Programarea orientată pe obiecte

SERIA CB

DOCUMENTAȚIA

Includeți pe scurt, la începutul programului, câteva explicații (în comentarii) despre ce face programul vostru, care este scopul lui și tehnicile utilizate

Introduceți comentarii la începutul unei secțiuni și explicați fiecare pas major sau instrucțiunile mai dificil de înțeles

Pe lângă comentariile clasice (pe linie // și în bloc /* */), folosiți și comentarii de tip ***javadoc***, care încep cu /** și se termină cu */.

Ele pot fi extrase într-un fișier de documentație HTML folosind comanda ***Tools -> Generate javadoc***

Utilizați-le pentru a comenta o clasă sau o metodă și amplasați-le în așa fel încât să preceadă metoda sau clasa respectivă pentru a putea fi extrase.

STILUL DE PROGRAMARE

Folosiți **litere mici pentru variabile și metode** (*camelCase*)

Folosiți **litere mari pentru constante** (*UPPERCASE*)

Numele de clase trebuie să aibă prima literă mare (*NumeClasa*)

Pentru numele claselor, evitați folosirea celor care sunt predefinite în limbajul Java – cum ar fi clasa `Math`

Indentați codul și lăsați spațiu după fiecare operator și operand

Puteți folosi **atât next line style**, cât și **end-of-line style**

https://www.jetbrains.com/help/idea/code-style-java.html.

```
double[] tablou = new double[5]; //Initializare tablou
tablou[0] = 0.0;
tablou[1] = 1.0;
tablou[2] = 2.0;
tablou[3] = 3.0;
tablou[4] = 40;
```

TABLOURI UNIDIMENSIONALE

Pentru organizarea eficientă a datelor, Java și alte limbaje de nivel înalt, folosesc o structură numită tablou, ce stochează o colecție secvențială de dimensiune fixă cu elemente de același tip.

Următoarea secvență creează un tablou cu elemente de tipul `double`, de dimensiune `100`.

```
double[] numereDouble = new double[100];
```

Declararea unei variable de tip tablou se face cu următoarea sintaxă:

```
tip_element[] referință_tablou;
```

Spre deosebire de tipurile de date primitive, declararea unei variabile tablou nu alocă spațiu în memorie pentru aceasta.

Declararea creează doar o locație de stocare pentru referința tabloului. Dacă o variabilă nu conține o referință la un tablou, valoarea acesteia este `null`.

Nu pot fi atribuite elemente unui tablou, decât dacă acesta a fost în prealabil creat (și nu doar declarat). După ce o variabilă tablou este declarată, se poate crea un tablou prin apelarea operatorului `new`, cu următoarea sintaxă:

```
referinta_tablou = new tip_element[dimensiune];
```

Această instrucțiune realizează două lucruri: creează un tablou utilizând apelul lui `new` și asignează referința noului tablou variabilei `referinta_tablou`.

Declararea unei variabile tablou, crearea acestuia și asignarea referinței pot fi combinate într-o singură instrucțiune astfel:

```
tip_element referința_tablou[] = new tip_element[dimensiune];
```

METODELE DIN CLASA SCANNER

Pentru citirea tipurilor de date

`nextByte()`

Citește un întreg de tipul `byte`

`nextShort()`

`nextInt()`

`nextLong()`

`nextFloat()`

`nextDouble()`

`next()`

Citește un șir care se termină cu caracterul spațiu

`nextLine()`

Citește o linie de text, până la întâlnirea lui `Enter`

Parametrii de tipul tablou sunt transmiși prin referință, spre

Dimensiunea spațiului alocat tabloului poate fi obținută prin apelarea variabilei **`length`**: `referinta_tablou.length`.

De exemplu, `tablou.length` este `5`.

Indicii tabloului sunt cuprinși între `0` și `referinta_tablou.length - 1`.

Inițializarea tablourilor combină într-o singură instrucțiune declararea, crearea și asignarea valorilor, și are forma:

`tip_element[] referinta_tablou = {val0, val1, ..., valk};`

De exemplu, instrucțiunea:

```
double[] lista = {1.9, 2.9, 3.4, 3.5};
```

este echivalentă cu:

```
double[] lista = new double[4];
lista[0] = 1.9;
lista[1] = 2.9;
lista[2] = 3.4;
lista[3] = 3.5;
```

Java se folosește de **`System.out`** pentru a se referi la dispozitivul standard de ieșire (de obicei un terminal/consolă), și de **`System.in`** pentru a lucra cu fluxul de intrare (de obicei tastatura).

Pentru a scrie informație la consolă, se utilizează metoda `println` ce afișează o valoare primitivă sau un șir de caractere (`String`).

Citirea de la consolă nu este direct suportată de Java, dar se poate folosi **clasa `Scanner`** pentru a crea un obiect ce preia valori de la `System.in`, de exemplu:

```
Scanner s = new Scanner(System.in);
```

Clasa Scanner se află în **pachetul `java.util`**, care trebuie importat.

O metodă poate returna un tablou. Următoarea metodă copiază elementele tabloului `tablou` dat ca parametru și returnează tabloul `copie`.

```
public static int[] copiere(int[] tablou){
        int copie[] = new int[tablou.length];
        for (int i = 0; i < tablou.length; i++)
            copie[i] = tablou[i];
        return copie;
    }
```

deosebire de tipurile primitive, care sunt transmise **prin valoare**.

Pasarea unui **tablou ca argument al unei metode** determină modificarea valorilor lui să fie vizibile și în afara metodei. JVM stochează tablourile într-o zonă de memorie numită *heap*, utilizată pentru alocare dinamică, și în care blocurile de memorie sunt alocate și eliberate în mod arbitrar.

java.utils.Arrays

`sort()`-sortarea

`binarySearch()` căutarea binară

`equals(tablou1, tablou2)` verifică dacă `tablou1` și `tablou2` au conținut identic

`fill(tablou, 0)` umple întregul tablou cu valori de `0`.

Semnul plus (+) are două semnificații:

- operația aritmetică de adunare
- concatenarea șirurilor de caractere (string-uri).

  Dacă unul dintre operanzi este un **non-string** (întreg, float, etc.), acesta este **automat convertit la `String`** și concatenat cu celelalte șiruri de caractere.

Pentru copierea unui tablou în altul, instrucțiunea **`tablou2`**

Următorul exemplu presupune **inițializarea unui tablou cu valori citite de la consolă**:

```
double[] tablou = new double [10];
java.util.Scanner s = new java.util.Scanner(System.in);
for (int i = 0; i < tablou.length; i++) {
    tablou[i] = s.nextDouble();
}
```

Inițializarea cu valori aleatoare (random) cuprinse între `0` și `100` se realizează astfel:

```
for (int i = 0; i < tablou.length; i++) {
    tablou[i] = Math.random() * 100;
}
```

Afișarea elementelor tabloului se poate face parcurgând elementele astfel:

```
for (int i = 0; i < tablou.length; i++) {
    System.out.print(tablou[i] + " ");
}
```

Un tablou de tipul `char[]` poate fi afișat folosind o singură instrucțiune `println`:

```
char[] nume = {'A', 'n', 'a'};
System.out.println(nume);
```

Pe lângă modalitățile de iterare cu ajutorul instrucțiunilor repetitive (`while`, `for` și `do-while`), Java suportă **`for each loop`** sau **`enhanced loop`**, care permite traversarea tabloului secvențial, fără a folosi o variabilă index.

```
for (double u: tablou) {
    System.out.println(u);
}
```

Accesarea unui index din tablou aflat în afara intervalului `[0; referinta_tablou.length-1]` generează excepția `ArrayIndexOutOfBoundsException`.

Dacă indexarea pornește de la `1` în loc de `0`, atunci apare eroarea numită off-by-one.

În continuare, metoda `afisareTablou` are ca parametru un tablou și realizează afișarea valorilor lui.

```
public static void afisareTablou(int[] tablou) {
    for (int i = 0; i < tablou.length; i++) {
        System.out.print(tablou[i] + " ");
    }
}
```

Un posibil apel al acestei metode se poate realiza astfel:

```
afisareTablou(new int[]{3, 1, 2, 6, 4, 2});
```

În acest caz nu avem o referință explicită pentru tablou, prin urmare acesta este un **tablou anonim**.

`= tablou1` nu va copia conținutul tabloului către care indică referința **`tablou1`** în **`tablou2`**, ci **va copia referința lui.**

Astfel, `tablou1` și `tablou2` vor indica către același tablou, iar `tabloul` va mai fi referit deloc (deci nu va mai putea fi accesat). El devine gunoi (*garbage*) și va fi colectat automat de către Java Virtual Machine (JVM).

Înainte de atribuire, `tablou1` și `tablou2` indică către două locații de memorie diferite. După atribuire, referința tabloului `tablou1` este pasată și variabilei `tablou2`.
Există două modalități corecte de a copia conținutul unui tablou în altul:

- **Utilizarea unei instrucțiuni repetitive** care copiază element cu element din primul tablou în al doilea tablou.
- **Folosirea metodei `arraycopy` din clasa `System`**

```
arraycopy
(sursa, src_pos, dest, dest_pos,
lungime);
System.arraycopy
(sursa, 0, dest, 0, sursa.length);
```

Metoda nu alocă spațiu pentru `dest`. Acest tablou trebuie creat și alocat în prealabil. După apelul metodei `arraycopy`, `sursa` și `target` vor avea același conținut, dar locații de memorie diferite

Tablouri bidimensionale

Declararea unei variabile bidimensionale se face astfel:

```
tip_element[][] referinta_tablou;
```

sau

```
tip_element referinta_tablou[][]; // versiune corectă, dar mai puțin preferabilă
```

De exemplu:

```
int[][] matrice;
```

sau

```
int matrice[][];
```

Pentru crearea unui tablou bidimensional cu `5` linii și `5` coloane, în care indecșii merg de la `0` la `4` atât pentru linii, cât și pentru coloane, se procedează astfel:

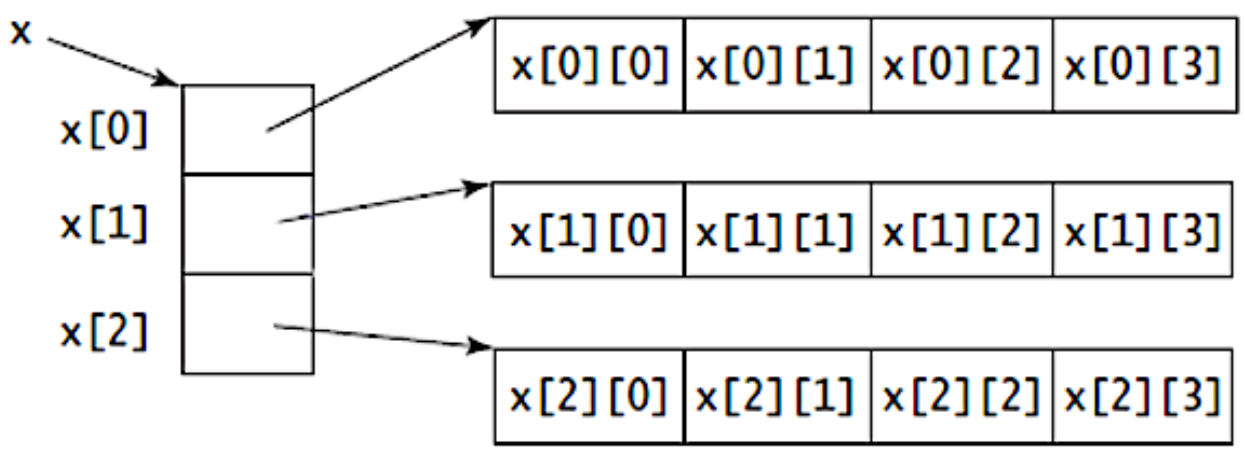
```
int[] matrice = new int[5][5];
```

Declararea, crearea și inițializarea unui tablou într-o singură linie de cod se face:

```
int[][] tablou={{1, 2, 3},{4, 5, 6},{7, 8, 9},{10, 11, 12}};
```

Un tablou bidimensional este un tablou în care fiecare element este un tablou (a se vedea figura). Dimensiunea tabloului, care reprezintă numărul de elemente al său, se poate obține cu apelul `x.length, x[0].length, x[1].length, ..., x[x.length-1].length`.

Dacă `x = new int[3][4], x.length = 3, x[0].length = 4, x[1].length = 4, x[2].length = 4`.



Rândurile unui tablou bidimensional pot avea diverse lungimi:

```
int[][] tablou = { {1, 2, 3, 4, 5}, {2, 3, 4, 5}, {3, 4, 5}, {4, 5}, {5} };
```

Inițializarea unui tablou bidimensional cu valori citite de la consolă se face astfel:

```
java.util.Scanner s = new Scanner(System.in);
int[][] matrice = new int[10][10];
for (int linie = 0; linie < matrice.length; linie++) {
    for (int col = 0; col < matrice[linie].length; col++) {
        matrice[linie][col] = s.nextInt();
    }
}
```

Afișarea unui tablou bidimensional se poate face astfel:

```
for (int linie = 0; linie < matrice.length; linie++) {
    for (int col = 0; col < matrice[linie].length; col++) {
        System.out.print(matrice[linie][col] + " ");
    }
    System.out.println();
}
```

Modalitatea în care tablourile bidimensionale pot fi pasate ca argumente ale metodelor, este similară tablourilor unidimensionale.

```
public static int suma(int[][] m) {
    int s = 0;
    for (int linie = 0; linie < m.length; linie++) {
        for (int col = 0; col < m[linie].length; col++) {
            s += m[linie][col];
        }
    }
    return s;
}
```

Iar în metoda `main`:

```
int[][] m = new int[2][3];
// inițializare valori aleatoare cuprinse între 0 și 100 în matricea m
for (int linie = 0; linie < m.length; linie++) {
    for (int col = 0; col < m [linie].length; col++) {
        m [linie][col] = (int)(Math.random() * 100);
    }
}
// apelul metodei sum

System.out.println("Suma=" + suma(m));
```

În Java se pot crea și tablouri multidimensionale sau n-dimensionale.

```
double[][][] x = new double[2][3][5];
```

`x` este un tablou tridimensional, ce conține pe `x[0]` și `x[1]` care sunt tablouri bidimensionale. `x[0]` și `x[1]` conțin fiecare câte `3` elemente (`x[0][0]`, `x[0][1]`, `x[0][2]`, care sunt tablouri unidimensionale cu `5` elemente fiecare, respectiv `x[1][0]`, `x[1][1]`, `x[1][2]`, care la fel, conțin câte `5` elemente fiecare).

APLICAȚII

1. Scrieți următorul cod într-un fișier numit **`ComputeArea.java`**. Rulați programul.

```
public class ComputeArea {
/** Main method */
public static void main(String[] args) {
    double radius;
    double area;
    radius = 20; // Assign a radius radius = 20;
    area = radius * radius * 3.14159; // Compute area
    System.out.println("Area for the circle of R=" +
        radius + " is " + area); // Display results
    }
}
```

2. Pornind de la fișierul anterior, creați un nou fișier **`ComputeAreaPerimeter.java`** în care veți integra următoarele sarcini:
    a. Citirea razei cercului să se facă de la tastatură
    b. Calculul ariei cercului (din exercițiul anterior)
    c. Calcului perimetrului cercului
    d. Modificați programul astfel încât utilizatorul să fie întrebat ce dorește să calculeze: aria sau perimetrul. Utilizatorul va introduce un caracter `"A"` sau `"P"`, care va declanșa una sau cealaltă dintre opțiuni.
    e. Introduceți de la tastatură o rază negativă. Ce se întâmplă? Modificați programul astfel încât să semnaleze un mesaj de eroare atunci când raza introdusă este negativă.
    f. Modificați programul astfel încât utilizatorul să poată introduce mai multe raze de la tastatură (într-o buclă continua), iar pentru fiecare rază introdusă să poată alege dacă dorește calculul ariei sau a perimetrului. Se va ieși din buclă atunci când valoarea razei introduce este negativă.

3. Tablouri
    a. Se citește de la tastatură o variabilă întreagă `n` (`n <= 10`) si apoi se citesc perechi de valori de tip `String` si `double`, ce vor reprezenta numele si media unui student. Sa se stocheze valorile in doi vectori, `studenti` (de tip `String[]`) si `mediiFinale` (de tip `double[]`) si sa se afișeze la consola.

    Dacă de la tastatură se introduc următoarele valori:

```
n = 3;
Ana Popescu
9.9
Maria Ionescu
9.8
Ion Ionescu
9.7
```

    Atunci, rezultatul va fi:

```
Ana Popescu  Maria Ionescu Ion Ionescu
 9.9 9.8 9.7
```

    b. Se citește de la tastatură o variabilă întreagă `n` (`n <= 10`), apoi se citesc elementele a două matrici pătratice de dimensiune `n`. Se cere să se determine și să se afișeze pe ecran matricea sumă.

Referințe:

1. Oana Balan, Mihai Dascalu. **Programarea orientata pe obiecte in Java**. Editura Politehnica Press, 2020
2. Bert Bates, Kathy Sierra. **Head First Java: Your Brain on Java - A Learner's Guide 1st Edition**. O'Reilly Media. 2003
3. Herbert Schildt. **Java: A Beginner's Guide**. McGraw Hill. 2018
4. Barry A. Burd. **Java For Dummies**. For Dummies. 2015
5. Joshua Bloch. **Effective Java**. ISBN-13978-0134685991. 2017
6. Harry (Author), Chris James (Editor). **Thinking in Java: Advanced Features (Core Series) Updated To Java 8**
7. Learn Java online. Disponibil la adresa: https://www.learnjavaonline.org/
8. Java Tutorial. Disponibil la adresa: https://www.w3schools.com/java/
9. The Java Tutorials. Disponibil la adresa: https://docs.oracle.com/javase/tutorial/