

## Rad sa tekstualnim fajlovima

Tekstualni fajlovi su oni koji sadrže samo tekst napisan u jednom ili više redova i moгу se ručno kreirati i čitati korišćenjem raznih uslužnih programa (npr. Notepad). Potrebno je napomenuti da ovi fajlovi sadrže samo tekst (znakove), a ne i podatke o formatiranju za taj tekst (font, razmak između redova itd.) pa se Word fajlovi ne smatraju za tekstualne fajlove, već za binarne fajlove.

Učitavanje iz tekstualnog fajla je veoma slično učitavanju sa tastature. Zbog toga se i u jednom i u drugom slučaju koristi klasa **BufferedReader**, samo što joj se, pri inicijalizaciji, prosleđuje drugi tip ulaznog toka (a ne standardni ulazni tok - tastatura). U pitanju je klasa **FileReader** koja omogućava učitavanje pojedinačnih slova iz tekstualnog fajla. Konstruktoru klase **FileReader** se, kao ulazni parametar, prosleđuje naziv tekstualnog fajla iz kojeg se želi učitavati. Fajl sa unetim nazivom mora postojati, jer će u suprotnom konstruktor baciti izuzetak. I u ovom slučaju, klasa **BufferedReader** obezbeđuje privremenu memoriju koja povećava efikasnost učitavanja.

```
BufferedReader naziv_promenljive =  
    new BufferedReader(new FileReader(...naziv fajla...));
```

Učitavanje se vrši pozivanjem “readLine” metode, koja učitava jedan red znakova iz tekstualnog fajla. Učitavanje se može vršiti sve dok u fajlu postoji još redova. Kada se stigne do kraja fajla, metoda umesto String vrednosti vraća vrednost null.

Sve što se učitava iz tekstualnog fajla je u formi String vrednosti. Ako je potrebno učitati neke brojeve ili boolean vrednosti iz tekstualnog fajla, mora se izvršiti konverzija na isti način kao i kod učitavanja sa tastature - korišćenjem odgovarajućih “parse” metoda klase **Double**, **Integer** i **Boolean**.

Kada se završi sa učitavanjem iz tekstualnog fajla, potrebno je zatvoriti tok za učitavanje. To se radi pozivom metode “close”.

### Primer 1

Napisati klasu **FileIO** koja ima:

- Javnu statičku metodu koja učitava tekst iz tekstualnog fajla “tekst.txt” i ispisuje ga na ekranu.

```
import java.io.*;  
  
public class FileIO {  
  
    public static void ucitajIIsipisi() {  
        try {  
            BufferedReader in =  
                new BufferedReader(new FileReader("tekst.txt"));  
  
            boolean kraj = false;  
            String s = "";  
  
            while (!kraj) {  
                String pom = in.readLine();
```

```

        if (pom == null) kraj = true;
        else s=s+pom+" ";
    }

    in.close();

    System.out.println(s);
} catch (Exception e) {
    System.out.println("Greska: "+e.getMessage());
}
}
}

```

Upisivanje u tekstualni fajl u Javi se vrši korišćenjem **PrintWriter** klase. Konstruktoru ove klase se prosleđuje objekat klase **BufferedWriter**. Ova klasa samo dodaje privremenu memoriju izlaznom toku klase **FileWriter**. Ako fajl čiji naziv je unet kao ulazni parametar konstruktora klase **FileWriter** ne postoji, biće napravljen.

```

PrintWriter naziv_promenljive =
    new PrintWriter(
        new BufferedWriter(
            new FileWriter(...naziv fajla...)));

```

Upisivanje u tekstualni fajl se vrši pozivanjem **metode “write”** koja kao ulazni parametar prima **String** koji je potrebno upisati. Potrebno je primetiti da ova metoda pri upisivanju **ne dodaje znak za kraj reda**. Drugim rečima, ako je potrebno dodati znak za kraj reda, on se mora ručno dodati na kraj **String** vrednosti koja se upisuje. **Znak za kraj reda se u Javi označava sa '\n'.**

## Primer 2

Dodati u klasu **FileIO**:

- Javnu statičku metodu koja u fajl “tekst2.txt” upisuje dva **Stringa** - “Danas je lep dan.” i “Možda će temperatura biti iznad nule.” i to tako da svaki **String** bude u posebnom redu.

Napisati klasu **TestFileIO** koja poziva metode klase **FileIO**.

```

public static void upisiStringove() {
    try{
        PrintWriter out = new PrintWriter(
            new BufferedWriter(new FileWriter("tekst2.txt")));

        out.write("Danas je lep dan."+'\n');
        out.write("Mozda ce temperatura biti iznad nule."+'\n');

        out.close();
    } catch (Exception e) {
        System.out.println("Greska: "+e.getMessage());
    }
}

public class TestFileIO {

    public static void main (String[] args){

        FileIO.ucitajIIspisi();
        FileIO.upisiStringove();
    }
}

```

```
}
```

```
}
```

## Zadaci

### Zadatak 1

Napraviti klasu **IORadSaTekstom** koja ima:

- Javnu statičku metodu koja učitava tekst iz fajla "tekst.txt" i prikazuje ga na ekranu.
- Javnu statičku metodu koja učitava tekst iz fajla i prikazuje ga na ekranu. Ime fajla iz kojeg se učitava tekst je dato u vidu ulaznog argumenta. Ako je kao ime fajla unet null String, potrebno je baciti izuzetak sa odgovarajućom porukom.
- Javnu statičku metodu koja učitava tekst iz fajla i na ekranu prikazuje broj reči u tekstu. Ime fajla iz kojeg se učitava tekst je dato u vidu ulaznog argumenta. Ako je kao ime fajla unet null String, potrebno je baciti izuzetak sa odgovarajućom porukom.
- Javnu statičku metodu koja učitava tekst iz fajla "tekst2.txt" i proverava da li se određena reč nalazi u tekstu. Metoda vraća TRUE ako se reč nalazi u tekstu a FALSE ako se ne nalazi (metoda vraća FALSE i ako se desi neka greška). Tražena reč je data u vidu ulaznog argumenta. Potrebno je obuhvatiti i one situacije u kojima se reč pojavljuje na početku ili na kraju rečenice (rečenica se uvek završava tačkom).
- Javnu statičku metodu koja učitava tekst iz fajla "tekst3.txt" i na ekranu ispisuje broj rečenica u tekstu. Rečenica može da se završi tačkom, znakom pitanja ili znakom uzvika.
- Javnu statičku metodu koja kao ulazni argument prima String i upisuje ga u fajl "tekst4.txt".
- Javnu statičku metodu koja kao ulazni argument prima String i u fajl "tekst5.txt" upisuje samo one reči iz tog String-a koje imaju više od 5 slova.
- Javnu statičku metodu koja kao ulazni argument prima String i u fajl "tekst6.txt" upisuje samo one reči iz tog String-a koje počinju velikim slovom "A".

Napraviti klasu **TestIORadSaTekstom** koja poziva metode klase IORadSaTekstom.

### Rešenje:

```
import java.io.*;

public class IORadSaTekstom {

    public static void ucitajIISPisi(){
        try{
            BufferedReader in =
                new BufferedReader (new FileReader("tekst.txt"));

            boolean kraj = false;
            String s = "";

            //Posto se unapred ne zna koliko linija teksta postoji
            //u fajlu, koristi se while petlja i indikator (kraj).
            //Kada se dodje do kraja fajla metoda readLine vraca
            //null String, indikator kraj dobija vrednost true i
            //while petlja se prekida. Tekst koji se ucitava se
            //sve vreme (liniju po liniju) dodaje u jedan String (s).
            while (!kraj){
                String pom = in.readLine();
                if (pom == null) kraj = true;
                else s=s+pom+" ";
            }

            in.close();

            System.out.println(s);
        } catch (Exception e){
            System.out.println("Greska: "+e.getMessage());
        }
    }
}
```

```

public static void ucitajIISPisi(String fajl){
    if (fajl == null)
        throw new RuntimeException("Ime fajla ne moze biti null");

    try{
        BufferedReader in =
            new BufferedReader (new FileReader(fajl));

        boolean kraj = false;
        String s = "";

        while (!kraj){
            String pom = in.readLine();
            if (pom == null) kraj = true;
            else s=s+pom+" ";
        }

        in.close();

        System.out.println(s);
    } catch (Exception e){
        System.out.println("Greska: "+e.getMessage());
    }
}

public static void ucitajIISPisiBrojReci(String fajl){
    if (fajl == null)
        throw new RuntimeException("Ime fajla ne moze biti null");

    try{
        BufferedReader in =
            new BufferedReader (new FileReader(fajl));

        boolean kraj = false;
        String s = "";

        while (!kraj){
            String pom = in.readLine();
            if (pom == null) kraj = true;
            else s=s+pom+" ";
        }

        in.close();

        String[] reci = s.split(" ");
        System.out.println("Broj reci u tekstu je: "+reci.length);
    } catch (Exception e){
        System.out.println("Greska: "+e.getMessage());
    }
}

public static boolean ucitajIProveriRec(String rec){
    try{
        BufferedReader in =
            new BufferedReader (new FileReader("tekst2.txt"));

        boolean kraj = false;
        String s = "";

        while (!kraj){
            String pom = in.readLine();
            if (pom == null) kraj = true;
            else s=s+pom+" ";
        }
    }
}

```

```

    }

    in.close();

    String[] reci = s.split(" ");
    for (int i=0; i<reci.length; i++)
        if (reci[i].equalsIgnoreCase(rec) ||
            reci[i].equals(rec+".")) return true;

    return false;
} catch (Exception e) {
    System.out.println("Greska: "+e.getMessage());
    return false;
}
}

public static void učitajIIspisiBrojRecenica() {
    try {
        BufferedReader in =
            new BufferedReader (new FileReader("tekst3.txt"));

        boolean kraj = false;
        String s = "";

        while (!kraj) {
            String pom = in.readLine();
            if (pom == null) kraj = true;
            else s=s+pom+" ";
        }

        in.close();

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

        System.out.println("Broj recenica u tekstu je: "+brojac);
    } catch (Exception e) {
        System.out.println("Greska: "+e.getMessage());
    }
}

public static void upisiString(String tekst) {
    try {
        PrintWriter out = new PrintWriter(
            new BufferedWriter(new FileWriter("tekst4.txt")));

        out.write(tekst);

        out.close();
    } catch (Exception e) {
        System.out.println("Greska: "+e.getMessage());
    }
}

public static void upisiReciSaViseOdPetSlova(String tekst) {
    try {
        PrintWriter out = new PrintWriter(
            new BufferedWriter(new FileWriter("tekst5.txt")));

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

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

```

```

        if (reci[i].length()>5)
            out.write(reci[i]+" ");

        out.close();
    } catch (Exception e) {
        System.out.println("Greska: "+e.getMessage());
    }
}

public static void upisiReciSaVelikimSlovomA(String tekst){
    try{
        PrintWriter out = new PrintWriter(
            new BufferedWriter(new FileWriter("tekst6.txt")));

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

        for(int i=0; i<reci.length;i++)
            if (reci[i].charAt(0)=='A')
                out.write(reci[i]+" ");

        out.close();
    } catch (Exception e) {
        System.out.println("Greska: "+e.getMessage());
    }
}

}

public class TestIORadSaTekstom {

    public static void main(String[] args) {

        IORadSaTekstom.ucitajIIspisi();
        IORadSaTekstom.ucitajIIspisi("tekst.txt");
        IORadSaTekstom.ucitajIIspisiBrojReci("tekst.txt");
        if(IORadSaTekstom.ucitajIProveriRec("lep"))
            System.out.println("Rec 'lep' se nalazi u tekstu");
        else
            System.out.println("Rec 'lep' se ne nalazi u tekstu");
        IORadSaTekstom.ucitajIIspisiBrojRecenica();

        IORadSaTekstom.upisiString("Danas je bas lep dan.");
        IORadSaTekstom.upisiReciSaViseOdPetSlova(
            "Automobil je brzi od bicikla");
        IORadSaTekstom.upisiReciSaVelikimSlovomA(
            "Automobili su mnogo brzi danas nego pre 50 godina");
    }
}

```

## Zadatak 2

Napraviti klasu **IOAnalizatorTeksta** koja ima:

- Javnu statičku metodu koja učitava tekst iz fajla i na ekranu ispisuje tekst tako da svaka reč bude u posebnom redu i da između svaka dva slova bude po jedan blanko znak. To znači da bi se npr. reč “kola” napisala kao “k o l a”. Ime fajla iz kojeg se učitava tekst je dato u vidu ulaznog argumenta. Ako je kao ime fajla unet null String, potrebno je baciti izuzetak sa odgovarajućom porukom.
- Javnu statičku metodu koja učitava tekst iz fajla “tekst7.txt” i ispisuje ga na ekranu tako da se u svakom redu nalazi tačno 10 reči. Naravno, izuzetak je poslednji red u kome može biti i manje reči.
- Javnu statičku metodu koja učitava tekst iz fajla. Potrebno je reči iz tog teksta svesti na 5 slova na sledeći način: reči koje sadrže više od 5 slova skratiti 'odsecanjem' viška slova; reči koje sadrže manje od 5 slova 'produžiti' dodavanjem poslednjeg slova onoliko puta koliko je potrebno da se dostigne dužina od 5 slova; reči koje sadrže tačno pet slova ostaviti takvim kakve jesu. Onda je potrebno na ekranu ispisati uporedni prikaz

svih reči iz teksta u originalnom i izmenjenom obliku. Prikaz bi trebalo da bude oblika: "originalna reč -> izmenjena reč": "danas -> danas", "dan -> dannn", "Prelepo -> Prele". Ime fajla iz kojeg se učitava tekst je dato u vidu ulaznog argumenta. Ako je kao ime fajla unet null String, potrebno je baciti izuzetak sa odgovarajućom porukom.

- Javnu statičku metodu koja kao ulazni argument ima String koji predstavlja tekst i String koji predstavlja reč i vraća broj ponavljanja te reči u tekstu. Trebalo bi obuhvatiti i one situacije kada je reč na početku ili na kraju rečenice (rečenica se uvek završava tačkom).
- Javnu statičku metodu koja učitava tekst iz fajla "tekst9.txt", i u fajl "tekst10.txt" upisuje samo one reči koje se više puta pojavljuju u tekstu.
- Javnu statičku metodu koja učitava tekst iz fajla "tekst8.txt", prebrojava reči, pronalazi najkraću i najdužu reč i u fajl "izvestaj.txt" unosi izveštaj koji se sastoji iz tri posebna reda teksta koji su u sledećoj formi: "Ukupan broj reci: ##", "Najduza rec: ##", "Najkraca rec: ##".

Napraviti klasu **TestIOAnalizatorTeksta** koja poziva metode klase IOAnalizatorTeksta.

**Rešenje:**

```
import java.io.*;

public class IOAnalizatorTeksta {

    public static void ucitajIIispisiUOdvojenomRedu(String fajl){
        if (fajl == null)
            throw new RuntimeException("Ime fajla ne moze biti null");

        try{
            BufferedReader in =
                new BufferedReader (new FileReader(fajl));

            boolean kraj = false;
            String s = "";

            while (!kraj){
                String pom = in.readLine();
                if (pom == null) kraj = true;
                else s=s+pom+" ";
            }

            in.close();

            //Posto se svaka rec ispisuje u posebnom redu, prvo
            //je potrebno podeliti tekst na reci. To se radi
            //koriscenjem split naredbe.
            String[] reci = s.split(" ");

            //For petlja je potrebna da bi se svaka rec ispisala posebno.
            for (int i=0; i<reci.length;i++){

                //Ova druga, "unutrasnja" for petlja prolazi kroz svaku
                //pojedinačnu rec(String)i ispisuje je slovo po slovo (izmedju
                //svaka dva slova stavlja po jedan prazan znak). Primetite
                //da se koristi print a ne println naredba da bi se sva
                //slova jedne reci ispisala u istom redu.
                for (int j=0; j<reci[i].length();j++)
                    System.out.print(reci[i].charAt(j)+" ");

                //Ova naredba služi samo da prebaci ispisivanje u sledeci
                //red kada se ispise neka rec (da bi se sledeca rec ispisala
                //u novom redu).
                System.out.println();
            }
        }
    }
}
```

```

    } catch (Exception e) {
        System.out.println("Greska: "+e.getMessage());
    }
}

public static void ucitajIIispisiPoDesetReci(String fajl){
    if (fajl == null)
        throw new RuntimeException("Ime fajla ne moze biti null");

    try{
        BufferedReader in =
            new BufferedReader (new FileReader(fajl));

        boolean kraj = false;
        String s = "";

        while (!kraj){
            String pom = in.readLine();
            if (pom == null) kraj = true;
            else s=s+pom+" ";
        }

        in.close();

        //Posto je potrebno ispisato po deset reci u redu,
        //prvo se ceo tekst podeli na reci.
        String[] reci = s.split(" ");

        //Ova for petlja prolazi kroz niz String-ova koji
        //predstavlja reci u tekstu i ispisuje ih u istom
        //redu (komanda print). Tek kada naidje na svaku desetu, on nju
        //ispise i prebaci dalje ispisivanje u sledeci red (println).
        //Potrebno je приметiti da se kod ispisivanja u istom redu
        //dodaje po jedan blanko znak posle svake reci da se
        //reci ne bi "lepile" jedna za drugu. Takodje, utvrdjivanje
        //svake desete reci se vrši tako sto se proverava da li je
        //indeks deljiv sa deset (stoji (i+1)%10 a ne i%10 jer je
        //po definiciji 0 deljiva sa 10 pa bi se, greskom, u prvom redu
        //ispisala samo prva rec).
        for (int i=0; i<reci.length;i++){
            if (((i+1)%10) == 0) System.out.println(reci[i]);
            else System.out.print(reci[i]+" ");
        }

    } catch (Exception e) {
        System.out.println("Greska: "+e.getMessage());
    }
}

public static void ucitajIIispisiReciSvedeneNaPetSlova(String fajl){
    if (fajl == null)
        throw new RuntimeException("Ime fajla ne moze biti null");

    try{
        BufferedReader in =
            new BufferedReader (new FileReader(fajl));

        boolean kraj = false;
        String s = "";

        while (!kraj){
            String pom = in.readLine();
            if (pom == null) kraj = true;

```



```

        else s=s+pom+" ";
    }

    in.close();

    String[] reci = s.split(" ");

    //For petlja je potrebna da bi se svaka rec ispisala posebno.
    //Ako je duzina reci pet slova ili vise, ispisuje se samo prvih
    //pet slova reci (ako rec ima tacno 5 slova, ispisuje se cela rec).
    //Ako rec ima manje od 5 slova, prvo se napravi pomocni String koj
    //se sastoji od originalne reci i dodatih 5 poslednjih slova te
    //originalne reci. Ovako formiran String je sigurno duzi od 5 slova
    //pa se onda kao rezultat ispisuje samo prvih 5 slova pomocnog
    //String-a - sto je upravo ta izmenjena rec koja se trazi.
    for (int i=0; i<reci.length;i++)
        if (reci[i].length()>=5)
            System.out.println(reci[i]+" -> "
                                +reci[i].substring(0, 5));
        else{
            char poslednjeSlovo =
                reci[i].charAt(reci[i].length()-1);

            String pom = reci[i]+poslednjeSlovo+poslednjeSlovo+
                poslednjeSlovo+poslednjeSlovo+poslednjeSlovo;

            System.out.println(reci[i]+" -> "+pom.substring(0, 5));
        }

    } catch (Exception e) {
        System.out.println("Greska: "+e.getMessage());
    }
}

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

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

    return broj;
}

public static void ucitajIUpisiReciKojeSePonavljaju(){
    try{
        BufferedReader in =
            new BufferedReader (new FileReader("tekst9.txt"));

        boolean kraj = false;
        String s = "";

        while (!kraj){
            String pom = in.readLine();
            if (pom == null) kraj = true;
            else s=s+pom+" ";
        }

        in.close();

        PrintWriter out = new PrintWriter(
            new BufferedWriter(new FileWriter("tekst10.txt")));
    }
}

```

```

        String[] reci = s.split(" ");

        for (int i=0; i<reci.length;i++)
            if (brojPonavljanja(s, reci[i])>1)
                out.write(reci[i]+" ");

        out.close();

    } catch (Exception e) {
        System.out.println("Greska: "+e.getMessage());
    }
}

public static void učitajIUpisiIzvestajORecima() {
    try {
        BufferedReader in =
            new BufferedReader (new FileReader("tekst8.txt"));

        boolean kraj = false;
        String s = "";

        while (!kraj) {
            String pom = in.readLine();
            if (pom == null) kraj = true;
            else s=s+pom+" ";
        }

        in.close();

        PrintWriter out = new PrintWriter(
            new BufferedWriter(new FileWriter("izvestaj.txt")));

        String[] reci = s.split(" ");

        //Znak "\n" predstavlja kraj reda. Kada se ovaj znak
        //upise u fajl, dalje upisivanje teksta se vrši u
        //sledecem redu.
        out.write("Ukupan broj reci je: "+reci.length+"\n");

        String maxRec = reci[0];
        for (int i=0; i<reci.length;i++)
            if (reci[i].length()>maxRec.length())
                maxRec = reci[i];

        out.write("Najduza rec je: "+maxRec+"\n");

        String minRec = reci[0];
        for (int i=0; i<reci.length;i++)
            if (reci[i].length()<minRec.length())
                minRec = reci[i];

        out.write("Najkraca rec je: "+minRec);

        out.close();

    } catch (Exception e) {
        System.out.println("Greska: "+e.getMessage());
    }
}

}

public class TestIOAnalizatorTeksta {

```

```

    public static void main(String[] args) {

        IOAnalizatorTeksta.ucitajIIspisiUOdvojenomRedu("tekst.txt");
        IOAnalizatorTeksta.ucitajIIspisiPoDesetReci("tekst.txt");
        IOAnalizatorTeksta.ucitajIIspisiReciSvedeneNaPetSlova("tekst.txt");
        IOAnalizatorTeksta.ucitajIUpisiReciKojeSePonavljaJu();
        IOAnalizatorTeksta.ucitajIUpisiIzvestajORecima();

    }

}

```

### Zadatak 3

Napraviti klasu **Djak** koja ima:

- Privatni atribut ime.
- Privatni atribut prezime.
- Privatni atribut ocena.
- Odgovarajuće javne get i set metode za ove atribute. Ime i prezime ne smeju da budu null, a ocena mora da bude u rasponu od 1 do 5. U slučaju unosa nedozvoljenih vrednosti, potrebno je baciti izuzetak sa odgovarajućom porukom.
- Redefinisani metodu toString klase Object koja vraća String sa svim podacima o đaku bez propratnog teksta i to u formi: "IME PREZIME OCENA".
- Redefinisani equals metodu klase Object koja kao ulazni argument dobija objekat klase Object. Metoda prvo proverava da li je uneti objekat klase Djak, pa ako nije baca izuzetak. Metoda vraća true ako su ime i prezime jednaki imenu i prezimenu unetog đaka. Ako ime i prezime nisu jednaki, metoda vraća false.

Napraviti klasu **Odeljenje** koja ima:

- Privatni atribut djaci koji predstavlja listu objekata klase Djak. Ovu listu je potrebno odmah inicijalizovati.
- Javnu metodu ucitajIzFajla koja ucitava podatke o đacima iz tekstualnog fajla "djaci.txt" i puni listu. Podaci o svakom đaku su upisani u poseban red u fajlu u formi "IME PREZIME OCENA". Pri učitavanju je potrebno obratiti pažnju na to da se u odeljenju jedan isti đak ne može više puta pojavljivati (pri pojavi duplikata ne treba vršiti unos u listu).
- Javnu metodu koja u tekstualni fajl "odlicni\_djaci.txt" upisuje samo podatke o onim đacima koji imaju ocenu 5. Podaci o svakom đaku se upisuju u poseban red u formi "IME PREZIME OCENA".
- Javnu metodu koja u tekstualni fajl "losi\_djaci.txt" upisuje samo podatke o onim đacima koji imaju ocenu 1 ili 2. Podaci o svakom đaku se upisuju u poseban red u formi "IME PREZIME OCENA".
- Javnu metodu koja na ekranu ispisuje podatke o svim đacima iz odeljenja.

Napraviti klasu **TestOdeljenje** koja kreira jedan objekat klase **Odeljenje** i testira njegove metode.

**Rešenje:**

```

public class Djak {

    private String ime;
    private String prezime;
    private int ocena;

    public String getIme() {
        return ime;
    }

    public void setIme(String ime) {
        if (ime == null)
            throw new RuntimeException("Ime ne moze biti null");

        this.ime = ime;
    }

    public int getOcena() {
        return ocena;
    }

    public void setOcena(int ocena) {
        if (ocena < 1 || ocena > 5)

```

```

        throw new RuntimeException("Ocena mora biti u rasponu 1-5");

        this.ocena = ocena;
    }
    public String getPrezime() {
        return prezime;
    }
    public void setPrezime(String prezime) {
        if (prezime == null)
            throw new RuntimeException("Prezime ne moze biti null");

        this.prezime = prezime;
    }

    public String toString(){
        return ime+" "+prezime+" "+ocena;
    }

    public boolean equals (Object o){
        if (!(o instanceof Djak))
            throw new RuntimeException("Morate uneti objekat klase Djak");

        Djak dj = (Djak) (o);
        if (ime.equals(dj.getIme()) &&
            prezime.equals(dj.getPrezime()))
            return true;
        else
            return false;
    }
}

import java.io.*;
import java.util.LinkedList;

public class Odeljenje {

    private LinkedList <Djak> djaci = new LinkedList<Djak>();

    public void ucitajIzFajla(){
        try{
            BufferedReader in =
                new BufferedReader(new FileReader("djaci.txt"));

            boolean kraj = false;

            while (!kraj){
                String pom = in.readLine();
                if (pom == null)
                    kraj = true;
                else{
                    Djak dj = new Djak();

                    String ime = pom.substring(0, pom.indexOf(' '));
                    String prezime = pom.substring(pom.indexOf(' ')+1,
                        pom.lastIndexOf(' '));
                    int ocena = Integer.parseInt(
                        pom.substring(pom.lastIndexOf(' ')+1));

                    dj.setIme(ime);
                    dj.setPrezime(prezime);
                    dj.setOcena(ocena);

                    if (!djaci.contains(dj)) djaci.add(dj);
                }
            }
        }
    }
}

```

```

        }
    }

    in.close();
} catch (Exception e) {
    System.out.println("Greska: "+e.getMessage());
}

}

public void upisiOdlicne () {
    try {
        PrintWriter out =
            new PrintWriter(new FileWriter("odlicni_djaci.txt"));

        for (int i=0; i<djaci.size();i++)
            if (djaci.get(i).getOcena() == 5)
                out.write(djaci.get(i)+"\n");

        out.close();
    } catch (Exception e) {
        System.out.println("Greska: "+e.getMessage());
    }
}

public void upisiLose () {
    try {
        PrintWriter out =
            new PrintWriter(new FileWriter("losi_djaci.txt"));

        for (int i=0; i<djaci.size();i++)
            if (djaci.get(i).getOcena() <= 2)
                out.write(djaci.get(i)+"\n");

        out.close();
    } catch (Exception e) {
        System.out.println("Greska: "+e.getMessage());
    }
}

public void ispisi() {
    for (int i=0; i<djaci.size();i++)
        System.out.println(djaci.get(i));
}

}

public class TestOdeljenje {

    public static void main(String[] args) {

        Odeljenje o = new Odeljenje();

        o.ucitajIzFajla();
        o.upisiOdlicne();
        o.upisiLose();

        o.ispisi();

    }

}

```

#### Zadatak 4

Napraviti klasu **Proizvod** koja ima:

- Privatni atribut sifra (ceo broj).
- Privatni atribut naziv.
- Privatni atribut kolicina (ceo broj).
- Odgovarajuće get i set metode za ove attribute. Naziv ne sme biti null, a količina i šifra moraju biti brojevi koji su veći od nule ili eventualno nula. U slučaju unosa nedozvoljenih vrednosti, potrebno je baciti izuzetak sa odgovarajućom porukom.
- Redefinisati toString metodu klase Object koja vraća String sa podacima o proizvodu bez dodatnog teksta u formi "SIFRA NAZIV KOLICINA".
- Redefinisati equals metodu klase Object koja kao ulazni argument dobija objekat klase Object. Metoda prvo proverava da li je uneti objekat klase Proizvod, pa ako nije baca izuzetak. Metoda vraća true ako je šifra jednaka šifri unetog proizvoda. Ako šifre nisu jednake, metoda vraća false.

Napraviti klasu **Magacin** koja ima:

- Privatni atribut proizvodi koji predstavlja listu objekata klase Proizvod.
- Javni konstruktor koji inicijalizuje atribut proizvodi.
- Javnu metodu koja učitava podatke o proizvodima iz tekstualnog fajla "proizvodi.txt" i puni listu. Podaci o svakom proizvodu su dati u posebnom redu u formatu "SIFRA NAZIV KOLICINA". Pri učitavanju je potrebno obratiti pažnju na to da se u listi ne smeju ponavljati isti proizvodi (pri pojavi duplikata ne treba vršiti unos u listu).
- Javnu metodu koja sastavlja listu nabavke i upisuje je u fajl "nabavka.txt". Listu nabavke čine oni proizvodi kojih ima vrlo malo u magacinu tj. količina im manja od 5. Tekst nabavke treba da se sastoji od jednog reda koji predstavlja zaglavlje (npr. "Proizvodi koje je potrebno nabaviti") i više redova koji predstavljaju proizvode koje je potrebno nabaviti. Podatke o svakom proizvodu je potrebno staviti u poseban red u formi "RBR SIFRA NAZIV KOLICINA" gde RBR predstavlja redni broj stavke u nabavci.
- Javnu metodu koja učitava podatke iz tekstualnog fajla "dopuna.txt" i dopunjava količine proizvoda u magacinu. Ovaj fajl predstavlja proizvode koji su upravo pristigli u magacin, pa se ažuriranje vrši na sledeći način. Ako je pristigao novi proizvod (ne postoji u magacinu) onda se dodaje u listu. Ako su stigle dodatne količine već postojećeg proizvoda, onda se one dodaju na postojeću količinu.
- Javnu metodu koja na ekranu ispisuje podatke o proizvodima u magacinu.

Napraviti klasu **TestMagacin** koja kreira jedan objekat klase Magacin i poziva njegove metode.

**Rešenje:**

```
public class Proizvod {  
  
    private int sifra;  
    private String naziv;  
    private int kolicina;  
  
    public int getKolicina() {  
        return kolicina;  
    }  
    public void setKolicina(int kolicina) {  
        if (kolicina < 0)  
            throw new RuntimeException(  
                "Kolicina ne sme biti manja od nule");  
  
        this.kolicina = kolicina;  
    }  
    public String getNaziv() {  
        return naziv;  
    }  
    public void setNaziv(String naziv) {  
        if (naziv == null)  
            throw new RuntimeException("Naziv ne sme biti null");  
  
        this.naziv = naziv;  
    }  
    public int getSifra() {  
        return sifra;  
    }  
}
```

```

    public void setSifra(int sifra) {
        if (sifra < 0)
            throw new RuntimeException("Sifra ne sme biti manja od nule");

        this.sifra = sifra;
    }

    public String toString(){
        return sifra+" "+naziv+" "+kolicina;
    }

    public boolean equals (Object o){
        if (!(o instanceof Proizvod))
            throw new RuntimeException("Morate uneti objekat klase Proizvod");

        Proizvod p = (Proizvod) (o);
        if (sifra == p.getSifra())
            return true;
        else
            return false;
    }
}

import java.io.*;
import java.util.LinkedList;

public class Magacin {

    private LinkedList<Proizvod> proizvodi;

    public Magacin(){
        proizvodi = new LinkedList<Proizvod>();
    }

    public void učitajIzFajla(){
        try{
            BufferedReader in =
                new BufferedReader(new FileReader("proizvodi.txt"));

            boolean kraj = false;

            while (!kraj){
                String pom = in.readLine();
                if (pom == null)
                    kraj = true;
                else{
                    Proizvod p = new Proizvod();

                    int sifra =
                        Integer.parseInt(pom.substring(0,
                            pom.indexOf(' ')));
                    String naziv =
                        pom.substring(pom.indexOf(' ')+1,
                            pom.lastIndexOf(' '));
                    int kolicina = Integer.parseInt(
                        pom.substring(pom.lastIndexOf(' ')+1));

                    p.setSifra(sifra);
                    p.setNaziv(naziv);
                    p.setKolicina(kolicina);

                    if (!proizvodi.contains(p)) proizvodi.add(p);
                }
            }
        }
    }
}

```

```

        }

        in.close();
    } catch (Exception e) {
        System.out.println("Greska: "+e.getMessage());
    }
}

public void sastaviListuNabavke() {
    try {
        PrintWriter out =
            new PrintWriter(new FileWriter("nabavka.txt"));

        out.write("Potrebno je nabaviti sledece proizvode\n");

        int rb=1;
        for(int i=0;i<proizvodi.size();i++)
            if (proizvodi.get(i).getKolicina()< 5) {
                out.write(rb+" "+proizvodi.get(i)+"\n");
                rb++;
            }

        out.close();
    } catch (Exception e) {
        System.out.println("Greska: "+e.getMessage());
    }
}

public void dopuniIzFajla() {
    try {
        BufferedReader in =
            new BufferedReader(new FileReader("dopuna.txt"));

        boolean kraj = false;

        while (!kraj) {
            String pom = in.readLine();
            if (pom == null)
                kraj = true;
            else {
                Proizvod p = new Proizvod();

                int sifra =
                    Integer.parseInt(pom.substring(0,
                        pom.indexOf(' ')));

                String naziv =
                    pom.substring(pom.indexOf(' ')+1,
                        pom.lastIndexOf(' '));

                int kolicina = Integer.parseInt(
                    pom.substring(pom.lastIndexOf(' ')+1));

                p.setSifra(sifra);
                p.setNaziv(naziv);
                p.setKolicina(kolicina);

                if (proizvodi.contains(p)) {
                    Proizvod stariProizvod =
                        proizvodi.get(proizvodi.indexOf(p));

                    stariProizvod.setKolicina(stariProizvod.getKolicina()+p.getKolicina());
                }
                else
                    proizvodi.add(p);
            }
        }
    }
}

```



```

        }

        in.close();
    } catch (Exception e) {
        System.out.println("Greska: "+e.getMessage());
    }
}

public void ispisi() {
    for (int i=0; i<proizvodi.size(); i++)
        System.out.println(proizvodi.get(i));
}

}

public class TestMagacin {

    public static void main(String[] args) {

        Magacin m = new Magacin();

        m.ucitajIzFajla();

        m.ispisi();

        m.sastaviListuNabavke();

        m.dopuniIzFajla();

        m.ispisi();

    }

}

```