

# Programare avansata pe obiecte – laborator 1

Alina Puscasu

[alina.puscasu@endava.com](mailto:alina.puscasu@endava.com)

[https://github.com/alina-puscasu/pao\\_lab\\_2022](https://github.com/alina-puscasu/pao_lab_2022)

Limbajul Java prezinta urmatoarele **caracteristici**:

- Este un **limbaj de programare orientat obiecte**
- Este un **limbaj portabil** => **independent de platforma**
- Programele Java sunt **interpretate** => este utilizata o masina virtuala Java (JVM) care interpreteaza un cod compilat al programelor: *.java (fisier cod sursa) ---> javac (compilator) ---> .class (bytecode) ---> java (interpretor, emulator de cod bytecode = JVM) ---> OS (sistem de operare)*
- Este un **limbaj case sensitive**

## Structura fisier .java

1. Cand cream o clasa, fisierul .java are numele singurei clase publice din aceasta. O sa invatam mai tarziu mai multe despre modificatorii de access.
  - a. **class** e un cuvânt cheie folosit pentru a defini o clasa
  - b. **Numele** pe care i-l dam clasei trebuie sa respecte urmatoarele reguli:
    - i. Sa inceapa cu litera mare
    - ii. Nu poate incepe cu cifre, dar poate contine cifre
    - iii. Nu poate contine spatii si operatori (sau caractere speciale: #)
    - iv. Nu pot fi folosite cuvinte cheie ale limbajului
2. Clasele Java sunt grupate in **pachete**. Declaratia de import ii spune compilatorului ce pachet sa caute pentru a gasi o clasa.
  - a. Exista un pachet special **java.lang** care este importat automat.
  - b. O buna practica este sa invatam sa folosim pachete care ne ajuta sa evitam conflicte si ne usureaza reutilizarea codului.
  - c. Numele pachetelor trebuie sa fie lowercase; intre diferite pachete separarea se face cu .
3. Metoda principala – **main** – entry point al programului

Element	Unde se afla in fisier .java
Package	Prima linie din fisier
Import	Dupa package
Class declaration	Dupa import
Fields/methods/blocks	Oriunde in clasa
Comentarii: <i>//, /**/, /**</i>	Inainte si dupa package, inainte si dupa class, inainte, in si dupa metode

Conventii nume fields/methods/parameters:

- Numele poate contine litere (mari sau mici), numere, \$ sau \_
  - Este best practice ca numele de variabile sa inceapa cu litera mica
  - Este best practice sa folosim camelCase: daca numele contine mai multe cuvinte, primul este cu litera mica iar urmatoarele cu litera mare
  - Nu pot fi folosite cuvinte cheie ale limbajului
- Atentie, Java este case sensitive!

**Var** – permite sa nu specificam un tip pentru variabile, acesta va fi automat alocat de compilator

<https://dzone.com/articles/finally-java-10-has-var-to-declare-local-variables>

<https://www.geeksforgeeks.org/var-keyword-in-java/>

## Tipuri de date

### Tipurile primitive

- byte, short, int, long, float, double, char, Boolean
- sunt alocate in zona de memorie de tip stiva
- <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

```
public class Primitives {
    /* NO Floating point - can be written in base 10 / binary / octal / hexadecimal*/

    // 8 bit: [-128...127]
    byte b1 = 10;
    byte b2 = 023; // octal

    // 16 bit
    short s1 = 10;
    short s2 = 0xFF; // hexa

    // 32 bit; 4 bytes
    int i1 = 10;
    int i2 = 0b10110; // binary

    // 64 bit; 8 bytes
    long l1 = 10;

    /* Floating point - can only be in decimal form */
    // 32 bit floating point; by default in java orice valoare cu virgula e considerata de tip double
    // -> de aceea folosim un literal (f, F) pentru float
    float f1 = 123.45f;

    // 64 bit floating point; by default in java orice valoare cu virgula e considerata de tip double
    double d1 = 123.456;

    // 16 bit
    char c1 = 'a';
    char c2 = '\n';

    boolean k1 = false;
    boolean k2 = true;

    /** numeric literals can have underscores inside them, for separating groups. However, a numeric literal cannot start
     * or end with an underscore (Eg: _52 is an invalid number). */
    int underscoreInt= 1____234; // it represents the number 1234
}
```

- valori default:
  - o byte/short/int – 0
  - o long – 0L
  - o boolean – false
  - o char – ‘\u0000’
  - o float – 0.0f
  - o double – 0.0
- **Wrapper Classes**
  - contin un set de metode utilitare
  - sunt folosite pentru a folosi primitivele pe post de obiecte, atunci cand este necesar

Primitiva	Wrapper Class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

- numim **auto boxing** cand trecem dintr-un tip de date primitive in clasa wrapper corespunzatoare - numim **unboxing** procesul invers

### Tipuri de date referinta

- tablouri, clase si interfete
- alocate dinamic prin operatorul new in zona de memorie HEAP

## Operatori

Se clasifica in urmatoarele categorii (in ordinea precedentei):

1. Postfix increment si decrement: **expr++, expr--**
2. Prefix increment si decrement si unary: **++expr, --expr, +expr, -expr, ~, !**
3. Multiplicativi: **\*, /, %**
4. Aditivi: **+, -, concatenare stringuri: +**. Asociativitatea este de la stanga la dreapta!
5. Shiftare: **<<, >>, >>>**
6. Relationali: **<, <=, >, >=, instanceof**
7. Egalitate: **==, !=**
8. Bitwise AND (**&**)

9. Bitwise exclusive OR (^)
10. Bitwise inclusive OR (|)
11. Logical AND (&&)
12. Logical OR (||)
13. Ternary: x = (expression) ? value if true : value if false
14. Asignare: =, +=, -=, \*=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=

## Instructiunile limbajului Java

### 1. Decizionale

- a. If/else
- b. Switch (case)

Valori acceptate: primitive (char, int, short, byte) si clasele wrapper asociate, enum, string

### 2. Repetitive

- a. For/ for each
- b. While/do-while

Do while mai intai executa body-ul metodei, apoi evalueaza conditia

### 3. Salt

- a. Break

Permite intreruperea unei bucle for/do-while/while sau iesirea din switch

- b. Continue

Permite trecea la urmatoarea iteratie a unui ciclu for/do-while/while, ignorand restul instructiunilor din iteratia curenta

## Tablouri

- Structura de date care contine mai multe valori
- Putem avea tablouri de obiecte sau tablouri de tablouri
- Fiecare valoare poate fi accesata prin indice
- Lungimea este stabilita la momentul crearii si poate fi accesata prin variabila length care este de tip primitiv int
- Elementele sunt numerotate de la 0 la n-1 unde n este numarul de elemente
- Accesarea unui element in afara limitelor va genera o exceptie in timpul rularii, `ArrayIndexOutOfBoundsException`
- Clasa `java.util.Arrays` ofera metode utile in lucrul cu vectori
  - `Sort` -> sorteaza ascendent
  - `Equals` -> testarea egalitatii valorilor a 2 vectori (acelasi numar de elemente si pentru fiecare indice valorile corespunzatoare din cei doi vectori sunt egale)
  - `binarySearch` -> cautarea binara a unei anumite valori
  - alte metode: <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Arrays.html>

## Exercitii

1. Scrieti un program care sa afișeze toate numerele pare din intervalul  $[0, n]$ , unde  $n$  este un numar citit de la tastatura.
2. Scrieți un program care sa compare doua numere  $a$  și  $b$  citite de la tastatura si sa afiseze numarul mai mare.
3. Scrieți o metoda care sa calculeze factorialul unui numar  $n$  citit de la tastatura.
4. Fiind dat un numar  $n$ , scrieti o metoda care insumeaza toti multiplii de 3 si 5 pana la  $n$  (inclusiv).
5. Cititi de la tastatura  $n$  numere. Elementele citite vor fi organizate in doi vectori diferiti, in functie de paritate (de ex: elementele pare in vectorul  $x$ , iar cele impare in vectorul  $y$ ).
6. Cititi de la tastatura  $n$  note, numere intregi, intr-un vector. Cand cititi valoarea -1 de la tastatura, citirea notelor se opreste si programul afiseaza media acestora.
7. Sa se afiseze al  $n$ -lea termen din seria Fibonacci, unde  $n$  este citit de la tastatura.