

## Clase, obiecte și tablouri

### 1 Scopul lucrării

Obiectivele acestei sesiuni de laborator sunt:

- Înțelegerea conceptului de tablou Java și utilizarea sa corespunzătoare

### 2 Tablouri

Un tablou poate stoca valori de același fel. Fiecare valoare poate fi accesată prin specificarea unui indice. "Tablou" în Java înseamnă *aproximativ* același lucru ca tablou, matrice sau vector în matematică. Spre deosebire de matematică, un tablou Java trebuie declarat și trebuie să i se aloce o cantitate fixă de memorie.

#### 2.1 Declararea unui tablou

Un tablou este ca alte variabile – trebuie declarat, adică trebuie specificat tipul elementelor din tablou. Toate elementele trebuie să fie de același tip.. Se scrie numele tipului elementelor, apoi "[ ]", apoi numele variabilei tablou. Declarația alocă doar suficient spațiu pentru o referință la un tablou (tipic 4 octeți), dar nu creează efectiv obiectul tablou.

```
String[] args; // args este un tablou de String
int[] scores; // scores este un tablou de int
JButton[] bs; // bs este un tablou de JButton
```

**Nu se specifică dimensiunea în declarație.** Spre deosebire de alte limbaje, nu se pune niciodată dimensiunea tabloului în declarație, deoarece o declarație de tablou specifică doar tipul elementelor și numele variabilei.

**Alocați obiectul tablou cu new.** Un tablou se creează cu new. Exemplul următor creează un tablou de 100 elemente de tipul int, de la a[0] la a[99].

```
int[] a; // Declara a ca fiind un tablou de int
a = new int[100]; // Aloca un tablou de 100 int
```

Acestea sunt adesea combinate într-o singură linie.

```
int[] a = new int[100]; // Declara si aloca.
```

#### Indici

Indicii sunt incluși între paranteze pătrate „[ ]”.  $x_i$  din matematica este  $x[i]$  în Java.

Gamele pentru indici încep întotdeauna la zero deoarece Java provine în mare parte din C++, limbaj care avea un motiv bun pentru folosirea lui zero (aritmetica cu poanteri pe tablouri). Nu este modul în care numără de obicei oamenii, dar asta e.

Java verifică întotdeauna legalitatea indicilor pentru a se asigura ca indicele este  $\geq 0$  și mai mic decât numărul de elemente din tablou. Dacă indicele este în afara acestei game, Java arunca o excepție de tipul `ArrayIndexOutOfBoundsException`. Acest comportament este net superior celui din C și C++, limbaje care permit referințe în afara gamei corecte. În consecință, programele Java sunt mult mai puțin susceptibile la erori și curențe de securitate decât programele C/C++.

### Lungimea unui tablou

Fiecare tablou are o variabilă instanță constantă (final) care conține lungimea tabloului. Puteți afla câte elemente poate păstra un tablou scriindu-i numele urmat de **.length**. În exemplul anterior, `a.length` ar fi 100. Țineți minte că acesta este numărul de elemente din tablou, cu unul mai mult decât indicele maxim.

### Idiomul Java pentru ciclarea peste un tablou

Cea mai frecventă folosire a lui `.length` se regăsește condiția de testat în buclele `for`. Spre exemplu, în fragmentul de cod următor variabila `i` va traversa întreaga gamă de indici a tabloului `a`.

```
for (int i=0; i < a.length; i++)
{
    . . .
}
```

Dacă aveți nevoie doar *să referiți valoarea* fiecărui element, puteți folosi bucla oarecum mai simplă din Java 5, care ține evidența indicelui și atribuie valori succesive unei variabile (`v` în acest exemplu).

```
for (int v : a)
{
    . . .
}
```

### Exemplu, versiunea 1 – Adunarea elementelor unui tablou

Fragmentul următor creează un tablou și pun 1000 de valori aleatoare în el. Cea de a doua buclă adună toate cele 1000 elemente. Ar fi fost mai bine să le adăugăm în prima buclă, dar prin această scriere avem două exemple de bucle.

```
int[] a;           // Declară un tablou de int
a = new int[1000]; // Crează un tablou de 1000 int.

//... Atribuie valori aleatoare fiecarui element.
for (int i=0; i < a.length; i++)
{
    a[i] = (int)(Math.random() * 100000); // Numarul aleatoriu in gama 0-99999
}

//... Aduna toate valorile din tablou.
int sum = 0;       // Initializeaza suma totala la 0.
for (int i=0; i<a.length; i++)
{
    sum = sum + a[i]; // Aduna urmatorul element la total
}
```

### Exemplu, versiunea 2 – Adunarea tuturor elementelor unui tablou în Java 5

Este la fel ca cel de mai sus, dar folosește bucla „`for each`” din Java 5 pentru a face însumarea, lucru care ne eliberează de folosirea unui indice, dacă ceea ce avem nevoie este să obținem toate valorile succesive. Acest fel de buclă doar ne obține valorile, așa că nu poate fi folosit pentru a stoca valorile în prima buclă de mai sus.

```
. . .
int sum = 0;           // Initializeaza suma totala la 0.
for (int v : a)
{
    sum = sum + v; // Aduna urmatorul element la total
}
```

## Valori inițiale pentru elementele de tablou – zero/null/false

La alocarea unui tablou (cu new), toate elementele *primesc o valoare inițială*. Această valoare este 0 dacă tipul este numeric (int, float, ...), false pentru boolean și null pentru toate tipurile de obiecte.

## 2.2 Inițializarea tablourilor

La declararea unui tablou, puteți și să alocați un obiect tablou preinițializat în aceeași instrucțiune. În acest caz, nu furnizați mărimea tabloului fiindcă Java numără valorile din inițializare pentru a determina mărimea. Spre exemplu,

```
// stil Java 1.0 - mai scurt, dar poate fi folosit DOAR IN DECLARATII
String[] days = {"Su", "Mo", "Tu", "We", "Th", "Fr", "Sa"};
```

## Variabilele tablou sunt referite la tablouri

La declararea unei variabile tablou, Java rezervă memorie suficientă doar pentru o referință (numele Java pentru adresa sau poanter) la un obiect tablou. Referințele necesită în mod tipic doar 4 octeți. La crearea unui obiect tablou cu new se returnează o referință, iar acea referință poate fi asignată unei variabile. La asignarea unui tablou la un altul, doar referința este copiată. Spre exemplu:

```
int[] a = new int[] {100, 99, 98};
int[] b;
// "a" poanteaza spre un tablou, iar "b" nu poanteaza spre nimic
b = a;      // Acum b se refera la ACELASI tablou ca si "a"
b[1] = 0;   // Schimba si a[1] deoarece a si se refera la acelasi tablou.
// Atât a cât si b se refera la acelasi tablou cu valorile {100, 0, 98}
```

## 2.3 Noțiuni mai complexe

### Tablouri anonime

Java 2 a adăugat tablourile anonime, care vă permit să creați un tablou de valori nou oriunde în program, nu doar într-o inițializare într-o declarație.

```
// Acest stil anonim de tablou poate fi folosit si in alte instructiuni.
String[] days = new String[] {"Su", "Mo", "Tu", "We", "Th", "Fr", "Sa"};
```

Sintaxa tablourilor anonime poate fi folosită și în alte părți ale programului. Spre exemplu:

```
// In afara declaratiei puteti face aceasta atribuire.
x = new String[] {"Su", "Mo", "Tu", "We", "Th", "Fr", "Sa"};
```

Trebuie să aveți grijă să nu creați aceste tablouri anonime în bucle sau ca variabile locale, deoarece fiecare folosire a lui new va crea un alt tablou.

### Alocarea dinamică

Deoarece tablourile sunt alocate dinamic, valorile de inițializare pot fi expresii arbitrare. Spre exemplu, apelul următor creează două tablouri noi pe care le transmite ca parametri lui drawPolygon.

```
g.drawPolygon(new int[] {n, n+45, 188}, new int[] {y/2, y*2, y}, 3);
```

### Declarații de tablouri în stil C

Java vă permite și să scrieți parantezele pătrate după numele variabilei în loc să le scrieți după tip. Acesta este modul în care se scriu declarațiile de tablouri în C, *dar nu constituie un stil bun pentru Java*. Sintaxa

C poate arata foarte urât având o parte a declarației înainte de variabilă și o parte după. Java are un stil mult mai curat, în care toată informația de tip poate fi scrisă fără a folosi o variabilă. Sunt situații în care acest stil – Java – este singura notație care se poate folosi.

```
int[] a;    // stil Java -- bun
int a[];    // stil C -- legal, dar nerecomandat
```

## 2.4 Probleme frecvente la tablouri

Câteva probleme frecvent întâlnite la folosirea tablourilor sunt:

- Se uita că indicii încep de la zero.
- Se scrie `a.length()` în loc de `a.length`. Metoda `length()` este folosită la String, nu la tablouri.
- Declararea unui tablou cu mărime. D.e., `int[100] a;` în loc de `int[] a = new int[100];`

## 2.5 Metode de bibliotecă pentru tablouri

Există metode de bibliotecă, statice pentru manipularea tablourilor în clasa **java.util.Arrays**.

<b>Arrays.asList()</b>	Returnează un List (listă) pe baza tabloului.
<b>Arrays.toString()</b>	Returnează o formă „citibilă” a tabloului.
<b>Arrays.binarySearch()</b>	Execută o căutare binară pe un tablou sortat.
<b>Arrays.equals</b>	Compară două tablouri dacă sunt egale..
<b>Arrays.fill()</b>	Umple tot tabloul sau o subgamă cu o valoare.
<b>Arrays.sort()</b>	Sortează un tablou.

În plus, există metoda `System.arraycopy()`.

### Inversarea unui tablou

Această versiune a inversării folosește doi indici: unul care începe la stânga (începutul) tabloului și un altul care începe la dreapta (sfârșitul) tabloului. Se poate folosi și o buclă care merge spre mijlocul tabloului.

```
//===== reverse
public static void inverseaza(int[] b)
{
    int stinga = 0;           // indexul elementului celui mai din stânga
    int dreapta = b.length-1; // indexul elementului celui mai din dreapta

    while (stinga < dreapta)
    {
        // interchimba elementele din stinga si dreapta
        int temp = b[stinga];
        b[stinga] = b[dreapta];
        b[dreapta] = temp;

        // deplaseaza limitele spre centru
        stinga++;
        dreapta--;
    }
} //sfârșit metoda inverseaza
```

O buclă for care să meargă spre mijloc ar înlocui cele 8 instrucțiuni de mai sus cu ceea ce urmează. Ambele bucle sunt la fel de rapide, așa că alegeți-o pe cea care vi se pare mai de înțeles.

```
for (int stinga=0, int dreapta=b.length-1; stinga<dreapta; stinga++, dreapta--) {
    // interchimba primul cu ultimul
    int temp = b[stinga]; b[stinga] = b[dreapta]; b[dreapta] = temp;
}
```

## 2.6 Tablouri multidimensionale

Tablourile din Java sunt de fapt tablouri liniare, unidimensionale. Cu toate acestea, se pot construi tablouri cu mai multe dimensiuni, întrucât există posibilități de creare a lor.

Exemplele care urmează folosesc toate tablouri bidimensionale, dar sintaxa și codificarea se poate extinde pentru oricâte dimensiuni. Prin convenție, tablourile bidimensionale au linii (orizontale) și coloane (verticale). Primul indice selectează linia (care este un tablou monodimensional în sine), iar cel de-al doilea selectează elementul în acea linie/acei tablou.

### Vizualizarea tablourilor bidimensionale

Să presupunem că avem un tablou, *a*, cu trei linii și patru coloane.

```
a[0][0] a[0][1] a[0][2] a[0][3]
a[1][0] a[1][1] a[1][2] a[1][3]
a[2][0] a[2][1] a[2][2] a[2][3]
```

Tablourile bidimensionale sunt vizualizate de obicei matrice, cu linii și coloane. Diagrama următoare arată un tablou cu indicii corespunzători.

```
+-----+
| a[0] | -> | [0] | [1] | [2] | [3] |
+-----+
+-----+
| a[1] | -> | [0] | [1] | [2] | [3] |
+-----+
+-----+
| a[2] | -> | [0] | [1] | [2] | [3] |
+-----+
```

În Java tablourile bidimensionale sunt implementate ca tablou monodimensional de tablouri monodimensionale.

### Declararea și alocarea unui tablou bidimensional

De exemplu, tabla pentru tic-tac-toe. Va avea (într-un caz foarte limitat) trei rânduri – primul indice și trei coloane – al doilea indice și va conține un *int* în fiecare element.

```
int[][] board = new int[3][3];
```

### Valori inițiale

Se pot asigna valori inițiale tabloului la declararea într-un mod foarte asemănător cu cel pentru tablourile monodimensionale. Dimensiunile sunt calculate din numărul de valori.

```
int[][] board = new int[][] {{0,0,0},{0,0,0},{0,0,0}};
```

Trebuie să dați o valoare unui element înainte de a-l folosi, fie printr-o inițializare, fie printr-o asignare.

### Exemplu – desenarea tablei de tic-tac-toe

Adesea este cel mai ușor să se folosească tablourile bidimensionale cu bucle for imbricate. Spre exemplu, codul care urmează desenează tabla de tic-tac-toe printr-o metodă paint. Codul presupune că o celulă are latura de 10 pixeli și că un număr pozitiv înseamnă un X, iar unul negativ reprezintă un O.

```
for (int row=0; row<3; row++)
{
    for (int col=0; col<3; col++)
    {
        if (board[row][col] > 0) { // deseneaza X
            g.drawLine(col*10, row*10, col*10+8, row*10+8);
            g.drawLine(col*10, row*10+8, col*10+8, row*10+8);
        }
        else if (board[row][col] < 0)
        { // deseneaza O
            g.drawOval(col*10, row*10, 8, 8);
        }
    }
}
```

## 3 Mersul lucrării

- 3.1. Studiați și înțelegeți textul și exemplele date.
- 3.2. Definiți o clasă Matrice care să implementeze operațiile de adunare, scădere, înmulțire de matrice, precum și împărțirea cu un scalar.
- 3.3. Implementați o clasă TablaSah care să păstreze poziții pe tabla de șah. Figurile și pionii sunt și ei clase. Verificați corectitudinea mutărilor.