PROGRAMARE AVANSATĂ PE OBIECTE

Conf.univ.dr. Radu BORIGA

Conținut tematic

- Introducere în limbajul Java
- Clase și obiecte. Extinderea claselor. Polimorfism
- Tablouri. Șiruri de caractere
- Clase abstracte. Interfețe
- Excepții
- Fluxuri de intrare/ieșire
- Colecții de date
- Lambda expresii
- Fire de executare
- Socket-uri
- Interfețe grafice
- Lucrul cu baze de date
- Servlet-uri. Java Server Pages (JSP)
- RESTful Web Services

Bibliografie

- **Joshua Bloch, *Effective Java* (3rd edition), Addison-Wesley Professional, 2018**
- **Raul Gabriel Urma, *Modern Java in action*, 2018**
- **Raul Gabriel Urma, *Java 8 in action*, 2014**
- **Bruce Eckel, *Thinking in Java*, 2012**
- **Ștefan Tanasă, Cristian Olaru, Ștefan Andrei, *Java de la 0 la expert, Ed. Polirom*, 2011**

- **Tutoriale:**
  - https://docs.oracle.com/javase/tutorial/index.html
  - http://www.tutorialspoint.com/java/

Notare

- 50% → Laborator (minim nota 5)
- 50% → Examen (minim nota 5)

Tematica cursului 1

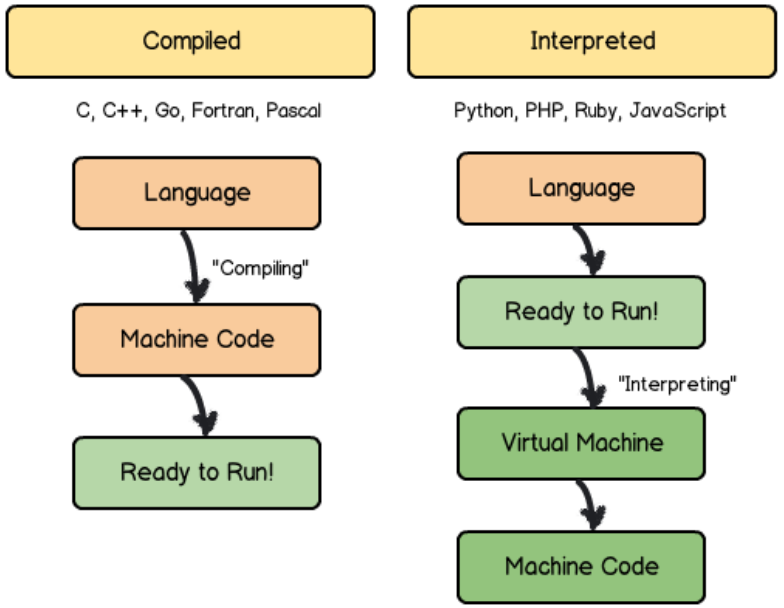
- Prezentarea generală a platformei Java
- Structura unui program
- Tipuri de date și operatori
- Literali
- Instrucţiuni
- Pachete de clase
- Operații de citire/scriere

Scurt istoric al limbajului Java

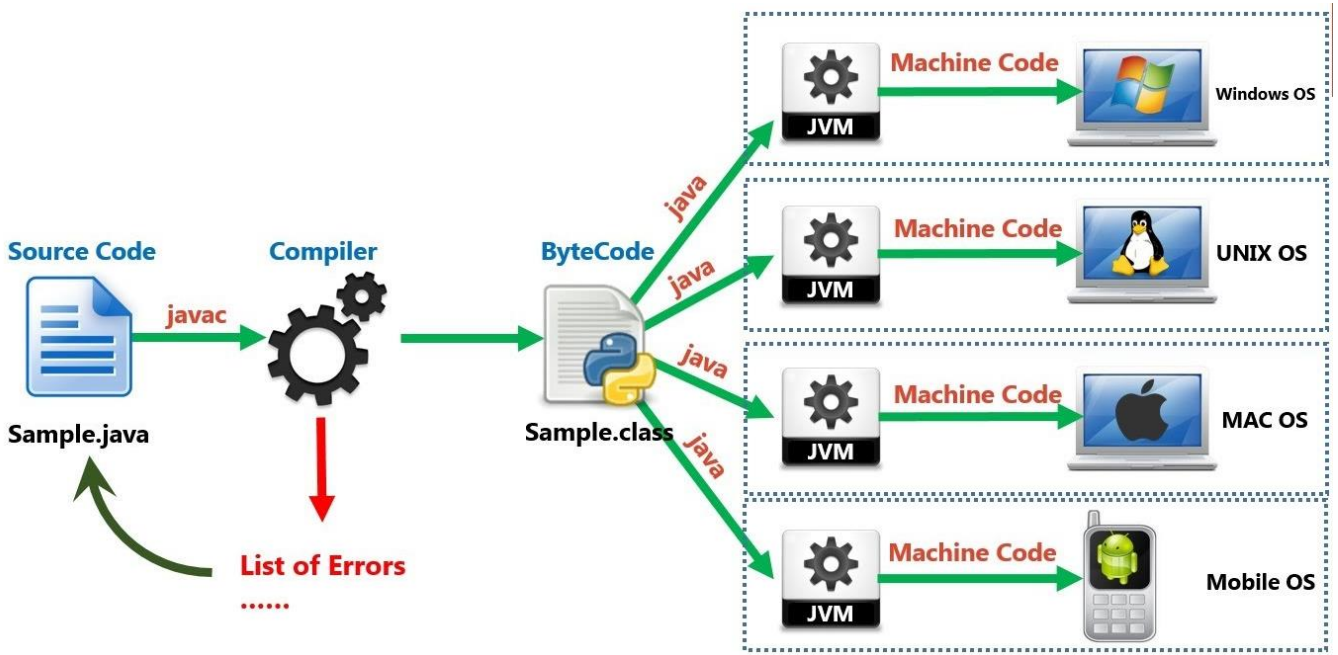
- În anul 1991 firma Sun Microsystems finanțează proiectul Green, coordonat de James Gosling.
- Specificațiile noului limbaj, inițial denumit OAK, iar apoi Java 1.0 sunt finalizate în anul 1995.
- În anul 1995 compania Sun Microsystems vinde licența firmelor IBM, Microsoft, Adobe și Netscape.
- În 2009 Sun Microsystems este cumpărată de Oracle.
- Java 1.1 (1997): JDBC și JIT
- .....
- Java 8 (2014): lambda expresii și programare funcțională
- .....
- Java 19 (2022): fire de executare virtuale, programare concurentă structurată etc.

Caracteristici

- Limbaj compilat și interpretat



- Write Once, Run Anywhere



Caracteristici

- Bytecode-ul reprezintă un set de instrucțiuni specifice JVM

C++/Java	X86 ASM	Java bytecode (mnemonics)	Java bytecode (hexadecimal)

int add(int a, int b)
{
return a+b;
}
	mov eax, byte[ebp-4]	iload_0	0x1A

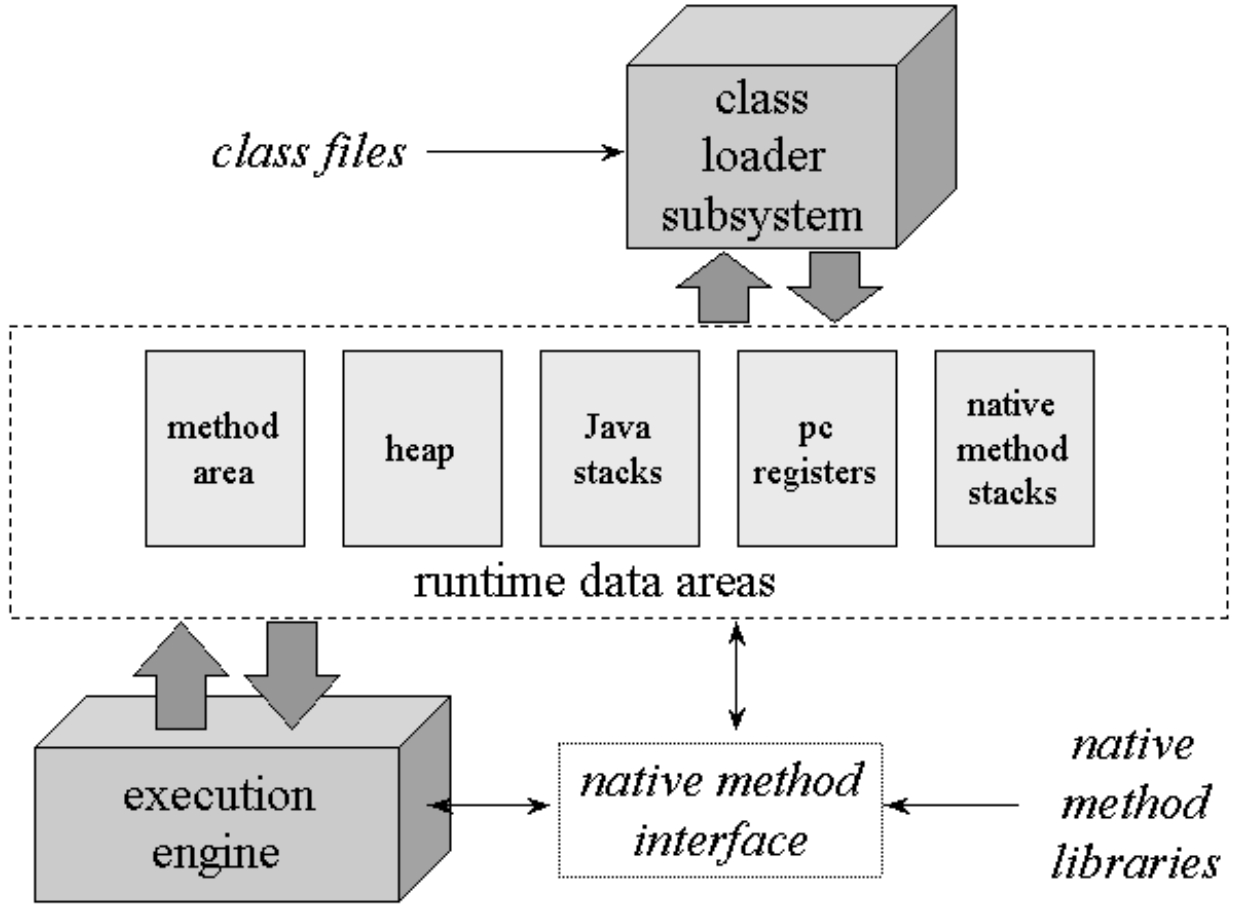
	mov edx, byte[ebp-8]	iload_1	0x1B
	add eax, edx	iadd	0x60
	ret	ireturn	0xAC

Caracteristici

➢ Java Virtual Machine (JVM)

- **Class loader:** program care încarcă în memorie bytecode-ul unei aplicații Java

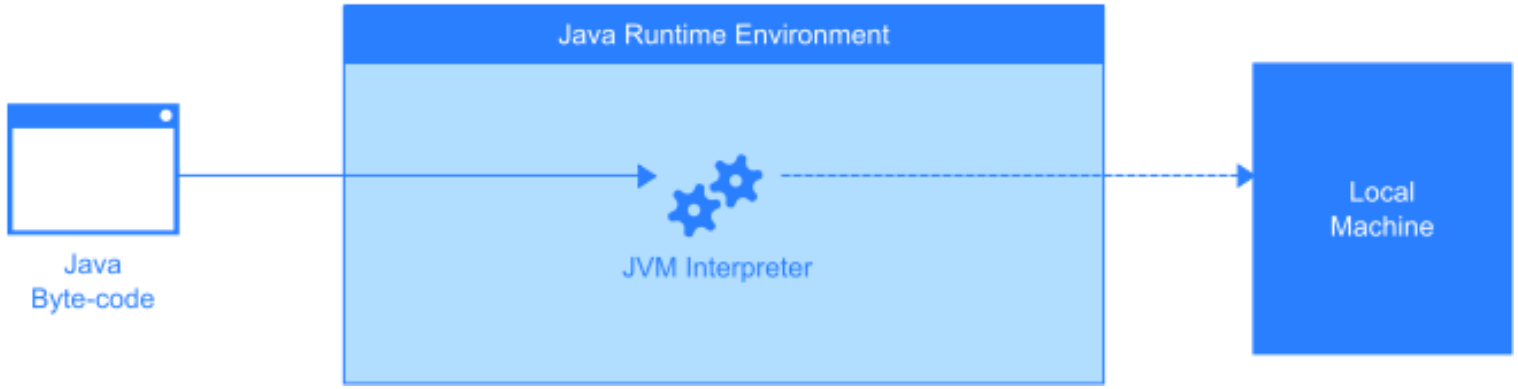
- **Execution engine:** execută instrucțiunile din bytecode-ul încărcat



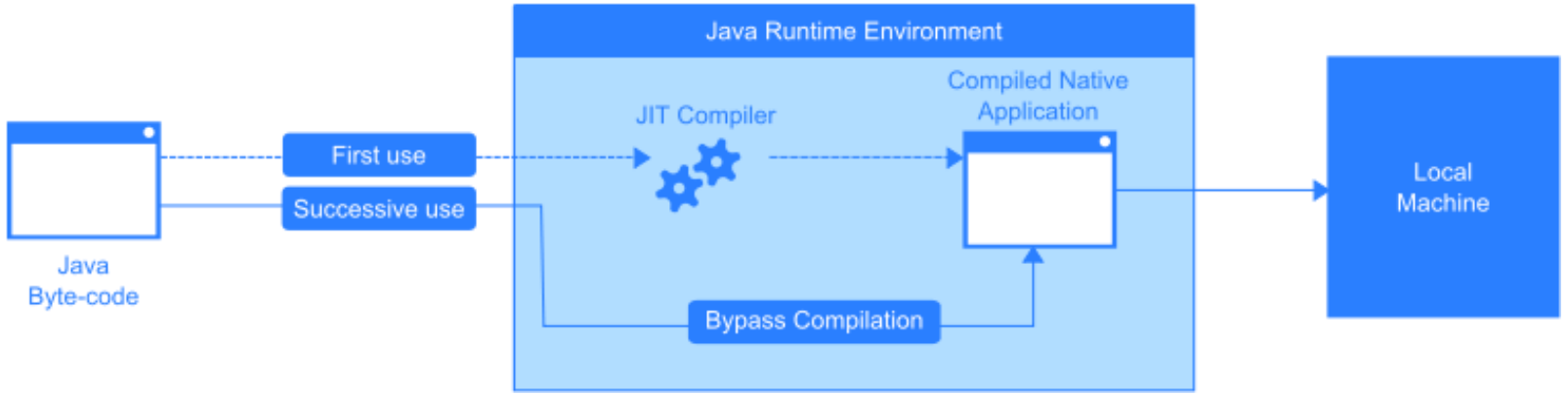
Caracteristici

Execution engine

- **Interpretator**: interpretează și execută bytecode-ul



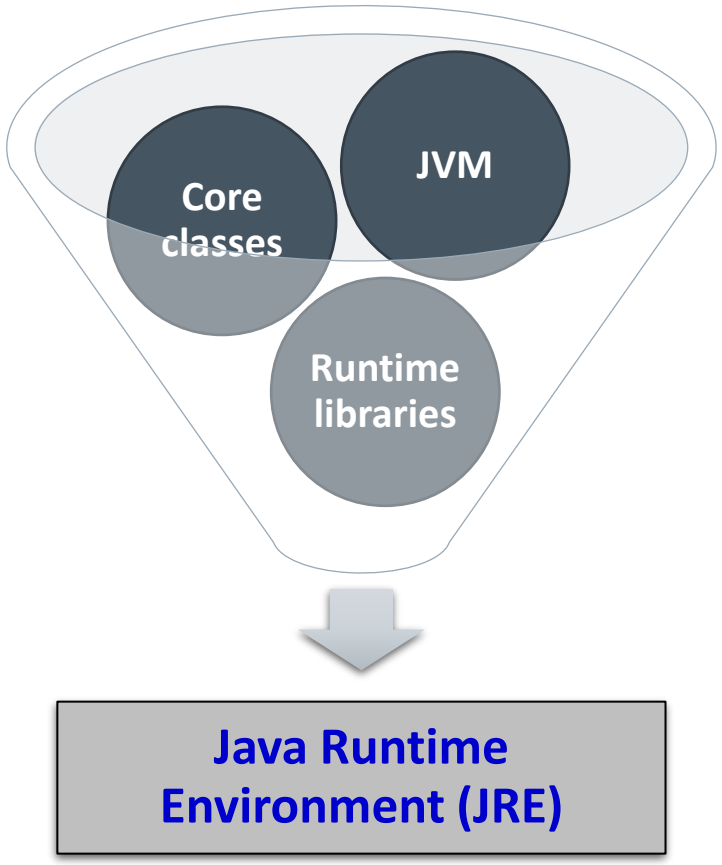
- **Compilator Just-in-Time (JIT):** transformă bytecode-ul care se execută frecvent în cod mașină nativ, specific procesorului gazdă

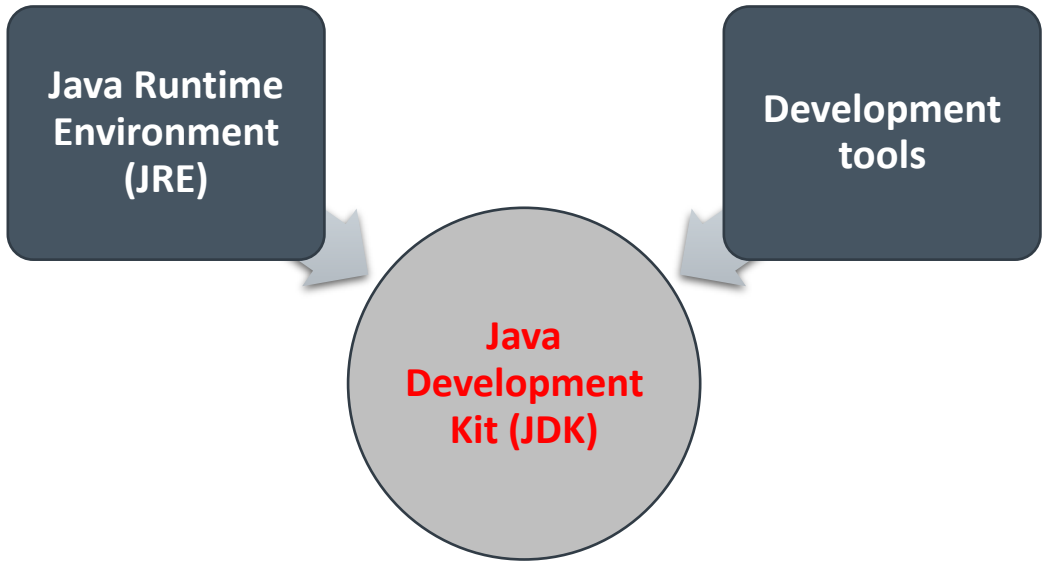


Caracteristici

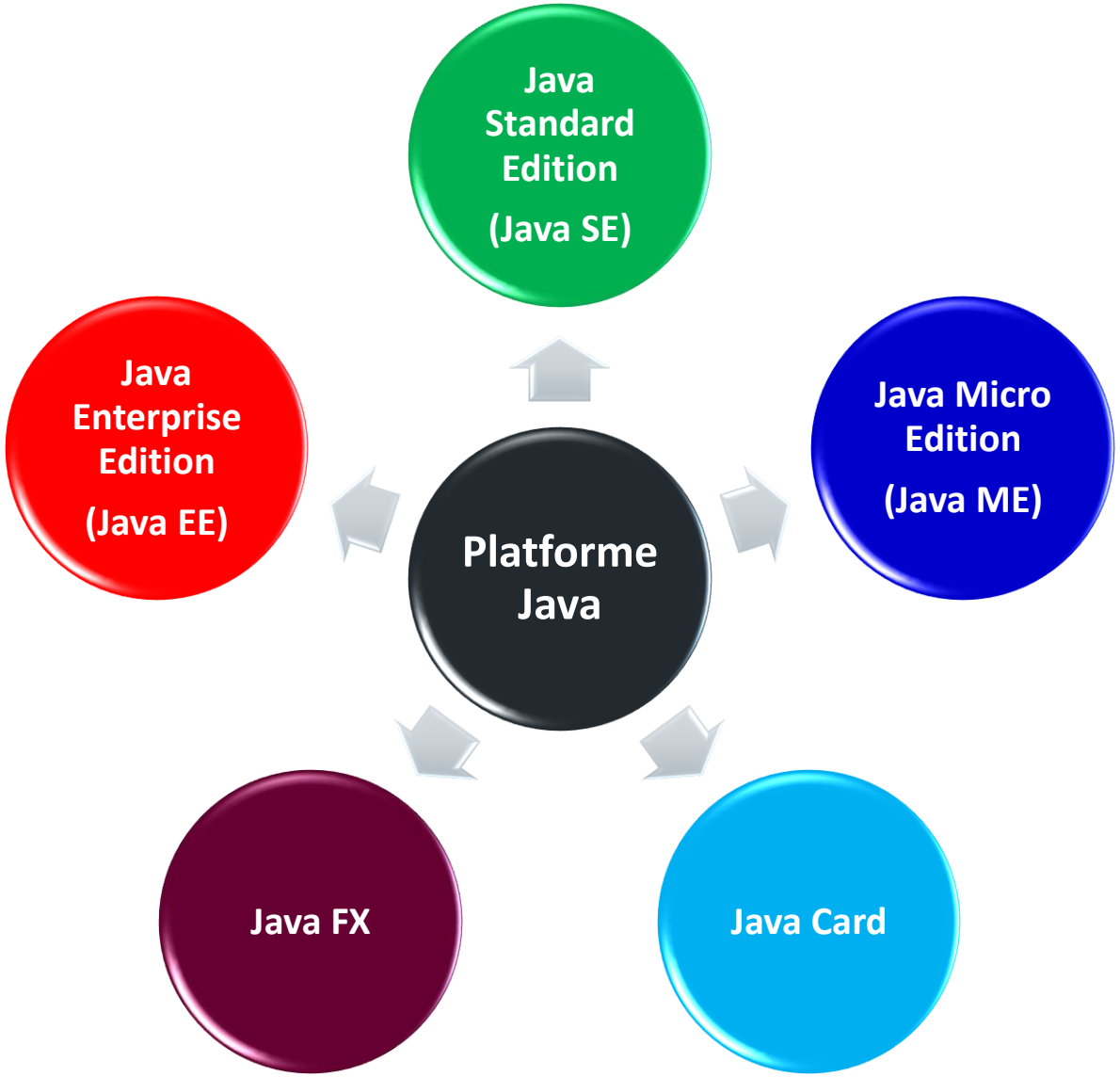
Limbaj orientat pe obiecte	• Orice program conține cel puțin o clasă • Nu mai există funcții independente
Simplu	• Au fost eliminate concepte precum: pointeri, supraîncărcarea operatorilor, moștenirea multiplă, structuri/uniuni etc.
Robust	• Management automat al memoriei • Strong data-typed • Mecanism standard de tratare a exceptiilor
Sigur	• Securitatea platformei • Sandbox
Permite programarea concurentă și distribuită	• Suport nativ pentru fire de executare (thread-safe) • Biblioteci care conțin primitive specifice

Software necesar





Platforme Java



Setul de caractere

- Setul de caractere: Unicode (65536 simboluri)
- Un caracter se reprezintă pe 2 octeţi.
- Unicode este compatibil cu 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

Literali

- **Întregi**
  - pot fi scrișI în baza 10, baza 2 (prefix **0b** sau **0B**), baza 8 (prefix **0**) sau baza 16 (prefix **0x** sau **0X**)
  - implicit se reprezintă pe 4 octeţi (**`int`**)
  - pentru **`long`** (8 octeţi) se adaugă sufixul **L** sau **l**
- **Numere cu virgulă mobilă**
  - implicit se reprezintă pe 8 octeţi (**`double`**)
  - pentru **`float`** se adaugă sufixul **F** sau **f**
- **Boolean:** **`true`** sau **`false`**
- **Caractere:** **`'A'`**, număr întreg, **`'\u0041'`** (format unicode baza 16)
- **Șiruri de caractere:** **`"Test"`**
- **`null`**

Cuvinte cheie

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

Keywords that are not currently used

const	goto

TIPURI DE DATE

1. **Tipurile primitive:**

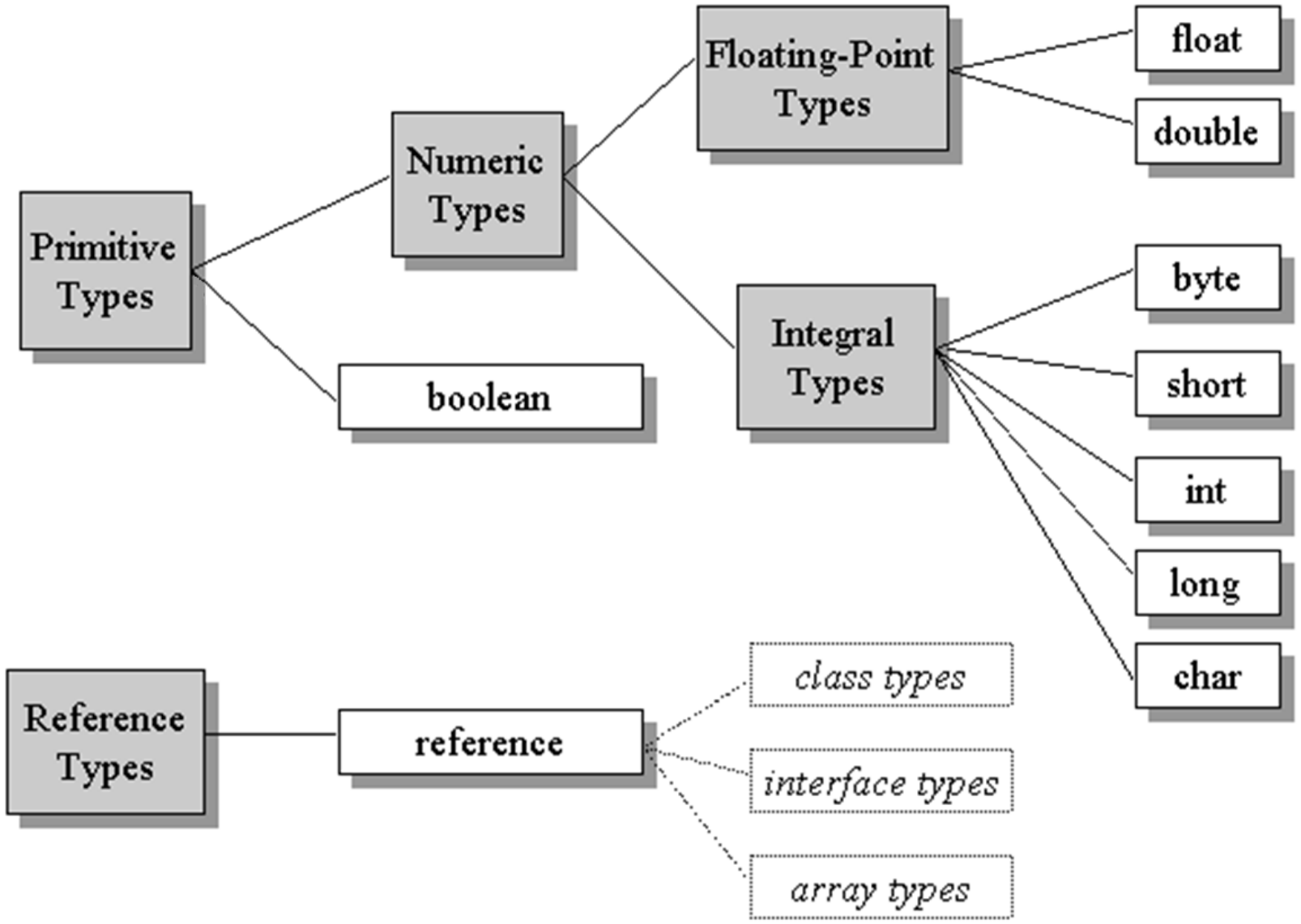
- sunt alocate în zona de memorie de tip stivă
- tipuri numerice (număr întreg, număr “real”), tipul boolean

2. Tipuri de date de referință

- sunt obiecte alocate dinamic prin operatorul `new` in zona de memorie HEAP
- tablouri, clase şi interfeţe

3. Tipul `null`

Tipuri de date



Tipuri de date primitive

- Fiecare tip de dată primitiv are asociată o clasă corespunzătoare (wrapper), care permite transformarea unei variabile de tip primitiv într-un obiect.

Primitive Types					
Type Name	Wrapper class	Value	Range	Size	Default Value
byte	java.lang.Byte	integer	−128 through +127	8-bit (1-byte)	0
short	java.lang.Short	integer	−32,768 through +32,767	16-bit (2-byte)	0
int	java.lang.Integer	integer	−2,147,483,648 through +2,147,483,647	32-bit (4-byte)	0
long	java.lang.Long	integer	−9,223,372,036,854,775,808 through +9,223,372,036,854,775,807	64-bit (8-byte)	0
float	java.lang.Float	floating point number	±1.401298E−45 through ±3.402823E+38	32-bit (4-byte)	0.0
double	java.lang.Double	floating point number	±4.94065645841246E−324 through ±1.79769313486232E+308	64-bit (8-byte)	0.0
boolean	java.lang.Boolean	Boolean	true or false	8-bit (1-byte)	false
char	java.lang.Character	UTF-16 code unit (BMP character or a part of a surrogate pair)	'\u0000' through '\uFFFF'	16-bit (2-byte)	'\u0000'

Clase înfășurătoare

Câmpuri statice

- MIN_VALUE
- MAX_VALUE
- SIZE (biți)
- BYTES (octeți)

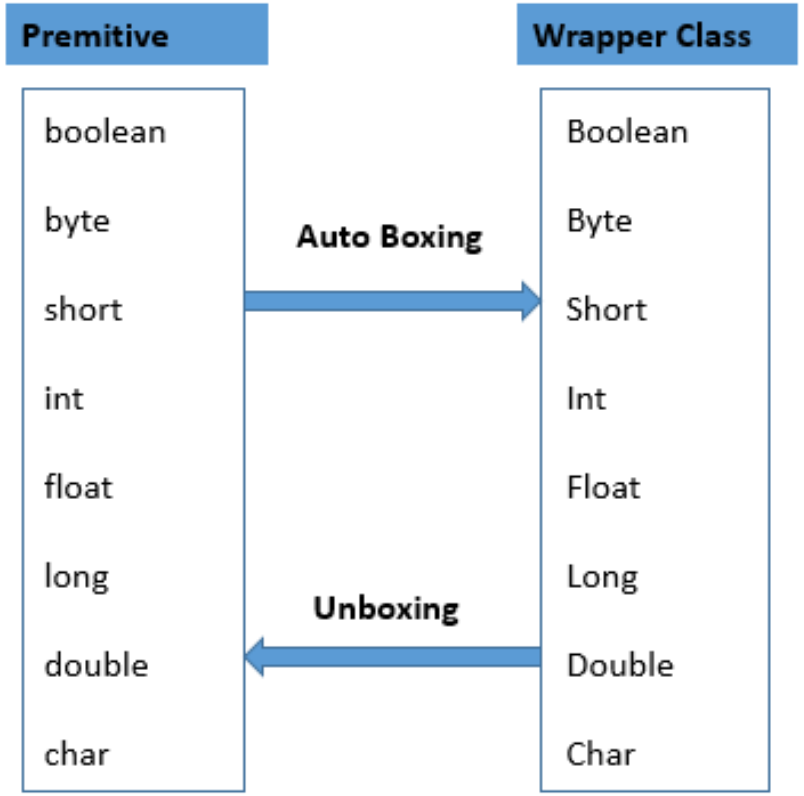
Constructori

- Cu argument de tip primitiv
- Cu argument de tip String

Metode

- tipValue()
- compare(tip x,tip y)
- compareTo(Tip ob)
- parseTip(String s)
- toString()
- toString(tip x)
- valueOf(tip x)
- valueOf(String s)

Autoboxing/unboxing




```
class Main {
    public static void main(String [] args) {
        // Boxing
        Integer a = 2;

        // UnBoxing
        int s = 5 + a;
    }
}
```

Declararea variabilelor

- La declararea variabilelor de tip primitiv nu se atribuie implicit o valoare reziduală!

- Variabilele locale trebuie să fie inițializate înainte de a fi utilizate, iar datele membre unei clase sunt inițializate implicit cu valori nule de tip!

Operatori

Precedence	Operator	Operand type	Description
1	++,	Arithmetic	Increment and decrement
1	+, -	Arithmetic	Unary plus and minus
1	~	Integral	Bitwise complement
1	!	Boolean	Logical complement
1	(*type*)	Any	Cast
2	*, /, %	Arithmetic	Multiplication, division, remainder
3	+, -	Arithmetic	Addition and subtraction
3	+	String	String concatenation
4	<<	Integral	Left shift
4	>>	Integral	Right shift with sign extension
4	>>>	Integral	Right shift with no extension
5	<, <=, >, >=	Arithmetic	Numeric comparison
5	instanceof	Object	Type comparison
6	==, !=	Primitive	Equality and inequality of value
6	==, !=	Object	Equality and inequality of reference
7	&	Integral	Bitwise AND
7	&	Boolean	Boolean AND
8	^	Integral	Bitwise XOR
8	^	Boolean	Boolean XOR
9	|	Integral	Bitwise OR
9	|	Boolean	Boolean OR
10	&&	Boolean	Conditional AND
11	||	Boolean	Conditional OR
12	?:	N/A	Conditional ternary operator
13	=	Any	Assignment

Operatori

- Operatorii * și & nu au semnificații specifice pointerilor!

- Nu există operatorul `sizeof`, dimensiunea în octeți pe care se reprezintă tipurile de date primitive se poate determina prin apelul metodei `BYTES` încapsulată în clasele wrapper.

- Operatorul binar `instanceof` testează dacă un obiect este o instanță a unei clase:

```
bool numeObiect instanceof NumeClasa
```

- Operatorii logici && (and), respectiv || (or) sunt furnizați și în varianta cu testare fără scurtcircuitare, respectiv & și |.

Instrucțiuni

Decision-making	Looping	Branching
• if-then • if-then-else • switch	• while • do-while • for	• break • continue • return

Observații

- Pentru instrucțiunea `for` există, pe lângă forma clasică (`loop for`), și forma `enhanced for`:

```
int[] numere = {1,2,3,4,5,6,7,8,9,10};
for (int x : numere)
     System.out.println(x);
```

Aplicații stand alone

- 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 încapsulata funcția principală `main` cu sintaxa :

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

Etapele realizării unei aplicații Java

1. Scrierea codului sursă

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

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

2. Salvarea fişierelor sursă

- se va face în fișiere care au obligatoriu extensia **.java**
- denumirea fișierului este aceeași cu cea a clasei publice

3. Compilarea aplicaţiei

- pentru compilare se utilizează compilatorul **javac**:

```
javac Test.java
```

4. Rularea aplicaţiei

- se realizează folosind interpretorul **java:**

```
java Test
```

Pachete de clase

- **Pachet** = colecţie de clase şi interfeţe salvate în același director.

- **Utilitatea pachetelor:**
  - organizarea fișierelor sursă
  - găsirea şi utilizarea mai uşoară a claselor
  - evitarea conflictelor de nume
  - controlul accesului

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

Utilizarea pachetelor de clase

1. Specificarea numelui complet

`java.numePachet.NumeClasa` // pentru pachete standard

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

Exemplu:

- **Scanner** - numele clasei care permite citirea formatată
- **java.util** - pachetul din care face parte
- **java.util.Scanner** - numele complet al clasei

```
java.util.Scanner in=new java.util.Scanner(System.in);
```

Utilizarea pachetelor de clase

2. Importul unei clase dintr-un pachet

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

Exemplu:

```
import java.util.Scanner;
Scanner in=new Scanner(System.in);
```

➢Se specifică clauza import pentru fiecare clasa importată!

```
import java.util.Scanner;
import java.util.Collections;
```

Utilizarea pachetelor de clase

3. Importul întregului pachet

```
import numePachet.*;
```

Exemplu:

```
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!!!

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