

Introducere în Java

- Ce este Java ?
- De ce Java ?
- Primul program
- Structura lexicală
- Tipuri de date
- Variabile
- Controlul execuției
- Vectori
- Șiruri de caractere
- Argumente de la linia de comandă



Ce este Java ?

- Limbaj de programare
- Platformă de lucru
- Sun Microsystems
- 1995
- James Gosling
- Duke



De ce Java ?

The Financial Express published some numbers on Java and how it stands today after 10 years of growth. Quoting from the article, here they are:

- Globally, over 4.5 million software developers work on Java.
- Java is a \$100 billion dollar per year industry.
- \$2.2 billion is invested yearly in Java application servers and 110 \$billion in related IT.
- There are 579 million Java enabled phones.
- Seven out of 10 wireless applications currently under constructions will use Java technology runtime environment.
- The Java mobile game market is estimated at around \$3 billion.
- Around 750 million Java cards have been deployed globally.

Limbaajul de programare Java

- **Simplitate**
- **Ușurință** în crearea de aplicații complexe
- **Robustețe** - \nexists pointeri, administrarea automată a memoriei, GC
- **Complet orientat pe obiecte**
- **Securitate**
- **Neutralitate arhitecturală**
- **Portabilitate**
- **Compilat și interpretat**
- **Performanță**
- **Modelat după C și C++**

Platforme de lucru Java

- **Java SE (Standard Edition)**
Aplicații independente, appleturi, Java Web Start
- **Java ME (Micro Edition)**
Programarea dispozitivelor mobile
- **Java EE (Enterprise Edition)**
Aplicații complexe, pe mai multe nivele pentru sisteme eterogene, cu baze de date distribuite, etc.

Aplicații și servicii Web: servle-turi, pagini JSP, etc.

Distribuțiile Java sunt oferite **gratuit**
<http://java.sun.com>

Compilat și interpretat

Limbajele de programare:

- **Interpretate**

- + : simplitate, portabilitate

- : viteza de execuție redusă

- **Compilate**

- + : execuția extrem de rapidă

- : lipsa portabilității

Java: compilat + interpretat

Compiler: surse - cod de octeți

Interpreter: execută codul de octeți

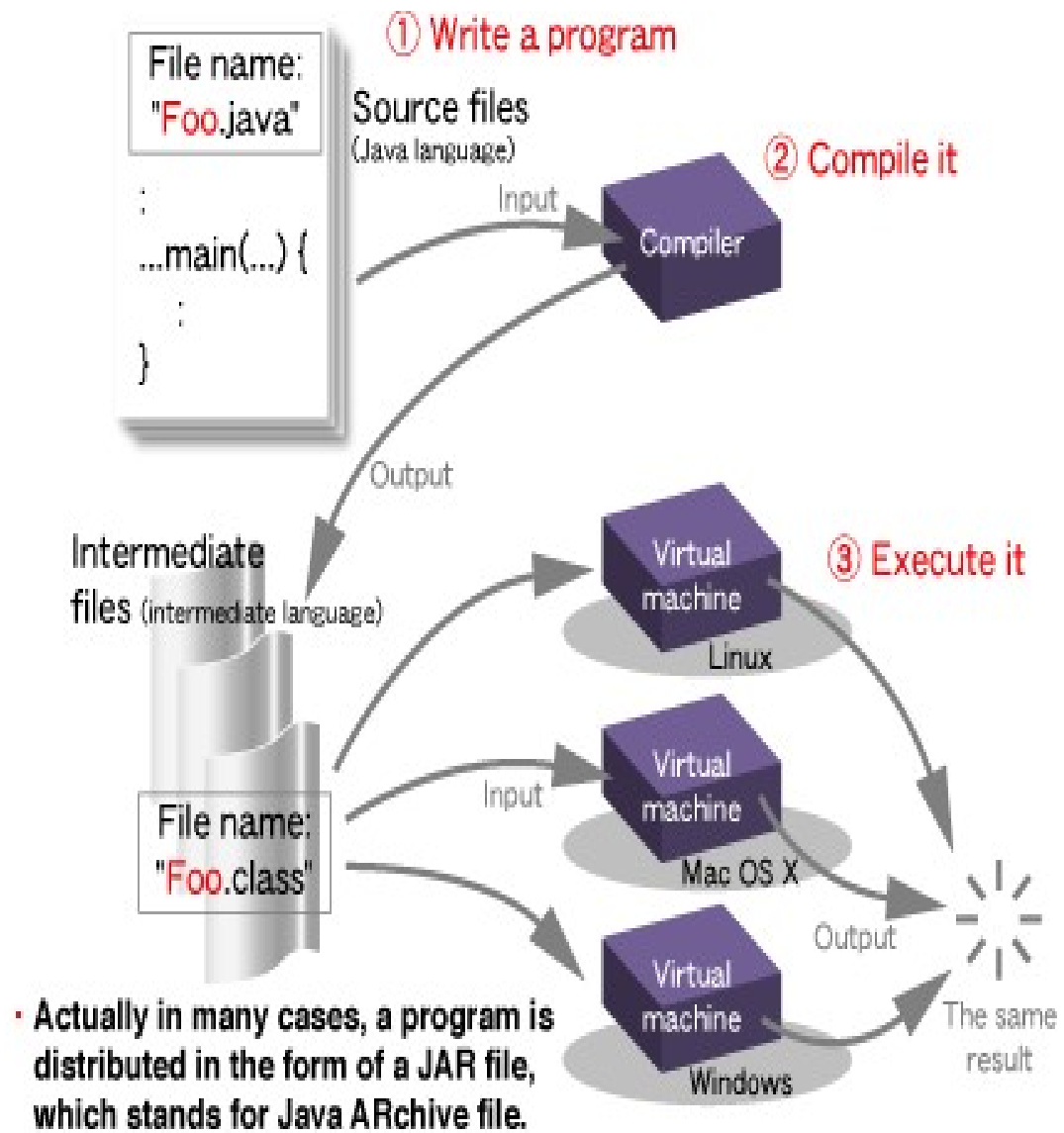
Cod octeți \neq Cod mașină

Cod mașină - procesor

Cod octeți - JVM

JVM = Java Virtual Machine

Imagine de ansamblu



Primul program

1. Scriererea codului sursă

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

2. Salvarea fișierelor sursă

C:\intro\FirstApp.java

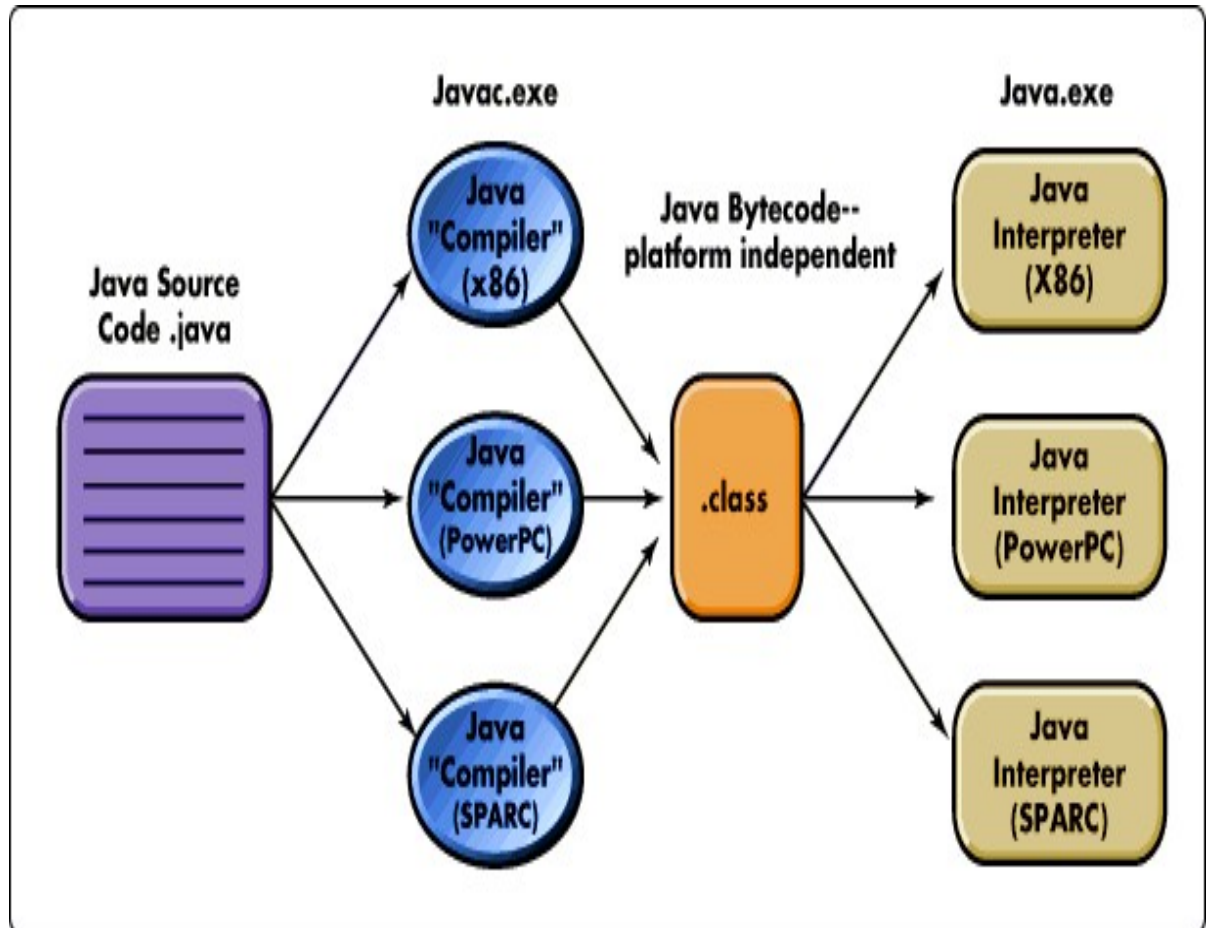
3. Compilarea aplicației

```
javac FirstApp.java  
va rezulta: FirstApp.class
```

4. Rularea aplicației

```
java FirstApp
```

Javac, Java



Dezasamblarea codului

javap -c FirstApp

Compiled from "FirstApp.java"

```
class FirstApp extends java.lang.Object{
FirstApp();
```

Code:

```
0: aload_0
1: invokespecial #1;
   //Method java/lang/Object."<init>":()V
4: return
```

```
public static void main(java.lang.String[]);
```

Code:

```
0: getstatic #2;
   //Field java/lang/System.out:Ljava/io/PrintStream;
3: ldc #3;
   //String Hello world!
5: invokevirtual #4;
   //Method java/io/PrintStream.println:(Ljava/lang/St
8: return
```

```
}
```

Obfuscare

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.

`\uxxxx`

`\u03B1` - `\u03C9`: α - ω

<http://www.unicode.org>

Cuvinte cheie

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

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

Incepând cu 1.5: **enum**.

Literali

- **Intregi** (10, 16 -0x, 8-0)
- **Flotanți**: 1.0, 2e2, 3f, 4D
- **Logici**: true, false
- **Character**: 'J', 'a', 'v', 'a', '\n'
- **Șiruri de caractere**: "Duke"

Separatori

() [] ; , .

Operatori

- atribuirea: =
- matematici:
+, -, *, /, %, ++, --
lval op= rval: x += 2 n -= 3
x++, ++x, n--, --n
- logici: &&(and), ||(or), !(not)
- relaționali: <, <=, >, >=, ==, !=
- pe biți: &(and), |(or), ^ (xor),
~ (not)
- de translație: <<, >>, >>> (shift
la dreapta fără semn)
- *if-else*: expresie-logica ? val-true
: val-false

- operatorul `,` (virgulă) folosit pentru evaluarea secvențială a operațiilor:

```
int x=0, y=1, z=2;
```

- operatorul `+` pentru concatenarea șirurilor

```
String s1="Ana";  
String s2="mere";  
int x=10;  
System.out.println(s1 + " are " + x + " " + s2);
```

- operatori pentru conversii
(*cast*) : (*tip-de-data*)

```
int a = (int)'a';  
char c = (char)96;  
int i = 200;  
long l = (long)i; //widening conversion  
long l2 = (long)200;  
int i2 = (int)l2; //narrowing conversion
```

Comentarii

Există trei feluri de comentarii:

- Comentarii pe mai multe linii, închise între `/* și */`.
- Comentarii pe mai multe linii care țin de documentație, închise între `/** și */`.
Textul dintre cele două secvențe este automat mutat în documentația aplicației de către generatorul automat de documentație **javadoc**.
- Comentarii pe o singură linie, care încep cu `//`.

Tipuri de date

Tipuri primitive - Tipuri referință.

Tipurile primitive sunt:

- **aritmetice**
 - întregi: `byte` (1 octet), `short` (2), `int` (4), `long` (8)
 - reale: `float` (4), `double` (8)
- **caracter**: `char` (2)
- **logic**: `boolean` (`true` și `false`)

Vectorii, clasele și interfețele sunt tipuri referință.

Nu există: `pointer`, `struct` și `union`.

Variabile

- Declararea variabilelor:
`Tip numeVariabila;`
- Inițializarea variabilelor:
`Tip numeVariabila = valoare;`
- Declararea constantelor:
`final Tip numeVariabila;`

```
final double PI = 3.14;  
int valoare = 100;  
long numarElemente = 12345678L;  
String bauturaMeaPreferata = "apa";
```

- a. Variabile membre
- b. Parametri metodelor
- c. Variabile locale, declarate într-o metodă
- d. Variabile locale, declarate într-un bloc
- e. Parametrii de la tratarea excepțiilor

```
class Exemplu {
    int a;
    public void metoda(int b) {
        a = b;
        int c = 10;
        for(int d=0; d < 10; d++) {
            c --;
        }
        try {
            a = b/c;
        } catch(ArithmeticException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

Controlul execuției

- 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*:

Instrucțiuni de decizie

if-else

```
if (expresie-logica) {  
    ...  
}  
if (expresie-logica) {  
    ...  
} else {  
    ...  
}
```

switch-case

```
switch (variabila) {  
    case valoare1:  
        ...  
        break;  
    case valoare2:  
        ...  
        break;  
    default:  
        ...  
}
```

Instrucțiuni de salt

for

```
for(initializare; expresie-logica; pas-iteratie) {  
    //Corpul buclei  
}
```

```
for(int i=0, j=100 ; i < 100 && j > 0; i++, j--) {  
    ...  
}
```

while

```
while (expresie-logica) {  
    ...  
}
```

do-while

```
do {  
    ...  
}  
while (expresie-logica);
```

Alte instrucțiuni

- break
- continue
- return [valoare]
- *numeEticheta*:

Nu există goto

Pot fi definite etichete folosite astfel:
break numeEticheta
continue numeEticheta

Exemplu de folosire a etichetelor

```
i=0;
eticheta:
while (i < 10) {
    System.out.println("i="+i);
    j=0;
    while (j < 10) {
        j++;
        if (j==5) continue eticheta;
        if (j==7) break eticheta;
        System.out.println("j="+j);
    }
    i++;
}
```

Vectori

- **Declararea**

```
Tip[] numeVector; sau  
Tip numeVector[];
```

- **Instanțierea**

```
numeVector = new Tip[nrElemente];
```

- **Inițializarea (opțional)**

```
String culori[] = {"Rosu", "Galben"};
```

- **Declarare și instanțiere**

```
Tip[] numeVector = new Tip[nrElemente];
```

- **Instanțiere și inițializare**

```
metoda( new String[]{"Rosu", "Galben"} );
```

- **Exemplu**

```
int[] v = new int[10];  
//aloca spatiu pentru 10 intregi: 40 octeti  
char c[] = new char[10];  
//aloca spatiu pentru 10 caractere: 20 octeti
```

Tablouri multidimensionale

Tablou multidimensionala = vector de vectori.

```
Tip matrice[] [] = new Tip[nrLinii][nrColoane];
```

`matrice[i]` este linia i a matricii și reprezintă un vector cu *nrColoane* elemente iar `matrice[i][j]` este elementul de pe linia i și coloana j .

Dimensiunea unui vector

Variabila **length**:

```
int []a = new int[5];  
// a.length are valoarea 5
```

```
int m[] [] = new int[5][10];  
// m[0].length are valoarea 10
```

Copierea vectorilor

```
int a[] = {1, 2, 3, 4};
int b[] = new int[4];

// Varianta 1
for(int i=0; i<a.length; i++)
    b[i] = a[i];

// Varianta 2
System.arraycopy(a, 0, b, 0, a.length);

// Nu are efectul dorit
b = a;
```

Sortarea vectorilor - clasa Arrays

Metode din `java.util.Arrays`:

- **sort** - (*QuickSort* - $O(n \log(n))$)

```
int v[]={3, 1, 4, 2};
java.util.Arrays.sort(v);
// Sorteaza vectorul v
// Acesta va deveni {1, 2, 3, 4}
```

- **binarySearch**
- **equals**
- **fill**

Vectori cu dimensiune variabilă și eterogeni: Vector, ArrayList, etc. din pachetul `java.util`.

Șiruri de caractere

- `char[]`
- `String`
- `StringBuilder`
- `StringBuffer`

```
String s = "abc";  
String s = new String("abc");  
char data[] = {'a', 'b', 'c'};  
String s = new String(data);
```

Concatenarea: `+`

```
String s1 = "ab" + "cd";  
String s2 = s1 + 123 + "xyz"
```

Testarea egalității: `equals`

```
if (nume.equals("duke")) { ... }
```

Argumente de la linia de comandă

Trimiterea argumentelor

```
java NumeAplicatie [arg0 arg1 . . .]
java Test Java "Hello Duke" 1.5
```

Primirea argumentelor

```
public static void main (String args[])
/* args[0] va fi primul argument trimis:
   "Java"
   args[1] va fi urmatorul argument:
   "Hello Duke"
   ...
*/
```

Numărul argumentelor

```
public static void main (String args[]) {
    int numarArgumente = args.length ;
}
```

Exemplu

```
public class Salut {
    public static void main (String args[]) {
        if (args.length == 0) {
            System.out.println(
                "Numar insuficient de argumente!");
            System.exit(-1); //termina aplicatia
        }
        String nume = args[0]; //exista sigur
        String prenume;
        if (args.length >= 1)
            prenume = args[1];
        else
            prenume = ""; //valoare implicita
        System.out.println("Salut " + nume + " " + prenume);
    }
}
```

Afişarea argumentelor

```
public class Afisare {  
    public static void main (String[] args) {  
        for (int i = 0; i < args.length; i++)  
            System.out.println(args[i]);  
    }  
}
```

java Afisare Hello Java

Hello
Java

java Afisare "Hello Java"

Hello Java

Argumente numerice

Argumentele de la linia de comandă sunt **șiruri de caractere**

Transformarea șir - număr se face cu metode de tipul:

- `Integer.parseInt`
- `Double.parseDouble`
- `Float.parseFloat`
- ...

```
public class Power {  
    public static void main(String args[]) {  
        double numar = Double.parseDouble(args[0]);  
        int putere = Integer.parseInt(args[1]);  
        System.out.println("Rezultat=" +  
            Math.pow(numar, putere));  
    }  
}
```

Bibliografie

- *The Java Language Specification*, James Gosling, Bill Joy, Guy Steele, Gilad Bracha
- *The Java Virtual Machine Specification*, Tim Lindholm, Frank Yellin
- *Thinking in Java*, Bruce Eckel
- `java.sun.com`
- `google:java`

- *Curs practic de Java*, C. Frasinaru
- `www.infoiasi.ro/~acf/java`
- `acf.info.uaic.ro/acf`

Adrese utile

- **www.java.com**
Toată lumea
- **www.java.net**
Comunitate
- **java.sun.com**
Dezvoltatori
- **kguru.com**
Cursuri
- **jars.com**
Resurse
- **javaworld.com, javareport.com**
Articole
- **javaalmanac.com**
Exemple de cod