Curs 2 POO
© Conf.dr. Crenguta M. Puchianu

Introducere

- Variabile
- Conversii
- Constante
- Expresii
- Instructiuni

Declararea variabilelor

- Declaraţiile de variabile sunt instrucţiuni care au ca rol:
  - alocarea de spaţiu pentru variabile,
  - introducerea în program de identificatori cu care se accesează aceste variabile,
  - asocierea unor atribute variabilelor,
  - iniţializarea variabilelor.

DeclaratieVariabila ::= { Modificator } TipDate Identificator;
| {Modificator} TipDate Identificator [=Expresie] {,Identificator [=Expresie]};

- Modificatori de vizibilitate: public, protected, private
- Alti modificatori: static, final

- Exemple: private int imprumut;
  private double a=2.3;
  public String mesaj, nume = "Ion";
  protected boolean gata = true;

- Regula. In Java, o variabila are o singura declaratie. Aceasta instructiune trebuie sa apara inainte de prima utilizare a ei.

Tipuri de date

- In Java exista doua categorii de tipuri de date: primitive si construite.

Nume tip	Descriere	Reprezentare în calculator
Tipuri întregi		
byte	Întreg de un byte	8-biţi (1 byte)
short	Întreg scurt	16-biţi (2 bytes)
int	Întreg	32-biţi (4 bytes)
long	Întreg lung	64-biţi (8 bytes)
Tipuri reale		
float	Real cu precizie simplă	32-biţi conform standardului IEEE 754 (4 bytes)
double	Real cu precizie double	64-biţi conform standardului IEEE 754 (8 bytes)
Alte tipuri		
char	Un caracter	16-biţi conform standardului Unicode (2 bytes)
boolean	Valoare logică (true sau false)	1 bit/1 byte

Tipuri de date construite

- Aceste tipuri reprezintă entităţi complexe ce se pot construi folosind mecanisme oferite de limbajul Java şi care pot fi:
  - clase
  - Tablouri (arrays)
  - interfeţe
  - sau tipuri enumerate (enum).
- Aceste tipuri pot fi definite de programator sau definiţia lor poate fi extrasă dintr-o bibliotecă existentă de tipuri Java. În ultimul caz avem de-a face cu o reutilizare de tipuri.

După categoria de tipuri cu care se declară, variabilele Java pot fi:

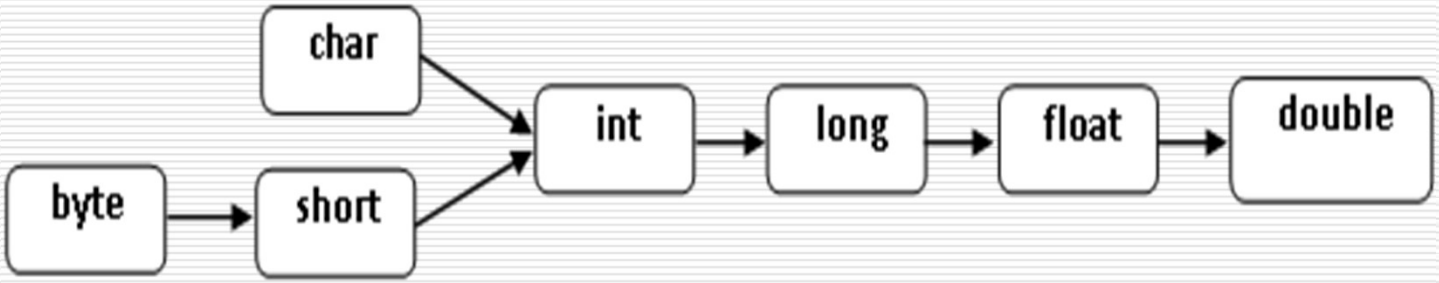
1. variabile de tip primitiv,
2. variabile de tip referinţă (în engleză, reference) la un tip T construit existent. Sunt variabile care conţin valoarea stânga (adresa de memorie) a unei variabile de tip T.

Exemple:
```
private Persoana p;
public String s;
public Test t;
```

- Un caz particular de valoare de tip referinţă este null. Ea indică, pentru orice tip referinţă, inexistenţa unui obiect referit.

Conversii ale valorilor de tip primitiv de date

- **Conversii implicite**. Java realizează, dacă contextul utilizării valorii o cere, conversii implicite a valorilor de tip primitiv. Dacă din context rezultă că pentru a se putea continua calculul o valoare de un anumit tip (a unei variabile sau a unei expresii) trebuie să fie convertită într-o valoare echivalentă de alt tip şi dacă această conversie este prevazută de limbaj ca implicită, atunci compilatorul Java introduce operaţii care convertesc automat valoarea.



Conversiile automate sunt facute de compilator în următoarele cazuri:

1. la atribuirea unei valori de un tip unei variabile de un alt tip
2. la apelul de metodă, la efectuarea transmiterii parametrilor, dacă tipul argumentelor este diferit de tipul parametrilor formali, şi

3. în calculul de expresii aritmetice, când o operaţie aritmetică (+, -, *, /, etc.) trebuie efectuată între două valori de tipuri diferite.

Conversii explicite

- **Casting** este o cerere explicită de realizare a unei conversii de tipuri de date. Ea se realizează precedând valoarea de convertit cu un "cast" format din numele tipului în care se doreşte să se facă conversia, cuprins între paranteze: **(T) expresie**

```
byte x = 127;
short x = (byte) 128;
int pixeli = (int)(latime/scala);
char c= (char) 65;
```

- În asemenea cazuri compilatorul va efectua conversia cerută chiar dacă ea produce şi o pierdere de informaţie (ca în cazul conversiei de la float la int, sau de la double la int). În cazul unei conversii imposibile (de la boolean la int, de exemplu), compilatorul semnalează eroare de compilare.

Constante

- O **constanta** este un element textual din program care reprezintă o valoare fixă ce nu se poate modifica în cursul execuţiei programului. Valoarea constantei rezultă direct din modul în care este scrisă sau dintr-o declaraţie precedentă în cazul constantelor simbolice. De exemplu, în instrucţiunea: int an = 2022; 2022 este o constantă.
- În funcţie de tipul de date de care aparţin, constantele pot fi:
  - constante numerice întregi, de exemplu: 16547 –654 987634872L 12e2
  - constante numerice reale, de exemplu: 2.25 -4556.43 3.1415927F 12.5e6 19E-65
  - constante booleene: true, false
  - constante caractere, de exemplu: ‘a’ ‘#’ ‘5’ ‘\u377’ ‘\u4323’

Notă. Caracterul special ‘\’ (backslash) este folosit în următoarele secvenţe “escape” pentru reprezentarea constantelor caracter necesare formatării informaţiilor afişate pe ecranul monitorului:

‘\n’ Newline. Poziţionează cursorul la începutul liniei următoare.

‘\t’ Tab orizontal. Aşează cursorul la următorul tab.

‘\r’ Carriage return. Aşează cursorul la începutul liniei curente, trecând în modul suprascriere.

Constante (cont.)

‘\f’ Form feed. Avansează cu o pagină.
‘\\’ Backslash. Este folosit pentru a afişa caracterul ‘\’.
‘\’’ Apostrof. Este folosit pentru a afişa caracterul apostrof.
‘\”’ Ghilimele. Este folosit pentru a afişa caracterul ghilimele.

- constante şir de caractere. Sintaxa lor este: un sir de caractere cuprins între ghilimele “…”. Aceste constante sunt unicul caz de constante-obiect în Java: valoarea unei asemenea constante nu este o valoare a unui tip primitiv ca în cazurile de mai sus, ci o valoare referinţă care referă un obiect al clasei **String**. Obiectul conţine şirul de caractere conţinut între ghilimele. Exemplu: “acesta este un sir.”

String s= “ hello “;

- ***Constantele simbolice*** sunt declarate în Java ca nişte variabile, doar că sunt precedate de modificatorul final.

final double PI;
PI= 3.14159265358979323 8;

- Uneori constantele sunt declarate ca variabile clasă ca MAX_VAL:

static final int MAX_VAL = 14567;

Expresii

O ***expresie*** este descrierea unui calcul a cărui evaluare produce o valoare unică, de un tip de date bine stabilit. Componentele unei expresii sunt: purtătorii de valori, numiti *operanzi*, şi simbolurile folosite pentru operaţii, numite *operatori*.
Operanzii unei expresii pot fi variabile, constante, apeluri de metode care returneaza o valoare sau alte expresii.

Operatori de comparaţie		**Operatori logici**		**Operatori pe şiruri**	
<	mai mic	&&	şi (evaluare scurtă)	+	concatenare
<=	mai mic sau egal	||	sau (evaluare scurtă)	**Operatori de deplasare**	
>	mai mare	&	şi (evaluare completă)	<<	deplasare la stânga
>=	mai mare sau egal	|	sau (evaluare completă)	>>	deplasare la dreapta cu extensie semn
==	egal	^	sau exclusiv (evaluare completă)	>>>	deplasare la dreapta cu extensie 0
!=	diferit	!	not		

Operatori (cont.)

Operatori aritmetici		Operatori multifuncţionali			
*	înmulţire	++	incrementează cu 1	/=	împarte cu valoarea specificată
/	împărţire	--	decrementează cu 1	%=	modulo valoarea specificată
%	modulo	+=	incrementează cu valoarea specificată	&=	"şi" pe biţi cu valoarea specificată
+	adunare	-=	decrementează cu valoarea specificată	|=	"sau" pe biţi cu valoarea specificată
-	scădere	*=	multiplică cu valoarea specificată	^=	"xor" pe biţi cu valoarea specificată
		<<=	deplasează stânga cu valoarea specificată	>>>=	deplasează dreapta fără semn cu valoarea specificată
		>>=	deplasează dreapta cu semn cu valoarea specificată	(tip)	conversie explicită a valorii operandului la tipul specificat in paranteze
instanceof		determină dacă un obiect aparţine unei clase			
new		creează un nou obiect			
?:		decizie			
,		secvenţă			

Instructiuni

Instrucţiunile sunt comenzi ale limbajului de programare ce modifică starea programului.

Instructiuni elementare:

- expresii:
  - expresii de atribuire
  - utilizarea lui ++ sau –-
  - apeluri de metode
  - expresii de creare a obiectelor: new String(“Buna ziua”)

- Instructiuni de declarare a variabilelor

Instructiuni de control:

- Instructiunea secventa – bloc {}

- Instructiuni de ramificare a controlului:

InstrucţiuneCondiţională ::= **if** (expresieLogica) instrucţiune1
[**else** instrucţiune2]

Instructiunea switch Java SE <12 vs Java SE >=12

InstrucţiuneDeSelectie ::= **switch (***E***){**
case v1: expresie_1;| *ListaInstrucţiuni_1*

case vn-1:
case vn: expresie_n;| *ListaInstrucţiuni_n*
[**default**: expresie_def;| *ListaInstrucţiuniDef*]
}

tip(E)={byte, short, int, char, enum, Byte, Short, Integer, Character, String}

Observatie: Instructiunea switch in general trebuie sa foloseasca **break**

Exemplu. Afisarea anotimpului unei luni date.

```
switch(luna){
        case "decembrie":
        case "ianuarie":
        case "februarie": System.out.println("Iarna"); break;
        case "martie":
        case "aprilie":
        case "mai": System.out.println("Primavara"); break;
        case "iunie":
        case "iulie":
        case "august": System.out.println("Vara"); break;
        case "septembrie":
        case "octombrie":
        case "noiembrie":System.out.println("Toamna");
    }
```

Expresia switch Java SE >=12

Versiunea Java SE 12:
ExpresieDeSelectie ::= **switch (***E***){**
case v1 **->** expresie_1;| *ListaInstrucţiuni_1*
case vi,vi+1, …vj **->** expresie_i;| *ListaInstrucţiuni_i*
…
case vn **->** expresie_n;| *ListaInstrucţiuni_n*
[**default ->** expresie_def;| *ListaInstrucţiuniDef*]
}

tip(E)={byte, short, int, char, enum, Byte, Short, Integer, Character, String}

Versiunea Java SE 13:Expresia switch poate folosi instructiunea **yield valoare**

```
String a = switch(luna){
  case "decembrie", "ianuarie", "februarie" -> "Iarna";
  case "martie", "aprilie", "mai" -> { System.out.println("A venit!");
                                       yield "Primavara";}
  case "iunie", "iulie", "august" -> "Vara";
  case "septembrie", "octombrie", "noiembrie" -> "Toamna";
  default -> "Eroare de anotimp";
  } ;
  System.out.println(a);
```

Instructiuni repetitive

InstrucţiuneWhile ::= **while (***E***)** *Instrucţiune* , unde E este o expresie logica

InstrucţiuneDoWhile ::= **do**
Instrucţiune
while (*E***);**

InstrucţiuneFor ::= **for** ([*ExpresieInitiala*]**;** [*ExpresieLogica*]**;** [*ExpresieIncrement*])
Instrucţiune

InstrucţiuneFor ::= **for** (*Declaratie***:** *Expresie*) *Instructiune*

Exemplu: *for(String s: args) System.out.println(s);*

Execuţia unei instrucţiuni repetitive se întrerupe cu **break** sau **continue**.