

# Curs 1 PPOO

Prof. univ. dr. Cristian CIUREA

Departamentul de Informatică și Cibernetică Economică

[cristian.ciurea@ie.ase.ro](mailto:cristian.ciurea@ie.ase.ro)

# Structură evaluare

## Curs:

- ▶ 60% (examen la calculator)

## Seminar:

- ▶ 20% (proiect, una din 5 teme alocată din oficiu)
- ▶ 20% (test grilă)

# Să ne cunoaștem...

E-business 2022/2023	Nr. Stud
<b>Academia de Studii Economice din București</b>	<b>55</b>
Facultatea de Business și Turism	2
Facultatea de Cibernetică, Statistică și Informatică Economică	52
Facultatea de Management	1
<b>Universitatea "Alexandru Ioan Cuza" din Iași</b>	<b>2</b>
Facultatea de Informatică	1
Facultatea de Biologie	1
<b>Universitatea "Dunărea de Jos" Din Galați</b>	<b>1</b>
Facultatea de Automatică, Calculatoare, Inginerie Electrică și Electronică	1
<b>Universitatea din București</b>	<b>6</b>
Facultatea de Administrație și Afaceri	3
Facultatea de Matematică și Informatică	3
<b>Universitatea Națională de Arte "George Enescu" din Iași</b>	<b>1</b>
Facultatea de Arte Vizuale și Design	1
<b>Universitatea Politehnica Din București</b>	<b>3</b>
Facultatea de Automatică și Calculatoare	1
Facultatea de Știința și Ingineria Materialelor	2
<b>Universitatea Tehnică Din Cluj-Napoca</b>	<b>1</b>
Facultatea de Automatică și Calculatoare	1
<b>Total</b>	<b>69</b>

**Total = 17 studenți non-CSIE**

# Să ne cunoaștem...

► <http://www.cristianciurea.ase.ro/JavaTest.aspx>

# Ce obiective are disciplina?

- ▶ programare orientată obiect în Java;
- ▶ proiectare aplicații eficiente folosind principii de internaționalizare (I18N), documentare cod sursă (JavaDoc), design patterns (Singleton), etc.

# Ce vom învăța?

- ▶ Tipuri de date, variabile
- ▶ Masive
- ▶ Structuri de control
- ▶ Transferul parametrilor
- ▶ Clase, polimorfism, clase abstracte, interfete
- ▶ String si immutable
- ▶ Exceptii try-catch
- ▶ JavaDoc
- ▶ Internationalizare
- ▶ Colectii
- ▶ Genericitate
- ▶ Fisiere, serializare
- ▶ Singleton, Callback
- ▶ Fire de executie, procese
- ▶ Lucrul cu baze de date
- ▶ Spring, Swing, Java FX

# Java fundamentals

- ▶ JDK, JRE, JVM
- ▶ Concepte de bază ale JVM
- ▶ Utilizarea unui IDE - NetBeans/Eclipse/IntelliJ
- ▶ Structura unei aplicații Java
- ▶ Pachete
- ▶ Pași de dezvoltare a unei aplicații Java
- ▶ Compilare în linie de comandă
- ▶ Tipuri de date în Java
- ▶ Variabile în Java
- ▶ Boxing / Unboxing
- ▶ Masive în Java
- ▶ Stiva și Heap

# JDK, JRE, JVM

- ▶ **JDK - Java Development Kit:** formează subsetul SDK, care are responsabilitatea pentru scrierea și rularea programelor Java.
- ▶ **JRE - Java Runtime Environment:** conține API-uri adecvate, împachetate împreună cu JVM.
- ▶ **JVM - Java Virtual Machine:** este o mașină virtuală, care poate executa Java bytecode. Este componenta pentru execuția codului sursă a platformei software Java.



# Noutăți Java 8... Java 17

- ▶ **Java 8:** expresii lambda pentru accesare elemente colectii
- ▶ **Java 9:** modularizare JDK
- ▶ **Java 10:** release March 20, 2018
  - ▶ Experimental Java-based JIT compiler
- ▶ **Java 11:** release September 25, 2018
  - ▶ Epsilon: a no-op garbage collector
  - ▶ HTTP Client has been standardized ([java.net.http](http://java.net/http))
  - ▶ support for Unicode 10
- ▶ **Java 12:** release March 19, 2019
  - ▶ Switch Expressions, și multe altele
- ▶ **Java 13:** release September 17, 2019
  - ▶ sute de îmbunătățiri mai mici și mii de remedieri de erori

# Noutăți Java 8... Java 17

## ► Java 14:

- JDK 14 - released on March 17, 2020.
- Java 14 include alte caracteristici noi, precum și sute de îmbunătățiri mai mici și mii de remedieri de erori.

## ► Java 15:

- JDK 15 - released on September 15, 2020.
- Java 15 adaugă de ex. suport pentru șirurile de caractere (literals) pe mai multe linii (alias blocuri de text).

## ► Java 16:

- JDK 16 - released on March 16, 2021. Java 16 elimină opțiunile de compilare Ahead-of-Time (și Graal JIT).

## ► Java 17:

- JDK 17 is the long-term support (LTS) release since September 2021.

# Noutăți Java 8... Java 17

## ► Java 17:

- Java 17 este a doua versiune de suport pe termen lung (LTS) de la trecerea la noua cadență de lansare de 6 luni (prima fiind Java 11).

## ► Java 18:

- JDK 18 was released on March 22, 2022.

## ► Java 19:

- JDK 19 was released on 20 September 2022.

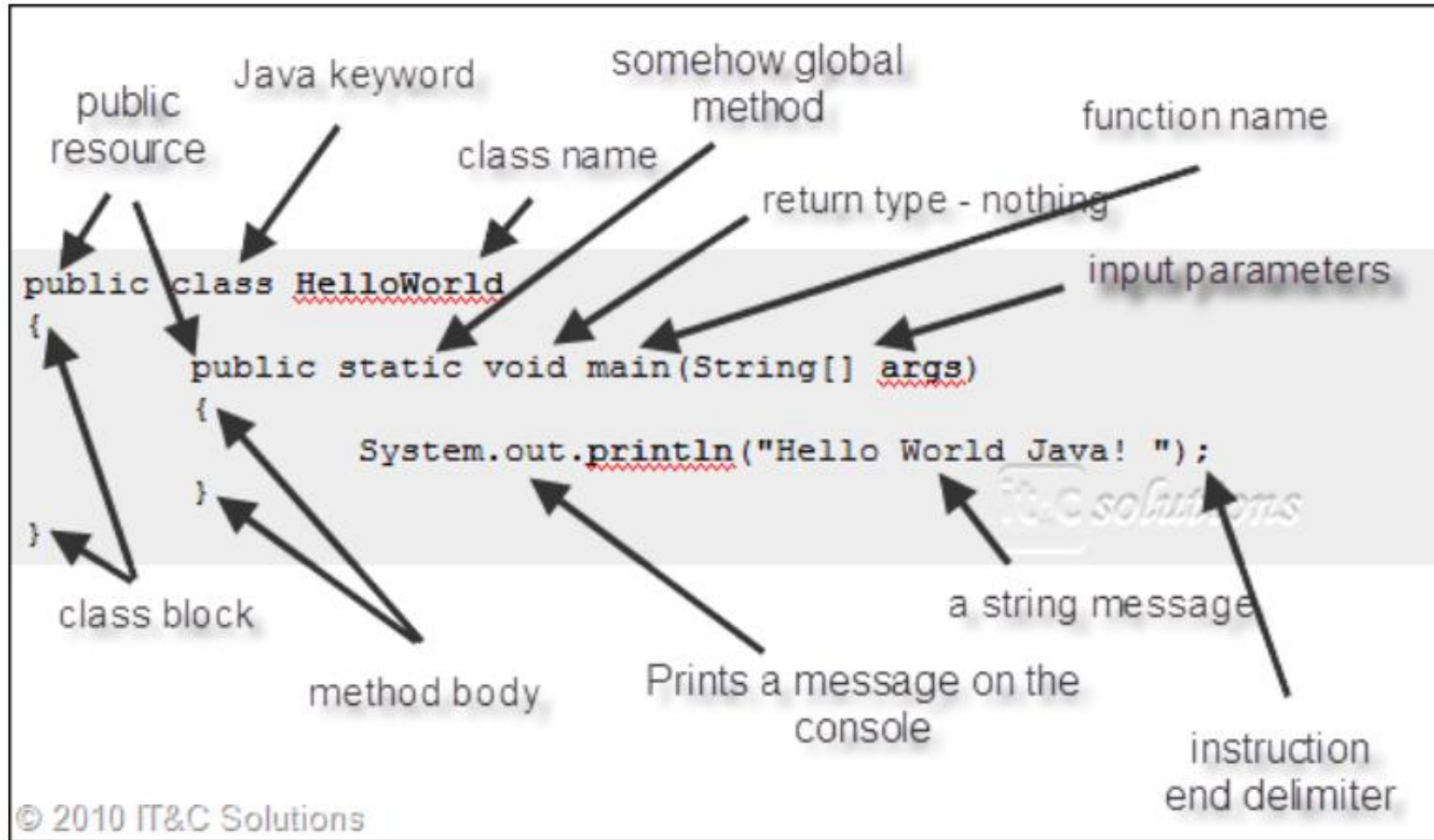
## ► Java 20:

- Java 20 is scheduled for release in March 2023.

# Utilizarea unui IDE

- ▶ NetBeans: <https://netbeans.org/>
- ▶ Eclipse: <https://www.eclipse.org/>
- ▶ IntelliJ IDEA: <https://www.jetbrains.com/idea/>

# Structura unei aplicații Java



# Structura unei aplicații Java

- ▶ comentariile pe o linie sunt definite prin //
- ▶ comentariile pe mai multe linii incluse între /\* și \*/
- ▶ delimitatorul pentru instrucțiuni este ;
- ▶ instrucțiunile pot fi incluse în blocuri de cod definite prin { }
- ▶ limbajul Java este case sensitive, variabila **vb** este diferită de **Vb** și **VB**

# Structura unei aplicații Java

- ▶ Totul se definește în interiorul unei clase;
- ▶ Nu se pot defini variabile globale sau metode în afara clasei (ca în C/C++);
- ▶ Conținutul unei clase este încadrat între { și };
- ▶ Clasa care include funcția *main()* are aceeași denumire cu fișierul de cod sursă care o conține.

# Pachete

- ▶ Un pachet este o grupare de clase înrudite, interfețe, enumerări și tipuri de adnotări care oferă protecție la acces și gestiune a spațiului de nume.
- ▶ Pentru a crea un pachet, se alege un nume pentru acel pachet și se pune o declarație de pachet cu acest nume în partea de sus a fiecărui fișier sursă care îl conține.

*//in the Graphic.java file*

```
package graphics;
```

```
public abstract class Graphic { ... }
```



# Pachete

- ▶ Numele pachetelor sunt scrise cu toate literele mici, pentru a evita un conflict cu numele de clase sau interfețe.
- ▶ Companiile folosesc numele inversat aferent domeniului de Internet pentru a începe denumirile de pachete: `com.example.mypackage` pentru un pachet denumit `mypackage` creat de un programator din compania `example.com`.
- ▶ Pachetele aferente limbajului Java încep cu `java.` sau `javax.`

# Pachete

- ▶ Pentru a importa un anumit membru în fișierul curent, se pune o declarație de import la începutul fișierului.
- ▶ `import graphics.Rectangle;`
- ▶ Pentru a importa toate tipurile conținute într-un pachet, se utilizează declarația de import cu asterisc (\*).
- ▶ `import graphics.*;`

# Pachete

Package class modifiers are:

- **default** (when you don't use anything) - - the class is visible in the package;
- **public** - the class is visible anywhere

*different packages*

```
package p1;  
  
public class Class1  
{ }  
  
class Class2  
{ }
```



```
package p2;  
  
import p1.*;  
  
public class Other  
{  
    Class1 c1;  
    Class2 c2;  
}
```

*only the public class is visible*

*same package*

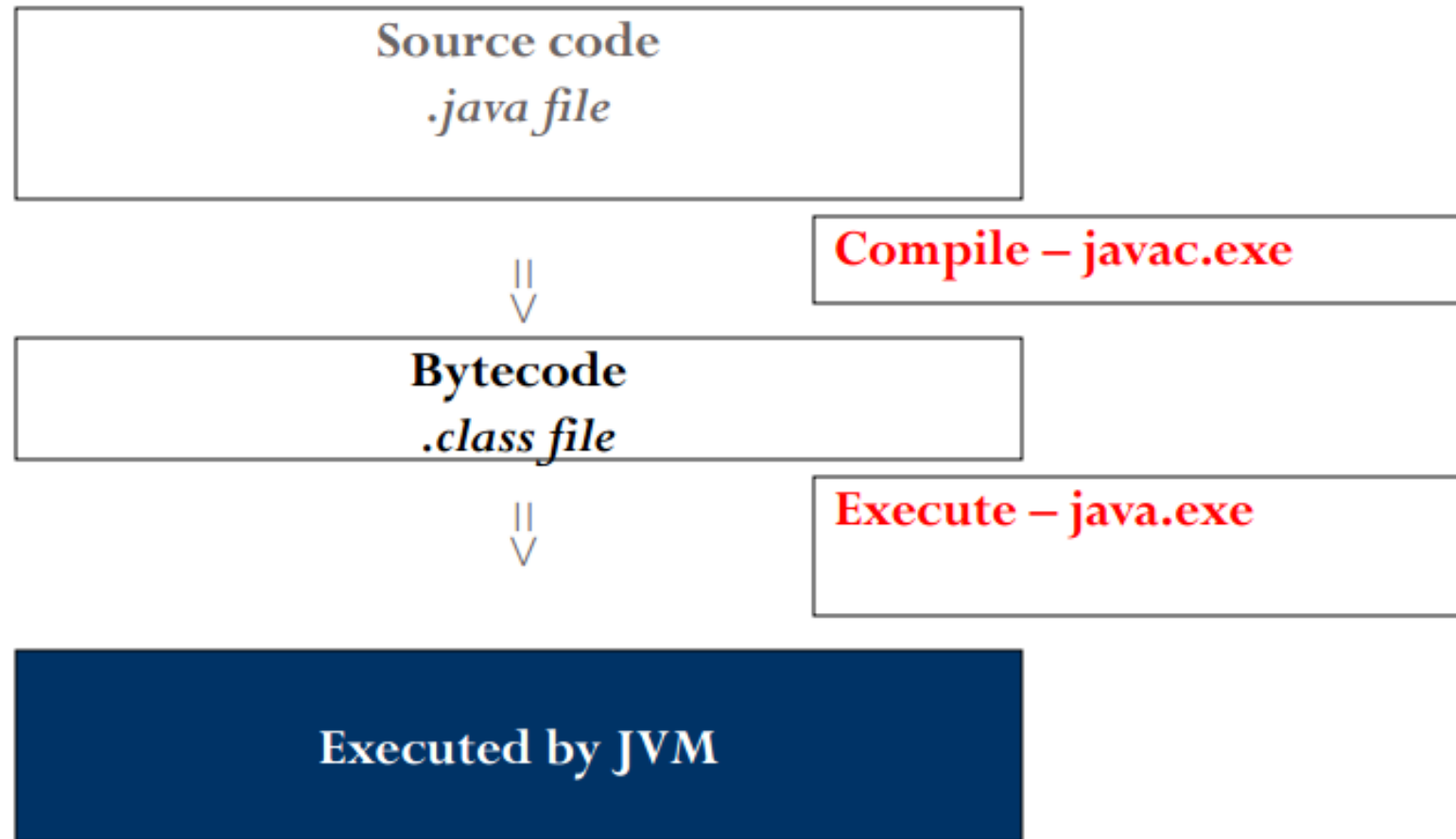
```
package p1;  
  
public class Class1  
{ }  
  
class Class2  
{ }
```



```
package p1;  
  
public class Other  
{  
    Class1 c1;  
    Class2 c2;  
}
```

*all classes are visible*

# Pași de dezvoltare a unei aplicații Java



# Pași de dezvoltare a unei aplicații Java

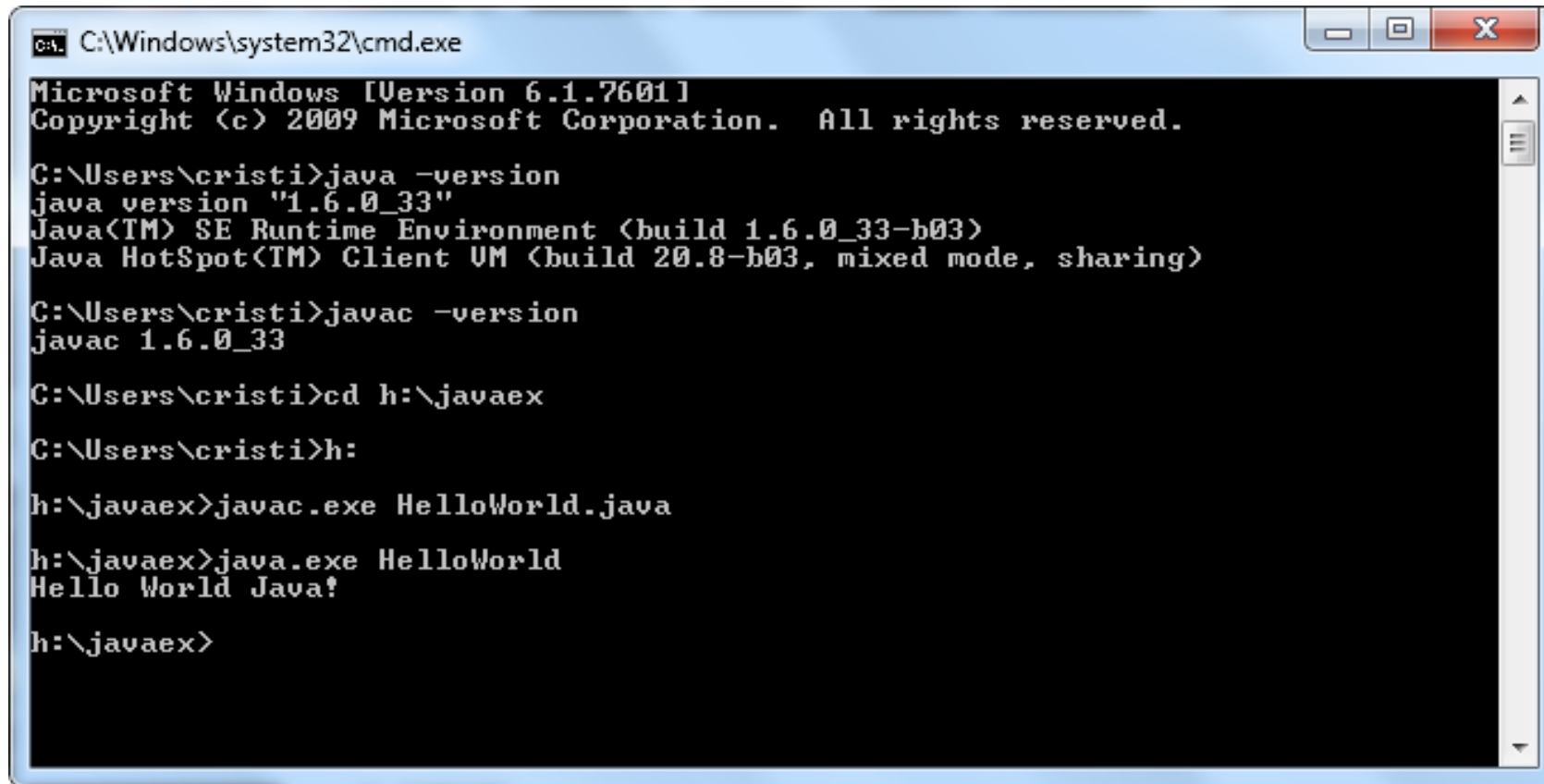
Instrumente necesare:

- ▶ Un editor ASCII simplu (Notepad, Notepad++, etc.) pentru scrierea fișierelor de cod sursă;
- ▶ Compilatorul Java (**javac.exe**) pentru a compila fișierele de cod sursă (**.java**) și a obține fișiere bytecode cu extensia **.class**;
- ▶ Mașina virtuală JVM (**java.exe**) pentru a executa aplicațiile Java.

# Pași de dezvoltare a unei aplicații Java

- ▶ Componentele **javac.exe** și **java.exe** se obțin prin instalarea Java SDK (JDK) de pe site-ul Oracle;
- ▶ Cele două executabile se găsesc în calea:  
**C:\Program Files\Java\jdk1.7.0\_71\bin**  
sau  
**C:\Program Files\Java\jdk-12.0.2\bin**  
în funcție de versiunea JDK instalată.

# Compilare în linie de comandă



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\cristi>java -version
java version "1.6.0_33"
Java(TM) SE Runtime Environment (build 1.6.0_33-b03)
Java HotSpot(TM) Client VM (build 20.8-b03, mixed mode, sharing)

C:\Users\cristi>javac -version
javac 1.6.0_33

C:\Users\cristi>cd h:\javaex

C:\Users\cristi>h:

h:\javaex>javac.exe HelloWorld.java

h:\javaex>java.exe HelloWorld
Hello World Java!

h:\javaex>
```

# Tipuri de date în Java

Value data type	Size	Range for signed values	Category
<b>byte</b>	<b>1 byte</b>	-128 → 127	integer
<b>short</b>	<b>2 bytes</b>	-32768 → 32767	integer
<b>int</b>	<b>4 bytes</b>	-2147483648 → 2147483647	integer
<b>long</b>	<b>8 bytes</b>	-9,223,372,036,854,775,808 → 9,223,372,036,854,775,807	integer
<b>float</b>	<b>4 bytes</b>	7 significant digits	real simple precision
<b>double</b>	<b>8 bytes</b>	15 significant digits	real double precision
<b>char</b>	<b>2 bytes</b>	'\u0000' → '\uffff' 0 → 65535	16 bits Unicode char
<b>boolean</b>	<b>1 bit</b>	true or false	logic value



# Variabile în Java

- ▶ Variabile de tipuri primitive (int, float, char, boolean, etc.);
- ▶ Variabile de tipuri referențiale (obiecte, interfețe, enumerări, învelitori (wrappers) pentru tipuri primitive).

# Variabile în Java

- ▶ Denumirile de variabile trebuie să înceapă cu o literă, simbolul underscore (\_) sau simbolul dolar (\$);
- ▶ Denumirile de variabile nu pot să înceapă cu o cifră;
- ▶ După primul caracter, se pot utiliza cifre în denumirile de variabile;
- ▶ Denumirile de variabile nu pot fi un cuvânt cheie sau rezervat din limbajul Java;
- ▶ Se pot defini mai multe variabile simultan.

# Variabile în Java

- ▶ Denumirile de variabile sunt alese de programator, dar pentru eficiență se pot respecta următoarele convenții de nume:
  - ▶ Notăția ungară;
  - ▶ Camel Case;
  - ▶ Java mixed case.

```
int iBooksNumber; //Hungarian Notation  
int BooksNumber; //CamelCase  
int booksNumber; //Java mixed case
```

# Variabile în Java

- Java is strong type language;

**float** vb2 = 23.5; *//compilation error - possible loss of precision*

**int** vb3 = 45.6; *//compilation error - possible loss of precision*

**boolean** test = 23; *//compilation error - incompatible types*

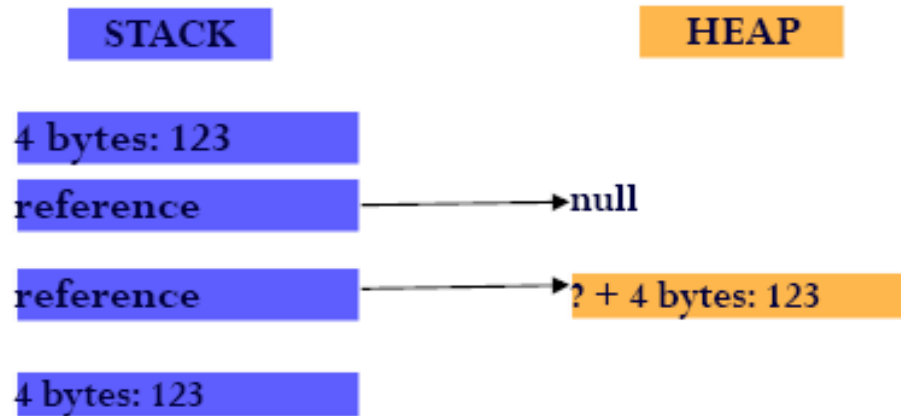
# Variabile în Java

```
public static void main()  
{  
int sum;  
//local variable declared in main  
method  
sum = sum + 10;  
//compiler error  
//variable sum might not have been  
initialized  
}
```

# Boxing / Unboxing

Converting a value type into a reference type and backwards is done by **boxing / unboxing**

```
public class BoxUnbox
{
    static void main(String[] args)
    {
        int i = 123;
        Integer iObject;
        iObject = i;
        int j = iObject;
    }
}
```



# Masive în Java

- ▶ un masiv este o formă particulară de obiect Java care este utilizat pentru a stoca o listă de elemente omogene (fiecare element al masivului are același tip ca tipul de bază);
- ▶ numărul de elemente din masiv este fix și definește lungimea sa;
- ▶ în Java, masivele sunt un tip special de obiecte - instanțe ale clasei Array;
- ▶ Java permite utilizarea operatorului index [ ] pentru a avea acces la elementele unui masiv;
- ▶ operatorul new acceptă ca parametru dimensiunea masivului.

# Masive în Java

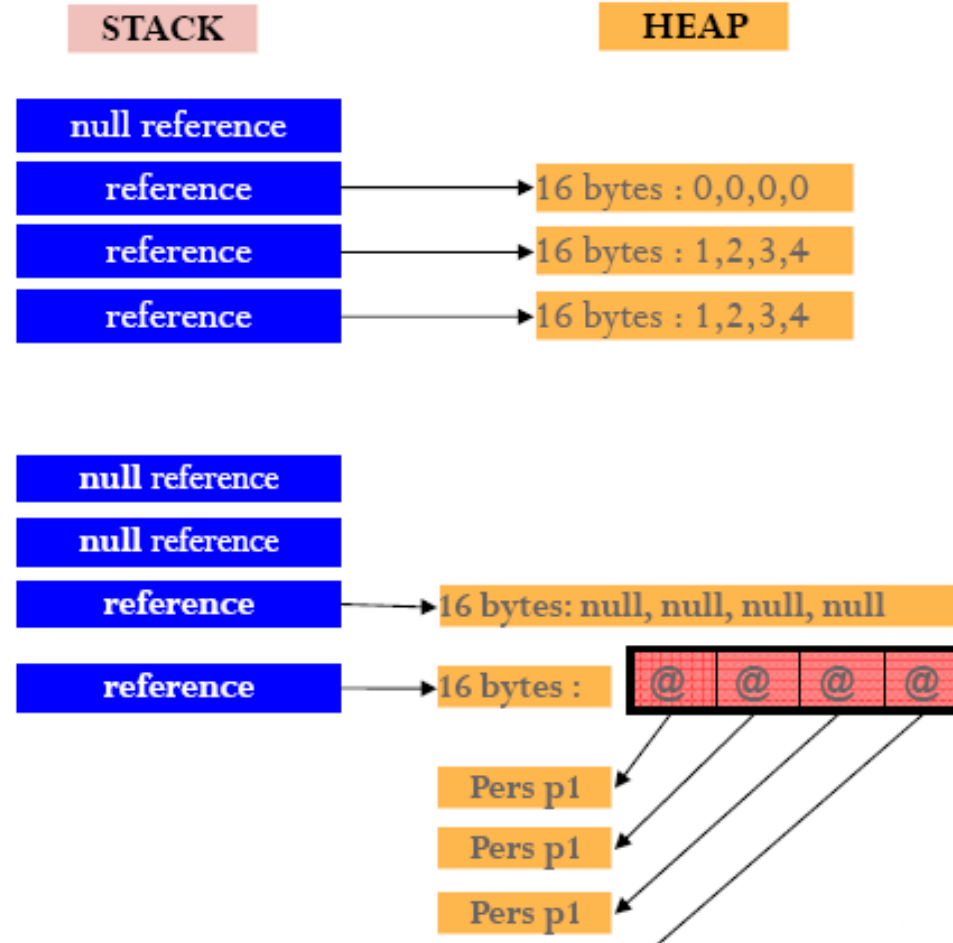
- ▶ Definirea masivelor unidimensionale (vectori) în Java:
  - ▶ `base_type[ ] array_name;`
  - ▶ `base_type array_name[ ];` //stil similar cu C/C++
- ▶ Inițializarea vectorilor în Java:
  - ▶ Se definește vectorul;
  - ▶ Se alocă spațiu de memorie pentru el;
  - ▶ Se inițializează elementele.
- ▶ Accesarea și prelucrarea vectorilor în Java:
  - ▶ Accesul la elemente cu operatorul index `[ ]`;
  - ▶ Numărul de elemente este gestionat cu proprietatea *length*.



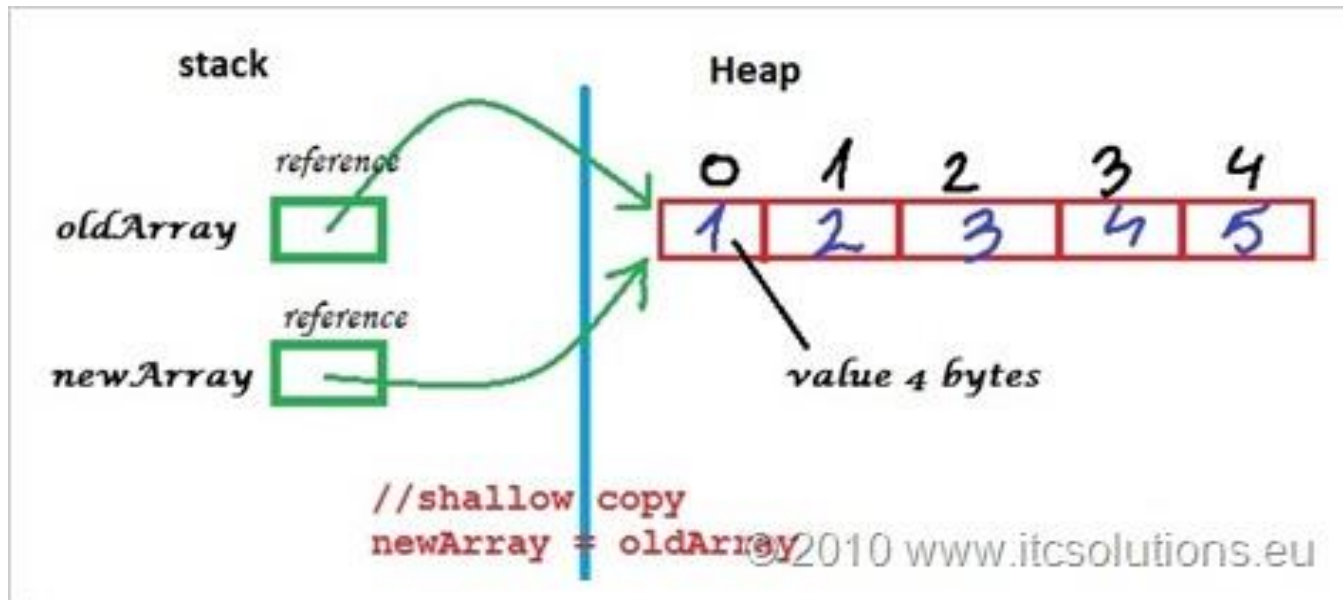
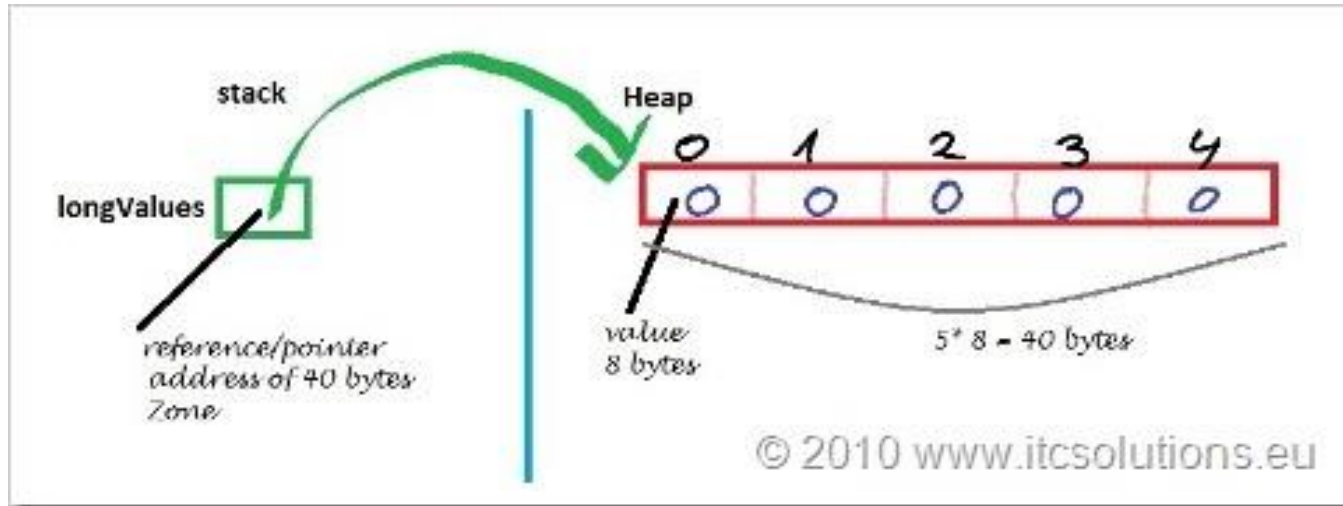
# Masive în Java

```
int [ ] vect;  
vect = new int[4];  
int [ ] vect2 = {1,2,3,4};  
int [ ] vect3 = new int[] {1,2,3,4};
```

```
Pers p1;  
Pers [ ] vectPers;  
vectPers = new Pers[4];  
vectPers = new Pers[4] {p1,p2,p3,p4};
```



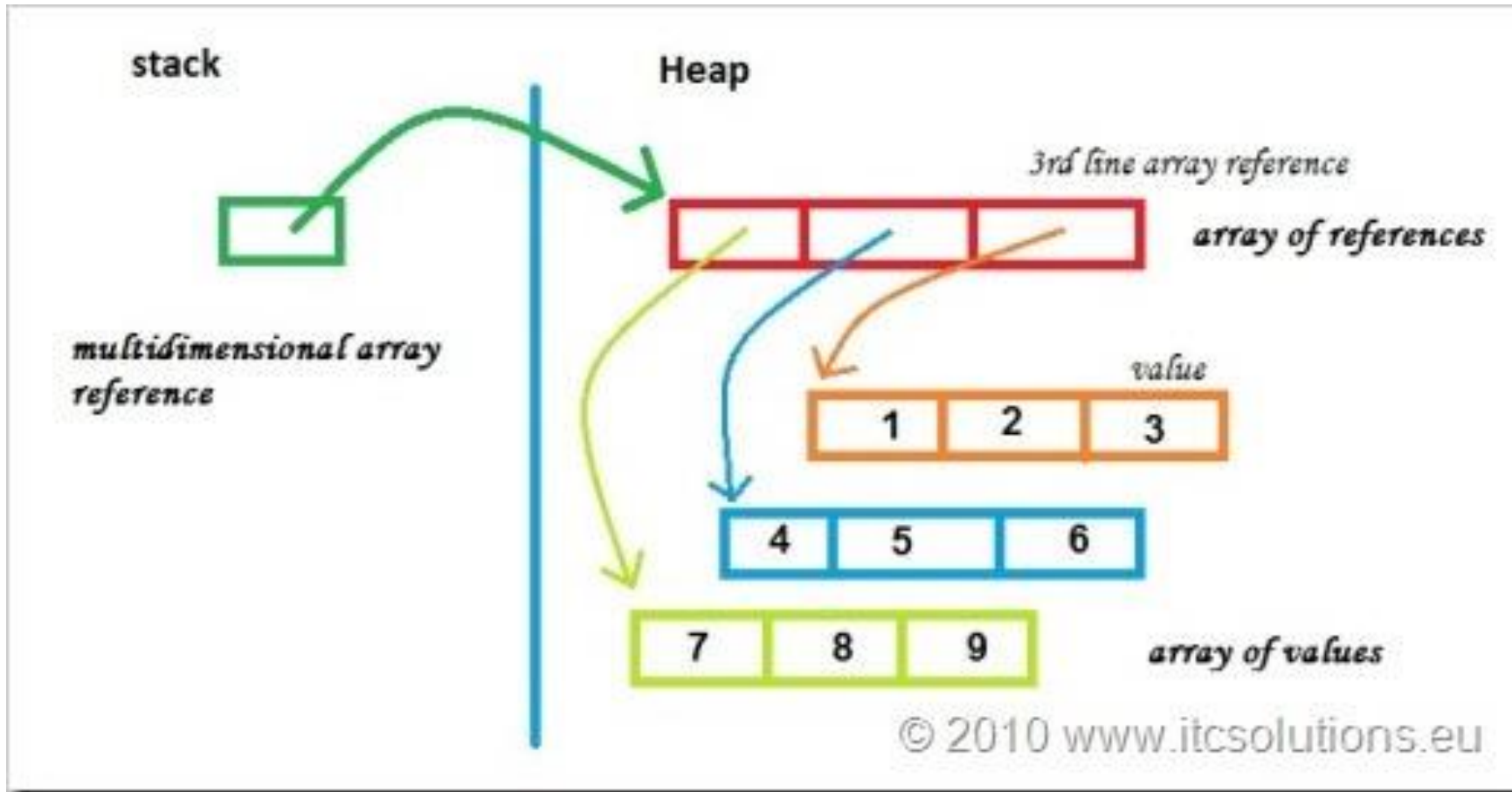
# Masive în Java



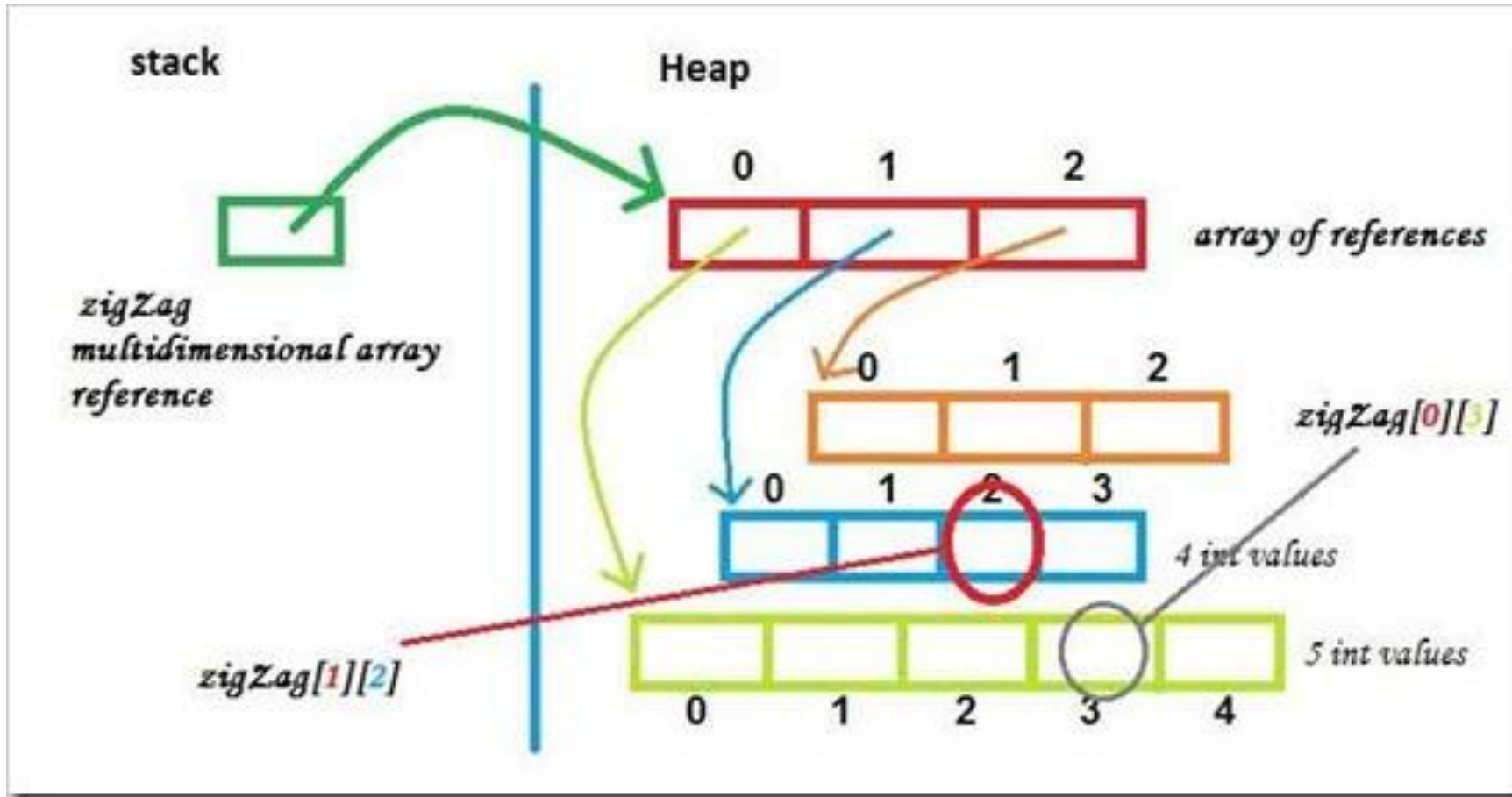
# Masive în Java

- ▶ Definirea masivelor bidimensionale (matrice) în Java:
  - ▶ `base_type[ ][ ] matrix_name;`
  - ▶ `base_type matrix_name[ ][ ];` //stil similar cu C/C++
- ▶ Inițializarea matricilor în Java:
  - ▶ Se definește vectorul de vectori (liniile matricei);
  - ▶ Se alocă spațiu de memorie pentru el și se inițializează numărul de elemente aferent primei dimensiuni;
  - ▶ Se rezervă spațiu de memorie pentru fiecare vector care va stoca valori; deoarece fiecare vector este prelucrat separat, este posibil să se stabilească diferite dimensiuni și să se creeze o matrice în zig-zag;
  - ▶ Se inițializează elementele.

# Masive în Java



# Masive în Java



# Stiva și Heap

## Stiva:

- ▶ un spațiu de memorie rezervat de sistemul de operare pentru procesul aplicației;
- ▶ dimensiunea stivei este fixă și este determinată în faza de compilare pe baza declarațiilor de variabile și alte opțiuni de compilare;
- ▶ este important să se stabilească faptul că stiva este limitată, iar dimensiunea sa este fixă (odată ce procesul a început, nu se mai poate modifica mărimea stivei);
- ▶ de cele mai multe ori, stiva este folosită pentru a stoca variabile ale funcțiilor sau metodelor (argumente de intrare și variabile locale);
- ▶ fiecare metodă are propria sa stivă, inclusiv metoda `main()`, care este, de asemenea, o funcție.
- ▶ o metodă există pe stivă numai pe timpul duratei de viață a acelei metode: din momentul apelului până la momentul returnării valorilor;

# Stiva și Heap

## Heap:

- ▶ un spațiu de memorie gestionat de sistemul de operare și utilizat de procese pentru a obține spațiu suplimentar la execuție;
- ▶ această zonă de memorie este globală, ceea ce înseamnă că orice proces o poate folosi (desigur, procesele nu pot citi sau scrie în zona Heap rezervată altui proces);
- ▶ rolul acestei zone de memorie este de a oferi resurse suplimentare pentru procesele care au nevoie de acest spațiu suplimentar la execuție (de exemplu, o aplicație simplă Java care construiește un vector cu valori de la consolă);
- ▶ spațiul necesar în timpul execuției unui proces este determinat de funcții cum ar fi *new* (aceeași funcție utilizată pentru a crea obiecte în Java), care sunt folosite pentru a obține spațiu suplimentar în Heap.

# Stiva și Heap

```
class Student {  
    int age; //instance variable  
    String name; //instance variable  
    public Student() {  
        this.age = 0;  
        name = "Anonymous";  
    }  
    public Student(int Age, String Name) {  
        this.age = Age;  
        setName(Name);  
    }  
    public void setName(String Name) {  
        this.name = Name;  
    }  
}
```

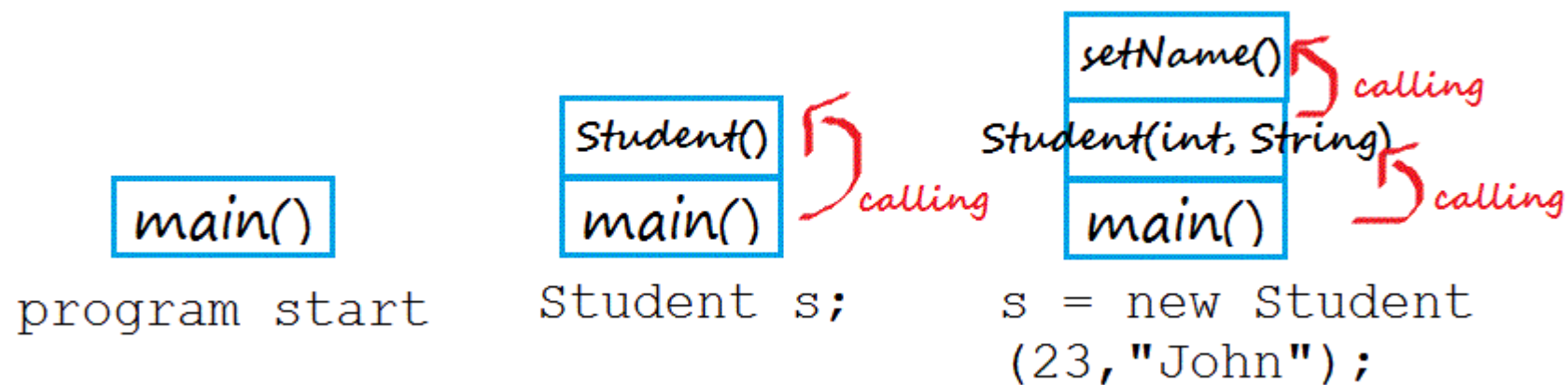


# Stiva și Heap

```
public class Main {  
    public static void main(String[] args) {  
        Student s; //local variable - reference  
        s = new Student(23,"John");  
        int noStudents = 1; //local variable  
    }  
}
```

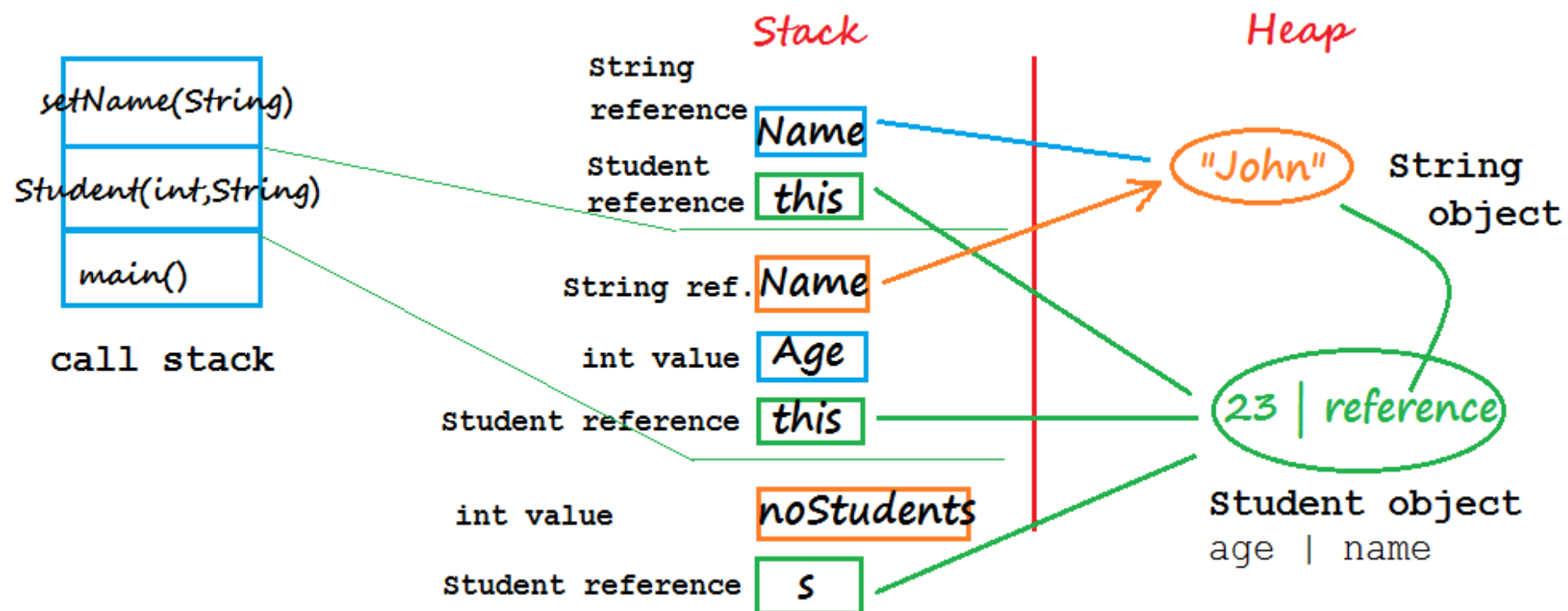
# Stiva și Heap

## ► Exemplu apel stivă:



# Stiva și Heap

- Valorile de pe stivă și din Heap:



# Stiva și Heap

Regula generală în ceea ce privește locul în care variabilele sunt plasate în memorie:

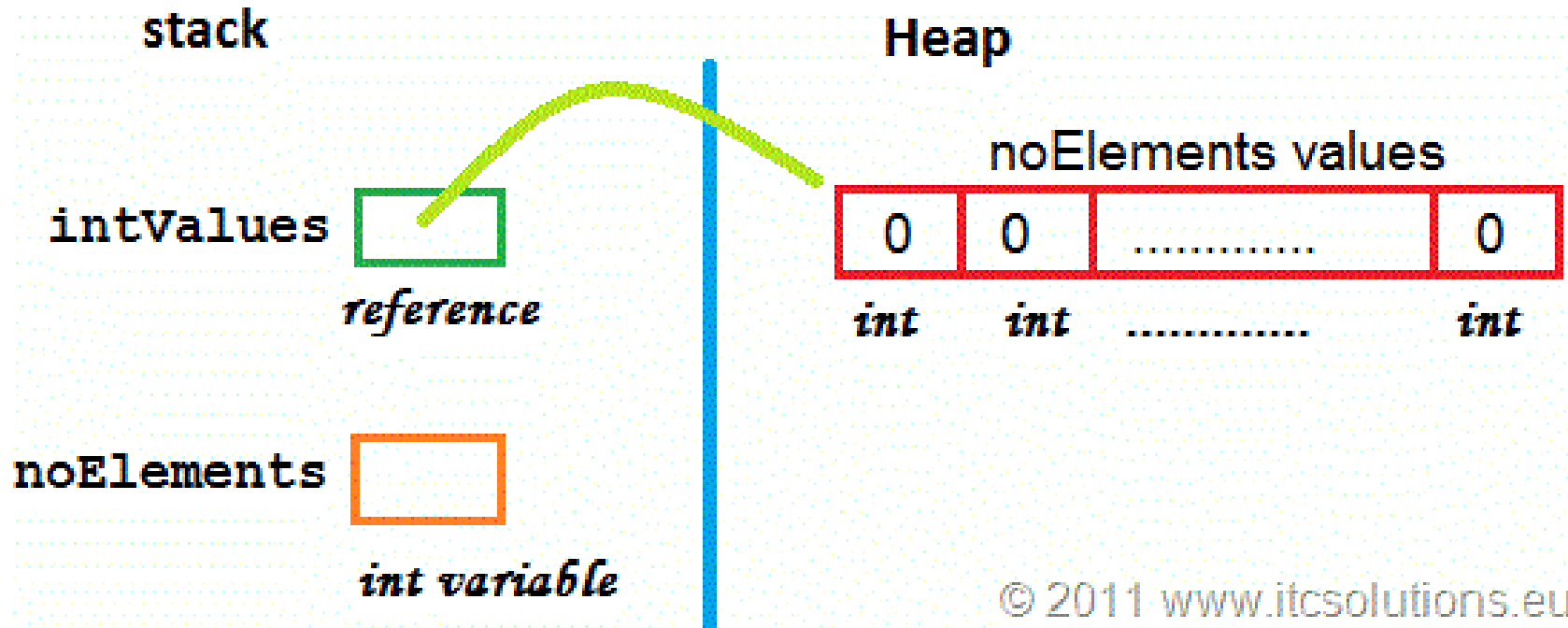
- ▶ variabilele declarate în blocuri de funcții (în Java nu se pot declara variabile globale, cum ar fi în C sau C++) sau în lista lor de parametri sunt plasate pe stivă (stiva funcției);
- ▶ orice valoare creată cu operatorul *new* este plasată în Heap.

# Stiva și Heap

```
try {  
    int[] intValues;  
    int noElements = 0;  
    System.out.println("The array elements number (0 - 255):"); //read  
number of elements  
    noElements = System.in.read();  
    intValues = new int[noElements]; //print the values - all with 0 default  
value  
    for(int i=0; i < intValues.length; i++) System.out.print(" "+intValues[i]);  
}  
catch(IOException ex) {  
    System.out.println(ex.getMessage());  
}
```

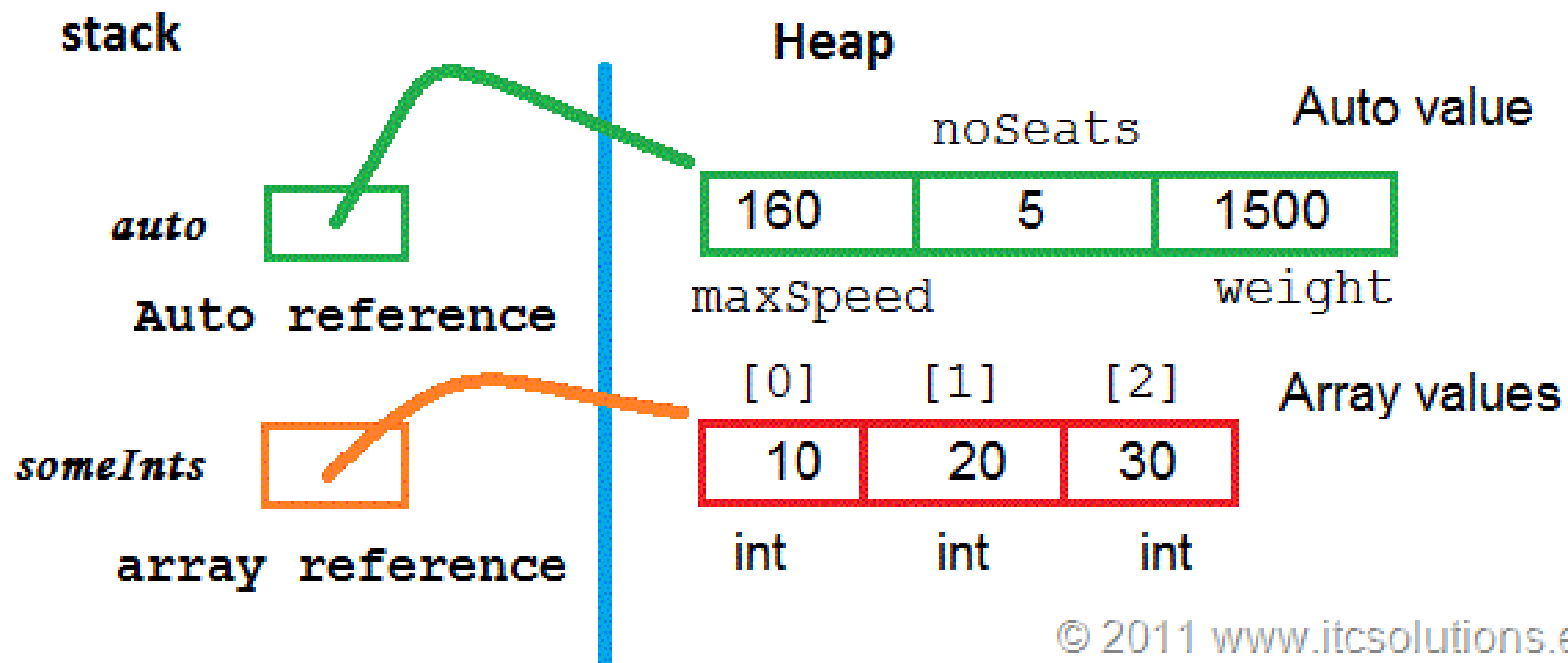
# Stiva și Heap

- Valorile obiectelor și referințele în memorie:



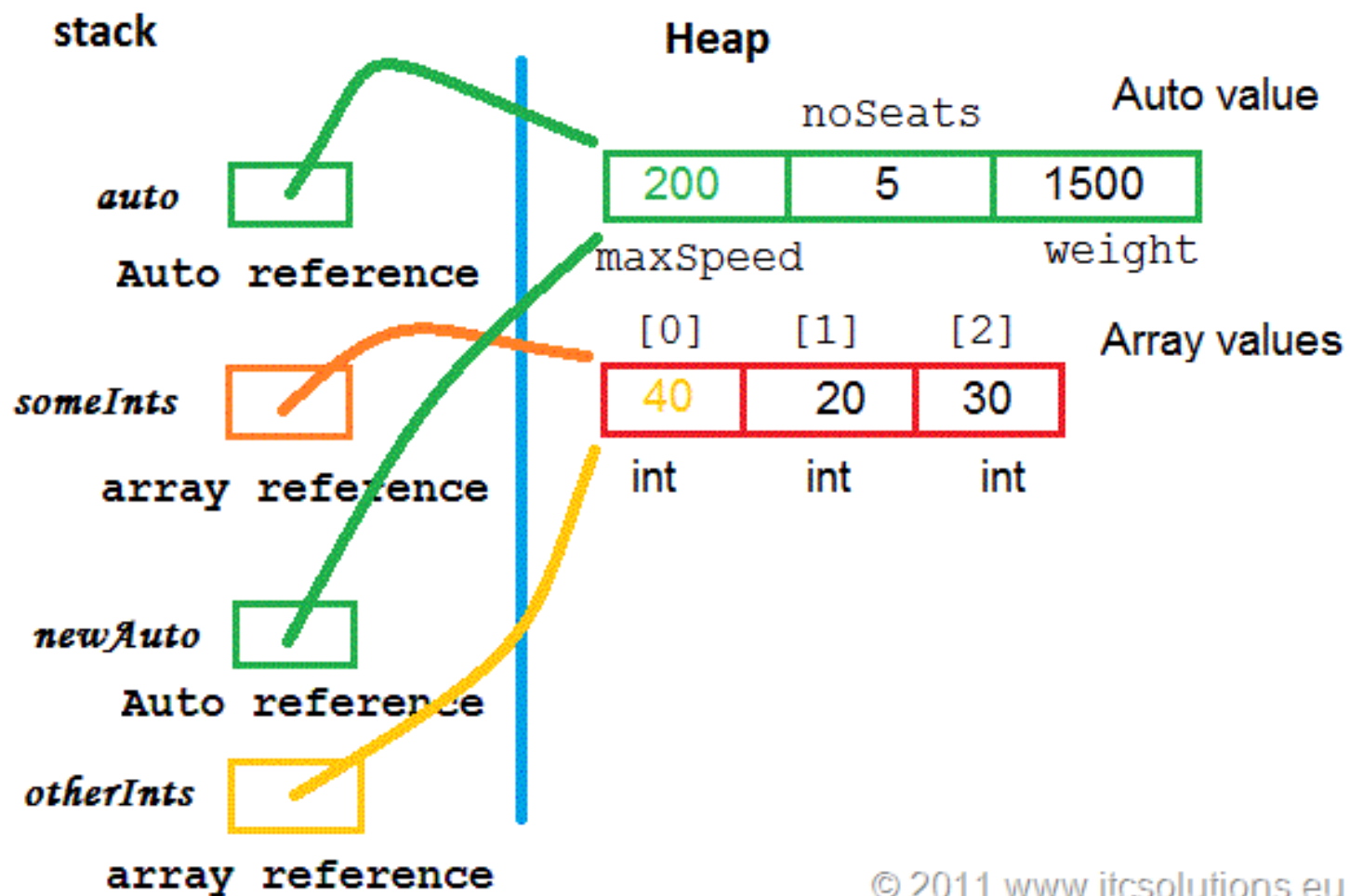
# Stiva și Heap

- Valorile obiectelor și referințele în memorie:



# Stiva și Heap

- Valorile obiectelor gestionate prin două referințe:





# Bibliografie

- ▶ [1] Jonathan Knudsen, Patrick Niemeyer - *Learning Java*, 3<sup>rd</sup> Edition, O'Reilly.
- ▶ [2] <http://www.itcsolutions.eu>
- ▶ [3] <http://www.acs.ase.ro>
- ▶ [4] <http://docs.oracle.com/javase/tutorial/index.html>