

Klasa String

U dosadašnjim primerima su se nizovi znakova predstavljali korišćenjem primitivnog tipa “char” i oznake za nizove (“char[]”). Ovo, međutim, nije najbolji način za rad sa nizovima znakova. Često je potrebno nad određenom reči ili izrazom uraditi neke složene operacije - izdvojiti samo jedan deo reči ili izraza, proveriti da li se u izrazu nalazi neka reč, zameniti neko slovo nekim drugim, spojiti neke reči u rečenice itd. Ako bi se koristio “char” niz, sve ove operacije bi trebalo posebno napisati u vidu metoda. Čak bi se i poređenje jednakosti dva niza slovo po slovo moralo implementirati na isti način. U Javi postoji predefinisana klasa za rad sa nizovima znakova koja već ima implementirane mnoge od ovih funkcionalnosti. To je **klasa String**.

Deklaracija String promenljive se vrši na isti način kao i za bilo koju drugu promenljivu (String je klasa pa njen naziv počinje velikim slovom):

```
String nazivPromenljive;
```

Sa obzirom na to da je u pitanju klasa, **svaka promenljiva ovog tipa je objekat pa se mora inicijalizovati** pre nego što se može koristiti. Inicijalizacija se može uraditi na nekoliko načina. Prvo, moguće je inicijalizovati objekat klase String korišćenjem konstruktora.

```
String nazivPromenljive = new String("neki niz znakova");
```

Konstruktor ove klase kao ulazni parametar prima neku String vrednost. Potrebno je primetiti da se **String vrednosti uvek pišu pod dvostrukim znacima navoda** (npr. “Rečenica broj 1”). Nasuprot tome, “char” vrednosti se pišu pod jednostrukim znacima navoda (npr. 'A').

String promenljive se mogu inicijalizovati i na kraći način. Efekat prethodne i naredne naredbe za inicijalizaciju je isti.

```
String nazivPromenljive = "neki niz znakova";
```

Ovakav vid inicijalizacije je **moguće ostvariti samo kada su u pitanju String objekti i ne važi za druge klase**. Moguć je i još jedan oblik inicijalizacije String promenljive koji je koristan u onim situacijama kada je niz promenljivih tipa “char” potrebno pretvoriti u String objekat iste sadržine. U ovom slučaju, konstruktor klase String kao ulazni parametar dobija niz promenljivih tipa “char”.

```
char[] niz;  
...
```

```
String s = new String(niz);
```

Efekat inicijalizacije String objekta je isti kao i za objekte drugih klasa. Pre inicijalizacije, promenljiva ima “null” vrednost. Pri inicijalizaciji se u memoriji računara zauzima prostor odgovarajuće veličine za smeštanje konkretne String vrednosti (Slika 1).

String promenljiva1;

promenljiva1

null

String nije inicijalizovan i još uvek ne može da se koristi.

promenljiva1 = "Recenica 1";

promenljiva1

30

30

"Recenica 1"

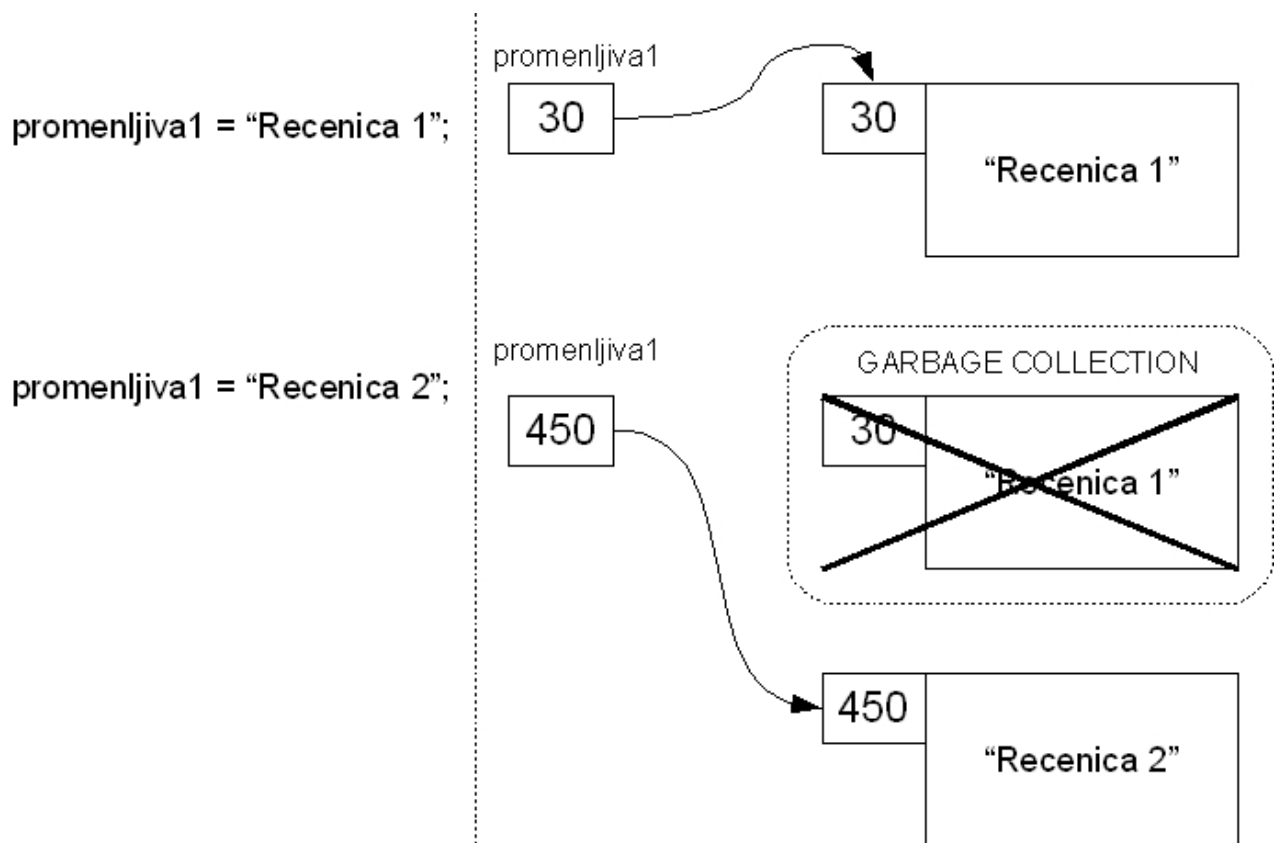
String je inicijalizovan i može da se koristi.

Slika 1: Inicijalizacija String promenljive

Pored "null" vrednosti, String promenljiva može da sadrži i **prazan String**. To je String vrednost koja **ne sadrži nijedan znak (čak ni prazna mesta) ali je promenljiva ipak inicijalizovana**.

```
String nazivPromenljive = "";
```

Potrebno je skrenuti pažnju i na to šta se dešava kada se String promenljivoj koja već ima neku vrednost dodeli nova vrednost. Iako se čini logičnim da će se iskoristiti memorijski prostor koji je već zauzet, to se ne dešava. Umesto toga, **inicijalizuje se novi String objekat i on prima novu vrednost, a objekat koji sadrži staru vrednost se oslobađa pomoću "garbage collection" mehanizma** (Slika 2).



Slika 2: Dodeljivanje nove vrednosti String promenljivoj

String vrednosti su, u stvari, nizovi znakova pa se često dešava da je potrebno spojiti više vrednosti u jedan String. U pitanju je **nadovezivanje ("concatenation") String vrednosti** i vrši se korišćenjem operatora za nadovezivanje (koji se piše isto kao i znak za sabiranje - "+"). Ovaj operator se može koristiti isključivo sa String vrednostima.

```
nazivPromenljive = "neki niz znakova" + "drugi niz znakova";
```

Operator za nadovezivanje String vrednosti je zapravo već korišćen u okviru "println" komande da bi se formirala neka složena izlazna poruka. Tu se može primetiti još jedna karakteristika ovog operatora a to je da će **sve vrednosti koje nisu String tipa biti pretvorene u odgovarajući niz znakova i nadovezane na postojeću String vrednost**. Jedini preduslov je da je **makar jedna vrednost u celom izrazu String**.

Primer 1

Evo nekoliko primera naredbi za nadovezivanje String vrednosti.

```
String s1 = "Danas";
String s2 = " je lep dan.";
```

```
//String s3 ce da dobije vrednost
//"Danas je lep dan."
String s3 = s1 + s2;
```

```
//Vrednost promenljive temperatura ce da
//se automatski pretvori u String vrednost
```

```
//"15" pa ce s4 da dobije vrednost
//"Napolju je 15 stepeni.".
int temperatura = 15;
String s4 = "Napolju je "+temperatura+" stepeni.";

//Ovo ce da bude greska jer nijedan element
//izraza nije String vrednost.
//String s5 = 12 + 15;

//Umesto toga, trebalo bi izraz napisati na jedan
//od sledecih nacina da bi promenljiva s5
//imala vrednost "1215"

//Prvi element je String pa je sve u redu.
String s5 = "12" + 15;

//Drugi element je String pa je sve u redu.
s5 = 12 + "15";

//"Podmetnut" je prazan String koji nece
//da utice na krajnju vrednost s5 ali ce
//izraz biti ispravan.
s5 = "" + 12 + 15;
```

Poređenje jednakosti za proste tipove podataka (int, boolean, char, double) je relativno jednostavno i vrši se pomoću operatora za poređenje jednakosti ("=="). Sa obzirom na to da je String klasa, **String vrednosti se ne mogu porediti korišćenjem operatora za poređenje jednakosti**. Ako bi se to uradilo, poredile bi se samo adrese objekata u memoriji a ne i sam sadržaj. Umesto toga, potrebno je koristiti **"equals" metodu** koja poredi dva String slovo po slovo i vraća "true" ako su jednaki a "false" ako nisu. Pri tome, obraća se pažnja na velika i mala slova. Ova metoda se koristi na sledeći način:

```
promenljiva1.equals(promenljiva2)
```

Operator za poređenje jednakosti se može koristiti jedino kada se želi proveriti da li String promenljiva ima "null" vrednost tj. da li je inicijalizovana ili ne.

```
promenljiva1 == null;
```

Primer 2

Ovaj primer prikazuje šta se dešava ako se dve String vrednosti porede na pravilan i nepravilan način.

```
String s1 = "Recenica 1";
String s2 = new String("Recenica 1");

if (s1 == s2)
    System.out.println("Jednaki su");
else
    System.out.println("Nisu jednaki");

if (s1.equals(s2))
```

```

        System.out.println("Jednaki su");
    else
        System.out.println("Nisu jednaki");

```

U prvoj IF komandi se vrednosti porede na nepravilan način pa će da se ispiše da nisu jednake iako je sadržaj obe promenljive isti. U drugoj IF komandi se poziva equals metoda koja vrši poređenje slovo po slovo pa će se ispisati da su jednake vrednosti u pitanju.

Već je rečeno da String klasa omogućava rad sa nizovima znakova na jednostavniji način, pružajući dodatne funkcionalnosti. Međutim, nekada je potrebno pojedinačno proći kroz sve znakove u String promenljivoj kao da je u pitanju običan niz znakova. Svaki znak String vrednosti ima svoj indeks a indeksi kreću od nule i tu se vidi sličnost sa običnim nizovima. Međutim, znakovima se ne pristupa korišćenjem uglastih zagrada već pozivanjem **metode “charAt”**. Dužina String vrednosti se dobija pozivanjem **metode “length”**.

Primer 3

Napraviti klasu PrebrojavanjeZnakova koja ima:

- *Statičku metodu prebroj koja kao ulazni parametar prima String i vraća koliko puta se u njemu pojavljuje slovo 'a'.*

```

class PrebrojavanjeZnakova {

    static int prebroj(String s){
        int brojac = 0;

        for (int i=0; i<s.length();i++)
            if(s.charAt(i) == 'a')
                brojac++;

        return brojac;
    }

}

```

Iz primera se može videti da se kroz String prolazi uz pomoć FOR petlje na sličan način kao kroz običan niz. Brojač petlje se kreće od nula (indeks prvog znaka) do “s.length()-1” (indeks poslednjeg znaka). Pojedinačnom znaku se pristupa pozivanjem “charAt” metode kojoj se prosleđuje indeks željenog znaka.

String klasa ima i niz drugih metoda za rad sa nizovima znakova. Njihov naziv i opis su dati u sledećoj tabeli.

Naziv	Opis
charAt(int indeks)	Vraća znak iz String promenljive koji je na određenoj poziciji
compareTo(String s)	Poredi dve String vrednosti leksikografski (abecedno) i vraća pozitivan broj ako je prva vrednost “posle” druge vrednosti, negativan broj ako je “pre” a nulu ako su jednaki.
compareToIgnoreCase(String s)	Radi isto što i prethodna metoda, ali ne obraća pažnju na to da li su u pitanju velika ili mala slova.
endsWith(String s)	Proverava da li se prvi String završava nizom slova koji se

	nalazi u Stringu s i vraća "true" ako je to tačno, a "false" ako nije.
equals(Object o)	Proverava da li je prvi String jednak unetom String-u i vraća "true" ako jeste a "false" ako nije. Kao ulazni parametar ova metoda bi trebalo da dobije String bez obzira na to što piše da je ulazni parametar tipa Object.
equalsIgnoreCase(String s)	Radi isto što i prethodna metoda samo se ne obraća pažnja na velika i mala slova (na primer, "DAN" i "dan" će da budu protumačeni kao jednake vrednosti).
indexOf(char c)	Pronalazi i vraća indeks prvog pojavljivanja znaka koji je unet kao ulazni parametar u String vrednosti. Ako se taj znak ne može naći, vraća -1.
indexOf(String s)	Radi isto što i prethodna metoda samo traži pojavljivanje niza znakova.
indexOf(char c, int indeks)	Pronalazi i vraća indeks prvog pojavljivanja znaka koji je unet kao ulazni parametar u String vrednosti. Pretraživanje počinje od pozicije u String-u koja je uneta u vidu indeksa.
indexOf(String s, int indeks)	Radi isto što i prethodna metoda samo traži pojavljivanje niza znakova.
lastIndexOf(char c)	Pronalazi i vraća indeks poslednjeg pojavljivanja znaka koji je unet kao ulazni parametar u String vrednosti. Ako se taj znak ne može naći, vraća -1.
lastIndexOf(String s)	Radi isto što i prethodna metoda samo traži pojavljivanje niza znakova.
lastIndexOf(char c, int indeks)	Pronalazi i vraća indeks poslednjeg pojavljivanja znaka koji je unet kao ulazni parametar u String vrednosti. Pretraživanje počinje od pozicije u String-u koja je uneta u vidu indeksa.
lastIndexOf(String s, int indeks)	Radi isto što i prethodna metoda samo traži pojavljivanje niza znakova.
length()	Vraća dužinu String vrednosti.
replace(char c1, char c2)	Pronalazi sva pojavljivanja znakova koji su jednaki vrednosti promenljive c1 u Stringu i menja ih znakom iz promenljive c2.
split(String s)	Deli početni String na više String vrednosti (vraća niz String vrednosti). Deljenje se vrši na onim mestima gde se naide na izraz koji je dat kao ulazni argument. Ako se npr. kao ulazni argument unese prazno mesto (" "), podeliće početnu vrednost na pojedinačne reči.
startsWith(String s)	Proverava da li prvi String počinje nizom slova koji se nalazi u Stringu s i vraća "true" ako je to tačno, a "false" ako nije.
substring(int pocetniIndeks)	Vraća deo početne String vrednosti počev od znaka sa indeksom "pocetniIndeks" pa do kraja.
substring(int pocetniIndeks, int krajnjiIndeks)	Vraća deo početne String vrednosti počev od znaka sa indeksom "pocetniIndeks" pa do znaka sa indeksom "krajnjiIndeks" ali ne uključujući taj znak.

toLowerCase()	Pretvara sva velika slova String vrednosti u mala i vraća tako izmenjen String.
toUpperCase()	Pretvara sva mala slova String vrednosti u velika i vraća tako izmenjen String.
trim()	“Odseca” znakove koji predstavljaju prazna mesta sa početka i kraja String vrednosti i vraća tako izmenjen String.

Zadaci

Zadatak 1

Napraviti klasu **AnalizatorTeksta**. Ova klasa bi trebalo da ima:

- Atribut tekst koji je tipa String.
- Konstruktor koji postavlja vrednost atributa tekst na “nepoznat”.
- Konstruktor koji postavlja vrednost atributa tekst na vrednost koja je data u vidu ulaznog argumenta. Dodeljivanje se vrši isključivo ako je ulazna vrednost različita od null. U suprotnom, atribut tekst dobija vrednost “nepoznat” i ispisuje se poruka o grešci na ekranu.
- Metodu **getTekst** koja vraća trenutnu vrednost atributa tekst.
- Metodu **setTekst** koja postavlja vrednost atributa tekst na novu vrednost. Ovaj novi tekst je dat u vidu ulaznog argumenta. Dodeljivanje se vrši isključivo ako je novi tekst različit od null. U suprotnom, ispisuje se poruka o grešci na ekranu.
- Metodu koja na kraj teksta dodaje novi tekst. Novi tekst je dat u vidu ulaznog argumenta. Prvo je potrebno proveriti da li je vrednost novog teksta null. Ako jeste, ispisuje se poruka o grešci na ekranu. Ako nije, proverava se da li je vrednost atributa tekst pre toga bila “nepoznat”. Ako jeste, atribut tekst dobija vrednost unetog teksta, a ako nije novi tekst se dodaje na kraj.
- Metodu koja na početak teksta dodaje novi tekst. Novi tekst je dat u vidu ulaznog argumenta. Prvo je potrebno proveriti da li je vrednost novog teksta null. Ako jeste, ispisuje se poruka o grešci na ekranu. Ako nije, proverava se da li je vrednost atributa tekst pre toga bila “nepoznat”. Ako jeste, atribut tekst dobija vrednost unetog teksta, a ako nije novi tekst se dodaje na početak.
- Metodu koja proverava da li je tekst koji se unosi kao ulazni argument isti kao i postojeći tekst i ispisuje poruku o tome na ekranu.
- Metodu koja proverava da li je tekst koji se unosi kao ulazni argument isti kao i postojeći tekst i ispisuje poruku o tome na ekranu ali bez obzira na to da li su slova u okviru teksta velika ili mala. Primer: “Danas je lep dan.” i “DANAS je lep dan.” bi trebalo da se protumače kao jednaki tekstovi.
- Metodu koja proverava da li je tekst koji se unosi kao ulazni argument pre, posle ili jednak u odnosu na postojeći tekst (leksikografski gledano) i ispisuje poruku o tome na ekranu. Primer: Tekst “Dan” je pre teksta “Mrak” jer je slovo D leksikografski pre slova M. Tekst “Dan” je posle teksta “Abba” je je slovo D veće od A. Tekst “Dan” je pre teksta “Danas” jer ovaj drugi tekst ima više slova.

Napraviti klasu **TestAnalizatorTeksta** koja kreira dva objekta klase AnalizatorTeksta: prvi korišćenjem prvog konstruktora (konstruktor bez argumenata), a drugi korišćenjem drugog konstruktora i to sa ulaznom vrednošću “Danas je lep dan”. Prvom objektu dodeliti vrednost teksta “Nebo je plavo.” i dodati na kraj teksta “Sunce sija.”. Proveriti da li je tekst drugog objekta jednak “lep dan”. Ispisati tekst oba objekta na ekranu.

Rešenje

```
class AnalizatorTeksta {

    String tekst;

    AnalizatorTeksta() {
        tekst = "nepoznat";
    }

    AnalizatorTeksta(String t) {
        if (t != null)
            tekst = t;
        else {
            System.out.println("String koji ste uneli je prazan!");
            tekst = "nepoznat";
        }
    }
}
```

```

    }
}

String getTekst() {
    return tekst;
}

void setTekst(String novi_tekst) {
    if (novi_tekst != null) tekst = novi_tekst;
    else System.out.println("String koji ste uneli je prazan");
}

void dodajNaKraj(String novi_tekst) {
    if (novi_tekst == null)
        System.out.println("String koji ste uneli je prazan");
    else {
        if (tekst.equals("nepoznat"))
            tekst = novi_tekst;
        else
            tekst = tekst + novi_tekst;
    }
}

void dodajNaPocetak(String novi_tekst) {
    if (novi_tekst == null)
        System.out.println("String koji ste uneli je prazan");
    else {
        if (tekst.equals("nepoznat"))
            tekst = novi_tekst;
        else
            tekst = novi_tekst + tekst;
    }
}

void proveraNednakosti (String novi_tekst){
    if (tekst.equals(novi_tekst))
        System.out.println("Tekstovi su jednaki");
    else
        System.out.println("Tekstovi nisu jednaki");
}

void proveraNednakosti2 (String novi_tekst){
    if (tekst.equalsIgnoreCase(novi_tekst))
        System.out.println("Tekstovi su jednaki");
    else
        System.out.println("Tekstovi nisu jednaki");
}

void leksikografskaProvera (String novi_tekst){
    if (tekst.compareTo(novi_tekst)>0)
        System.out.println("Tekst je posle novog teksta");
    if (tekst.compareTo(novi_tekst)<0)
        System.out.println("Tekst je pre novog teksta");
    if (tekst.compareTo(novi_tekst)==0)
        System.out.println("Tekst je jednak novom tekstu");
}

}

class TestAnalizatorTeksta {

    public static void main (String[] args){

```



```

AnalizatorTeksta a1 = new AnalizatorTeksta();
AnalizatorTeksta a2 = new AnalizatorTeksta("Danas je lep dan.");

a1.setTekst("Nebo je plavo.");
a1.dodajNaKraj("Sunce sija.");

a2.proveraJednakosti("lep dan");

System.out.println(a1.getTekst());
System.out.println(a2.getTekst());

    }
}

```

Zadatak 2

Napraviti klasu **AnalizatorSlovaUTekstu** koja ima:

- Atribut tekst koji je tipa String koji ima početnu vrednost “nepoznat”.
- Metodu **getTekst** koja vraća trenutnu vrednost atributa tekst.
- Metodu **setTekst** koja postavlja vrednost atributa tekst na novu vrednost. Ovaj novi tekst je dat u vidu ulaznog argumenta. Dodeljivanje se vrši isključivo ako je novi tekst različit od null. U suprotnom, ispisuje se poruka o grešci na ekranu.
- Metodu koja vraća dužinu teksta tj. broj znakova u tekstu.
- Metodu koja proverava i vraća broj pojavljivanja malog slova “a” u tekstu.
- Metodu koja proverava i vraća broj pojavljivanja nekog slova u tekstu. Slovo koje se traži je dato u vidu ulaznog argumenta.
- Metodu koja proverava i vraća broj rečenica u tekstu. Svaka rečenica se obavezno završava tačkom, znakom upita ili uzvičnikom.
- Metodu koja proverava i vraća broj samoglasnika u tekstu. Potrebno je prebrojati i velika i mala slova (a,e,i,o,u,A,E,I,O,U).
- Metodu koja vraća slovo (znak) koje se najviše puta pojavljuje u tekstu. Pri tome, ne treba brojati prazna mesta u tekstu.
- Metodu koja vraća slovo (znak) koje se namanje puta pojavljuje u tekstu. Pri tome, ne treba brojati prazna mesta u tekstu.
- Metodu koja vraća redni broj (indeks) prvog pojavljivanja slova “a” u tekstu.
- Metodu koja vraća redni broj (indeks) poslednjeg pojavljivanja slova “a” u tekstu.
- Metodu koja vraća redni broj (indeks) prvog pojavljivanja nekog slova u tekstu. Slovo je dato u vidu ulaznog argumenta.
- Metodu koja vraća redni broj (indeks) poslednjeg pojavljivanja nekog slova u tekstu. Slovo je dato u vidu ulaznog argumenta.
- Metodu koja vraća redni broj (indeks) drugog pojavljivanja slova “a” u tekstu.
- Metodu koja vraća redni broj (indeks) preposlednjeg pojavljivanja slova “a” u tekstu.

Napisati klasu **TestAnalizatorSlovaUTekstu** koja kreira jedan objekat klase AnalizatorSlovaUTekstu i dodeljuje mu tekst “Dan je lep. Da li je lep? Da, stvarno je LEP!”. Ispisati na ekranu: broj samoglasnika u tekstu, broj rečenica i redni broj drugog pojavljivanja slova a.

Rešenje

```

class AnalizatorSlovaUTekstu {

    String tekst = "nepoznat";

    String getTekst() {
        return tekst;
    }

    void setTekst(String novi_tekst) {
        if (novi_tekst != null)
            tekst = novi_tekst;
        else
            System.out.println("String koji ste uneli je prazan");
    }
}

```

```

}

int duzinaTeksta() {
    return tekst.length();
}

int brojPojavljivanjaSlovaA() {
    int brojac = 0;
    for (int i = 0; i < tekst.length(); i++)
        if (tekst.charAt(i) == 'a')
            brojac++;
    return brojac;
}

int brojPojavljivanjaSlova(char c) {
    int brojac = 0;
    for (int i = 0; i < tekst.length(); i++)
        if (tekst.charAt(i) == c)
            brojac++;
    return brojac;
}

int brojRecenica() {
    int broj_recenica = 0;
    for (int i = 0; i < tekst.length(); i++)
        if ((tekst.charAt(i) == '.') || (tekst.charAt(i) == '!')
            || (tekst.charAt(i) == '?'))
            broj_recenica++;
    return broj_recenica;
}

int brojSamoglasnika() {
    int broj_s = brojPojavljivanjaSlova('a') +
        brojPojavljivanjaSlova('e')
        + brojPojavljivanjaSlova('i') +
        brojPojavljivanjaSlova('o')
        + brojPojavljivanjaSlova('u') +
        brojPojavljivanjaSlova('A')
        + brojPojavljivanjaSlova('E') +
        brojPojavljivanjaSlova('I')
        + brojPojavljivanjaSlova('O') +
        brojPojavljivanjaSlova('U');
    return broj_s;
}

char maxPojavljivanjeSlova() {
    int max_broj = 0;
    char max_slovo = tekst.charAt(0);

    for (int i = 0; i < tekst.length(); i++)
        if ((tekst.charAt(i) != ' ')
            && (brojPojavljivanjaSlova(tekst.charAt(i)) > max_broj)) {
            max_broj = brojPojavljivanjaSlova(tekst.charAt(i));
            max_slovo = tekst.charAt(i);
        }

    return max_slovo;
}

char minPojavljivanjeSlova() {
    int min_broj = tekst.length()+1;
    char min_slovo = tekst.charAt(0);

    for (int i = 0; i < tekst.length(); i++)

```

```

        if ((tekst.charAt(i) != ' ')
            && (brojPojavljivanjaSlova(tekst.charAt(i)) < min_broj)) {
            min_broj = brojPojavljivanjaSlova(tekst.charAt(i));
            min_slovo = tekst.charAt(i);
        }

        return min_slovo;
    }

    int prviRedniBrojSlovaA() {
        // Metoda indexOf vraca indeks prvog pojavljivanja
        // trazenog slova u Stringu. Ako tog slova nema u
        // Stringu, metoda vraca -1.
        return tekst.indexOf('a');
    }

    int poslednjiRedniBrojSlovaA() {
        // Metoda lastIndexOf vraca indeks poslednjeg pojavljivanja
        // trazenog slova u Stringu. Ako tog slova nema u
        // Stringu, metoda vraca -1.
        return tekst.lastIndexOf('a');
    }

    int prviRedniBrojSlova(char c) {
        return tekst.indexOf(c);
    }

    int poslednjiRedniBrojSlova(char c) {
        return tekst.lastIndexOf(c);
    }

    int drugiRedniBrojSlovaA() {
        // Metoda indexOf(char c,int i) vraca indeks prvog pojavljivanja
        // trazenog slova u Stringu ali pocev od pozicije i. Ako tog slova
        // nema u Stringu, metoda vraca -1.
        int prvi_indeks = tekst.indexOf('a');
        if (prvi_indeks == -1)
            return -1;
        else
            return tekst.indexOf('a', prvi_indeks + 1);
    }

    int pretposlednjiRedniBrojSlovaA() {
        // Metoda lastIndexOf(char c,int i) vraca indeks poslednjeg
        // pojavljivanja
        // trazenog slova u Stringu ali od pocetka stringa
        // završno sa pozicijom i. Ako tog slova nema u
        // Stringu, metoda vraca -1.
        int poslednji_indeks = tekst.lastIndexOf('a');
        if (poslednji_indeks == -1)
            return -1;
        else
            return tekst.lastIndexOf('a', poslednji_indeks - 1);
    }
}

class TestAnalizatorSlovaUTekstu {

    public static void main(String[] args) {

        AnalizatorSlovaUTekstu a = new AnalizatorSlovaUTekstu();

        a.setTekst("Dan je lep. Da li je lep? Da, stvarno je LEP!");
    }
}

```

```

        //Broj samoglasnika u tekstu bi trebalo da bude 12
        System.out.println("Broj samoglasnika u tekstu je: "+
            a.brojSamoglasnika());

        //Broj recenica u tekstu bi trebalo da bude 3
        System.out.println("Broj recenica u tekstu je: "+a.brojRecenica());

        //Indeks drugog pojavljivanja slova 'a' bi trebalo da bude 13
        System.out.println("Redni broj drugog pojavljivanja slova 'a' je: "+
            a.drugiRedniBrojSlovaA());
    }
}

```

Zadatak 3

Napraviti klasu **AnalizatorReciIRecenica** koja ima:

- Atribut tekst koji je tipa String koji ima početnu vrednost “nepoznat”.
- Metodu **getTekst** koja vraća trenutnu vrednost atributa tekst.
- Metodu **setTekst** koja postavlja vrednost atributa tekst na novu vrednost. Ovaj novi tekst je dat u vidu ulaznog argumenta. Dodeljivanje se vrši isključivo ako je novi tekst različit od null. U suprotnom, ispisuje se poruka o grešci na ekranu.
- Metodu koja vraća deo teksta bez prvog slova.
- Metodu koja vraća deo teksta bez prva četiri slova.
- Metodu koja vraća deo teksta bez prve reči. Svake dve reči u tekstu su odvojene jednim praznim mestom. Ako tekst ima samo jednu reč, potrebno je vratiti null vrednost.
- Metodu koja vraća deo teksta bez prve rečenice. Sve rečenice se završavaju tačkom. Ako tekst nema nijednu rečenicu, potrebno je vratiti null.
- Metodu koja vraća poslednju reč iz teksta. Svake dve reči u tekstu su odvojene jednim praznim mestom. Ako tekst ima samo jednu reč, potrebno je vratiti ceo tekst.
- Metodu koja vraća samo poslednju rečenicu teksta. Sve rečenice se završavaju tačkom. Ako tekst nema nijednu rečenicu, ili ako ima jednu rečenicu, potrebno je vratiti null.
- Metodu koja vraća deo teksta bez prvog i poslednjeg slova.
- Metodu koja vraća deo teksta od trećeg do pretposlednjeg slova (uključujući treće ali ne uključujući pretposlednje slovo).
- Metodu koja vraća drugu reč iz teksta. Svake dve reči u tekstu su odvojene jednim praznim mestom. Ako tekst ima samo jednu reč, potrebno je vratiti null.
- Metodu koja proverava da li se novi tekst nalazi na početku teksta (tj. da li tekst počinje izrazom datim u novom tekstu). Ako počinje, metoda vraća TRUE, u suprotnom FALSE. Ako je novi tekst jednak null metoda vraća FALSE. Novi tekst je dat u vidu ulaznog argumenta.
- Metodu koja proverava da li se novi tekst nalazi na kraju teksta (tj. da li se tekst završava izrazom datim u novom tekstu). Ako se završava, metoda vraća TRUE, u suprotnom FALSE. Ako je novi tekst jednak null metoda vraća FALSE. Novi tekst je dat u vidu ulaznog argumenta.

Napraviti klasu **TestAnalizatorReciIRecenica** koja kreira jedan objekat klase AnalizatorReciIRecenica i unosi u njega tekst: “Sunce sija. Nebo je plavo. Nema nijednog oblaka.”. Ispisati na ekranu poslednju reč i rečenicu iz teksta.

Rešenje

```

class AnalizatorReciIRecenica {

    String tekst = "nepoznat";

    String getTekst() {
        return tekst;
    }

    void setTekst(String novi_tekst) {
        if (novi_tekst != null)
            tekst = novi_tekst;
        else
            System.out.println("String koji ste uneli je prazan");
    }
}

```

```

}

String bezPrvogSlova() {
    String rezultat = tekst.substring(1);
    return rezultat;
}

String bezPrvaCetiriSlova() {
    String rezultat = tekst.substring(4);
    return rezultat;
}

String bezPrveReci() {
    int indeks_prvog_praznog_mesta = tekst.indexOf(' ');
    if (indeks_prvog_praznog_mesta == -1) return null;
    String rezultat = tekst.substring(indeks_prvog_praznog_mesta+1);
    return rezultat;
}

String bezPrveRecenice() {
    int indeks_prve_tacke = tekst.indexOf('.');
    if (indeks_prve_tacke == -1) return null;
    String rezultat = tekst.substring(indeks_prve_tacke+1);
    return rezultat;
}

String poslednjaRec() {
    int indeks_poslednjeg_praznog_mesta = tekst.lastIndexOf(' ');
    if (indeks_poslednjeg_praznog_mesta == -1) return tekst;
    String rezultat = tekst.substring(indeks_poslednjeg_praznog_mesta+1);
    return rezultat;
}

String poslednjaRecenica() {
    int indeks_poslednje_tacke = tekst.lastIndexOf('.');
    if (indeks_poslednje_tacke == -1) return null;

    int indeks_pretposlednje_tacke =
        tekst.lastIndexOf('.', indeks_poslednje_tacke-1);
    if (indeks_pretposlednje_tacke == -1) return null;

    String rezultat = tekst.substring(indeks_pretposlednje_tacke+1);
    return rezultat;
}

String bezPrvogIPoslednjegSlova() {
    String rezultat = tekst.substring(1, tekst.length()-1);
    return rezultat;
}

String odTrecegDoPretposlednjegSlova() {
    String rezultat = tekst.substring(2, tekst.length()-2);
    return rezultat;
}

String drugaRec() {
    int indeks_prvog_praznog_mesta = tekst.indexOf(' ');
    if (indeks_prvog_praznog_mesta == -1) return null;

    int indeks_drugog_praznog_mesta = tekst.indexOf(' ',
        indeks_prvog_praznog_mesta+1);
    //Situacija kada tekst sadrzi samo dve reci. Druga rec je
    //onda od prvog praznog mesta do kraja Stringa.
    if (indeks_drugog_praznog_mesta == -1)

```

```

        return tekst.substring(indeks_prvog_praznog_mesta+1);

        return tekst.substring(indeks_prvog_praznog_mesta+1,
                                indeks_drugog_praznog_mesta);
    }

    boolean daLiPocinjeSa (String nt){
        if (nt == null) return false;

        if (tekst.startsWith(nt)) return true;
        else return false;
        //Umesto ove druge if naredbe moglo je da
        //stoji samo:
        //return tekst.startsWith(nt);
    }

    boolean daLiZavrsavaSa (String nt){
        if (nt == null) return false;

        if (tekst.endsWith(nt)) return true;
        else return false;
        //Umesto ove druge if naredbe moglo je da
        //stoji samo:
        //return tekst.endsWith(nt);
    }
}

class TestAnalizatorReciIRecenica {

    public static void main(String[] args) {

        AnalizatorReciIRecenica a = new AnalizatorReciIRecenica();

        a.setTekst("Sunce sijava. Nebo je plavo. Nema nijednog oblaka.");

        System.out.println("Poslednja rec iz teksta je: "+a.poslednjaRec());
        System.out.println("Poslednja recenica teksta je: "+
                            a.poslednjaRecenica());
    }
}

```

Zadatak 4

Napraviti klasu **BrojacReci** koja ima:

- Statičku metodu koja kao ulazni argument prima tekst i vraća ukupan broj reči u tekstu. Svake dve reči u tekstu su međusobno odvojene jednim praznim mestom.
- Statičku metodu koja prebrojava i vraća broj pojavljivanja reči “lep”. Tekst je dat u vidu ulaznog argumenta. Svake dve reči u tekstu su međusobno odvojene jednim praznim mestom.
- Statičku metodu koja prebrojava i vraća broj pojavljivanja određene reči u tekstu. I tekst i reč koja se traži su dati u vidu ulaznog argumenta. Potrebno je ne uzimati u obzir da li su u pitanju velika ili mala slova. Takođe, potrebno je uzeti u obzir situaciju kada je tražena reč na kraju rečenice (rečenica se uvek završava tačkom). Primer: U tekstu “Lep dan danas. Danas je lep dan.” reč “danas” se pojavljuje dva puta.
- Statičku metodu koja vraća reč koja se u tekstu pojavljuje najveći broj puta. Tekst je dat u vidu ulaznog argumenta.
- Statičku metodu koja vraća reč koja se u tekstu pojavljuje najmanji broj puta. Tekst je dat u vidu ulaznog argumenta.

Napraviti klasu **TestBrojacReci**. Za tekst “Danas je lep dan. Bas lep.” proveriti i ispisati na ekranu: reč koja se pojavljuje najmanji broj puta i reč koja se pojavljuje najveći broj puta.

Rešenje

```
class BrojacReci {

    static int brojReciUTekstu(String tekst) {
        String[] reci = tekst.split(" ");
        return reci.length;
    }

    static int brojReciLep(String tekst) {
        String[] reci = tekst.split(" ");
        int brojac = 0;

        for (int i = 0; i < reci.length; i++)
            if (reci[i].equals("lep"))
                brojac++;

        return brojac;
    }

    static int brojReci(String tekst, String rec) {
        String[] reci = tekst.split(" ");
        int broj = 0;

        for (int i = 0; i < reci.length; i++)
            if ((reci[i].equalsIgnoreCase(rec))
                || (reci[i].equalsIgnoreCase(rec + ".")))
                broj++;

        return broj;
    }

    static String maxRec (String tekst){
        String[] reci = tekst.split(" ");

        int max_broj_pojavljivanja = 1;
        String max_rec = reci[0];

        for (int i = 0; i < reci.length; i++)
            if (brojReci(tekst,reci[i]) > max_broj_pojavljivanja){
                max_rec = reci[i];
                max_broj_pojavljivanja = brojReci(tekst,reci[i]);
            }

        return max_rec;
    }

    static String minRec (String tekst){
        String[] reci = tekst.split(" ");

        //Svaka rec ce sigurno da se pojavljuje manje
        //puta od ukupnog broja reci u tekstu +1.
        //Zato je to pocetna vrednost za min broj pojavljivanja.
        int min_broj_pojavljivanja = reci.length+1;
        String min_rec = reci[0];

        for (int i = 0; i < reci.length; i++)
            if (brojReci(tekst,reci[i]) < min_broj_pojavljivanja){
                min_rec = reci[i];
                min_broj_pojavljivanja = brojReci(tekst,reci[i]);
            }
    }
}
```

```

        return min_rec;
    }
}

class TestBrojacReci {

    public static void main(String[] args) {

        String min_rec = BrojacReci.minRec("Danas je lep dan. Bas lep.");
        String max_rec = BrojacReci.maxRec("Danas je lep dan. Bas lep.");

        System.out.println("Rec koja se najmanje puta pojavljuje u tekstu je: "+
            min_rec);
        System.out.println("Rec koja se najvise puta pojavljuje u tekstu je: "+
            max_rec);

    }

}

```

Zadatak 5

Potrebno je kreirati klasu **Avion** koja ima:

- Atribut sedišta koji je tipa niz Stringova. Ako je sedišta slobodno, odgovarajuća vrednost elementa niza je null. Ako nije, odgovarajući element niza sadrži ime i prezime putnika koji sedi na tom sedištu u formatu "IME PREZIME". Primer: "Milan Milanovic", "Bogdan Arsic" itd.
- Konstruktor koji inicijalizuje atribut sedišta. Avion ima tačno 50 mesta. Na početku su sva sedišta slobodna tj. potrebno je, u okviru konstruktora, inicijalizovati niz tako da sva sedišta budu slobodna.
- Metodu koja vraća ime i prezime putnika koji se nalazi na određenom sedištu. Broj sedišta je dat kao ulazni argument. Ako je sedišta slobodno, metoda vraća null.
- Metodu koja uvodi putnika na određeno mesto. Redni broj mesta je dat u vidu ulaznog argumenta. Ime i prezime putnika su takođe ulazni argumenti i dati su u vidu jednog Stringa (u formatu "IME PREZIME"). Ako je sedišta slobodno, putnik se uvodi na to mesto i ispisuje se poruka na ekranu koja sadrži ime i prezime putnika i broj mesta na koje je uveden. Ako sedišta nije slobodno, ispisuje se poruka o grešci koja sadrži redni broj mesta, ime putnika koji već sedi na tom mestu i ime putnika koji nije mogao biti raspoređen na to mesto.
- Metodu koja izvodi iz aviona putnika koji sedi na određenom sedištu. Broj sedišta je dat kao ulazni argument. Ako je sedišta bilo zauzeto, potrebno ga je osloboditi i ispisati poruku o tome na ekranu (poruka bi trebalo da sadrži i ime i prezime putnika koji se izvodi, kao i redni broj sedišta). Ako je sedišta već bilo slobodno, potrebno je ispisati poruku na ekranu o tome.
- Metodu koja proverava i vraća koliko ima slobodnih mesta.
- Metodu koja proverava i vraća koliko ima zauzetih mesta.
- Metodu koja na ekranu ispisuje imena (bez prezimena) svih putnika koji se prezivaju "Arsic",
- Metodu koja proverava i vraća koliko ima putnika koji se zovu "Nikola".
- Metodu koja na ekranu ispisuje status svakog sedišta u avionu. Ako je sedišta zauzeto, ispisuje se redni broj i ime i prezime putnika koji tu sedi, a ako je slobodno onda samo redni broj i poruka da je slobodno.

Potrebno je kreirati klasu TestAvion koja kreira dva objekta klase Avion. U prvi avion se uvode putnici "Petar Petrovic", "Natasa Jankovic" i "Nikola Arsic" na prvo, poslednje i 20. mesto. U drugi avion se uvode putnici "Masa Petrovic", "Nikola Tomic" i "Nikola Trajkovski" na 2., 10. i 20. mesto. Ispisati status svih sedišta u oba aviona. Ispisati imena svih putnika prvog aviona koji se prezivaju "Arsic" i broj putnika drugog aviona koji se zovu "Nikola".

Rešenje

```

class Avion {

    String[] sedista;

    Avion() {
        sedista = new String[50];
        for (int i=0; i<50; i++) sedista[i] = null;
    }
}

```



```

String getPutnik (int br_sedista){

    return sedista[br_sedista];
}

void uvediPutnika(int br_sedista,String ime_prezime){
    if (sedista[br_sedista] == null) {
        sedista[br_sedista] = ime_prezime;
        System.out.println("Putnik "+ime_prezime+" je uveden na "+
            br_sedista+". sediste");
    }
    else System.out.println("Putnik "+ime_prezime+" nije uveden na "+
        br_sedista+". sediste jer tu vec sedi putnik "+sedista[br_sedista]);
}

void izvediPutnika(int br_sedista){
    if (sedista[br_sedista] == null)
        System.out.println("Sediste br. "+br_sedista+" je vec bilo slobodno");
    else {
        System.out.println("Putnik "+sedista[br_sedista]+
            " koji je sedeo na sedistu br. "+br_sedista+" je izveden");
        sedista[br_sedista] = null;
    }
}

int brojSlobodnihMesta (){
    int broj_s = 0;
    for (int i=0; i<50; i++)
        if (sedista[i] == null) broj_s++;
    return broj_s;
}

int brojZauzetihMesta (){
    int broj_z = 0;
    for (int i=0; i<50; i++)
        if (sedista[i] != null) broj_z++;
    return broj_z;
}

void ispisiImenaArsica(){
    for (int i=0; i<50; i++)
        if (sedista[i]!=null){

            String prezime =
                sedista[i].substring(sedista[i].indexOf(' ')+1);

            if (prezime.equals("Arsic")){
                String ime =
                    sedista[i].substring(0,sedista[i].indexOf(' '));
                System.out.println(ime);
            }
        }
}

int brojPutnikaSaImenomNikola (){
    int brojac = 0;
    for (int i=0; i<50; i++)
        if (sedista[i]!=null){
            String ime = sedista[i].substring(0,
                sedista[i].indexOf(' '));
            if (ime.equals("Nikola")) brojac++;
        }
}

```

```

        return brojac;
    }

    void ispisiStatus() {
        for (int i=0; i<50; i++)
            if (sedista[i]==null)
                System.out.println("Sediste br. "+i+": slobodno");
            else
                System.out.println("Sediste br. "+i+": "+sedista[i]);
    }
}

class TestAvion {

    public static void main(String[] args) {

        Avion a1 = new Avion();
        Avion a2 = new Avion();

        a1.uvediPutnika(0, "Petar Petrovic");
        a1.uvediPutnika(49, "Natasa Jankovic");
        a1.uvediPutnika(19, "Nikola Arsic");

        a2.uvediPutnika(1, "Masa Petrovic");
        a2.uvediPutnika(9, "Nikola Tomic");
        a2.uvediPutnika(19, "Nikola Trajkovski");

        a1.ispisiStatus();
        a2.ispisiStatus();

        a1.ispisiImenaArsica();
        System.out.println("Broj putnika koji se zovu Nikola je: "+
            a2.brojPutnikaSaImenomNikola());
    }
}

```

Zadatak 6

Potrebno je kreirati klasu **OceneStudenata** koja ima:

- Atribut ocene koji je tipa niz Stringova. Svaki element niza predstavlja po jednog studenta i njegovu ocenu u vidu Stringa. Taj string je u formatu "BROJ-INDEKSA IME PREZIME OCENA". Primer "0064/99 Maja Vukovic 10", "0123/07 Pera Peric 6", "0001/03 Mika Lazic 8".
- Konstruktor koji inicijalizuje atribut ocene na određenu vrednost koja je data u vidu ulaznog argumenta tipa String. U okviru ovog Stringa se nalaze ocene za više studenata odvojene tačka-zarezom npr. "0064/99 Maja Vukovic 10;0123/07 Pera Peric 6;0001/03 Mika Lazic 8". Potrebno je razdvojiti ovaj String i popuniti elemente niza. Ako je String null, ispisuje se greška na ekranu.
- Metodu koja ispisuje na ekranu podatke o svim studentima.
- Metodu koja ispisuje na ekranu samo podatke o onim studentima koji su pali na ispitu.
- Metodu koja ispisuje na ekranu samo podatke o onim studentima koji su položili ispit.
- Metodu koja ispisuje na ekranu samo podatke o onim studentima koji su fakultet upisali 1999 godine.

Napraviti i klasu **TestOceneStudenata** koja kreira jedan objekat klase Ocene Studenata i puni ga vrednostima "0067/99 Bojan Vukovic 10;0103/06 Jelena Jovic 6;0001/99 Mika Lazic 5". Ispisati na ekranu studente koji su položili ispit, zatim one koji nisu položili ispit i na kraju one koji su fakultet upisali 1999. godine.

Rešenje

```

class OceneStudenata {

    String[] ocene;

```

```

OceneStudenata(String o){
    if (o == null) System.out.println("Greska!");
    else ocene = o.split(";");
}

void ispisi (){
    for (int i=0; i< ocene.length; i++){
        System.out.println(ocene[i]);
    }

    void ispisiStudenteKojiSuPali (){
        for (int i=0; i< ocene.length; i++){
            String ocena = ocene[i].substring(ocene[i].lastIndexOf(' ') + 1);
            if (ocena.equals("5"))
                System.out.println(ocene[i]);
        }
    }

    void ispisiStudenteKojiSuPolozili (){
        for (int i=0; i< ocene.length; i++){
            String ocena = ocene[i].substring(ocene[i].lastIndexOf(' ') + 1);
            if (!ocena.equals("5"))
                System.out.println(ocene[i]);
        }
    }

    void ispisiStudenteKojiSuUpisali99 (){
        for (int i=0; i< ocene.length; i++){
            String god = ocene[i].substring(5, 7);
            if (god.equals("99"))
                System.out.println(ocene[i]);
        }
    }
}

class TestOceneStudenata {

    public static void main(String[] args) {

        OceneStudenata os =
            new OceneStudenata(
                "0067/99 Bojan Vukovic 10;0103/06 Jelena Jovic 6;0001/99 Mika Lazic 5");

        os.ispisiStudenteKojiSuPolozili();
        os.ispisiStudenteKojiSuPali();
        os.ispisiStudenteKojiSuUpisali99();
    }
}

```