



Laborator 4 - limbajul Java

Citirea datelor de la tastatura si afişarea datelor pe ecran

In limbajul Java nu exista instructiuni specializate pentru citirea/scrierea datelor. Aceste operatii se realizeaza prin intermediul unor metode existente in pachetele API ale limbajului. Intrarea si iesirea in Java se realizeaza cu ajutorul claselor de obiecte din **pachetul predefinit java.io**. Orice program care foloseste rutinele de intrare/iesire trebuie sa cuprinda instructiunea:

```
import java.io.*
```

Conceptul fundamental in operatiile de intrare/iesire in limbajul Java este fluxul de intrare/iesire (stream).

Daca stream-ul este de intrare, succesiunea de biti "curge" dinspre exterior (in acest caz, de la tastatura) catre memoria calculatorului.

Daca stream-ul este de iesire, secventa de biti "curge" dinspre memoria calculatorului catre exterior (in acest caz, catre ecran).

Java oferă trei fluxuri predefinite pentru operații I/O standard:

- **System.in** pentru intrarea standard de la tastatura
- **System.out** pentru ieșirea standard la ecranul calculatorului
- **System.err** pentru fluxul de erori

Pentru afişarea datelor la ecranul calculatorului se folosesc **metodele print si println**. Spre deosebire de C/C++ care dispun de un număr foarte mare de opțiuni de formatare, afişarea în Java se face exclusiv prin concatenare de String-uri fără nici o opțiune de formatare.

Observatie: String-urile sunt obiecte Java care descriu sirurile de caractere si le vom studia separat intr-o lectie viitoare. Sa retinem ca prin concatenarea a doua siruri se obtine un nou sir de caractere care uneste cele doua siruri initiale. Operatorul de concatenare a doua siruri de caractere folosit de Java este semnul + (plus).

Sintaxa folosita la apelul metodei **print** este:

```
System.out.print (<expresie>);
```

unde:

- <expresie> - este numele unei variabile de un tip de data sau este o expresie care foloseste operatorul de concatenare pentru siruri de caractere; daca nu toti operanzii din expresie sunt siruri de caractere, ci alte tipuri primitive de date atunci Java face o conversie temporara la tipul String.

Efectul apelului metodei **print** este acela ca se realizeaza afisarea la ecran a variabilei data ca parametru si nu se face salt la o linie noua.

Sintaxa folosita la apelul metodei **println** este:

```
System.out.println (<expresie>);
```

unde:

- <expresie> - este numele unei variabile de un tip de data sau este o expresie care foloseste operatorul de concatenare pentru siruri de caractere.



Efectul apelului metodei **println** este acela ca se realizeaza afisarea la ecran a variabilei data ca parametru si se face salt la o linie noua.

Metoda **println** se poate apela si fara parametri, adica in forma:

System.out.println();

caz in care se face numai un salt la o linie noua fara sa se afiseze nimic.

Se poate folosi urmatoarea combinatie de apeluri care este echivalenta cu **println()**:

System.out.print(<expresie>);

System.out.println();

Pentru citirea datelor de la tastatura procedura este mai anevoioasa.

Acest lucru se datoreaza in primul rand faptului ca programele java nu sunt concepute pentru a citi de la tastatura. In majoritatea cazurilor, programele Java isi preiau datele dintr-o interfata grafica, din forme HTML sau din fisiere.

Citirea datelor de la tastatura se realizeaza cu metoda **readLine**. Insa pentru citire trebuie sa construim un obiect *BufferedReader* dintr-un obiect *InputStreamReader* care la randul sau este construit din *System.in*.

Descrierea detaliata a acestor obiecte o vom face intr-un curs viitor dupa intelegerea conceptelor de clase si obiecte.

Sintaxa folosita la apelul metodei **readLine** este:

<nume_obiect>.readLine();

unde:

- <nume_obiect> - reprezinta o variabila de tipul obiectului *BufferedReader*.

Efectul apelului metodei readLine este urmatorul: preia caracterele de la intrare pana cand intalneste un terminator de linie sau sfarsit de fisier.

Metoda returneaza caracterele citite (din care extrage terminatorul de linie) ca sir de caractere de tip *String*. Daca primul caracter citit este terminatorul de linie, atunci metoda **readLine** returneaza valoarea *null*.

Urmatorul program (Afiseaza.java) ilustreaza modul de folosire al metodelor **println** si **readLine** pentru afisarea si respectiv citirea unor siruri de caractere:

```
/*
 * Afiseaza.java
 */
import java.io.*;
public class Afiseaza
{
    public static void main(String[] args) throws IOException
    {
        System.out.println("Bun venit in universul Java");
        System.out.print ("Introduceti un numar ");
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        String s = br.readLine();
    }
}
```



```
    int a = Integer.parseInt(s);  
    System.out.println(s);  
  }  
}
```

Soluție implementată în compilatorul online <https://www.jdoodle.com/online-java-compiler/> :

The screenshot shows a Java code editor with the following code:

```
1 import java.io.*;  
2 public class Afiseaza  
3 {  
4     public static void main(String[] args) throws IOException  
5     {  
6         System.out.println("Bun venit in universul Java");  
7         System.out.print ("Introduceti un numar ");  
8         InputStreamReader isr = new InputStreamReader(System.in);  
9         BufferedReader br = new BufferedReader(isr);  
10        String s = br.readLine();  
11        int a = Integer.parseInt(s);  
12        System.out.println(s);  
13    }  
14 }  
15
```

Below the code editor, the execution mode is set to "JDK 17.0.1". The "Stdin Inputs" field contains the value "12". The "Execute" button is highlighted in blue. The "Result" section shows the output:

```
Result  
CPU Time: 0.13 sec(s), Memory: 32156 kilobyte(s)  
Bun venit in universul Java  
Introduceti un numar 12
```

Nota: Metoda `Integer.parseInt(s)` aplicată șirului de caractere de la intrare realizează conversia șirului de caractere `s` într-un număr întreg de tip `int`. Pentru a converti un șir de caractere la un număr de tip `double` se poate folosi metoda `Double.parseDouble()`, iar pentru a converti un șir de caractere la un număr de tip `float` se poate folosi metoda `Float.parseFloat()`. Asupra acestor metode vom reveni în lecția despre șiruri de caractere.

Observație: Clauza `throws` utilizată în antetul metodei `main` este folosită pentru a specifica toate excepțiile (erorile) de I/O care nu sunt tratate în cadrul metodei `main` ci de către alte metode din clasele `java.io.*`. Modulul de tratare a excepțiilor (erorilor) vor fi descrise într-o lecție viitoare.

Metoda `System.in.read()` citește următorul caracter din fluxul de intrare (care poate conține mai multe caractere citite de la tastatură) și returnează caracterul citit ca un întreg (cuprins între 0 și 65535) sau -1 dacă s-a întâlnit terminatorul de linie (caracterul `'\r'`-carriage return).

Programul următor (`Afiseaza1.java`) ilustrează modul de folosire a acestei metode:

```
/*  
 * Afiseaza1.java  
 */
```



```
import java.io.*;
public class Afiseaza1
{
    public static void main(String[] args) throws IOException
    {
        char b;
        System.out.println("Bun venit in universul Java");
        b = (char) System.in.read();
        System.out.println(b);
    }
}
```

Solutie implementata in compilatorul online <https://www.jdoodle.com/online-java-compiler/> :

The screenshot displays an online Java compiler interface. At the top, the source code for 'Afiseaza1.java' is shown, including the import statement, class declaration, and the main method with its logic. Below the code editor, the execution environment is configured with 'JDK 17.0.1' and 'Interactive' mode. The 'Stdin Inputs' field contains the text 'Adrian'. An 'Execute' button is visible. The 'Result' section shows the output: 'Bun venit in universul Java' followed by the character 'A'. Performance metrics indicate a CPU time of 0.09 seconds and memory usage of 31792 kilobytes.



Probleme rezolvate:

Scrieți, compilați și executați toate exemplele din acest laborator

1. Programul afișează căutarea unui număr între generat aleator și afișarea numărului de încercări până la identificarea numărului respectiv.

```
import java.io.*;
public class Main
{
public static void main (String args[]) throws IOException
{
    int i;
    int tries = 0;
    int n = (int) (Math.random () * 10); // Math.random() returneaza numere reale in
    intervalul 0..1
    BufferedReader b = new BufferedReader (new InputStreamReader (System.in));
    do{
        tries++;
        System.out.print ("Dati numarul: ");
        String str = b.readLine ();
        i = Integer.parseInt (str); //conversie String -> int
        System.out.println ("Ati introdus " + i);
        if (i < n)
        {    System.out.println ("Prea mic !");    }
        else
            if (i > n)
                {    System.out.println ("Prea mare !");    }
    }while (i != n);
    System.out.println ("Ati ghicit din " + tries + " incercari !");
    }
}
```



Soluție implementată în compilatorul online:
https://www.onlinegdb.com/online_java_compiler

```
1- /*****
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
*****/
Online Java Compiler.
Code, Compile, Run and Debug java program online.
Write your code in this editor and press "Run" button to execute it.
import java.io.*;
public class Main
{
public static void main (String args[]) throws IOException
{
int i;
int tries = 8;
int n = (int) (Math.random () * 10); // Math.random() returneaza numere reale in intervalul 0..1
BufferedReader b = new BufferedReader (new InputStreamReader (System.in));
do{
tries++;
System.out.print ("Dati numarul: ");
String str = b.readLine ();
i = Integer.parseInt (str); //conversie String -> int
System.out.println ("Ati introdus " + i);
if (i < n)
{
System.out.println ("Prea mic !");
}
else
if (i > n)
{
System.out.println ("Prea mare !");
}
}while (i != n);
System.out.println ("Ati ghicit din " + tries + " incercari !");
}
}
```

```
input
Dati numarul: 12
Ati introdus 12
Prea mare !
Dati numarul: 4
Ati introdus 4
Prea mic !
Dati numarul: 10
Ati introdus 10
Prea mare !
Dati numarul: 8
Ati introdus 8
Prea mare !
Dati numarul: 6
Ati introdus 6
Prea mare !
Dati numarul: 5
Ati introdus 5
Ati ghicit din 6 incercari !

...Program finished with exit code 0
Press ENTER to exit console.□
```



2. Sa se rezolve ecuația: $a * x + b = 0$, a,b nr. reale.

Exemplu: pentru $a = 2$ si $b = 4$. Solutia este: -2.0

```
import java.io.*;
public class Main
{
    public static void main (String args[]) throws IOException
    {
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        String s1 = br.readLine();
        double a = Double.parseDouble(s1);
        System.out.println("numarul a = "+s1);
        String s2 = br.readLine();
        double b = Double.parseDouble(s2);
        System.out.println("numarul b = "+s2);
        if ( a == 0 )
            if(b == 0) System.out.println ("Infinitate de solutii !");
            else System.out.println ("Nu are solutie !");
        else
            {      System.out.print("Solutia x = ");      System.out.println (-b/a);      }
    }
}
```

Solutie implementata in compilatorul online:

https://www.onlinegdb.com/online_java_compiler

```
Main.java
1 //.....
2
3           Online Java Compiler.
4           Code, Compile, Run and Debug java program online.
5 Write your code in this editor and press "Run" button to execute it.
6
7 ...../
8
9 import java.io.*;
10 public class Main
11 {
12     public static void main (String args[]) throws IOException
13     {
14         InputStreamReader isr = new InputStreamReader(System.in);
15         BufferedReader br = new BufferedReader(isr);
16         String s1 = br.readLine();
17         double a = Double.parseDouble(s1);
18         System.out.println("numarul a = "+s1);
19         String s2 = br.readLine();
20         double b = Double.parseDouble(s2);
21         System.out.println("numarul b = "+s2);
22         if ( a == 0 )
23             if(b == 0) System.out.println ("Infinitate de solutii !");
24             else System.out.println ("Nu are solutie !");
25         else
26             {      System.out.print("Solutia x = ");      System.out.println (-b/a);      }
27     }
28 }
```



```
input
2
numarul a = 2
4
numarul b = 4
Solutia x = -2.0

...Program finished with exit code 0
Press ENTER to exit console.
```

3. Se citeste un numar natural n . Sa se calculeze $1 + 2 + \dots + n$.

Exemplu: pentru $n = 10$ se va afisa: **Suma este = 55**

```
import java.io.*;
public class Main
{
    public static void main (String args[]) throws IOException
    {
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        System.out.println("numarul n = ");
        String s1 = br.readLine();
        int n = Integer.parseInt(s1);
        int s=0,i;
        for(i=1;i<=n;i++)        s+=i;
        System.out.println ("Suma este = "+s);
    }
}
```




Soluție implementată în compilatorul online:
https://www.onlinegdb.com/online_java_compiler

```
Main.java
1- /*****
2
3           Online Java Compiler.
4       Code, Compile, Run and Debug java program online.
5   Write your code in this editor and press "Run" button to execute it.
6
7   *****/
8
9- import java.io.*;
10 public class Main
11 {
12     public static void main (String args[]) throws IOException
13     {
14         InputStreamReader isr = new InputStreamReader(System.in);
15         BufferedReader br = new BufferedReader(isr);
16         System.out.println("numarul n = ");
17         String s1 = br.readLine();
18         int n = Integer.parseInt(s1);
19         int s=0,i;
20         for(i=1;i<=n;i++)      s+=i;
21         System.out.println ("Suma este = "+s);
22     }
23 }
24
25
input
numarul n =
10
Suma este = 55
...Program finished with exit code 0
Press ENTER to exit console.[]
```



4. Se dă un număr n. Afișați figura din exemplu.

Exemplu

Intrare: 5

Ieșire:

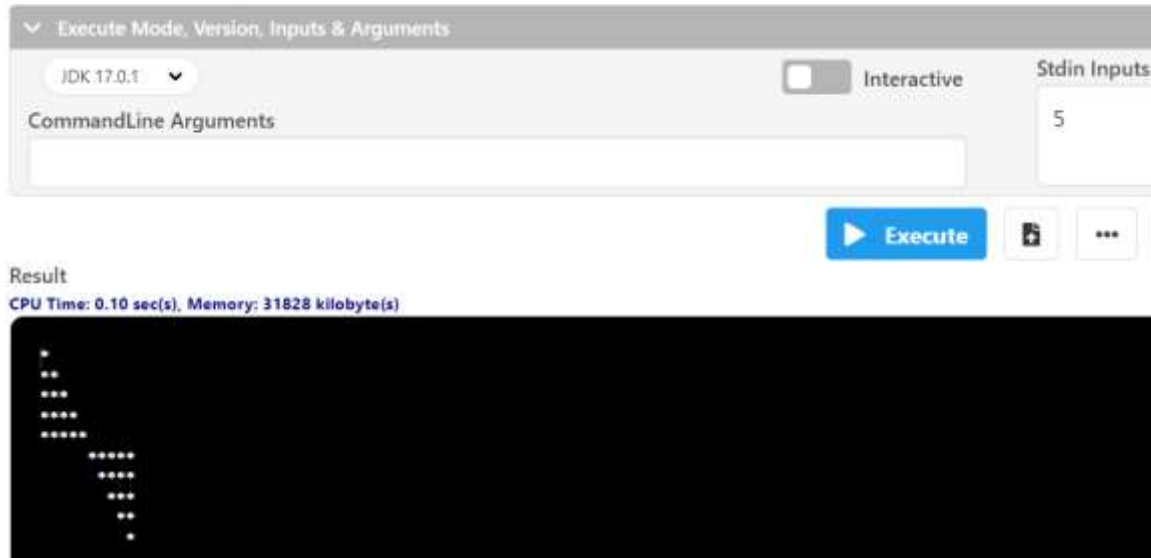
```
*
**
***
****
*****
*****
****
***
**
*
```

Explicație:

S-au afișat 2 triunghiuri, formate din 15 steluțe fiecare.

Soluție implementată în compilatorul online <https://www.jdoodle.com/online-java-compiler/> :

```
1 import java.io.*;
2 public class Main
3 {
4     public static void main (String args[]) throws IOException
5     {
6         InputStreamReader isr = new InputStreamReader(System.in);
7         BufferedReader br = new BufferedReader(isr);
8         //System.out.println("numarul n = ");
9         String s1 = br.readLine();
10        int n = Integer.parseInt(s1);
11        int m,y,i,j,copie;
12        // afisarea primului triunghi
13        y = n - 1;
14        for(i = 1; i <= n; i++){
15            for(j = 1; j <= i; j++)
16                System.out.print("*");
17            System.out.println ();
18        }
19        //afisarea celui de-al doilea triunghi
20        //se afiseaza caracterul spatiu
21        for(m = n; m > 0; m--){
22            for(copie = y; copie >= 0; copie--)
23                System.out.print(" ");
24            y++;
25            for(copie = m - 1; copie >= 0; copie--)
26                System.out.print("*");
27            System.out.println ();
28        }
29    }
30 }
```



5. Se citește un număr natural n . Să se afișeze o figură similară cu cea din exemplu.

Exemplu

Intrare: 3

Ieșire:

```
1
22
22
333
333
333
```

Explicație:

Cifra 1 s-a afișat o singura data, pe un singur rand.

Cifra 2 s-a afișat de 2 ori, pe 2 randuri.

Cifra 3 s-a afișat de 3 ori, pe 3 randuri.

```
import java.io.*;
```

```
public class Main
{
    public static void main (String args[]) throws IOException
    {
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        //System.out.println("numarul n = ");
        String s1 = br.readLine();
        int n = Integer.parseInt(s1);
        int i,j,k;

        for(j = 1; j <= n; j++)
```



```
{
  for(i = 1; i <= j; i++)
  {
    for(k = 1; k <= j; k++)
      if(k<=j) System.out.print(j);
    System.out.println ();
  }
}
```

Solutie implementata in compilatorul online <https://www.jdoodle.com/online-java-compiler/> :

The screenshot shows an online Java compiler interface. The code editor contains the following code:

```
1 import java.io.*;
2 public class Main
3 {
4     public static void main (String args[]) throws IOException
5     {
6         InputStreamReader isr = new InputStreamReader(System.in);
7         BufferedReader br = new BufferedReader(isr);
8         //System.out.println("numarul n = ");
9         String s1 = br.readLine();
10        int n = Integer.parseInt(s1);
11        int i,j,k;
12
13        for(j = 1; j <= n; j++)
14        {
15            for(i = 1; i <= j; i++)
16            {
17                for(k = 1; k <= j; k++)
18                    if(k<=j) System.out.print(j);
19                System.out.println ();
20            }
21        }
22    }
23 }
24
```

Below the code editor, there is a control panel with a dropdown menu set to "JDK 17.0.1", an "Interactive" checkbox, and a "CommandLine Arguments" input field. A blue "Execute" button is visible.

The "Result" section shows the output of the program:

```
1
22
22
333
333
333
```

At the bottom of the result section, it displays "CPU Time: 0.13 sec(s), Memory: 31604 kilobyte(s)".



Probleme propuse spre rezolvare

Lab4.1) Se dă n un număr natural. Să se afișeze un romb de latură n umplut cu caractere * iar spațiul spațiul exterior umplut cu #, ca în exemplu.

Exemplu

Intrare: 5

Ieșire:

```
#####  
#####  
#####  
#####  
#####  
#####  
#####  
#####  
#####  
#####
```

Lab4.2) Se citește un număr natural n cu o cifră. Afișați pe ecran o figură sub forma de romb formata cu numerele naturale de la 1 la n , ca în exemplu.

Exemplu

Intrare: 5

Ieșire:

```
 1  
 22  
3333  
444444  
55555555  
4444444  
33333  
 222  
 1
```

Lab4.3) Se citește numărul natural n , și două caractere c și d . Să se afișeze următorul pătrat, format din n linii și n coloane:

```
ccc...cc  
cdd...dc  
....  
cdd...dc
```



ccc...cc

Date de intrare: Programul citește de la tastatură numărul n și caracterele c d .

Date de ieșire: Programul afișează pe ecran pătratul descris mai sus.

Exemplu

Intrare:

4 * #

Ieșire:

##

##

Lab4.4) Se dă n. Afișați un triunghi cu latura de n steluțe gol înăuntru.

Date de intrare: Se va citi de la tastatură numărul n.

Date de ieșire: Se va afișa triunghiul cerut.

Exemplu

Intrare: 5

Ieșire:

```
*
 *
 * *
 *  *
 *   *
 *    *
 * * * * *
```

Lab4.5) Se dă un număr n. Afișați figura din exemplu.

Date de intrare: Programul citește de la tastatură numărul n.

Date de ieșire: Programul va afișa pe ecran figura.

Exemplu

Intrare: 4

Ieșire:

```
*           *
**          **
***         ***
****        ****
           ****
           ****
           ****
           ****
****       ****
***        ***
**         **
*          *
```



Explicație

S-au afișat 4 triunghiuri și un pătrat, figuri formate din caracterul steluță.

Bibliografie:

- [1] <http://www.pbinfo.ro> *Descrierea site-ului: "www.pbinfo.ro îți propune să rezolvi probleme de informatică, cu evaluator automat. Știi pe loc dacă soluția ta este corectă sau dacă trebuie să mai lucrezi la ea. Problemele sunt grupate după programa de informatică pentru liceu. Dar nu trebuie să fii la liceu ca să rezolvi aceste probleme. Poți fi elev de gimnaziu, student, profesor sau pur și simplu pasionat de informatică. De fapt, trebuie doar să vrei!!"*
- [2] <https://www.runceanu.ro/adrian>
- [3] Adrian Runceanu „Programarea și utilizarea calculatoarelor”, Editura Academica Brâncuși din Târgu-Jiu, 2003, ISBN 973-8436-44-3
- [4] Adrian Runceanu, Mihaela Runceanu, „Noțiuni de programare – limbajul C++”, Editura Academica Brâncuși din Târgu-Jiu, 2012, ISBN 978-973-144-550-2
- [5] Adrian Runceanu, Mihaela Runceanu, „Algoritmi implementati in limbajul C++. Volumul I – Algoritmi elementari”, Editura Academica Brâncuși din Târgu Jiu, 2021, ISBN 978-606-9614-06-8