Java

Curs 1

Danciu Gabriel Mihail

Septembrie 2018

Cuprins

Ce este Java?

- Inspirat din C++ cu o sintaxa similară
- Creat de o echipă la Sun Microsystems în 1991. Denumit iniţial *Oak*, redenumit *Java* în 1995
- Creat pentru a fi portabil: independent de platformă (Intel, Macintosh, sau UNIX)
- Orientat pe obiecte

Cum a influenţat Java Interneul?

- Applet
- Securitate
- Portabilitate

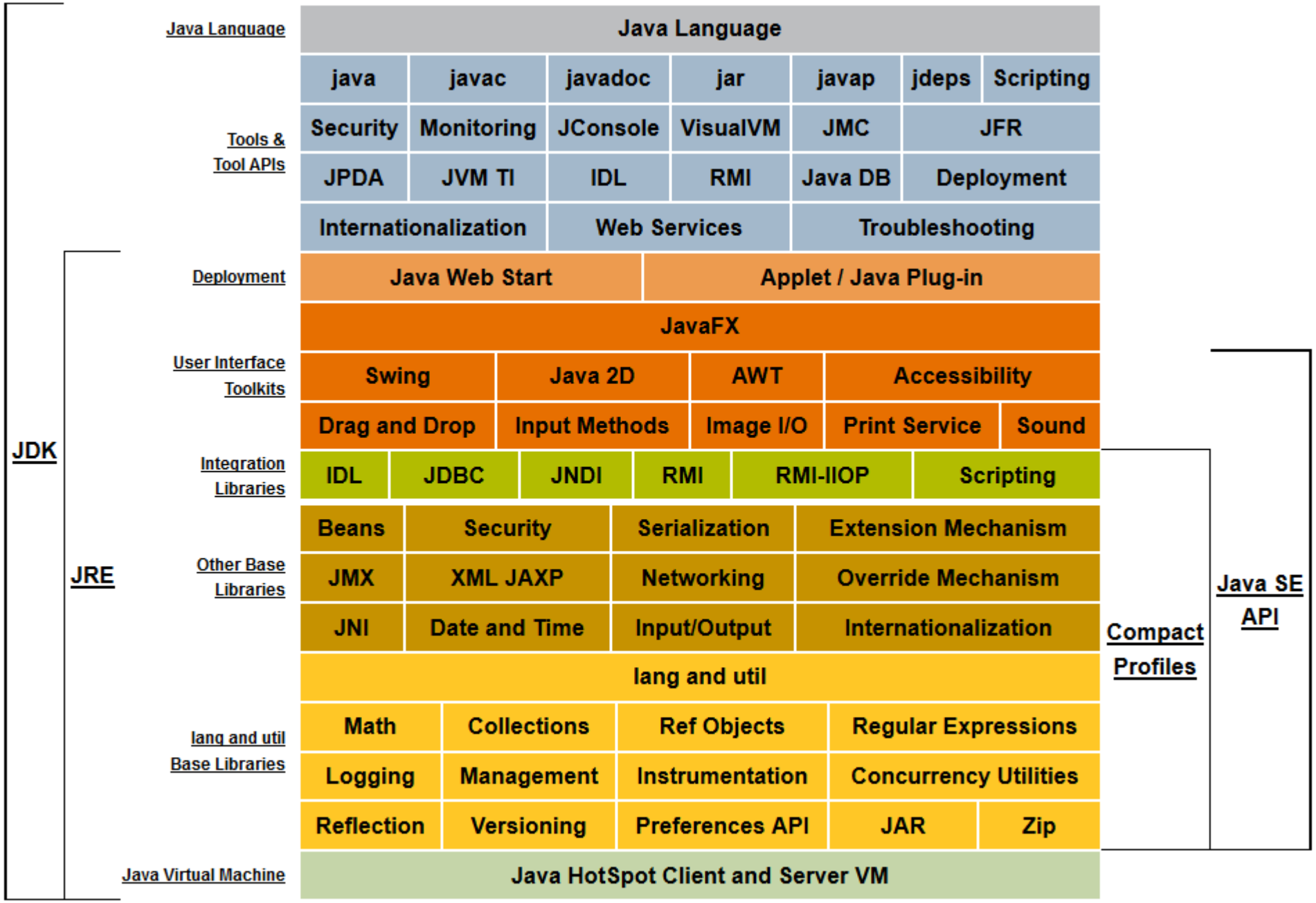
Ce este deosebit la Java?

- ByteCode
- Java Virtual Machine
- Just in Time
- Servlets

- Simplu
- Sigur
- Portabil
- Robust
- Suport de multithread
- Distribuit
- Dinamic

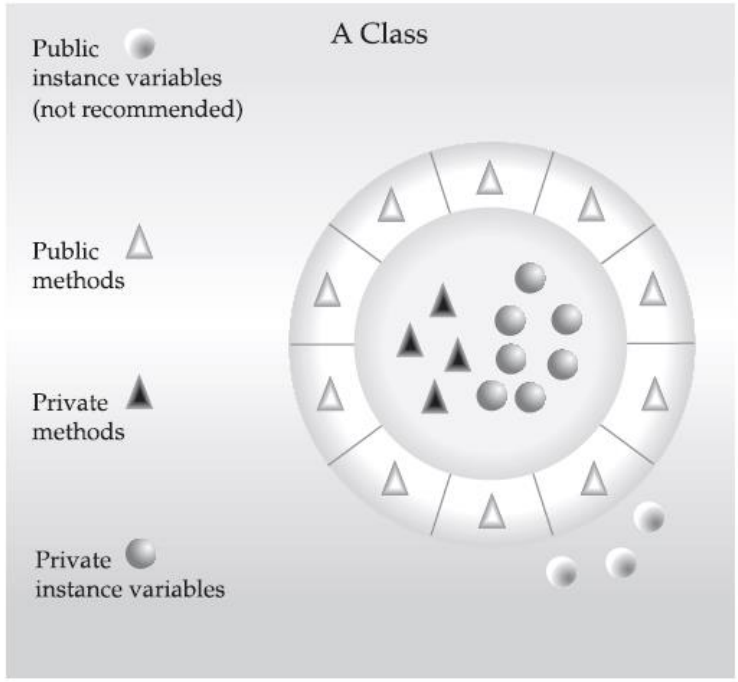
Evoluţia Java

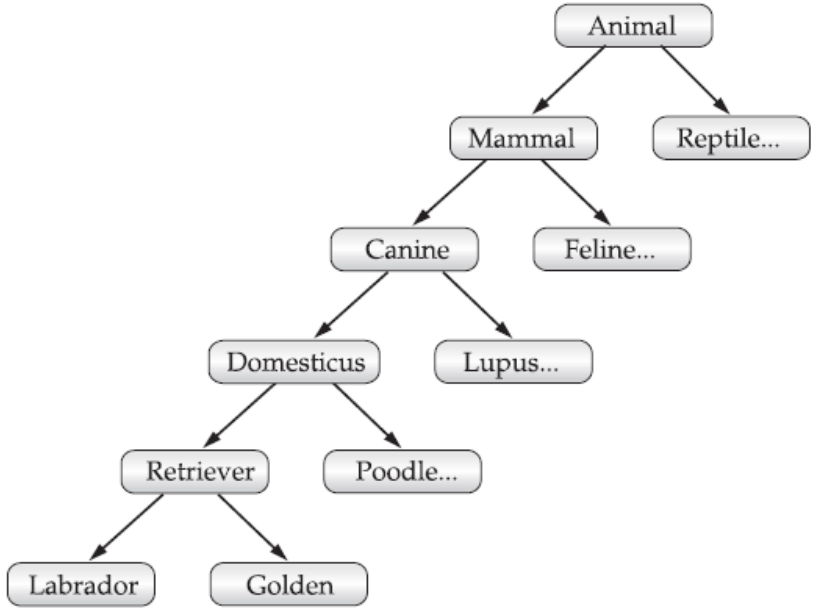
- Java 1.0
- Java 1.1 – componente noi + învechirea unora din 1.0
- Java 2 Platform S.E. – Swing, Collections
- J2SE 1.4 – assert, excepţii înlănţuite, I/O bazat pe canale
- J2SE 1.5 – tipuri generice, adnotări, autoboxing, varargs, concurrency, import static, I/O formatat, ClassDataSharing
- JSE 6 – imbunătăţirea API-ului anterior
- Java SE 7 – string admis în switch, întregi reprezentaţi prin literali, try-with-resources, inferenţa tipurilor de date via operatorul <>, parallel programing, nio
- Java SE 8 – lambda expressions, stream, functional interface
- JSE 9 - optional phase, link time, JMOD
- JSE10 – inferenţa variabilelor locale, ClassDataSharing extins, GC paralelizat, Java based JIT compiler


Java Language
Java Language
java
javac
javadoc
jar
javap
jdeps
Scripting
Tools & Tool APIs
Security
Monitoring
JConsole
VisualVM
JMC
JFR
JPDA
JVM TI
IDL
RMI
Java DB
Deployment
Internationalization
Web Services
Troubleshooting
Deployment
Java Web Start
Applet / Java Plug-in
JavaFX
User Interface Toolkits
Swing
Java 2D
AWT
Accessibility
Drag and Drop
Input Methods
Image I/O
Print Service
Sound
Integration Libraries
IDL
JDBC
JNDI
RMI
RMI-IIOP
Scripting
Other Base Libraries
Beans
Security
Serialization
Extension Mechanism
JMX
XML JAXP
Networking
Override Mechanism
JNI
Date and Time
Input/Output
Internationalization
lang and util
lang and util Base Libraries
Math
Collections
Ref Objects
Regular Expressions
Logging
Management
Instrumentation
Concurrency Utilities
Reflection
Versioning
Preferences API
JAR
Zip
Java Virtual Machine
Java HotSpot Client and Server VM
JDK
JRE
Compact Profiles
Java SE API

Object Oriented

- Process-oriented model: codul acţionează asupra datelor
- Object-oriented programming: datele controlează accesul
- Nivele multiple de abstractizare
- Principii OOP:
  - Încapsulare
  - Moştenire
  - Polimorfism





Un prim exemplu

Codul sursă:

```
/*

Acesta este codul sursă. Va fi salvat in fişierul
"Example.java".

*/

class Example {

// Your program begins with a call to main().

public static void main(String args[]) {

          System.out.println("This is a simple Java
          program.");

          }

}
```

Compilare:

```
C:\>javac Example.java
```

Rulare:

```
C:\>java Example
```

Rezultat:

```
This is a simple Java program.
```

If

```
class IfSample {
public static void main(String args[]) {
        int x, y;
        x = 10;
        y = 20;
        if(x < y)
        {       System.out.println("x is less than y");
                x = x * 2;
        }
        else
            System.out.println("x is not less than y");

        if(x == y) System.out.println("x now equal to y");
                x = x * 2;
        if(x > y) System.out.println("x now greater than y");

        // this won't display anything
        if(x == y) System.out.println("you won't see this");
        }
    }
```

Rezultat

```
x is less than y
x now equal to y
x now greater than y
```

For

```
class ForTest {
public static void main(String args[]) {
        int x;
        for(x = 0; x<10; x = x+1)
                System.out.println("This is x: " + x);

        String[] str = new String[]{"A","C","B","D"};
        for(String s : str)
        {
                System.out.println(s);
        }
        }
}
```

rezultat

```
This is x: 0
This is x: 1
This is x: 2
This is x: 3
This is x: 4
This is x: 5
This is x: 6
This is x: 7
This is x: 8
This is x: 9
A
C
B
D
```

Cuvinte cheie

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

Tipuri de dată

- Integer

Name	Width	Range
long	64	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
int	32	–2,147,483,648 to 2,147,483,647
short	16	–32,768 to 32,767
byte	8	–128 to 127

- Floating-point

Name	Width in Bits	Approximate Range
double	64	4.9e–324 to 1.8e+308
float	32	1.4e–045 to 3.4e+038

- Caracter 16 bit

- Boolean

Literali

- Intregi
  - int x = 0b1010;
  - int x = 123_456_789;
  - int x = 0b1101_0101_0001_1010;
- Floating-Point
  - float x = 6.022f;
  - double x= 314159E–05;
  - double a= 0x12.2P2;
  - double num = 9_423_497_862.0;
- Caractere
  - char x= '\141'; ⇔ char x =‘a’;
  - char x=‘*u0061*’; ⇔ char x=‘a’;

Escape Sequence	Description
\ddd	Octal character (ddd)
\uxxxx	Hexadecimal Unicode character (xxxx)
\'	Single quote
\"	Double quote
\\	Backslash
\r	Carriage return
\n	New line (also known as line feed)
\f	Form feed
\t	Tab
\b	Backspace

Variabile

- *type identifier* [= *value*][, *identifier* [= *value*] …];

Domeniu unei variabile

- Variabilele declarate într-un bloc, există şi sunt vizibile doar în acel bloc.
- Blocurile (domeniile de vizibilitate) pot fi imbricate.
- Un bloc în Java înseamnă fie o metodă, fie o clasă. De asemenea, în cadrul metodelor pot exista blocuri.
- Singurul mod in care se pot declara variabile globale în Java, se poate realiza prin folosirea **static.**

```
class Scope {
        public static void main(String args[]) {
                int x; // known to all code within main
                x = 10;
                if(x == 10) { // start new scope
                        int y = 20; // known only to this block
                        // x and y both known here.
                        System.out.println("x and y: " + x + " " + y);
                        x = y * 2;
                }
                // y = 100; // Error! y not known here
                // x is still known here.
                System.out.println("x is " + x);
        }
}
```

Rezultat:

```
x and y: 10 20
x is 40
```

Conversie între tipuri de date

- Conversia automată când:
  - Cele 2 tipuri sunt compatibile
  - Tipul destinaţie este mai mare decât cel sursă
- Conversia de tip *narrowing*
  - Prin cast de tipul: (*target-type*) *value.* Exemplu **int** la **byte**
  - Prin procedeul de trunchiere. Exemplu **float** la **int**
- Type promotion:
  - **byte**, **short**, **char** sunt automat convertite (promoted) la **int**
  - Dacă un operand este **float(double)**, expresia este convertită la **float(double)**

Conversii: exemplu

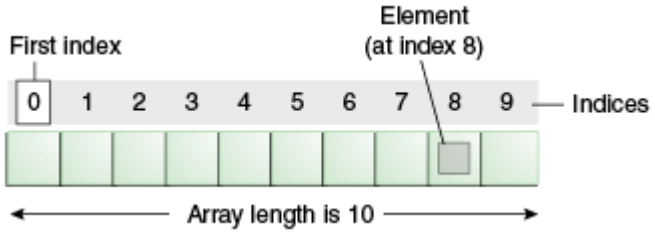
```
class Promote {
public static void main(String args[]) {
        byte b = 42;
        char c = 'a';
        short s = 1024;
        int i = 50000;
        float f = 5.67f;
        double d = .1234;
        double result = (f * b) + (i / c) - (d * s);
        System.out.println((f * b) + " + " + (i / c) + " - " + (d * s));
        System.out.println("result = " + result);
}
}
```

Rezultat

```
238.14 + 515 - 126.3616
result = 626.7784146484375
```

Array

- Cu o singură dimensiune:
  - Declarare *type var-name*[ ];
  - Iniţializare: *array-var* = new *type* [*size*];




```
class Array {
public static void main(String args[]) {
        int month_days[];
        month_days = new int[12];
        month_days[0] = 31;
        month_days[1] = 28;
        month_days[2] = 31;
        month_days[3] = 30;
        month_days[4] = 31;
        month_days[5] = 30;
        month_days[6] = 31;
        month_days[7] = 31;
        month_days[8] = 30;
        month_days[9] = 31;
        month_days[10] = 30;
        month_days[11] = 31;
        System.out.println("April has " + month_days[3] + " days.");
    //sau: int month_days[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31,30, 31 };

}
}
```

Array

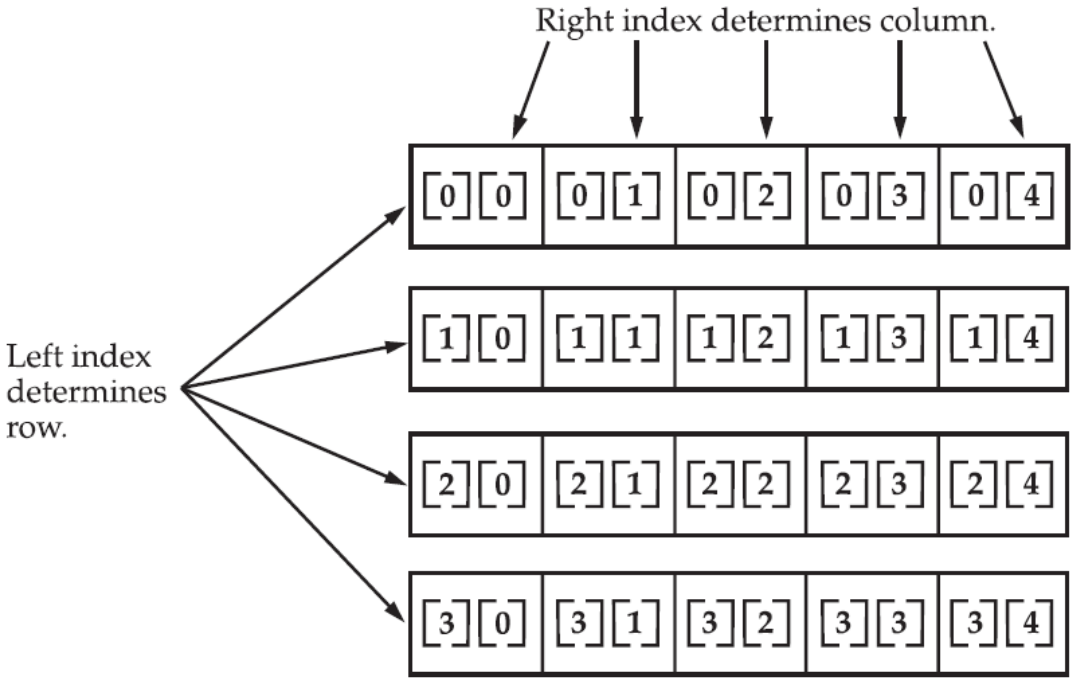
- multidimensionale

  int twoD[][] = new int[4][5];

Rezultat

```
0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19
```

```
// Demonstrate a two-dimensional array.
class TwoDArray {
    public static void main(String args[])
    {
        int twoD[][]= new int[4][5];
        int i, j, k = 0;
        for(i=0; i<4; i++)
        for(j=0; j<5; j++) {
        twoD[i][j] = k;
        k++;
        }
        for(i=0; i<4; i++) {
        for(j=0; j<5; j++)
        System.out.print(twoD[i][j] + " ");
        System.out.println();
    }
    }
}
```



Jagged Array

- Sintaxa permisivă
  - int al[] = new int[3];
  - int[] a2 = new int[3];
  - int[] nums, nums2, nums3;
  - int nums[], nums2[], nums3[];

```
public class JArray{
public static void main(String args[]) {
        int arr[][] = new int[3][];
        for(int i=0; i<3; i++)
        {
                arr[i] = new int[i+1];
                for(int j=0;j<arr[i].length;j++)


        System.out.print(arr[i][j] + " ");
                System.out.println();
        }
}
}
```

Rezultat

```
0
0  0
0  0  0
```

String

- Fields:
  - static Comparator<String> CASE_INSENSITIVE_ORDER
- Methods:
  - charAt(int index)
  - compareToIgnoreCase(String str)
  - contentEquals(StringBuffer sb)
  - format(Locale l, String format, Object... args)
  - getBytes(Charset charset)
  - …

```
class MakeString {
public static void main(String args[]) {
        char c[] = {'J', 'a', 'v', 'a'};
        String s1 = new String(c);
        String s2 = new String(s1);
        System.out.println(s1);
        System.out.println(s2);
}
}
```

String - ascii

```
class SubStringCons {
public static void main(String args[]) {
     byte ascii[] = {65, 66, 67, 68, 69, 70 };
     String s1 = new String(ascii);
     System.out.println(s1);
     String s2 = new String(ascii, 2, 3);
     System.out.println(s2);
     System.out.println(s2.length());
}
}
```

String - substring

```
class getCharsDemo {
public static void main(String args[]) {
   String s = "This is a demo of the getChars method.";
   int start = 5;
   int end = 14;
   char buf[] = new char[end - start];
   s.getChars(start, end, buf, 0);
   System.out.println(buf);
}
}
```

String - cautare

```
class indexOfDemo {

public static void main(String args[]) {

String s = "Now is the time for all good men " + "to come to the aid of their
country.";

System.out.println(s);

System.out.println("indexOf(t) = " + s.indexOf('t'));

System.out.println("lastIndexOf(t) = " + s.lastIndexOf('t'));

System.out.println("indexOf(me) = " + s.indexOf("me"));

System.out.println("matches me = " + s.matches("(.*)me"));

System.out.println("matches me = " + s.matches("(.*)me(.*)"));}

}
```

Now is the time for all good men to come to the aid of their country.
indexOf(t) = 7
lastIndexOf(t) = 65
indexOf(me) = 13
matches me = false
matches me = true

String
alte metode

Method	Description
int codePointAt(int *i*)	Returns the Unicode code point at the location specified by *i*.
int codePointBefore(int *i*)	Returns the Unicode code point at the location that precedes that specified by *i*.
int codePointCount(int *start*, int *end*)	Returns the number of code points in the portion of the invoking **String** that are between *start* and *end*–1.
boolean contains(CharSequence *str*)	Returns **true** if the invoking object contains the string specified by *str*. Returns **false** otherwise.
boolean contentEquals(CharSequence *str*)	Returns **true** if the invoking string contains the same string as *str*. Otherwise, returns **false**.
boolean contentEquals(StringBuffer *str*)	Returns **true** if the invoking string contains the same string as *str*. Otherwise, returns **false**.
static String format(String *fmtstr*, Object ... *args*)	Returns a string formatted as specified by *fmtstr*. (See Chapter 19 for details on formatting.)
static String format(Locale *loc*, String *fmtstr*, Object ... *args*)	Returns a string formatted as specified by *fmtstr*. Formatting is governed by the locale specified by *loc*. (See Chapter 19 for details on formatting.)
boolean isEmpty( )	Returns **true** if the invoking string contains no characters and has a length of zero.
boolean matches(string *regExp*)	Returns **true** if the invoking string matches the regular expression passed in *regExp*. Otherwise, returns **false**.
int offsetByCodePoints(int *start*, int *num*)	Returns the index within the invoking string that is *num* code points beyond the starting index specified by *start*.
String replaceFirst(String *regExp*, String *newStr*)	Returns a string in which the first substring that matches the regular expression specified by *regExp* is replaced by *newStr*.
String replaceAll(String *regExp*, String *newStr*)	Returns a string in which all substrings that match the regular expression specified by *regExp* are replaced by *newStr*.
String[ ] split(String *regExp*)	Decomposes the invoking string into parts and returns an array that contains the result. Each part is delimited by the regular expression passed in *regExp*.

Exerciţii cu şiruri

- Dat fiind 2 şiruri de numere întregi, găsiţi numărul de elemente comune
  - Exemplu {12,54,2,78,36,22,92} şi {33, 6,19, 86, 54,44,20,78} => |2|
- Verificaţi dacă elementele din cadrul unui şir respectă principiul unui palindrom
  - Exemplu {23,56,79, 61,34,61,79,56,23} – da {21,44,37,90,21,44,37} – nu
- Se dă o matrice *M* (*m*×*n* linii şi coloane). Extrageţi un şir de *m* elemente, fiecare element fiind calculat ca suma celor *n* întregi din fiecare coloană a lui *M*.
  - Exemplu $\begin{matrix} 1 & 1 & 2 \\ 5 & 0 & 0 \\ 2 & 3 & 3 \end{matrix}$ => {8,4,5}

Exerciții cu String-uri

- Se dă șirul de string-uri

  static String arr[] = {"Now", "is", "the", "time", "for", "all", "good", "men", "to", "come", "to", "the", "aid", "of", "their", "country"};
  - Creați un alt șir de String-uri cu elementele lui *arr* sortate. Folosiți funcția *compareTo()* din cadrul clasei String
- Concatenați String-urile din șirul *arr* de mai sus, folosind operatorul “+”. Apoi concatenați folosind metoda *join()*. Rezultatul ar arăta astfel:
  - String str = “Now,is,the,time,for,all,good,men,to, come,to,the,aid,of,their,country”.
  - Găsiți și altă metodă de concatenare a String-urilor? Încercați să obțineți un șir precum *arr* pornind de la variabila *str.* Indiciu: folosiți funcția spli
- Se dau următoarele variabile:

  double d = 102939939.939E+20;

  boolean b = true;

  long l = 1232874;

  char[] arr = {'a', 'b', 'c', 'd', 'e', 'f','g’ };
  - Transformați-le în variabile de tip String folosind metoda *valueOf()* și afișați-le la consolă. Încercați să le transformați din String-uri în tipurile de dată originale.