja `konstantu`

²Nepromenljivi

§ Imutabilnost stringa - primer

```
public static void main(String[] args) {  
    String s1 = "Zdravo";  
    String s2 = "Zdravo";  
    String t1 = "studenti";  
    String t2 = t1;  
    String t3 = "studenti";  
}
```



§ Imutabilnost stringa - dobre strane



- > Potencijalno možemo uštedeti na memoriji
- > Usled imutabilnosti, sigurni smo da string neće biti promenjen
- > Time omogućavamo da jedna stringovna konstanta bude više puta referencirana

§ Imutabilnost stringa - loše strane



- > Izmene stringa zahtevaju alokaciju memorije u pravljenju novog stringa
- > Naše operacije su često takve da želimo izmeniti originalni string bez ikakvih kopiranja

§ Kako protiv novih stringova

- > Pretpostavimo da želimo da generišemo string koji sadrži 10000 jedinica

```
public static void main(String[] args) {  
    String nums = "";  
    for (int i = 0; i < 10000; i++)  
        nums = nums + 1;  
    System.out.println(nums);  
}
```

- > Potencijalno³ 10000 puta alociramo novi string
- > Na kraju petlje, za sobom smo ostavili 9999 stringova koji su dalje nepotrebni
- > Grubom i optimističnom računicom, potrošili smo oko 47 megabajta!

³Za slučaj da kompilator ne optimizuje

§ `StringBuilder`

- > `StringBuilder` predstavlja klasu kojom rešavamo prethodno izneti problem
- > Koristimo ga kada želimo da vršimo neku modifikaciju stringa ili ga gradimo dodavanjem drugih stringova
- > Omogućava nam udobne funkcije za modifikaciju stringa

§ StringBuilder - primer

```
public static void main(String[] args) {  
    // Pravimo objekat klase StringBuilder  
    // (sintaksa slicna kao za Scanner)  
    StringBuilder sb = new StringBuilder();  
  
    // Koristimo metod append koji dodaje  
    // u element na kraj stringa unutar 'sb' objekta  
    for (int i = 0; i < 10000; i++)  
        sb.append(1);  
  
    // Izvlacimo string iz StringBuilder objekta  
    String r = sb.toString();  
  
    System.out.println(r);  
}
```

§ Neke metode klase `StringBuilder`

Neke od korisnih metoda⁴:

- > `append(x)`
 - * dodaje `x` u `StringBuilder` objekat
- > `charAt(int i)`
 - * vraća karakter na indeksu `i`
- > `delete(int start, int end)`
 - * briše podstring od `start` do `end`
- > `deleteCharAt(int i)`
 - * briše karakter na indeksu `i`
- > `indexOf(String s)`
 - * vraća indeks prvog pojavljivanja `s` u stringu unutar `StringBuilder`-a
- > `toString()`
 - * vraća string sadržan u `StringBuilder` objektu

⁴Tipovi su negde uklonjeni radi ilustracije, pogledati dokumentaciju detaljnije

§ StringBuffer

- > Nudi istu funkcionalnost kao `StringBuilder`
- > Bezbedan je za rada sa nitima (eng. thread-safe)
- > Niže performanse od klase `StringBuilder` (zbog sinhronizacije potrebne za niti)
- > Na kursu svi programi imaju jednu nit te ćemo koristiti `StringBuilder`

Zahvalnica

Ovu prezentaciju je napravio kolega Nemanja Mićović.
Hvala kolegi Mićoviću što se saglasio da koristimo ovu prezentaciju.