Standardni ulaz i izlaz

Autori:
Goran Savić
Milan Segedinac

1. Klasa String

Objekat klase String sadrži niz karaktera i pruža skup često korišćenih operacija nad ovim nizom.

Objekat klase String je moguće kreirati na dva načina. Prvi način je pozivom konstruktora klase, kao što je prikazano u primeru.

```
String s = new String("Novi Sad");
```

Obzirom da je konstruktor funkcija, ova funkcija može da ima i parametre. Kao što vidimo, parametar konstruktora klase String je vrednost koja treba da bude sadržaj novokreiranog Stringa.

Drugi način kreiranja objekta klase String je korišćenjem literala kao u narednom primeru.

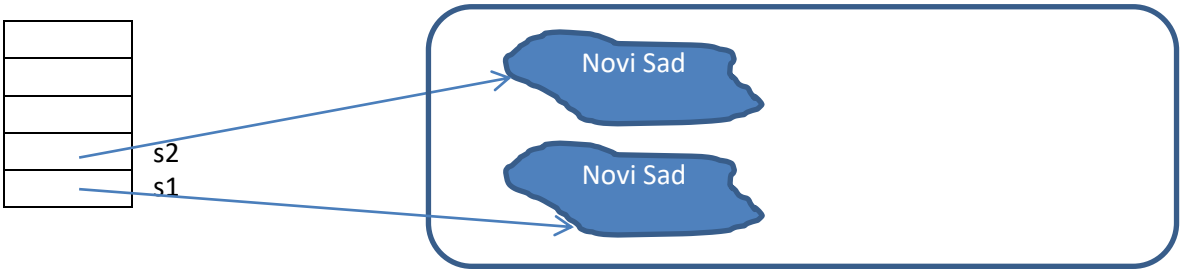
```
String s = "Novi Sad";
```

Prikazani izraz će takođe konstruisati objekat klase String.

Važno je imati u vidu da se, kao i svi drugi adresni tipovi, i String objekti skladište u heap memoriji. Konstruisanje novog Stringa podrazumeva alociranje memorije u heap memoriji i reference na taj objekat u stack memoriji. U skladu sa tim, poređenje string objekata poredi reference na stack memoriji. Dat je primer poređenja dva String objekta.

```
String s1 = new String("Novi Sad");
String s2 = new String("Novi Sad");
System.out.println(s1 == s2);
```

Prikazani primer ispisaće false, jer reference s1 i s2 nisu jednake (ne predstavljaju reference na istu memorijsku lokaciju). Promenljive skladište reference na dve različite memorijske lokacije u heap memoriji. Iako je sadržaj na tim lokacijama isti (Novi Sad), reč je o različitim objektima u memoriji. Ovo je ilustrovano na sledećoj slici.



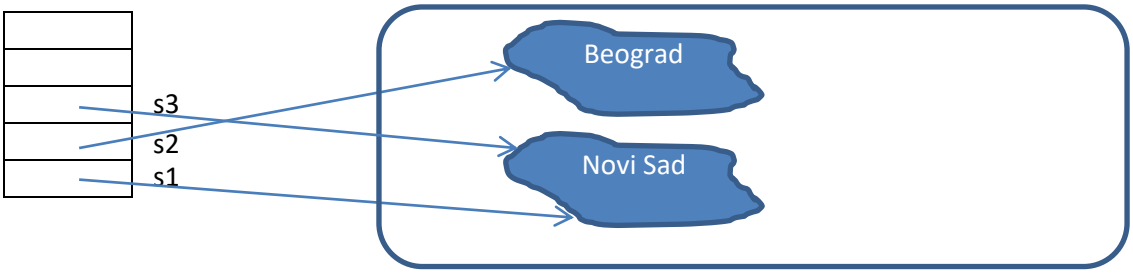
Ukoliko želimo da uporedimo sadržaj stringova, a ne njihovih referenci, tada koristimo equals() metodu klase String. Dat je primer korišćenja ove metode.

```
String s1 = new String("Novi Sad");
String s2 = new String("Novi Sad");
System.out.println(s1.equals(s2));
```

Prethodni primer će ispisati vrednost true, jer će metoda equals da uporedi karaktere koje stringovi sadrže i utvrdiće da su karakteri u oba stringa jednaki.

Kada je reč o String literalima, treba imati u vidu da Java kompajler sve String literale koji imaju istu vrednost predstavlja istim objektom. Dat je primer kreiranja tri Stringa putem literala i izgled memorije nakon kreiranja.

```
String s1 = "Novi Sad";
String s2 = "Beograd";
String s3 = "Novi Sad";
```



Kao što vidimo, promenljive s1 i s3 pokazuju na isti objekat jer koriste isti literal. Posledično, sada izraz `s1 == s3` vraća rezultat true.

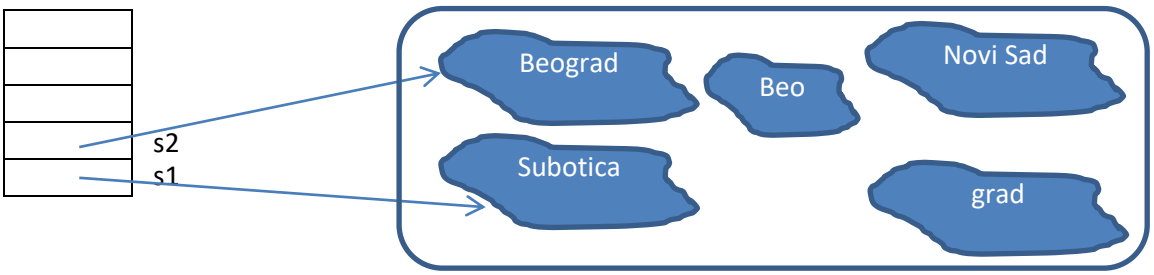
Moguće je vršiti i spajanje (eng. *concatenation*) stringova korišćenjem operatora +. Dat je primer spajanja stringova.

```
String s1 = "Novi";
String s2 = "Sad";
System.out.println(s1 + " " s2);
```

Prikazani primer ispisuje reč Novi Sad jer je izvršeno spajanje stringova.

Važna osobina stringova je da su neizmenjivi (eng. *immutable*). Dodela nove vrednosti stringu ili spajanje stringa uvek iniciraju kreiranje potpuno novog objekta klase String. Dat je primer dodele vrednosti Stringu, kao i spajanja stringova i prikazan je izgled memorije nakon izvršavanja ovih naredbi.

```
String s1 = "Novi Sad";
s1 = "Subotica";
String s2 = "Beo";
s2 += "grad";
```



Ukoliko želimo da proširujemo postojeći niz karaktera drugim karakterima, koriste se StringBuilder ili StringBuffer klasa. Korišćenje ovih klasa je važno sa stanovišta optimizacije zauzeća memorije. Dat je primer spajanja stringa korišćenjem StringBuilder klase.

```
StringBuilder sb = new StringBuilder();
sb.append("Beo");
sb.append("grad");
System.out.println(sb.toString());
```

Prikazani primer kreira samo jedan objekat klase StringBuilder u koji se redom dodaju novi karakteri pozivom metode append(). Moguće je preuzeti karaktere koje StringBuilder skladišti u formi objekta klase String pozivom metode toString() klase StringBuilder.

Operacije nad stringovima

Klasa String pruža metode za manipulaciju nad stringovima. U primeru su prikazane i objašnjene neke često korišćene metode.

```
String s = "Beograd";
int duzina = s.length(); // vraća broj karaktera u Stringu
                         // U ovom slučaju 7
char c = s.charAt(2); // vraća karakter sa prosleđenim indeksom
                      // U ovom slucju karakter o
int i = s.indexOf('g'); // vraća indeks na kojem se prosleđeni
                        // karakter nalazi. U ovom slučaju 3
String s1 = s.substring(2, 5); //vraća podstring između prosleđenog
                               // početnog i krajnjeg indeksa
                               // u ovom slučaju ogr
boolean b = s.startsWith("Beo"); // da li String počinje prosleđenim
                                 // karakterima. U ovom slučaju true
String s2 = s.toLowerCase(); // vraća String pretvoren u mala slova
                             // U ovom slučaju beograd
```

Ako je iz Stringa potrebno izdvojiti delove po nekom kriterijumu, koristi se metoda split(). Ova metoda kao parametar prima String koji predstavlja karaktere (ili šablon karaktera) po kojima je potrebno razdvojiti string. Metoda vraća delove Stringa u formi niza Stringova. Na primer, ako je potrebno u Stringu uočiti delove koji su razdvojeni praznim mestom, to je moguće realizovati sledećim primerom.

```
String s = "Novi Sad";
String[] delovi =  s.split(" ");
```

Niz delovi će sada imati dva elementa. Prvi element će imati vrednost Novi, a drugi element vrednost Sad.

Ostale metode klase String mogu se naći u zvaničnoj dokumentaciji ove klase na adresi https://docs.oracle.com/javase/7/docs/api/java/lang/String.html.

2. Standardni izlaz

Java omogućuje ispis podataka na standardni izlaz (monitor). Dat je primer metoda print() i println() koje se koriste za ispis podataka. Razlika između ove dve metode je u tome što metoda println dodaje i novi red na kraj ispisa.

```
System.out.println("Novi Sad");
System.out.print("Beograd");
```

Pomenućemo još i metodu printf() koja omogućuje formatirani ispis podataka. Metoda kao prvi parametar prima string koji treba ispisati. U stringu je moguće znakom % označiti promenljive koje će dinamički biti ubačene u string. Vrednosti ovih promenljivih se zatim navode kao naredni parametri metode. Dat je primer formatiranog ispisa podataka.

```
System.out.printf("Student %s %s ima prosek ocena %10.3f", ime, prezime, prosek);
```

Kao što vidimo, promenljive se navode znakom % nakon čega sledi oznaka tipa podataka. Npr. %s je oznaka za String, a %f je oznaka za float. Pri ispisu promenljive moguće je specificirati i format ispisa. Tako vidimo da je za float promenljivu označeno da zauzme 10 mesta pri ispisu i da prikaže tri decimale. Kao što smo objasnili, nakon stringa za ispis navode se promenljive koje će redom biti ubačene na mesta označena znakom % u stringu.

3. Standardni ulaz

Kao što možemo ispisivati podatke na standardni izlaz, tako je moguće i preuzeti unos od korisnika sa standardnog ulaza (tastatura). Standardni izlaz je predstavljen statičkim atributom in klase System. Preuzimanje podataka se može izvršiti korišćenjem objekta klase BufferedReader koji pri instanciranju dobija objekat klase InputStreamReader kao parametar. Dat je primer instanciranja objekta klase BufferedReader i preuzimanja podataka sa standardnog ulaza korišćenjem ovog objekta.

```
BufferedReader br = new BufferedReader(new
      InputStreamReader(System.in));
try {
      String s = br.readLine();
} catch (IOException e) {
      e.printStackTrace();
}
```

Metoda readLine() preuzima ulaz sa tastature u formi String objekta. Ova metoda može da izazove izuzetak tipa IOException ukoliko dođe do problema pri preuzimanju podataka sa standardnog ulaza. Iz tog razloga je poziv metode obuhvaćen try ... catch blokom.

String objekat koji je metodom preuzet je moguće konvertovati u različite vrednosti primitivnog tipa. Za svaki primitivni tip podatka postoji takozvana klasa omotač (eng. *wrapper*). Ove klase u sebi sadrže podatak primitivnog tipa i omogućuju operacije nad ovim podatkom. U tabeli je dat pregled često korišćenih primitivnih tipova i naziva odgovarajućih klasa omotača.

Primitivni tip	Klasa omotač
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

Između ostalog, ove klase omogućuju konverziju string vrednosti u vrednost odgovarajućeg primitivnog tipa putem metoda parseXXX. Dat je primer konverzije stringa u int vrednost korišćenjem klase Integer i metode parseInt.

```
int i = Integer.parseInt("23");
```

U promenljivu *i* će biti upisan broj 23.

Kada je reč o *wrapper* klasama za primitivne tipove, pomenimo i da postoji automatska konverzija iz objekta *wrapper* klase u promenljivu odgovarajućeg primitivnog tipa. Ovu operaciju zovemo *unboxing.* Takođe, ukoliko je potrebno, automatski se vrši konverzija promenljive primitivnog tipa u objekat odgovarajuće *wrapper* klase, što je postupak koji se naziva *boxing*. Dati su primeri *unboxing* i *boxing* mehanizma.

```
int i = 5;
Integer x = i; // boxing

Integer y = new Integer(9);
int m = y; // unboxing
```