Curs 2 PPOO

Conf. univ. dr. Cristian CIUREA

Departamentul de Informatică și Cibernetică Economică

cristian.ciurea@ie.ase.ro

Java fundamentals

- Structuri de control
- Enumerări
- Transferul parametrilor
- Mecanismul try-catch
- String și immutable
- JavaDoc
- Internaționalizare

Structuri de control

Flow control structures:

- if-then
- if-then-else
- do-while
- while-do
- for
- enhanced-for
- switch

Structuri de control

IF-THEN

```
if (condition)
{
   < statement 1 >
   < statement 2 >
}
```

- *condition* is a boolean expression or variable that has the value *true* or *false* .For example *30*> *10* or 10 == 30

- are not accepted conditions based on expressions or variables that have numeric values

Structuri de control

IF-THEN-ELSE

```
if (condition)
{
   < statement 1 >
   < statement 2 >
}
else
{
   < statement 1 >
   < statement 2 >
}
```

- an alternative is the conditional operator:

```
condition ? then_statement : else_statement
```

Structuri de control

DO-WHILE

```
do
{
   < statement 1>
   < statement 2>
} while (condition)
```

-the condition for exiting/staying in the loop is checked at the end of the loop block

- the structure will run at least once the iteration statements

Structuri de control

WHILE-DO

```
while (condition)
{
   < statement 1>
   < statement 2>
}
```

- the condition for exiting from/staying in the loop is checked before the first statement in the loop block is executed

Structuri de control

FOR

```
for(initialization; condition;
   iteration)
{
   < statement 1>
   < statement 2>
}
```

- like *do-while;*
- is more efficient, simply because the iteration and the initialization statements are included in the structure and not in the block;

Structuri de control

ENHANCED-FOR

```
for ( variable: iterable_collection ){
    < statement 1>
    < statement 2>
}
```

- used to iterate through a collection that implements the *java.lang.Iterable* interface

Structuri de control

SWITCH

```
switch (testValue) {
case constant1:
   <statement 1>
   break;
case constant2:
   < statement 2>
   break;
...
default:
```

- conditional structure with multiple branches;
- each *case* clause is closed with the *break* instruction because it provides the exit from the *switch* structure;
- break is optional;

Enumerări

- Definirea unei enumerări:
  - **enum EnumName {constant1, constant2, …, constantN};**
  - //ATENȚIE simbolul ; (punct și virgulă) de la sfârșit este opțional
- Declararea unei enumerări:
  - independent sau global, deoarece enumerările sunt văzute ca și clase
  - în interiorul unei alte clase
  - NU în interiorul metodelor

Enumerări

Reguli la definirea unei enumerări:

- simbolurile unei enumerări sau constantele sunt de obicei definite cu litere mari (cum ar fi DIESEL, BENZINA, etc.), pe baza convențiilor de nume ale Sun;
- simbolurile unei enumerări nu sunt întregi sau string-uri;
- enumerările declarate la nivel global pot fi definite doar default sau public (NU private sau protected), dar în acest ultim caz, în propriul fișier .java (aceeași regulă ca și pentru clase);
- enumerările declarate într-o altă clasă pot avea modificatori de acces (private, public, protected, default), care controlează vizibilitatea acestora în afara clasei părinte.

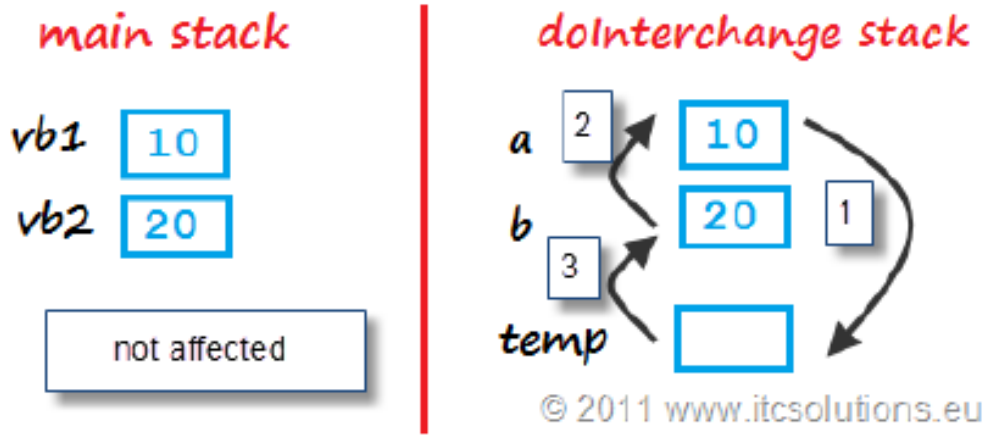
Enumerări

- enumerările sunt clase în Java și pot conține atribute, metode și constructori;
- fiecare enumerare furnizează o metodă statică, values(), utilizată pentru a itera peste constantele respectivei enumerări;
- simbolurile unei enumerări pot avea un corp constant de clasă specifică folosită pentru a înlocui o metodă generică.

Transferul parametrilor

- transfer parameters by their value

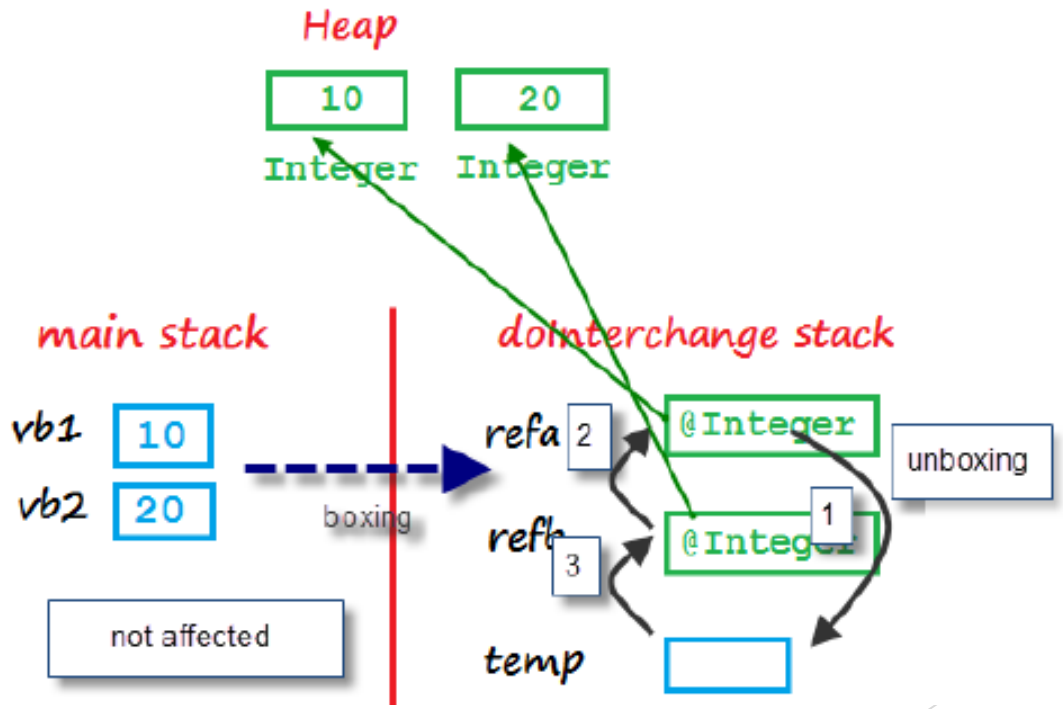
```
public static void doInterchange(int p, int q){
        int t = p;
        p = q;
        q = t;
}
```



Transferul parametrilor

- transfer parameters using wrappers

```
public static void doInterchange(Integer ra, Integer rb){
        int temp = ra;
        ra = rb;
        rb = temp;
  }
```



Mecanismul try-catch

- Permite gestiunea situațiilor excepționale care conduc la terminarea imediată a unui program;
- Necesar pentru a realiza programe fiabile și robuste;
- Implementare prin **try**, **catch**, **finally** și **throw**.

Mecanismul try-catch

```
try
        {//statements}
catch(exception_type_1)
        { //particular statements}
catch(exception_type_2)
        {// particular statements}
catch(Exception)
        { //general statements}
finally
        {//must-do statements}
```

Mecanismul try-catch

- Blocurile **try - catch - finally** pot fi incluse în alte blocuri **try**;
- Programatorul poate să-și definească propriile excepții prin derivarea de clase din Exception;
- Funcția **throw** generează o nouă excepție.

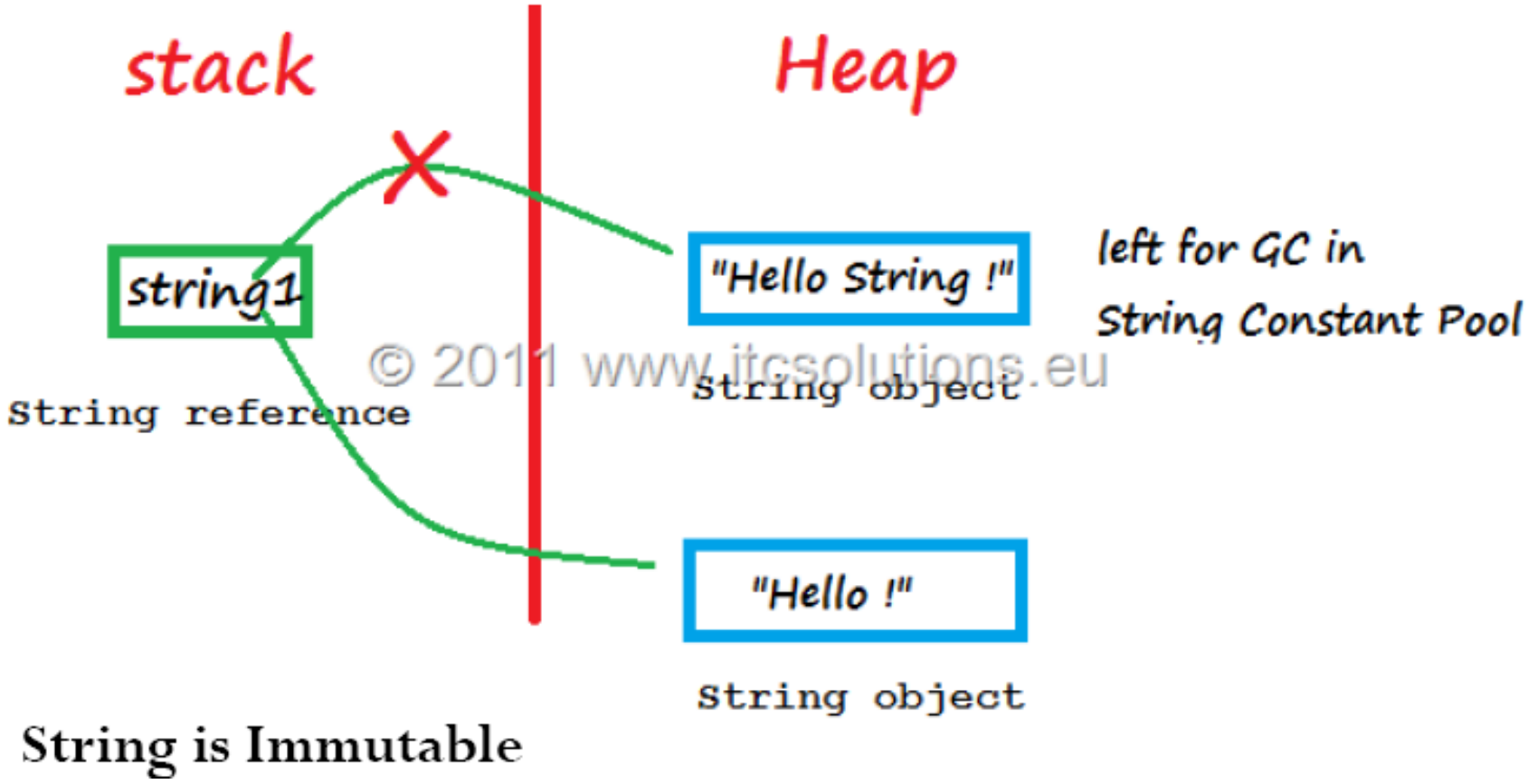
String și immutable

- În Java, fiecare caracter este o valoare Unicode pe 16 biți (2 B) și nu pe 1 B ca în C++;
- În Java, variabilele de tip string sunt gestionate prin obiecte din clasa String;
- În Java, sintaxa permite utilizarea obiectelor String ca tipuri de date primitive (se poate utiliza operatorul “=” pentru inițializarea acestora);
- În Java, variabilele de tip String sunt obiecte imuabile, adică odată ce au fost create nu-și mai pot modifica valoarea.

String și immutable

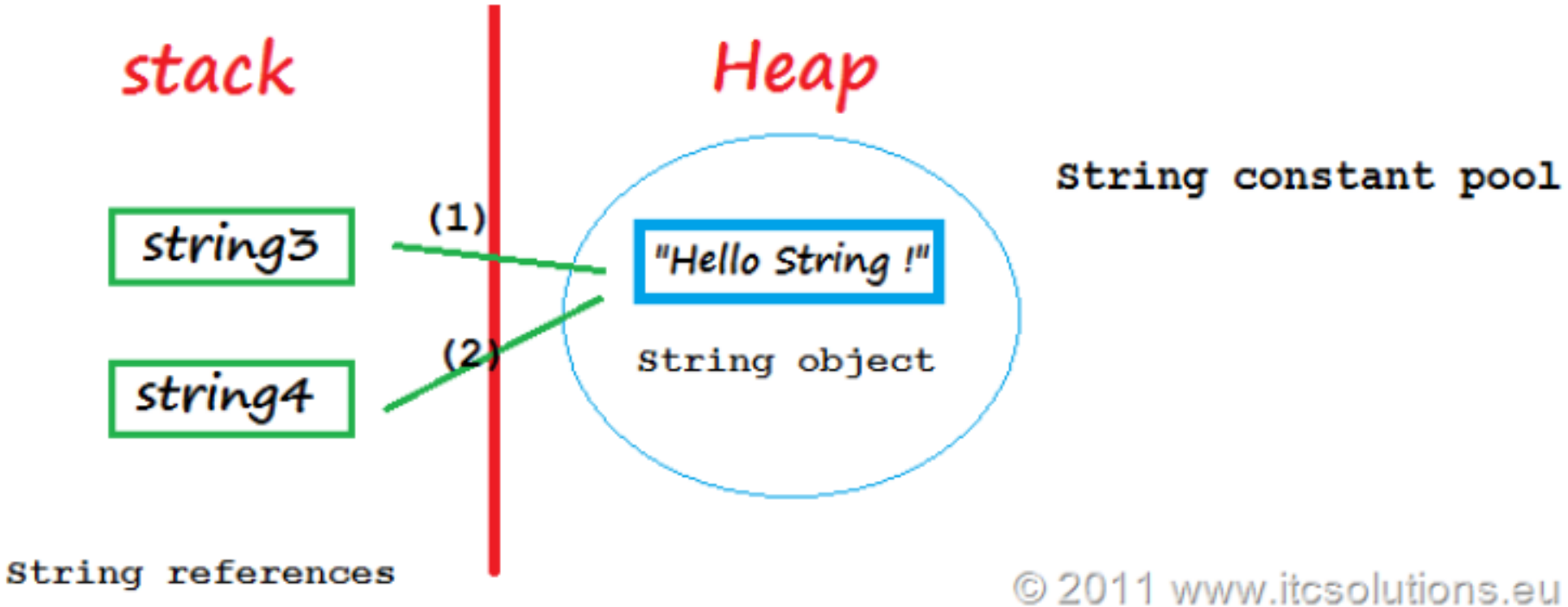
- Utilizarea operatorului “=” între două referințe String va copia valoarea referinței, nu valoarea obiectului;
- În Java, obiectele String sunt foarte speciale, pe de-o parte pentru că sunt imuabile și, pe de altă parte, pentru utilizarea eficientă a memoriei, JVM gestionează valorile String prin punerea lor într-o zonă specială de memorie denumită **String constant pool**.

String și immutable



String is Immutable

String și immutable



String și immutable

Method	Description
charAt()	returns the char at a given index; index takes values from 0 to length()-1;
concat()	appends a String to the end of another; the same as +
equals()	compare at case level 2 String values
length()	return the number of chars; IT IS NOT the length attribute of an array. IT IS A METHOD
replace()	replace occurences of a char with a given one
substring()	returns a substring
toLowerCase()	converts all chars to lowercase
toString()	returns the value of the String object
toUpperCase()	converts all chars to uppercase
trim()	remove whitespace from the end

These methods will NOT affect the current object.

String și immutable

StringBuilder și **StringBuffer**:

- cele două clase sunt aproximativ la fel;
- oferă o modalitate eficentă de lucru pentru operații de I/O pe fluxuri mari de date;
- valorile lor nu sunt stocate în *String constant pool*, ci se comportă ca obiecte normale.

String și immutable

Method	Description
append()	adds the argument to the end of the current object
delete()	deletes chars between a start and end index
insert()	inserts a value at a given offset
reverse()	reverse the value of the current object
toString()	returns the value of the StringBuilder or StringBuffer object

These method affects the value of the calling object, StringBuilder or StringBuffer.

JavaDoc

În activitatea de programare este important să se documenteze codul sursă, deoarece:

- vom avea o imagine clară asupra proiectelor complexe cu multe clase sau module;
- vom fi capabili, mai târziu, să înțelegem ce a fost făcut, astfel încât să putem modifica, adăuga sau șterge.

JavaDoc

- Pentru aplicațiile Java, documentația este furnizată de obicei în format HTML sub formă de arhive sau fișiere **.chm**.
- Pentru o documentație de cod sursă eficientă, proiectele JavaDoc sunt generate automat pe baza comentariilor de cod sursă cu ajutorul instrumentului **javadoc.exe**.

JavaDoc

Pentru a genera documentatia JavaDoc, comentariile de cod sursă trebuie să fie definite în conformitate cu anumite reguli:

- blocurile de comentarii JavaDoc încep cu /**
- blocurile de comentarii JavaDoc se termină cu */
- liniile de comentarii JavaDoc încep, prin convenție, cu *
- în timpul generării documentației JavaDoc, * sau spațiile de la începutul comentariului sunt ignorate
- comentariile sunt folosite pentru a genera documentația pentru metode sau clase
- deoarece documentația JavaDoc este în format HTML, comentariile pot conține tag-uri HTML folosite pentru a formata conținutul (de exemplu,
 pentru a trece pe linia următoare)

JavaDoc

Există o serie de tag-uri JavaDoc pentru clase sau atribute speciale ale metodelor:

JavaDoc Tag	Meaning	Description
@see	Name of associated class	Class, method
@author	Author	Class
@version	Version	Class
@param	Input parameters	Method
@return	Return value	Method
@exception	Generated exception	Method
@throws	Generated exception	Method
@deprecated	Defines the element as deprecated	Class, method
@since	The API version in which this element was included	Class, method

Internaționalizare

- **Internaționalizarea (i18n)** este procesul de proiectare a unei aplicații, astfel încât să poată fi adaptată la diferite limbi și regiuni fără modificări majore.
- **Localizarea (l10n)** este procesul de adaptare a software-ului pentru o regiune sau o anumită limbă prin adăugarea de componente specifice de localizare și traducerea textului.

Internaționalizare

Un program internaționalizat are următoarele caracteristici:

- același executabil poate rula oriunde în lume;
- elementele textuale, cum ar fi mesajele de stare și etichetele componentelor GUI nu sunt hardcodate în aplicație;
- suportul pentru adăugarea de noi limbi nu necesită recompilarea aplicației;
- datele calendaristice și monedele apar în formate care sunt conforme cu regiunea utilizatorului final și limba selectată.

Internaționalizare

- Un obiect **Locale** pentru localizare este un identificator pentru o anumită combinație de limbă și regiune. În cazul în care o clasă își variază comportamentul în funcție de elementele regionale, se spune ca este **locale-sensitive**.
- De exemplu, clasa **NumberFormat** este sensibilă la setările regionale. Astfel, NumberFormat poate returna un număr 902 300 (Franța) sau 902.300 (Germania) sau 902,300 (Statele Unite).

Bibliografie

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