Clase, obiecte şi tablouri

1. Scopul lucrării

Obiectivele acestei sesiuni de laborator sunt:
- Înțelegerea conceptului de tablou în Java
- Utilizarea tablourilor în aplicații Java

2. Tablouri

Un tablou poate stoca valori de acelaşi fel. Fiecare valoare poate fi accesată prin specificarea unui indice. "Tablou" în Java înseamnă *aproximativ* acelaşi lucru ca tablou, matrice sau vector în matematică. Spre deosebire de matematică, un tablou Java trebuie declarat şi 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 acelaşi tip. Se scrie numele tipului elementelor, apoi "[]", apoi numele variabilei tablou. Declaraţia alocă doar suficient spaţiu pentru o referinţă la un tablou (tipic 4 octeţi), 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 declaraţie. Spre deosebire de alte limbaje, nu se pune niciodată dimensiunea tabloului în declaraţie, deoarece o declaraţie de tablou specifică doar tipul elementelor şi numele variabilei.

Alocaţi 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 incluşi între paranteze pătrate „[]”. x_i din matematică este x[i] în Java.
Gamele pentru indici încep întotdeauna de la zero deoarece Java provine în mare parte din C++, limbaj care avea un motiv bun pentru folosirea lui zero (aritmetica cu pointeri pe tablouri).
Java verifică întotdeauna legalitatea indicilor pentru a se asigura ca indicele este >= 0 şi mai mic decât numărul de elemente din tablou. Dacă indicele este în afara acestei game, Java aruncă o excepţie de tipul `ArrayIndexOutOfBoundsException`. Acest comportament este net superior celui din C şi C++, limbaje care permit referinţe în afara gamei corecte. În consecinţă, programele Java sunt mult mai puţin susceptibile la erori şi carenţe de securitate decât programele C/C++.

Lungimea unui tablou
Fiecare tablou are o variabilă instanţă constantă (final) care conţine lungimea tabloului. Puteţi afla câte elemente poate păstra un tablou scriindu-i numele urmat de **.length**. În exemplul anterior, `a.length` ar fi 100. Ţineţi 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 regăseşte în condiţia de testat din 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 aveţi nevoie doar *să referiţi valoarea* fiecărui element, puteţi folosi bucla oarecum mai simplă din Java 5, care ţine evidenţa indicelui şi 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 şi pune 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;           // Declara un tablou de int
a = new int[1000]; // Creeaza 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 foloseşte bucla „for each" din Java 5 pentru a face însumarea, lucru care ne eliberează de folosirea unui indice, dacă ceea ce avem nevoie este doar să obţinem toate valorile succesive. Acest fel de buclă doar ne obţine valorile, aşa că nu poate fi folosită pentru a stoca valorile ca și 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 iniţiale pentru elementele de tablou – zero/null/false

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

2.2. Iniţializarea tablourilor

La declararea unui tablou, puteţi şi să alocaţi un obiect tablou preiniţializat în aceeaşi instrucţiune. În acest caz, nu furnizaţi mărimea tabloului fiindcă Java numără valorile din iniţializare 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 referinţă (numele Java pentru adresă sau pointer) la un obiect tablou. Referinţele necesită în mod tipic doar 4 octeţi. La crearea unui obiect tablou cu new se returnează o referinţă, iar acea referinţă poate fi asignată unei variabile. La asignarea unui tablou la un altul, doar referinţa este copiată. Spre exemplu:

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

2.3. Noţiuni mai complexe

Tablouri anonime
Java 2 a adăugat tablourile anonime, care vă permit să creaţi un tablou de valori nou oriunde în program, nu doar într-o iniţializare într-o declaraţie. Exemplu de tablou anonim:

```
new String[] {"Su", "Mo", "Tu", "We", "Th", "Fr", "Sa"}
```

Acest stil anonim de tablou poate fi folosit și în alte instrucțiuni, spre exemplu:
```
String[] days = new String[] {"Su", "Mo", "Tu", "We", "Th", "Fr", "Sa"};
```

Sintaxa tablourilor anonime poate fi folosită şi în alte părţi ale programului, spre exemplu:
```
x = new String[] {"Su", "Mo", "Tu", "We", "Th", "Fr", "Sa"};
```

Trebuie să aveţi grijă să nu creaţi astfel de 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 iniţializare pot fi expresii arbitrare. Spre exemplu, apelul următor creează două tablouri anonime 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);
```

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

C poate arăta foarte urât având o parte a declaraţiei înainte de variabilă şi o parte după. Java are un stil mult mai curat, în care toată informaţia de tip poate fi scrisă fără a folosi o variabilă. Sunt situaţii în care acest stil – Java – este singura notaţie 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 uită 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**.

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

În plus, există metoda System.arrayCopy() pentru a copia tot tabloul sau o subgamă a lui într-un alt tablou.

Inversarea unui tablou

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

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

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

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

O buclă for care să meargă spre mijloc ar înlocui cele 8 instrucţiuni de mai sus cu ceea ce urmează. Ambele bucle sunt la fel de rapide, aşa că alegeţi-o pe cea care vi se pare mai uşor de înţeles.

```
for (int stinga=0, int dreapta=b.length-1; stinga<dreapta; stinga++, dreapta--) {
    // interschimba stinga cu dreapta
    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.

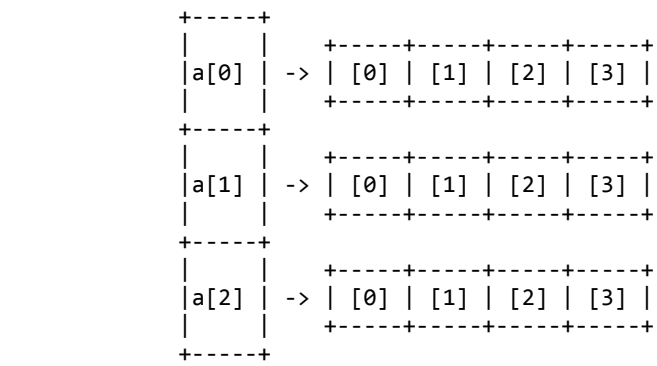
Exemplele care urmează folosesc toate tablouri bidimensionale, dar sintaxa și codificarea se poate extinde pentru oricâte dimensiuni. Prin convenție, tablourile bidimensionale au linii (orizontale) şi coloane (verticale). Primul indice selectează linia (care este un tablou unidimensional în sine), iar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

Vizualizarea tablourilor bidimensionale

Să presupunem ca avem un tablou, a, cu trei linii şi 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 ca o matrice, cu linii şi coloane. Diagrama următoare arată un tablou cu indicii corespunzători.



În Java, un tablou bidimensional este implementat ca un tablou unidimensional de tablouri unidimensionale.

Declararea şi alocarea unui tablou bidimensional

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

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

Valori iniţiale

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

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

Trebuie să daţi o valoare unui element al unui tablou înainte de a-l folosi, fie printr-o iniţializare, fie printr-o asignare.

Exemplu – desenarea tablei de tic-tac-toe

Adesea este cel mai uşor 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 şi 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  );
        }
        else if (board[row][col] < 0)
        { // deseneaza O
            g.drawOval(col*10, row*10, 8, 8);
        }
    }
}
```

3. Mersul lucrării

3.1. Studiaţi şi înţelegeţi textul şi exemplele date.

3.2. Definiţi o clasă Complex pentru lucrul cu numere complexe. Apoi definiți clasa Matrice de numere complexe care să implementeze operaţiile de adunare, scădere, înmulţire de matrice, precum şi înmulțirea cu un scalar. Construiți diagrama UML de clase a aplicației.

3.3. Implementaţi o clasă TablaSah care să păstreze poziţii pe tabla de şah. Figurile şi pionii sunt şi ei clase. Verificaţi corectitudinea mutărilor pe tabla de șah.