Ingineria Sistemelor de Programare

Structuri Lexicale

mihai.hulea@aut.utcluj.ro

2017

Entitatile lexicale Java

- Comentarii
- Cuvente cheie
- Identificatori
- Literali
- Separatori
- Operatori

https://docs.oracle.com/javase/specs/jls/se7/html/jls-3.html

Comentarii

- **/* ignore all between stars */**
  - La fel ca si in C
- **// ignore all till the end of this line**
  - La fel ca si in C++
- **/** this is a documentation comment */**
  - Trebuie sa apara imediat inainte definitiei unei clase, metode sau variabile

Comentarii pentru documentatie

- Utlizate de unelte precum **javadoc** pentru a produce documentatie HTML
- Optional pot fi incluse taguri precum **@param**, pentru descrierea parametrilor metodelor:

```
/** This method does what it feels like.
      @param  bar   This is a pointless argument. */
void foo (int bar) {. . .}
```

- Alte taguri **@returns**, **@see *name***

Cuvinte cheie rezervate

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

- goto nu este permis dar este rezervat
- null, true, si false semnificatii speciale

Tipuri de date

- Cum se declara o variabila ?

```
int i;
double x, y, z;
Color c;
```

- Doua tipuri de date:
  - *Tipuri primitive*
  - *Tipuri referinta*

Tipurile primitive

Type	Significance	Dimension	Minim	Maxim
byte	Integer	8 biţi	-128	+127
short	Integer cu semn	16 biţi	-32768	32767
int	Integer cu semn	32 biţi	-2147483648	2147483648
long	Integer cu semn	64 biţi	-2^{63}	$2^{63}-1$
char	Integer fără semn	16 biţi	Characters	Characters
float	Float point Virgulă mobilă în simplă precizie	32 biţi	$-3.40282347*10^{38}$	$3.40282347*10^{38}$
double	Virgulă mobilă în dublă precizie	64 biţi	$-1.7976931348623157*10^{308}$	$1.7976931348623157*10^{308}$
boolean	Val. booleană	32 biţi	false	true

Conversii intre tipuri

- Doua tipuri de conversii:
  - Widening (realizata automat de catre Java)
  - Narrowing (fortare prin cast)

```
int i = 13;
byte b = (byte) i; // Force the int to be converted to a byte

i = (int) 13.456; // Force this double literal to the int 13

long k = 10;
int y = (long)k; //eroare
```

Autoboxing si unboxing

- Conversia automata intre tipurile primitive si tipurile referinta asociate (wrapper)
- Tip primitiv > wrapper = autoboxing
- Wrapper > tip primitiv = unboxing

```
Character ch = 'a';

Integer i = new Integer(5);
int k = 3;
k = k + i;
```

Primitive type	Wrapper class
boolean	Boolean
byte	Byte
char	Character
float	Float
int	Integer
long	Long
short	Short
double	Double

Tipuri referinta

- Aceste tipuri sunt asociate cu obiectele si array-urile.
- Se numesc tipuri referinta pentru ca aceste variabile contin nu o valoare ci un pointer (o adresa) catre un obiect.
  - Variabilele de tip referinta care nu pointeaza catre nici un obiect au valoarea **null**.
- Tipurile referinta se pot divide in:
  - Class
  - Interface
  - Enum
  - Array

String—Un exemplu de tip referinta

- Java vine cu un set predefinit de clase printre care si clasa String pentru manipularea sirurilor de caractere.
- Declaratia clasei **String**:

  String s ; // variable declaration

- Prin declaratie nu se creeaza efectiv un obiect. Pentru a creea un obiect:

  s = new String(“This is the text”) ;

- Sau intr-o linie

  String s = new String (“This is the text.”) ;

Caracteristici ale clasei String

- Pot fi create si astfel:

  String s = “This is the text.” ;

- Obiectele String nu pot fi modificare dupa creare.
- Operatorul + folosit pentru concatenarea a doua stringuri
- Pentru a cauta intr-un string un substring
  - int p = s.indexOf(“the”);
  - s.contains(substring);
- Pentru a compara doua stringuri
  - if(s.equals(“This is the text”)

    System.out.println(“texte identice”);

    else

    System.out.println(“texte diferite”);

Tipul Array (vector)

- Un Array este o colectie de elemente de acelasi tip:

  int states[] ; // declaratie

  si:

  states = new int[128] ; // creatie

- Sau:

  int states[] = new int[128] ;

- Sau:

  int states[] = new int[128] ;

- Se poate declara un vector si astfell:

  int[] tab= {1, 3, 5, 2, 34};

- Lungimea unui array nu poate fi schimbata
- Indexul incepe la 0
- Java verifica si genereaza eroare daca se incearca acesare index inexistent
- Lungimea vectorului este preluata de *states.length*

Array de obiecte

- Poate fi creat vector de obiecte, ex:

```
Color manycolors[] = new Color[1024];
```

- Creaza un vector de referinte. Obiectele efective nu sunt construite.
- Obiectele trebuie construite manual:

```
for (int i = 0 ; i < 1024 ; i++)
    manycolors [i] = new Color() ;
```

Array cu mai multe dimensiuni

- *Moduri de declarare*:

```
int graph[][] = new int[2][];
graph[0] = new int[4];      // Row 0 has length 4
graph[1] = new int[7];      // Row 1 has length 7
...
graph[1][1] = 9;
```

- O matrice bidimensionala de caractere:

```
char icon[][] = new char [16][16];   // 16 by 16 array
```

  - Pot fi creati vectori asimetrici
    icon [8] = new char [17] ;

Operatori

=	Atribuire	a = b
>	Mai mare decât	a > b
<	Mai mic decât	a < b
<=	Mai mic sau egal cu	a <= b
>=	Mai mare sau egal cu	a >= b
==	Egal cu	a == b
!=	Nu este egal cu	a != b
!	Negaţie logică	!a
~	Negaţie logică pe biţi	~a
?:	Operatori condiţional	a ? expr1 : expr2
&&	ŞI (conditional AND)	a && b
||	SAU (conditional OR)	a || b

Operatori

++	Incrementare	a++ or ++a
--	Decrementare	a-- or --a
+	Adunare	a + b
-	Scădere	a - b or -b
*	Multiplicare	a * b
/	Împărţire	a / b
%	Modulo	a % b
&	ŞI pe biţi (bitwise AND)	a & b
|	SAU pe biţi (bitwise OR)	a | b
^	SAU EXCLUSIV pe biţi (XOR)	a ^ b
<<	Deplasare stânga	a << b
>>	Deplasare dreapta	a >> b
>>>	Deplasare dreapta cu umplere cu zero	a >>> b
+=	Atribuie rezultatul adunării	a += b

Operatori

-=	Atribuie rezultatul scăderii	a -= b
*=	Atribuie rezultatul înmulţirii	a *= b
/=	Atribuie rezultatul împărţirii	a /= b
&=	Atribuie rezultatul lui ŞI pe biţi	a &= b
|=	Atribuie rezultatul lui SAU pe biţi	a |= b
^=	Atribuie rezultatul lui SAU EXCLUSIV pe biţi	a ^= b
%=	Atribuie rezultatul lui modulo	a %= b
<<=	Atribuie rezultatul deplasării la stânga	a <<= b
>>=	Atribuie rezultatul deplasării la dreapta	a >>= b
>>>=	Atribuie rezultatul deplasării la dreapta cu umplere cu zero	a >>>= b

Operatorul “+” pentru String

```
String s1=”123”;
String s2=”abc”;
String s3;
s3=s1+s2;
System.out.println(s3);
```

Controlul fluxurilor: **if-else**

- Executie conditionala:

```
if (some Boolean expression) {
    statements to be executed if true
}
```

- Optional clauza **else**:

```
if (some Boolean expression) {
    statements to be executed if true
} else {
    statements to be executed if false
}
```

- Expresii imbricate:

```
if (some Boolean expression) { . . . }
else if (another Boolean expression) { . . . }
else { . . . }
```

Controlul fluxurilor: **bucla while**

- bucla **while** normala:

```
while (any Boolean) {
    Stuff to do
}
```

Examplu:

```
int i = 0 ;
while(i < a.length) {
    a [i] = i * i ;
    i++ ;
}
```

- **while** cu test la sfarsit:

```
do {
    What to do
 } while (another Boolean) ;
```

Controlul fluxurilor: **bucla for**

- Bucla for similara cu cea din C++:

```
for (declaration1 ; booleanExpression ; expressionList2) {
    Statements to do
}
```

- Exemplu:

```
for (int i = 0 ; i < a.length ; i++)
    a [i] = i * i ;

for (int x: a)
    System.out.println(x) ;
```

Controlul fluxurilor: **switch**

- Identic cu cel din C:

```
switch (expression) {
case Constant1:     // Do following if expression==Constant1
    Bunch of Stuff
    break;
case Constant2:     // Do following if expression==Constant2
    Bunch of Stuff
    break;
default:            // Do the following otherwise
    Bunch of Stuff
    break;
}
```

Incepand cu Java 7 pot fi utilizate String-uri in expresiile switch.

Controul fluxurilor: **break** si **continue**

- **break** determina parasirea fluxului **switch**, **while**, **do** sau **for**:

```
while (true)
    if (++i == a.length || a[i] == v) break ;
```

- **continue** trece la urmatoarea iteratie a fluxului **while**, **do** or **for**.

Sfarsit