

# **ELEMENTE AVANSATE DE PROGRAMARE**

**Conf.univ.dr. Ana Cristina DĂSCĂLESCU**

# Conținutul tematic

- **Introducere în Limbajul Java:** structura programelor Java, identificatori, variabile, operatori, instrucțiuni.
- **Concepte de programare orientată pe obiecte în Java:** Clase și obiecte, Moștenire, Polimorfism
- **Lucruri cu tablouri în Java. Șiruri de caractere.**
- **Excepții:** tratarea Excepțiilor.
- **Fluxuri de intrare/ieșire:** fluxuri primitive, fluxuri de procesare.
- **Interfețe:** definirea unei interfețe, extinderea interfețelor
- **Clase Abstracte**
- **Colecții de date**
- **Fire de execuție**
- **Lucrul cu baze de date în Java**
- **Tehnologia Java Servlet**
- **Interfețe Grafice (Awt, Swing)**

# Bibliografie

- H. Georgescu – **Introducere în universul Java**, Ed. Tehnică, 2002
- Paul Deitel, Harvey Deitel - **Java How To Program in Java**, 10th Edition, 2012
- Bruce Eckel - **Thinking in Java**, ISBN-13: 007-6092039389, 2012
- Tanasă, C. Olaru, Ș. Andrei - **Java de la 0 la expert**, Editura Polirom, 2011;
- C. Frăsinaru - **Curs practic de Java**, Editura Matrix Rom, 2010.
- Joshua Bloch – **Effective Java**, Addison Wesley, 2013
- Tutoriale: <http://www.tutorialspoint.com/java/>

# Examinare

- 20% Laborator
- 20% Proiect
- 60% Evaluare finală în sesiune
- Structura lucrării finale: Clase și obiecte, tablouri de date, șiruri de caractere, clase abstracte și interfețe, colecții de date, fluxuri, tratarea excepțiilor, fire de executare, interfețe grafice, serveturi și appleturi.

# Tematică Curs 1

- Tehnologia Java
- Structura unui program. Primul program
- Structura lexicală
- Tipuri de date
- Variabile
- Instrucțiuni
- Pachete de clase/Importul unei clase sau interfețe
- Crearea unui pachet
- Citirea datelor de la tastatură

# Tehnologia Java



- 3 bilioane de telefoane mobile au la baza tehnologie Java
- anual 930 milioane de descarcari pentru JRE

- În anul 1991 firma **Sun Microsystems** finanțază proiectul Green, coordonată de **James Gosling**.
- Specificațiile limbajului nou, inițial denumit **OAK**, iar apoi **Java 1.0** sunt finalizate în anul 1995
- În anul 1995 compania Sun Microsystems vinde licența firmelor IBM, Microsoft, Adobe și Netscape.
- În decembrie 1998-1999 apare **Java 2**
- Pe 8 Mai 2007, nucleul ce constituie Java este declarat free-ware/open-source
- În 2009-2010 Sun Microsystems este cumparata de Oracle
- În aprilie 2014 apare Java 8



# Tehnologia Java – Caracteristici principale

## ➤ Limbaj compilat și interpretat

- Programele Java sunt mai întâi compilate în fișiere intermediare (**byte code**) după care sunt interpretate în mediul de execuție Java.

## ➤ Limbaj independent de platforma (portabilitate)

- Masina virtuală Java - **Java Virtual Machine (JVM)** este mediul de execuție al limbajului Java.
- JVM este înglobată în platforma JDK (**Java Development Kit**)
- WORA – Write Once, Run Anywhere

# Tehnologia Java

## ➤Simplitate

- Elimină supraîncarcarea operatorilor, moștenirea multiplă, renunță la utilizarea variabilelor pointer.

## ➤Robustețe:

- Gestionează automat procesul de alocare a memoriei
- Verificarea dinamică a codului (**Just in time complilation- JIT**)

## ➤Permite programarea concurentă

- Fire de executare

## ➤Permite programarea distribuită

- Se folosesc sisteme precum RMI, CORBA



# În tehnologia Java sunt cuprinse patru componente:

**1. Limbajul Java propriu-zis.** Limbajul Java este orientat pe obiecte și se aseamăna din punct de vedere al sintaxei cu C++.

## **2. Un compilator**

- Compilatorul Java traduce instrucțiunile scrise în limbajul Java (stocate în fișiere cu extensia **.java**) în instrucțiuni generice “bytecode” (stocate în fișiere cu extensia **.class**) care sunt “înțelese” de mașina virtuală.

**3. Mașina virtuală**, denumită JVM (Java Virtual Machine) care interpretează fișierele cu extensia **.class** ce conțin “bytecode”.

**4. Pachete de clase Java.** Conține un set de componente utile care pot fi reutilizate în diverse aplicații informatice.

# Platormele de lucru Java

## ➤ Platforma **J2SE** (Standard Edition)

- Aplicații independente, aplicații cu interfețe grafice, aplicații Web (client, server)

## ➤ Platforma **J2ME** (Micro Edition)

- Dedicata dispozitivelor portabile sau cu resurse limitate (telefoane mobile, PDA-uri etc.)

## ➤ Platforma **J2EE** (Enterprise Edition)

- Aplicații de tip server, cu grad sporit de complexitate.
- Java EE include resursele disponibile în Java SE și adaugă elemente necesare aplicațiilor distribuite

## ➤ Distribuțiile Java sunt oferite gratuit

– <http://www.oracle.com>

# Tipuri de aplicatii Java

## 1) Aplicatii de sine statatoare (stand –alone)

- Nu necesită un browser Web
- Pot fi de două tipuri: aplicații de tip consolă, aplicații cu interfețe grafice (GUI)

## 2) Aplicatii Web

- Aplicații care se execută pe partea de client (applet)
- Aplicații care se execută pe partea de server (servleturi, JSP)

## 3) Aplicații distribuite

- Folosesc sisteme precum RMI (Remote Method Invocation), CORBA

## 4) Aplicații cu baze de date

- Pot fi dezvoltate prin aplicații de tipul 1, 2, 3

# Aplicații de sine stătătoare

- Sunt programe care pot fi executate în afara contextului unui browser Web.
- Conțin cel puțin o clasă, numită clasă de bază, declarată **public**, iar denumirea acesteia coincide cu numele fișierului;
- În cadrul clasei publice trebuie încapsulată funcția principală *main()* cu sintaxa :

```
public static void main (String args[])
```

# Etapele de realizare ale unei aplicații

## 1. Scrierea codului sursă

- se poate utiliza orice editor obișnuit sau un mediu specializat (NetBeans, Eclipse etc.)

```
class FirstApp
{
    public static void main( String args[])
    {
        System.out.println("Hello world!");
    }
}
```

## 2. Salvarea fișierelor sursă

- se va face în fișiere care au obligatoriu extensia **java**.

**Fisierul care conține codul sursă al clasei publice trebuie să aibă același nume cu cel al clasei!!!**

C:\intro\FirstApp.java

## 3. Compilarea aplicației

- pentru compilare se utilizează compilatorul **javac** din distribuția JDK.

javac FirstApp.java

## 4. Rularea aplicației

- se realizează cu interpretorul java, apelat pentru unitatea de compilare corespunzătoare clasei principale prin comanda:

java FirstApp



# Structura lexicală

## ➤ Setul de caractere: **Unicode**

- înlocuiește ASCII
- un caracter se reprezintă pe **2 octeți**
- conține 65536 de semne
- compatibil ASCII : primele 256 caractere sunt cele din ASCII
- este structurat în blocuri:
  - Basic, Latin, Greek, Arabic, Gothic, Currency, Mathematical, Arrows, Musical, etc.
  - <http://www.unicode.org>.

# Cuvinte cheie

➤ Cuvintele rezervate sunt, cu câteva excepții, cele din C++.

abstract	boolean	break	byte	case
catch	char	class	continue	default
do	double	else	extends	final
finally	float	for	if	implements
import	instanceof	int	interface	long
new	null	package	private	protected
public	return	short	static	super
switch	synchronized		this	throw
throws	try	void	while	

# Identificatori

➤ Sunt secvențe nelimitate de litere și cifre Unicode plus simbolul “\_”, ce încep cu o literă sau “\_”.

➤ Identificatorii nu au voie să fie identici cu cele rezervate.

**Example:** a1, FirstApp, factorial, etc.

## Convenție:

- Nume de clasa: prima literă mare ( exp. *Complex*)
- Nume de metodă: prima literă mică ( exp. *aduna*, *adunaComplex*)
- Nume variabilă: prima literă mică (exp. *var1*)
- Nume constantă: prima literă mare sau tot numele cu litere mari (exp. *Pi*, *PI*)

# Constante

## ➤ Intregi (10, 16 -0x, 8-0)

- normali - se reprezintă pe 4 octeți (32 biți)
- lungi - se reprezintă pe 8 octeți (64 biți) și se termină cu caracterul **L** (sau **l**).

## ➤ Flotanți: 1.0, 3f, 4D

- să aibă cel puțin o zecimală după virgulă
- să aibă sufixul **F** sau **f**

## ➤ Logici: true, false

# Constante

- Character: 'J', 'a', 'v', 'a', '\n'
- Secvențele escape predefinite în Java sunt:
  - '\b' : Backspace (BS)
  - '\t' : Tab orizontal (HT)
  - '\n' : Linie nouă (LF)
  - '\f' : Pagină nouă (FF)
  - '\r' : Inceput de rând (CR)
  - '\"' : Ghilimele
  - '\'' : Apostrof
  - '\\' : Backslash

# Operatori

➤ Java pune la dispoziția utilizatorului aproximativ 40 de operatori.

**Observație:** Programatorul poate defini metode noi, dar nu și operatori noi!!

1. Operatorii aritmetici:  $+$ ,  $-$ ,  $*$ ,  $/$ ,  $\%$ ,

2. Operatorul de atribuire:  $=$

Pe lângă operatorul  $=$  folosit standard pentru atribuire, mai pot fi folosiți și operatorii:

$+=$     $-=$     $/=$     $*=$     $\%=$

3. Operatorii logici:  $\&\&$ (and),  $\|\|$ (or),  $!$ (not)

3. Operatorii relaționali:  $<$ ,  $<=$ ,  $>$ ,  $>=$ ,  $==$ ,  $!=$



## 6. Operatorul + pentru concatenarea șirurilor

```
String s1="Sursa";  
String s2="Java";  
System.out.println(s1 + " " + s2); // Sursa Java
```

## 7. Operatori pentru conversii (cast) : (tip-de-data)

```
int a = (int) 'a';  
char c = (char) 96;
```

## 8. Operatorii de incrementare/decrementare: ++ --

# Tipuri de date

## 1. Tipurile primitive:

- **aritmetice**

- întregi: `byte` (1 octet), `short` (2), `int` (4), `long` (8)

- reale: `float` (4), `double` (8)

- **caracter**: `char` (2)

- **logic**: `boolean` (`true` și `false`)

## 2. Tipuri de date de referință

- **tabouri, clasele și interfețele**

- Valoarea unei variabile de acest tip este o referință (adresă de memorie) către valoarea reprezentată de variabila respectivă.

## 3. Tipul **null**

**Nu există: `pointer`, `struct` și `union`!!!**

# Variable

➤ Declararea variabilelor:

< șir de modificatori >   < tip >   < listă de identificatori > ;

unde < șir de modificatori > *specifică*

- un modificador de acces : **public**, **protected**, **private**
- unul din cuvintele rezervate: **static**, **final**;

**Exemplu:**

```
int x;  
static float v;
```

- O declarare poate să apară oriunde în cadrul unei clase, metode sau a unui bloc de inițializare.

## ➤ Inițializarea variabilelor:

```
Tip numeVariabila = valoare;
```

**Exemplu:** `int x=8, y=5;`

**Observație** Limbajul Java atribuie valori inițiale implicite variabilelor. Ele sunt:

- `false` (pentru **boolean**) ; `'\u0000'` (pentru **char**) ; **null** (pentru referințe)
- `0` (pentru orice tip întreg) ; `+0.0f` sau `+0.0d` (pentru **float** și **double**)

## ➤ Declararea constantelor:

```
final Tip numeVariabila;
```

**Exemplu:** `final double PI = 3.14;`

# Categorii de variabile

- Variabile membre ale unei clase.
- Parametrii metodelor – în limbajul Java un parametru poate fi transmis doar prin valoare!!!!
- Variabile locale, declarate într-o metodă, vizibile doar în metoda respectivă.
- Variabile locale, declarate într-un bloc, vizibile doar în blocul respectiv.

```
class Exemplu {  
    int a;    //data membra  
    public void metoda(int b)  
    {  
        a = b;  
        int c = 10; //variabilă locală  
        for(int d=0; d < 10; d++)  
            c --;  
    }  
}
```

variabilă locală, vizibilă doar în  
blocul for



# Instrucțiuni

## ➤ Instrucțiuni de decizie:

`if-else, switch-case`

## ➤ Instrucțiuni de salt:

`for, while, do-while`

## ➤ Instrucțiuni pentru tratarea excepțiilor:

`try-catch-finally, throw`

## ➤ Alte instrucțiuni:

`break, continue, return, label:`

# Ce este un pachet?

➤ **Pachet** = Colecție de clase și interfețe salvate în același director.

➤ Scopul:

- Organizarea fișierelor sursă
- Găsirea și utilizarea mai ușoară a claselor
- Evitarea conflictelor de nume
- Controlul accesului

Pachete standard Java: *java.lang* , *java.io*, *java.applet*, *java.awt*, *java.math*, *java.util*, *java.sql*, *java.net*, *java.security*

# Folosirea unui pachet

## 1. Specificarea numelui complet

**Java.numePachet.NumeClasa**// pentru pachete standard

**numePachet.NumeClasa**// pentru pachete definite de programator

### Exemplu:

- **Button** - numele clasei care modelează noțiunea de componentă grafică de tip buton
- **java.awt** - pachetul din care face parte
- **java.awt.Button** - numele complet al clasei

```
java.awt.Button b1=new java.awt.Button("Text");  
java.awt.TextField t1=new  
    java.awt.TextField("Java")
```

# Folosirea unui pachet

## 2. Importul unei clase dintr-un pachet

```
import java.numePachet.numeClasa;  
import numePachet.numeClasa;
```

Exemplu:

```
import java.awt.Button;  
import java.awt.TextField;  
    Button b1=new Button("Text");  
    TextField t1=new TextField("Java");
```

**Problemă – se specifica clauza import pentru fiecare clasa importată**

```
import java.awt.Button;  
import java.awt.TextField;  
import java.awt.List;
```

# Folosirea unui pachet

## 3. Importul întregului pachet

```
import numePachet.*;
```

```
import java.awt.*;
```

```
Button b1=new Button("Text");
```

```
TextField t1=new TextField("Java");
```

**Se poate importa orice pachet de clase care nu este standard!!!**

```
import    nume_pachet.*;
```

```
import    nume_pachet.nume_clasa;
```

**Pachetul java.lang se importă implicit în orice sursă java!!!**

# Structura unui fișier standard

Există trei elemente care pot să apară în fișierul sursă:

- declarații de pachete:

```
package nume_pachet;
```

- instrucțiuni de includere clase:

```
import java.nume_pachet.nume_clase;  
import nume_pachet.nume_clase;
```

- definiții de clase:

```
public class Id_clasa_1{corpul clasei;}  
class Id_clasa_2{corpul clasei;}
```



# Citirea datelor de tip primitiv de la tastatură

➤ Clasa `Scanner` din pachetul `java.util` conține metode pentru citirea formatată de la tastatură.

➤ Crearea unui obiect de tipul `Scanner` pentru citirea de la tastatură:

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

➤ Metode pentru citire formatată:

- `nextInt()` – citirea unui număr întreg
- `nextDouble()` – citirea unui număr real
- `next()` – citirea unui șir de caractere fără spații
- `nextLine()` – citirea unui șir de caractere cu spații