5 Rad sa stringovima

5.1 Klasa String

U Javi, stringovi se najčešće predstavljaju pomoću klase `String`. Način deklarisanja i inicijalizacije stringova koji smo koristili do sad je:

```
String str = "Petak, 22. februar";
```

Pored ovog načina, možemo koristiti i standardni način kreiranja stringa pomoću ključne reči new:

```
String str = new String("Petak, 22. februar");
```

Novi string se može inicijalizovati postojećim stringom:

```
String s1 = "Programiranje";
String s2 = new String(s1);
```

Novi string se može inicijalizovati i nizom `char` vrednosti, ili njegovim podnizom:

```
char s1[] = {'P', 'e', 't', 'a', 'k'};
String s2 = new String(s1);
String s3 = new String(s1, 1, 3);             // s3 = "eta"
```

Jednom kreiran `String` objekat se ne može promeniti. Klasa `String` je deklarisana kao `public final`.

U nastavku dajemo pregled osnovnih operacija sa `String` objektima.

5.1.1 Metode length, charAt i substring

Dužina stringa se dobija pomoću metode `length`, npr. `str.length()`.

Metoda `charAt` vraća karakter (podatak tipa `char`) sa određene pozicije u stringu. Na primer, poslednji karakter stringa `str` se dobija sa

```
str.charAt(str.length() - 1);
```

Kao i kod nizova, indeksiranje elemenata stringa počinje od 0. U slučaju nepostojeće pozicije, baca se izuzetak `StringIndexOutOfBoundsException`.

5.2 Klase StringBuilder i StringBuffer

Klasa StringBuilder služi za rad sa promenljivim stringovima, što nije dozvoljavala klasa String. Interno, StringBuilder objekti se tretiraju kao nizovi promenljive dužine koji sadrže sekvencu karaktera.

Postoje četiri konstruktora klase StringBuilder, od kojih ćemo pokazati tri:

```
StringBuilder s = new StringBuilder();
StringBuilder s = new StringBuilder(10);
StringBuilder s = new StringBuilder("Petak i Java");
```

Konstruktor bez argumenata pravi prazan StringBuilder objekat sa podrazumevanim kapacitetom od 16 karaktera.

Ako se navede ceo broj u pozivu konstruktora, taj broj će predstavljati kapacitet StringBuilder objekta.

Ako se u konstruktoru navede string, kreira se StringBuilder objekat sa karakterima tog stringa. Kapacitet tako formiranog stringa je za 16 veći do dužine stringa.

Metoda length vraća dužinu StringBuilder stringa, kao kod klase String.

Metoda capacity vraća broj karaktera koji se može smestiti u StringBuilder objekat bez alokacije dodatne memorije, dok metoda ensureCapacity osigurava da kapacitet StringBuilder objekta bude jednak argumentu te metode.

Metoda setLength povećava ili smanjuje dužinu StringBuilder stringa. Ako je argument metode manji od dužine StringBuilder stringa, dolazi do odsecanja karaktera. U suprotnom, ako je argument metode veći od dužine StringBuilder stringa, dolazi do nadovezivanja null karaktera '\u0000' na kraj tog stringa do tražene dužine. Ovo je ilustrovano sledećim naredbama:

```
StringBuilder s1 = new StringBuilder("Java");
StringBuilder s2 = new StringBuilder("Java");
int duzina = s1.length(), kapacitet = s1.capacity();
System.out.printf("Dužina %d, kapacitet %d\n", duzina, kapacitet);
s1.setLength(8);
s2.setLength(2);
System.out.printf("String veće dužine %s, a manje %s\n", s1, s2);
```

```
Dužina 4, kapacitet 20
String veće dužine Java    , a manje Ja
```

Još neke korisne metode klase StringBuilder su opisane u tabeli 5.2.

Metoda	Opis
charAt	Vraća karakter na poziciji argumentu metode. Pokušaj pristupanju karakteru van dozvoljenog opsega dovodi do izuzetka StringIndexOutOfBoundsException.

5.3 Klasa Character

Klasa Character je omotačka klasa za primitivni tip char. Većina metoda ove klase su statičke, radi jednostavne obrade pojedinačnih char vrednosti. Često korišćene metode su date u tabeli 5.3.

Metoda	Opis
isDefined	Vraća true ili false u zavisnosti od toga da li je karakter parametar definisan u Unicode skupu karaktera ili nije.
isDigit isLetter isLetterOrDigit	Vraćaju true ili false u zavisnosti od toga da li je karakter parametar cifra (isDigit), slovo (isLetter) ili jedno od to dvoje (isLetterOrDigit).
isLowerCase isUpperCase	Vraćaju true ili false u zavisnosti od toga da li je karakter parametar malo (isLowerCase) ili veliko slovo (isUpperCase).
toLowerCase toUpperCase	Pretvaraju veliko slovo u malo (toLowerCase) i obrnuto (toUpperCase), ili vraćaju originalni karakter ukoliko se date operacije ne mogu izvršiti.
digit forDigit	Pretvaraju karakter u cifre (digit) i cifre u karaktere (forDigit), u različitim brojnim sistemima. Osnova brojnog sistema je drugi parametar metoda, dok je predmetni karakter prvi parametar.

Tabela 5.3. Metode klase Character.

5.4 Regularni izrazi

Regularni izraz ili *regex* (eng. *regular expression*) je sekvenca karaktera koja predstavlja šablon za pretragu. Obično se koristi u string matching operacijama, tj. operacijama tipa "pronađi i zameni". Koncept regularnih izraza je uveo američki matematičar Stephen Kleene 1950.

Regularni izrazi pružaju vrlo elegantno rešenje za validaciju unosa korisnika i proveru da li podaci zadovoljavaju određeni format. Na primer, regularni izrazi nam mogu poslužiti za proveru formata mejl adrese, broja telefona, korisničke lozinke koja mora da sadrži određene karaktere, imena i prezimena osobe itd. Regularne izraze koriste mnogi tekst editori i programi za pretragu i manipulaciju tekstom, uključujući Microsoft Word.

Vrlo važna primena regularnih izraza je u kreiranju kompajlera, i to u domenu leksičke analize programa. Naime, prvi korak u proveri ispravnosti teksta programa je kreiranje niza značajnih stringova (tokena), što se vrši korišćenjem složenog regularnog izraza.

Mnogi programski jezici, uključujući Javu, PHP, Perl, Python, Ruby, JavaScript, podržavaju rad sa regularnim izrazima.

U Javi, regularni izraz je string koji se sastoji od karaktera literala i specijalnih simbola. Za rad sa regularnim izrazima, mi ćemo ovde objasniti nekoliko metoda iz klase `String`. Na raspolaganju su i klase `Pattern` i `Matcher`, koje ostavljamo čitaocu da ih sam istraži.

U nastavku opisujemo klase karaktera koje se mogu koristiti u regularnim izrazima.

5.4.1 Klase karaktera u regularnim izrazima

U tabeli 5.4 dajemo primere konstrukcija koje se koriste u regularnim izrazima. Izraz unutar uglastih zagrada menja jedan karakter.

Konstrukcija	Opis
`[abc]` ili `[a|b|c]`	Karakter a, b ili c.
`[^abc]`	Sve osim karaktera a, b ili c.
`[a-z]`	Sva mala slova (crtica označava opseg karaktera).
`[^a-z]`	Sve osim malih slova (^ negira opseg karaktera koji sledi).
`[a-zA-Z]` ili `[a-z[A-Z]]`	Sva slova, mala ili velika (unija skupova karaktera).
`[a-m&&e-q]`	Presek opsega karaktera a-m i e-q.

Tabela 5.4. Konstrukcije koje menjaju jedan karakter u regularnim izrazima.

Predefinisane klase karaktera (metakarakteri) koji se koriste u regularnim izrazima su dati u tabeli 5.5. Svaki od njih menja jedan karakter.

Metakarakter	Opis
`.`	Bilo koji karakter.
`\d`	Bilo koja cifra `[0-9]`.
`\D`	Sve osim cifri `[^0-9]`.
`\w`	Bilo koji word karakter `[a-zA-Z_0-9]`.
`\W`	Sve osim word karaktera `[^\w]`.
`\s`	Belina.
`\S`	Sve osim belina `[^\s]`.

Tabela 5.5. Metakarakteri u regularnim izrazima.

5.4.2 Kvantifikatori u regularnim izrazima

Konstrukcije i metakarakteri iz tabela 5.4 i 5.5 definišu šablone koji se odnose na jedan karakter u regularnom izrazu. Ukoliko želimo da se definisani karakter ponavlja više puta (npr. bar 3 slova, između 4 i 9 cifara), potrebno je koristiti kvantifikatore u regularnim izrazima. U pitanju su specijalni izrazi koji određuju broj pojavljivanja sekvence karaktera (jednog ili više uzastopnih karaktera) u stringu.

Kvantifikator se odnosi samo na sekvencu koja prethodi kvantifikatoru. Ako želimo proveru da li se složeniji izraz ponavlja u stringu, možemo koristiti male zagrade da grupišemo taj izraz. U tabeli 5.6 su dati Javini kvantifikatori pojavljivanja sekvenci u regularnim izrazima.

Kvantifikator	Opis
*	0 ili više uzastopnih pojavljivanja sekvence.
+	1 ili više uzastopnih pojavljivanja sekvence.
?	0 ili 1 pojavljivanja sekvence.
{n}	Tačno n uzastopnih pojavljivanja sekvence.
{n,}	Barem n uzastopnih pojavljivanja sekvence.
{n,m}	Između n i m uzastopnih pojavljivanja sekvence (uključeni n i m).

Tabela 5.6. Kvantifikatori pojavljivanja sekvenci u regularnim izrazima.

Kvantifikatori su pohlepni (eng. *greedy*), tj. traže maksimalan broj pojavljivanja podstringa (nastavljaju pretragu sve dok ima poklapanja). Ako se iza njih stavi karakter `?`, kvantifikatori postaju lenji, tj. traže minimalan broj pojavljivanja.

Primeri korišćenja klasa karaktera i kvantifikatora su dati u narednom poglavlju.

5.4.3 Regularni izrazi u metodama klase `String`

Klasa `String` ima nekoliko metoda za izvršavanje operacija sa regularnim izrazima, od kojih je najjednostavnija metoda `matches`. Ova metoda za parametar ima `String` objekat koji specificira regularan izraz i poziva se na `String` objektu za koji želimo da utvrdimo da li odgovara predmetnom regularnom izrazu. Metoda vraća `true` ili `false`. Navedimo nekoliko primera korišćenja ove metode.

`str.matches("\\d*")` proverava da li se string `str` sastoji samo od cifara.

`str.matches("[a-z]+")` proverava da li se string `str` sastoji samo od malih slova.

`str.matches("Java.*")` proverava da li string `str` počinje rečju `"Java"` i nakon toga može da sadrži proizvoljan broj (uključujući 0) proizvoljnih karaktera.

`str.matches("(Ta)+")` proverava da li se string sastoji od ponavljanja proizvoljnog broja slogova `"Ta"`, tj. stringovi `"Ta"`, `"TaTa"`, `"TaTaTa"`...

`str.matches("[A-Z][a-z]+")` proverava da li string predstavlja ime osobe. Ime je string koji počinje velikim slovom, a nakon toga dolazi jedno ili više malih slova.

`str.matches("[A-Z][a-z]+(-[A-Z][a-z]+)?")` proverava da li string predstavlja prezime osobe, tj. da li string sadrži jednu ili dve reči. Ako prezime ima dve reči, one su razdvojene crticom. Reči počinju velikim slovom, a ostalo su mala slova. Na primer, `"Popović"` ili `"Popović-Bajčeta"`.

`str.matches("06[7|8|9]\\d{6}")` proverava da li string predstavlja broj mobilnog telefona u Crnoj Gori (kôd mobilne mreže 067, 068 ili 069, nakon toga 6 proizvoljnih cifara).

Metoda `replaceFirst` menja prvu pojavu jednog podstringa u predmetnom stringu drugim podstringom, dok `replaceAll` menja sve pojave jednog podstringa u predmetnom