

## Operații de I/O

Pachetul `java.io` este un set de clase cu ajutorul cărora se realizează operațiile de intrare/iesire într-un program Java.

În Java operațiile de intrare/iesire se bazează pe conceptul de *flux de date (stream)*.

Un *flux de date* este o secvență de date care se deplasează din spate o sursă externă spre memorie - caz în care avem de a face cu un flux de intrare (input stream) sau din memorie spre o destinație externă - flux de iesire (output stream).

Pentru operațiile de intrare/iesire cele mai frecvente sursă externă este tastatura, iar destinația este ecranul monitorului. Acestea se mai numesc și suporturi standard de intrare, respectiv iesire. Corespunzător suporturilor standard, în Java există 2 obiecte predefinite: `System.in` pentru tastatura și `System.out` pentru monitor.

**Observatie:** `in` și `out` sunt variabile membru statice ale clasei `System` definită în pachetul `java.lang`. Tipurile acestor variabile sunt clase din pachetul `java.io`, și anume: `System.in` este o referință la clasa `InputStream`, iar `System.out` este o referință la clasa `PrintStream`.

Dacă dorim să citim sau să scriem date din/in fisiere de pe disc, trebuie create și utilizate alte obiecte ale unor claselor din pachetul `java.io`. În continuare vom studia posibilitatea de a efectua operații de intrare/iesire la nivel de linii de caractere (adică fisiere de text).

**1. Operația de citire** a liniilor de text Clasa care modelează citirea unei linii dintr-un flux de intrare este `BufferedReader`, prin operația `readLine`.

Aceasta operație nu are parametri, iar execuția ei are ca efect citirea din fluxul de intrare a unei secvențe de caractere până la întâlnirea terminаторului de linie. Operația returnează o referință la un obiect `String` care conține caracterele citite, dar fără să includă și terminatorul. Cu alte cuvinte, stringul returnat conține doar caracterele utile (semnificative) ale liniei. Dacă s-a ajuns la sfârșitul fluxului de intrare, operația returnează valoarea `null`. Dacă citirea nu se poate efectua, operația emite o excepție de tip `IOException`. De aceea, semnatura unei funcții care apelează metoda `readLine`, dar nu tratează eventualele erori de citire, trebuie să contină clauza `throws IOException`.

**Observatie:** una dintre cele mai frecvente erori într-un program Java este omisiunea clauzelor `throws` din antetul funcțiilor utilizatorului care apelează funcții predefinite dotate cu aceasta clauză. Acest lucru este semnalat la compilare.

Pentru a crea un obiect al clasei `BufferedReader` este necesar să furnizăm constructorului acestuia o referință la un obiect al clasei `InputStreamReader`. Constructorul acestuia, la rândul lui necesită:

- o referință la un obiect `FileInputStream`, dacă dorim să se facă dintr-un fisier de pe disc, sau
- referința `System.in`, dacă dorim să se facă de la tastatura.

Deci, dacă urmărează să citim dintr-un fisier al cărui nume este dat de o variabilă `String nume_fis`, va trebui să cream un obiect `BufferedReader` ca în secvență de mai jos:

```
BufferedReader flux_in = new BufferedReader(new InputStreamReader(new  
FileInputStream(nume_fis)));
```

Dacă citirea se va face de la tastatura, obiectul `BufferedReader` se crează astfel:

```
BufferedReader flux_in = new BufferedReader(new InputStreamReader(System.in));
```

În continuare citirea se va realiza cu apelul:

```
linie = flux_in.readLine();
```

unde `linie` este o referință `String`.

In urma apelului, ea va indica spre un obiect care contine caracterele citite. In functie de natura datelor reprezentate de aceste caractere, uneori pot fi necesare conversii de la *String* la alte tipuri, in vederea utilizarii datelor respective in diverse calcule.

**Exemplu:** vom prezenta un program care citeste numere intregi dintr-un fisier de text, calculeaza si afiseaza pe ecran suma acestora. Se presupune ca fiecare numar din fisier se afla pe cate o linie distincta, iar numele fisierului este dat ca argument la executia programului.

```
import java.io.*;
public class Suma {
    public static void main (String[ ] args) throws IOException {
        BufferedReader flux_in = new BufferedReader (new
            InputStreamReader (new FileInputStream(args[0])));
        int suma = 0;
        String linie;
        while ((linie = flux_in.readLine()) != null) // cat timp nu am
            // ajuns la sfarsitul fisierului
            suma+=Integer.parseInt(linie); //s-a convertit stringul la int
        System.out.println("Suma = "+suma);
        flux_in.close(); //se inchide fisierul (fluxul)
    }
}
```

**Tema:** sa se modifice programul de mai sus astfel incat citirea sa se faca de la tastatura.

## 2. Operatii de scriere a liniilor de text

Afisarea unei linii pe ecran este deja o operatie binecunoscuta, ea se realizeaza apeland metodele *print/println* definite in clasa *PrintStream*, pentru obiectul *System.out*.

Pentru a scrie o linie intr-un fisier de pe disc vom folosi aceleasi metode, dar va trebui sa cream un obiect separat al clasei *PrintStream*. Pentru aceasta trebuie sa furnizam constructorului clasei *PrintStream*, ca parametru, o referinta la un obiect *FileOutputStream*, asa ca in secventa de mai jos:

```
PrintStream flux_out = new PrintStream (new FileOutputStream(numefis));
unde numefis este o referinta la un obiect String ce contine numele fisierului.
```

**Exemplu:** vom prezenta un program care citeste dintr-un fisier de text o secventa de numere reale, dispuse cate unul pe linie, determina numarul lor, suma, media aritmetica, valoarea minima/maxima si tipareste aceste informatii intr-un alt fisier. Numele ambelor fisiere implicate se dau ca argumente la executia programului.

```
import java.io.*;
public class Statist {
    public static void main (String[ ] args) throws IOException {
        BufferedReader flux_in = new BufferedReader (new
            InputStreamReader (new FileInputStream(args[0])));
        PrintStream flux_out = new PrintStream (new FileOutputStream(args[1]));
        double suma = 0.0, min, max, val;
        int contor=0;
        String linie;
        while ((linie = flux_in.readLine()) != null) {
            contor++;
            // procesarea liniei
            // ...
            flux_out.println("Linie " + contor + ": " + linie);
            if (val < min) min = val;
            if (val > max) max = val;
            suma += val;
        }
        flux_out.println("Suma totala: " + suma);
        flux_out.println("Media aritmetica: " + (suma / contor));
        flux_out.println("Valoarea minima: " + min);
        flux_out.println("Valoarea maxima: " + max);
    }
}
```

```

        val = Double.parseDouble(linie); //conversia String -> double
        suma+=val;
        if(contor==1) { //s-a citit primul numar
            min = val; max = val;
        }
        else {
            if(val<min) min = val;
            if(val>max) max =val;
        }
    }
    flux_out.println("S-au citit "+contor+" valori");
    flux_out.println("Suma lor este "+suma);
    flux_out.println("Media aritmetica este "+(contor>0 ? suma/contor : 0.0));
    flux_out.println("Element minim: "+min);
    flux_out.println("Element maxim: "+max);
    flux_in.close(); flux_out.close();
}
}

```

### **3. Citirea fisierelor de text disponibile pe Internet**

In acest paragraf ne propunem sa prezentam cateva notiuni primare legate de accesul in citire la fisiere aflate pe Internet si care sunt disponibile public.

O retea de calculatoare este un grup de calculatoare care pot schimba informatii intre ele in mod nemijlocit. Calculatoarele din retea sunt interconectate prin cabluri. Internetul este o retea de retele de calculatoare care permite unui calculator dintr-o retea sa comunice cu alt calculator aflat in oricare din celelalte retele. Retelele ce compun Internetul sunt legate intre ele prin linii telefonice, fibre optice sau prin satelit. Comunicarea intre calculatoarele legate la Internet se bazeaza pe faptul ca fiecare asemenea calculator are o adresa de Internet unica. In principiu o asemenea adresa este reprezentata pe 4 octeti si poate fi scrisa sub forma:

*val\_octet1.val\_octet2.val\_octet3.val\_octet4*

Cum aceasta notatie nu este prea comoda pentru utilizator, adreselor de Internet li s-au asociat nume. De exemplu, calculatorului cu adresa:

*193.226.12.166*

ii este asociat numele:

*algoritm.info.uvt.ro*

Suportul software care face posibila comunicarea in retea dispune de un serviciu special care identifica adresa de Internet corespunzatoare unui nume dat.

Informatiile accesibile pe Internet sunt organizate in unitati numite resurse de retea. De regula o resursa este un fisier care poate contine: imagini, sechete audio, text etc, stocat pe un calculator conectat la Internet. O asemenea resursa este identificata in mod unic cu ajutorul asa-numitului URL (Universal Resources Locator) adica locator universal de resurse. Un URL are forma:

*protocol://adresaInternet/caleFisier*

Protocolul identifica tipul suportului software necesar accesarii datelor din resursa. Pentru accesarea paginilor web (fisiere html) protocolul este http. Alte protocoale cunoscute: ftp, file.

Revenind acum la limbajul Java, acesta ne permite sa scriem programe prin care sa citim date din resurse aflate pe Internet, daca aceste resurse sunt disponibile public.

In esenta, pentru a citi date dintr-un fisier aflat pe retea se parcurg urmatorii pasi:

- se stabileste o conexiune cu resursa dorita, precizand URL-ul acesteia,

- se creaza obiectul BufferedReader necesar
- se citesc datele.

Pentru primul pas se utilizeaza clase predefinite din pachetul *java.net*. Considerand ca fisierul pe care dorim sa-l citim este pagina al carei URL este <http://web.info.uvt.ro/index.html> dam mai jos un exemplu de program care citeste pagina respectiva, afisand pe ecran fiecare linie, precedata de numarul ei de ordine:

```

Import java.io.*;
import java.net.*;
public class CitireNet {
    public static void main (String[ ] args) throws Exception {
        URL urlObject = new URL("http://web.info.uvt.ro/index.html");
        //se creaza un obiect care reprezinta URL-ul
        URLConnection conex = urlObject.openConnection();
        //se initiaza comunicarea cu masina pe care se afla resursa
        BufferedInputStream buf = (BufferedInputStream) conex.getContent();
        //se creaza un obiect BufferedInputStream care va fi utilizat la citire;
        //atentie!! getContent NU returneaza continutul fisierului dorit
        BufferedReader flux_in = new BufferedReader (new InputStreamReader (buf));
        int k = 1;
        String linie;
        while ((linie = flux_in.readLine()) != null) { // cat timp nu am ajuns la sfarsitul
fisierului
            System.out.println("Linia "+k+": "+linie);
            k++;
        }
        flux_in.close(); //se inchide fisierul (fluxul)
    }
}

```