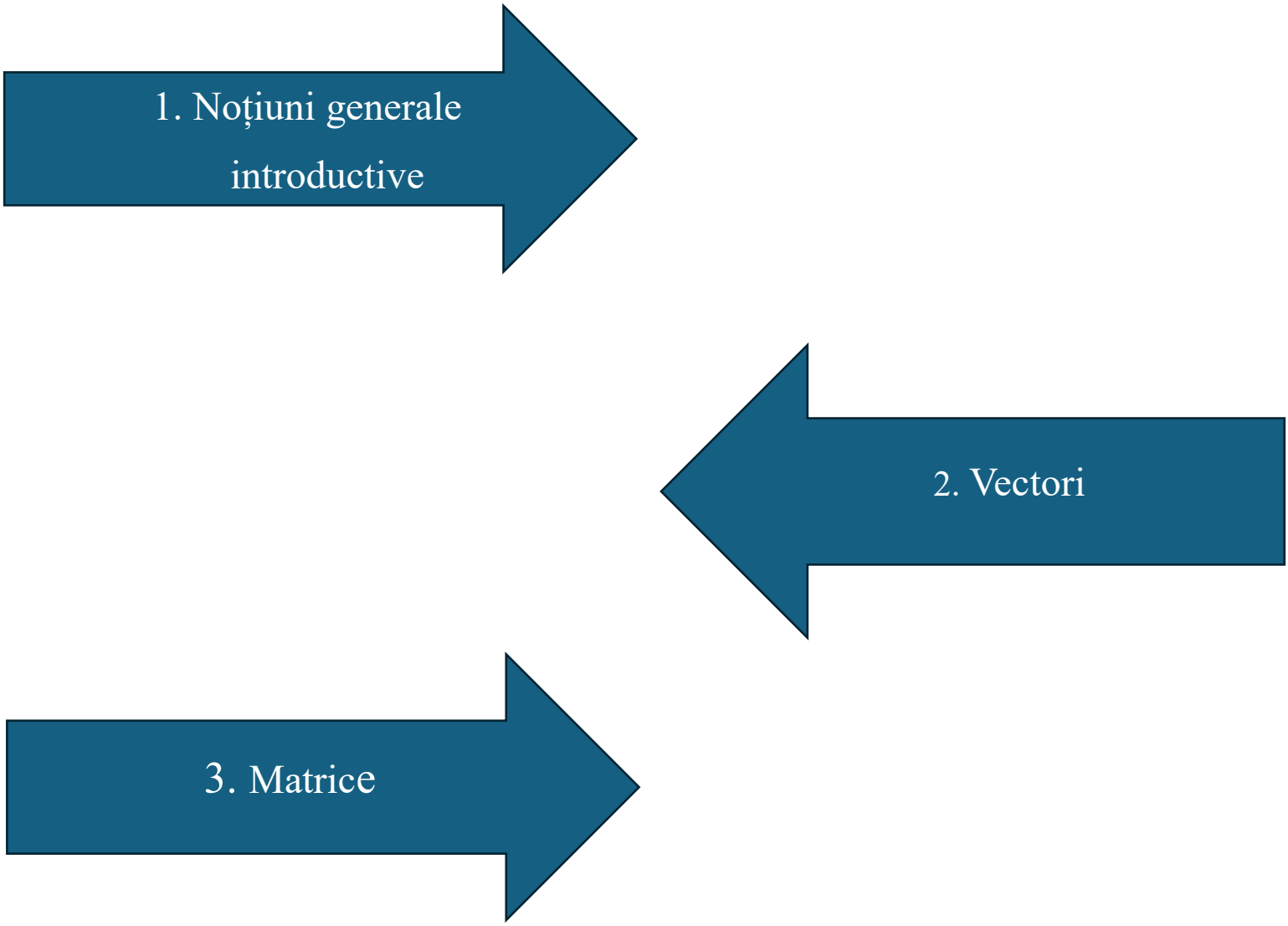

1. Noțiuni generale introductive
2. Vectori
3. Matrice

1. Scrie un program Java care calculează aria unui triunghi folosind formula lui Heron. Programul trebuie să solicite utilizatorului să introducă lungimile celor trei laturi ale triunghiului și să afișeze aria triunghiului calculată pe baza acestor lungimi

```java
package laborator2;
import java.io.*; // Importă toate clasele din pachetul java.io, necesare pentru operatiuni de intrare/iesire (I/O).
public class ArieTriunghi {
    public static void main(String[] args) throws IOException {
        double x, y, z, p, aria; // Declaratia variabilelor pentru laturile triunghiului (x, y, z), semiperimetru (p) si aria.
        String s; // Declaratia unei variabile string pentru a citi inputul de la utilizator.
        // Citește valoarea pentru latura x
        System.out.print("Introduceti x= ");
        InputStreamReader isrx = new InputStreamReader(System.in);
        BufferedReader brx = new BufferedReader(isrx);
        s = brx.readLine(); // Citește linia de text introdusă de utilizator
        x = Double.parseDouble(s); // Converteste textul citit într-un număr de tip double
        // Citește valoarea pentru latura y
        System.out.print("Introduceti y= ");
        InputStreamReader isry = new InputStreamReader(System.in);
        BufferedReader bry = new BufferedReader(isry);
        s = bry.readLine();
        y = Double.parseDouble(s);
        // Citește valoarea pentru latura z
        System.out.print("Introduceti z= ");
        InputStreamReader isrz = new InputStreamReader(System.in);
        BufferedReader brz = new BufferedReader(isrz);
        s = brz.readLine();
        z = Double.parseDouble(s);
        // Verifică dacă valorile introduse sunt valide pentru laturile unui triunghi
        if (x <= 0 || y <= 0 || z <= 0) {
            System.out.println("Numerele introduse nu constituie laturile unui triunghi");
        } else if (x + y <= z || x + z <= y || y + z <= x) {
            System.out.println("Numerele introduse nu constituie laturile unui triunghi");
        } else {
            // Calculează semiperimetrul triunghiului
            p = (x + y + z) / 2;
            // Calculează aria utilizând formula lui Heron
            aria = Math.sqrt(p * (p - x) * (p - y) * (p - z));
            // Afisează aria triunghiului
            System.out.println("Aria triunghiului = " + aria);
        }
    }
```

Figura 1. Codul Sursa – Aria unui triunghi java

IOExceptions

În declarația metodei principale public static void main(String[] args) throws IOException, expresia ***throws IOException*** are rolul de a gestiona excepțiile în Java.

Ce sunt excepțiile?

Excepțiile sunt evenimente care apar în timpul execuției unui program și care întrerup fluxul normal de execuție al programului. În Java, excepțiile sunt obiecte care reprezintă erori sau condiții neașteptate.

Tipuri de excepții

1. **Checked Exceptions (excepții verificate)** - Acestea sunt verificate la timp de compilare. Programele trebuie să gestioneze aceste excepții fie prin tratarea lor (folosind blocuri try-catch), fie prin declararea lor (folosind throws).

2. **Unchecked Exceptions (excepții neverificate)** - Acestea sunt subclase ale RuntimeException și nu sunt verificate la timpul de compilare. Ele pot fi tratate, dar nu este obligatoriu.

3. **Errors** - Acestea sunt subclase ale Error și indică probleme grave pe care un program tipic nu le poate gestiona.

- Variabila de tip `**String**` (`**s**`) stochează temporar textul introdus de utilizator, permițând conversia acestuia într-un număr `double` pentru a putea fi utilizat în calcule matematice ulterioare.

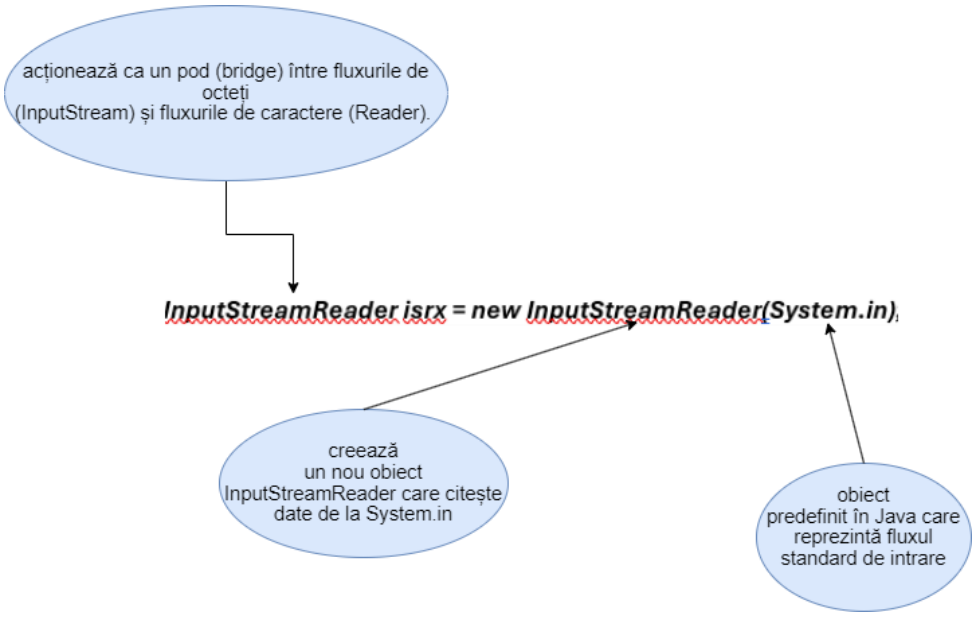


Figura 2. InputStreamReader explicații

Metoda readLine():

- readLine() este o metodă a clasei BufferedReader care citește o linie completă de text de la fluxul de intrare și o returnează ca un String.

- O linie este considerată terminată atunci când este întâlnit unul dintre următoarele: un caracter de sfârșit de linie ('\n'), un caracter de sfârșit de linie urmat de un caracter de sfârșit de linie ('\r\n'), sau sfârșitul fluxului (EOF).

Metoda parseDouble:

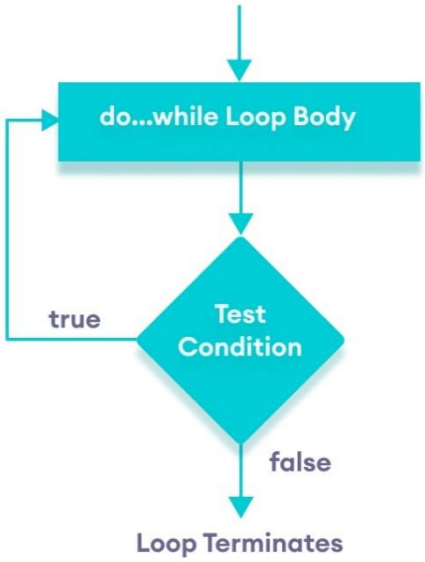
- parseDouble este o metodă statică a clasei Double. Metodele statice sunt apelate pe clasă, nu pe o instanță a clasei.
- Metoda parseDouble ia un String (s), încearcă să interpreteze conținutul său ca un număr real (cu virgulă mobilă) și returnează valoarea numerică echivalentă de tip double.

```
38      }
39      // Cod comentat care reprezintă o metodă separată pentru calcularea ariei unui triunghi
40      /*
41      public static double Triunghi(double a, double b, double c) {
42          double p = (a + b + c) / 2;
43          double aria = Math.sqrt(p * (p - a) * (p - b) * (p - c));
44          return aria;
45      }
46      */
47  }
48
```

Figura 3. Alta metoda de a calcula Aria

2. Scrie un program Java care să calculeze cifra de control a unui număr natural introdus de utilizator. Cifra de control este obținută prin repetarea procesului de calculare a sumei cifrelor numărului până când rezultatul este o singură cifră. Programul trebuie să solicite utilizatorului să introducă un număr natural și să afișeze atât suma cifrelor la fiecare pas, cât și cifra de control finală.

Vom folosi bucla repetitivă do..while.



Bucla **do-while** în Java este un tip de buclă de control care execută un bloc de cod cel puțin o dată, și apoi repetă execuția acelui bloc atât timp cât o anumită condiție este adevărată.

Spre deosebire de bucla while, condiția este verificată după ce blocul de cod a fost executat, asigurând astfel că codul din interiorul buclei este executat cel puțin o dată.

hello.java | sumanrnaturale.java | *SumaCifre.java | *Afisare.java | ArieTriunghi.java | *CifraControl.java

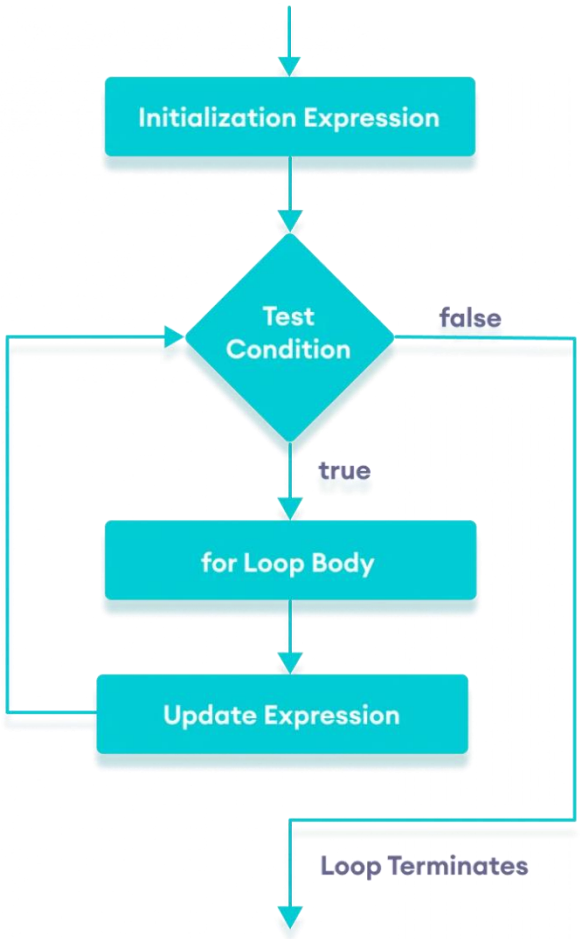
```java
package laborator2; // Pachetul în care se află clasa
import java.io.*; // Importă toate clasele din pachetul java.io
public class CifraControl // Declaratia clasei publice CifraControl
{
    public static void main(String[] args) throws IOException // Metoda principală, punctul de intrare al programului
    {
        int x, suma = 0; // Declararea variabilelor întregi x și suma, suma fiind initializată la 0
        String s; // Declararea variabilei de tip String pentru citirea inputului
        // Solicită utilizatorului să introducă un număr natural
        System.out.print("Introduceti numarul natural=  ");
        // Creează un InputStreamReader și un BufferedReader pentru a citi inputul de la tastatură
        InputStreamReader isrx = new InputStreamReader(System.in);
        BufferedReader brx = new BufferedReader(isrx);
        // Citește o linie de text introdusă de utilizator și o stochează în variabila s
        s = brx.readLine();
        // Convertește textul citit (String) într-un număr întreg și îl stochează în variabila x
        x = Integer.parseInt(s);
        // Buclă do-while pentru a calcula suma cifrelor numărului și cifra de control
        do {
            // Buclă while pentru a calcula suma cifrelor numărului x
            while (x != 0) {
                suma += x % 10; // Adună ultima cifră a lui x la suma
                x /= 10; // Elimină ultima cifră a lui x
            }
            // Afișează suma cifrelor numărului curent
            System.out.println("Suma cifrelor numarului: " + suma);
            // Actualizează x cu suma cifrelor sumei curente
            x = suma;
            // Resetează suma pentru următoarea iterație (dacă este necesar)
            suma = 0;
        } while (x > 9); // Continuă bucla dacă x are mai mult de o cifră
        // Afișează cifra de control a numărului, care este x după ultima iterație
        System.out.println("Cifra de control a numarului este: " + x);
    }
}
```

Figura 4. Cod Sursă

LABORATOR 2

Vector

În Java, un ***vector*** (sau tablou) este o structură de date care stochează o colecție de elemente de același tip într-o secvență ordonată, accesibile prin intermediul unui index numeric, care începe de la 0. Vectorii sunt de dimensiune fixă și sunt utilizati pentru a organiza și a accesa eficient grupuri de date.



În Java, instrucțiunea ***for*** este utilizată pentru a repeta un bloc de cod de un număr specificat de ori, prin inițializarea variabilei de control, evaluarea unei condiții care determină continuarea buclei și actualizarea variabilei de control la fiecare iterație.

```java
//Declararea
int[] vector;

//Initializare cu dimensiune fixa
int[] vector = new int[5]; // Creează un vector de întregi cu 5 elemente, toate initializate la 0

//Initializare cu valori specifice
int[] vector = {1, 2, 3, 4, 5}; // Creează un vector cu 5 elemente, fiecare având valorile 1, 2, 3, 4 și 5

//Initializare cu dimensiune si valori
int[] vector = new int[5]; // Declara și initializează un vector de 5 elemente
vector[0] = 10; // Setează valoarea 10 la prima pozitie
vector[1] = 20; // Setează valoarea 20 la a doua pozitie
// Continuă pentru restul elementelor

//Accesare elemente
System.out.println(vector[2]); // Afișează 3, care este valoarea de la indexul 2


//Iterare elemnte
for (int i = 0; i < vector.length; i++) {
    System.out.println(vector[i]);
}

for (int element : vector) {
    System.out.println(element);
}
```

Figura 4. Metode de declarare si lucru cu vectori

3. Scrieți un program Java care citește un număr natural pentru a determina dimensiunea unui vector, solicită utilizatorului să introducă acea cantitate de numere întregi pentru a popula vectorul și apoi afișează fiecare element al vectorului împreună cu indicele său, folosind două funcții: una pentru citirea unui număr întreg și alta pentru coordonarea întregului proces.

```java
package laborator2;
import java.io.*; // Importă toate clasele din pachetul java.io
public class Tablou { // Declaratia clasei publice Tablou
    public static void main(String [] args) throws IOException { // Metoda principală, punctul de intrare al programului
        System.out.println("Introduceti primul numar"); // Afișează un mesaj pentru a cere utilizatorului să introducă ur
        int x = citesteNr(); // Apelează metoda citesteNr() pentru a citi un număr de la tastatură și îl stochează în var
        System.out.println("Numarul introdus este:" + x); // Afișează numărul introdus
        System.out.println("Introduceti vectorul"); // Afișează un mesaj pentru a cere utilizatorului să introducă elemer

        int v[] = new int[x]; // Declară și initializează un vector de întregi cu dimensiunea x
        for (int i = 0; i < v.length; i++) { // Buclă for pentru a citi elementele vectorului
            v[i] = citesteNr(); // Citește un număr de la tastatură și îl stochează în vector la pozitia i
        }
        for (int i = 0; i < v.length; i++) { // Buclă for pentru a afișa elementele vectorului
            System.out.println("v(" + i + ")=" + v[i]); // Afișează elementul de la pozitia i din vector
        }
    }
    public static int citesteNr() throws IOException { // Metodă statică pentru citirea unui număr de la tastatură
        InputStreamReader isr = new InputStreamReader(System.in); // Creează un InputStreamReader pentru a citi inputul c
        BufferedReader br = new BufferedReader(isr); // Creează un BufferedReader pentru a citi linia de text
        String s = br.readLine(); // Citește linia de text introdusă de utilizator și o stochează în variabila s
        return Integer.parseInt(s); // Convertește textul citit într-un număr întreg și îl returnează
    }
}

```

Figura 5. Codul Sursă

În Java, **v.length** este o proprietate utilizată pentru a obține dimensiunea (numărul de elemente) a unui vector (array) numit v. Este important de reținut că length este un atribut (nu o metodă) al vectorului, deci nu se folosesc paranteze după length.

Matrice

O **matrice** (sau tablou bidimensional) este o structură de date care stochează date într-o formă de tabel, organizate în rânduri și coloane. Fiecare element al matricei este accesibil printr-o pereche de indecși: unul pentru rând și unul pentru coloană.

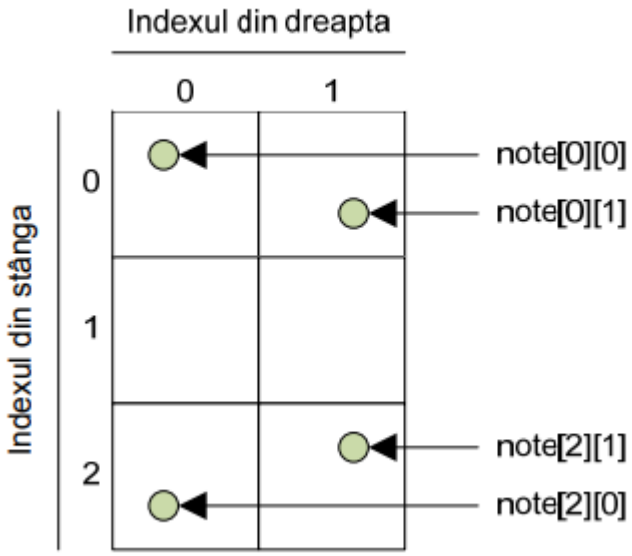


Figura 6. Elementele tablourilor multidimensionale

Declarația și Inițializarea Matricilor

Declarația unei matrice se face specificând tipul elementelor și dimensiunile matricei. De exemplu, pentru a declara o matrice de întregi cu 3 rânduri și 4 coloane:

int[][] matrice = new int[3][4];

Inițializarea unei matrice poate fi realizată fie în momentul declarației, fie ulterior:

int[][] matrice = {

{1, 2, 3, 4},

{5, 6, 7, 8},

{9, 10, 11, 12}

};

Accesarea Elementelor

Elementele unei matrici pot fi accesate utilizând indecșii corespunzători pentru rând și coloană. De exemplu, pentru a accesa elementul din rândul 2 și coloana 3:

int valoare = matrice[1][2]; // reține că indexarea începe de la 0

Parcurgerea unei Matrici

Pentru a parcurge toate elementele unei matrici, se pot folosi bucle for imbricate:

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

for (int j = 0; j < matrice[i].length; j++) {

System.out.print(matrice[i][j] + " ");

}

System.out.println();

}

4. Creează un program Java care citește un număr întreg `a` de la utilizator pentru a defini dimensiunea unui vector și a unei matrice bidimensionale `a x a`, apoi citește `a` valori întregi pentru vector și le afișează, ignorând al doilea număr introdus.

```java
package laborator2;
import java.io.*; // Importă clasele necesare pentru citirea inputului de la tastatură
public class Vector { // Declarația clasei publice Vector
    public static void main(String[] args) throws IOException { // Metoda principală, punctul de intrare al programului
        // Citește două numere întregi de la utilizator
        int a = citesteNr(); // Apelează metoda citesteNr() pentru a citi primul număr, care va fi dimensiunea vectorului
        int b = citesteNr(); // Apelează metoda citesteNr() pentru a citi al doilea număr, care nu este utilizat în acest cod
        // Afișează dimensiunea vectorului
        System.out.println("Dimensiunea vectorului este:" + a);
        // Declară și inițializează un vector de dimensiune a
        int v[] = new int[a];
        // Declară și inițializează o matrice bidimensională de dimensiune a x a
        int A[][] = new int[a][a];
        // Citește valorile pentru fiecare element al vectorului
        for (int i = 0; i < v.length; i++) { // Buclă pentru a parcurge fiecare index al vectorului
            v[i] = citesteNr(); // Citește un număr și îl stochează în vector la poziția i
        }
        // Afișează valorile stocate în vector
        for (int i = 0; i < v.length; i++) { // Buclă pentru a parcurge fiecare index al vectorului
            System.out.println("v(" + i + ")=" + v[i]); // Afișează valoarea elementului de la poziția i din vector
        }
    }
    // Metodă statică pentru citirea unui număr întreg de la tastatură
    public static int citesteNr() throws IOException {
        InputStreamReader isr = new InputStreamReader(System.in); // Creează un InputStreamReader pentru a citi inputul de la
        BufferedReader br = new BufferedReader(isr); // Creează un BufferedReader pentru a citi linia de text
        String s = br.readLine(); // Citește linia de text introdusă de utilizator și o stochează în variabila s
        return Integer.parseInt(s); // Convertește textul citit într-un număr întreg și îl returnează
    }
}
```

Figura 7. Cod sursa