Florida UNIVERSITÀRIA

Guia de Referència Ràpida de Java

Última Actualització: dilluns, octubre 6, 2014

Operadors Aritmètics

+	Suma
-	Resta
/	Divisió (int / float)
	2/3 = 0, 2.0/3.0 =.666667
*	Multiplicació
%	Mòdul (Rest de la divisió entera)

Operadors Relacionals/Igualtat

<	Menor que
<=	Menor o igual que
>	Major que
>=	Major o igual que
==	Igual a
!=	No igual / Distint que

Operadors Lògics

!	NO
&&	I (AND)
||	O (OR)

Operadors d'Assignació

=	assignació
+=	incrementa i assigna
-=	resta i assigna
*=	multiplica i assigna
/=	divideix i assigna
%=	rest de la divisió i assigna

Recordeu utilitzar els mètodes **equals( )** o **compareTo( )** quan compareu Strings en compte de l'operador de comparació.

String s1 = "abc", s2 = "def";

Comparació d'String:

Comparar si son iguals:

- s1.equals(s2) ó
- s1.compareTo(s2) == 0

Recordeu que el mètode compareTo( ) torna un d'aquests 3 valors:

- nombre negatiu, nombre positiu, 0

Comparar lexicogràficament:

- s1.compareTo(s2) < 0 (s1 abans s2)
- s1.compareTo(s2) > 0 (s1 després s2)

Recordeu distingir entre nombres enters i nombres reals (anomenats de punt flotant en Java). Aquestos s'emmagatzenen a la memòria de manera diferent i tenen diferents rangs de valors.

- enter: 2, 3, -5, 0, 8
- punt-flotant: 2.0, 0.5, -3., 4.653

Increment ++ /Decrement - - operadors utilitzats en mode prefix i posfix

++/--	mode prefix	- primer increm./decrem. la variable i després la utilitza
++/--	mode posfix	- primer utilitza la variable i després la increm./decrem.

Creació d'Objectes: (new) new int[10], new Alumne("Manel")

L'operador **new** crea un objecte i torna una referència a d'ell (adreça d'un objecte)

Tipus en Java [valor/referència]

Un tipus primitiu emmagatzema el valor d'una dada primitiva **int x = 3;**

Un tipus de referència emmag. l'adreça d'un objecte **Cercle c = new Cercle(2);**

Una variable de referència es crea utilitzant el nom de la classe: **Alumne unNouAlumne;**

Tipus de Dades Primitius (Java) Recordeu: **String** és un tipus de referència

boolean	flag / logic	true, **false**	[**boolean** literals]
char	character	'A', 'n', '!'	[**char** literals]
byte, short, **int**, long	integral	2, 3, 5000, 0	[**int** literals]
float, **double**	punt-flotant	123.456, .93	[**double** literals]

Default numeric literal types:

integral:	**int**	int x = 3;	//3 és un int literal
punt-flotant:	**double**	double y = 2.5;	//2.5 és un double literal

La construcció switch/case (**break** i **default** són opcionals)

Forma:

```
switch (expressió)
{
  case int-constant :
    sentència (es);
  [ break; ]

  case int-constant :
    sentència (es);
  [ break; ]

  [ default :
    sentència; ]

}
```

Exemple:

```
switch (elecció)
{
  case 0 :
     System.out.println( "Has triat 0." );
     break;

  case 1:
     System.out.println( "Has triat 1." );
     break;

  default :
     System.out.println(
          "No has triat ni 0 ni 1." );
}
```

L' "*expressió* " i la "*int-constant* " normalment són de tipus **int** o **char**. Java 7 permet utilitzar un String. switch(cadena) { case "bo": … }

Utilitza la paraula clau **break** per eixir de l'estructura (evitar arribar al "default"). Utilitza la paraula clau **default** per proporcionar un cas per defecte si cap dels casos anteriors no coincideixen (similar a la d'eixida "si no" en una sentència if-else-if).

Formes de la Sentència if

Simple if

```
if (expressió)
    sentència;
```

Exemple

```
if (x < y)
    x++;
```

if/else

```
if (expressió)
    sentència;
else
    sentència;
```

Exemple

```
if (x < y)
    x++;
else
    x--;
```

if/else if (nested if)

```
if (expressió)
    sentència;
else
    if (expressió)
        sentència;
    else
        sentència;
```

Exemple

```
if (x < y)
    x++;
else
    if (x < z)
        x--;
    else
        y++;
```

L' **"expressió"** entre parèntesis per a una **sentència if** o per a un **bucle** també s'anomena sovint com una **"condició"**

Per executar condicionalment més d'una sentència, s'ha de crear una **sentència composta** (bloc) adjuntant les declaracions entre claus (això també es compleix per als bucles):

Forma

```
if (expressió)
{
   sentència;
   sentència;
}
```

Exemple

```
if (x < y)
{
   x++;
   System.out.println( x );
}
```

Entrada utilitzant la classe Scanner

Scanner entrada = new Scanner (System.in); //entrada teclat

Mètodes d'entrada: next(), nextLine(), nextInt(), nextDouble()

Mètodes d'Eixida per a objectes System.out o PrintWriter

print(), println(), printf() [eixida amb format]

Entrada/ Eixida amb la classe JOptionPane [paquet javax.swing]

```
String nString; int num;

nString = JOptionPane.showInputDialog("Introd. un nombre ");
num = Integer.parseInt(nString);

JOptionPane.showMessageDialog(null, "Has introd. " + num);
```

Conversió des d'un String a un nombre utilitzant Classes Wrapper

```
double d = Double.parseDouble(dString);
float  f = Float.parseFloat(fString);
int    j = Integer.parseInt(jString);
```

Eixida amb format de Java [mètodes **printf( )** i **String.format( )**]

3 components: *format, string* i opcionalment: especificadors de format (fs) amb una llista de paràmetres (al)

- **fs**: " ... % [flags] [width] [precisió] especificador de format ... "
- **al**: llista d'expressions separades per comes.

Especificadors de Format: s (string), d (integer), f (punt-flotant)

Exemple: `System.out.printf("Total és %,10.2f\n", total);`

Conversions Numèriques Java I conversions Forçoses:

Automàtiques es realitzen implícitament.

```
double x;    int y = 100;
x = y;       // El valor de y es converteix implícitament a double.
```

Forçoses es fan explícitament mitjançant un Càsting.

```
double x = 100;    int y;
y = (int) x;    // El valor de x explícitament es converteix a un int
```

En expressions mixtes, la conversió numèrica succeeix implícitam. **double** és el tipus de dades primitiu "més alt", **byte** el "més baix".

Florida UNIVERSITÀRIA

Guia de Referència Ràpida de Java

Última Actualització: dilluns, octubre 6, 2014

El bucle while (bucle pre-test)

Forma:

```
inicialització;
while (test)
{
   sentència;
   actualització;
}
```

Exemple:

```
x = 0;
while (x < 10)
{
   sum += x;
   x++;
}
```

El bucle do-while (bucle post-test)

Forma:

```
inici;
do
{
   sentència;
   actualització;
} while (test);
```

Exemple:

```
x = 0;
do
{
   sum += x;
   x++;
} while (x < 10);
```

El bucle for (bucle pre-test)

Forma:

```
for (ini.; test; actual.)
{
    sentència;
}
```

Exemple:

```
for (int count=1; count<=10; count++)
{
    System.out.println( count );
}
```

bucle for-each:

```
for (parametre : collection)
    sentència;
```

```
int valors[ ] = {85, 92, 76, 66, 94};   //collection és un array de valors
for ( int valor : valors )              //valor és la variable que pren cadascun dels
   System.out.println(valor);           // elements de l'array valors
```

Seqüències d'Escapament

Caràcters Especials en Java

\n	caràcter de nova línia	'\n'
\t	caràcter tabulació	'\t'
\"	doble cometa	'\"'
\'	cometa simple	'\''
\\	barra Invertida	'\\'

Precedència d'Operadors

```
    (  )
----------
 *, /, %      [ matemàtics]
----------
   +, -
```

Operadors Lògics: `!, &&, ||`

(1) matemàtics (2) relacionals (3) lògics

Selecció i Estructures de Bucles

Selecció:

- Unaris o de selecció única
- Binaris o de selecció dual
- Estructura Case quan una variable pren alguns possibles valors
- Selecció Única
  - Una única condició
- Selecció composta
  - Multiples condicions unides pels operadors **i/o**

Bucles:

- Bucles **Pre-test** Java
- El test precedeix el cos
  - while
  - for
- Bucles **Post-test** Java
- El test segueix el cos
  - do-while

Control de bucles:

- Hi han 3 tipus d'expressions:
  - inicialització (inici)
  - test
  - actualització
- Bucles controlats per comptador, sabem el nombre exacte de vegades que s'ha d'executar
- Bucles controlats per sentinella, es repetiran fins que es complisca una condició
- Eixida abans que acabe:
  - Sentència **break**

Nota: **break** també es pot utilitzar en la sentència **switch**.

Arrays Java: Crear un array (2 maneres)

```
1. <type> <array-name>[ ] = new <type>[size];
2. <type> <array-name>[ ] = { <llista-inicialització> };
```

```
//crea un array de 20 elements.
       int    elMeuArray[ ] = new int[20];

//crea un array de 3 elements i l'inicialitza amb els valors de la llista.
       int    elMeuArray [ ] = { 1, 2, 3 };
       String personatges[ ] = { "Moe", "Bart", "Hommer" };

//assigna el valor del primer element de l'array a la variable entera x.
       int x = elMeuArray [0];

// assigna el valor de l'últim element de l'array a la variable entera y.
       int y = elMeuArray [elMeuArray.length-1 ];
```

Amb la classe ArrayList per a crear a dinàmicament arrays redimensionables.

La classe Arrays conté mètodes stàtics que poden utilitzar-se amb ArrayLists i arrays per a cercar, copiar, ordenar, comparar, etc.

int num[ ]; … <stmts> ….

Crear un nou array, inicialitzar-lo i assigat a num. **num = new int[ ]{1,2,3,4,5};**

Tots els arrays tenen un camp públic anomenat **`length`** que conté el nombre d'elements de l'array.

Amb aquesta declaració: `int x[][][];`

`x.length`	Indica el nombre d'elements de l'array en la primera dimensió.
`x[m].length`	Indica el nombre d'elements d'un array concret en la segona dimensió.
`x[m][n].length`	Indica el nombre d'elements d'un array concret en la tercera dimensió.

Mètodes Java: <tipus> <nom-mètode> ([<tipus> parametre1, [<tipus> parametre2, …]])

Els mètodes que no retornen cap valor tindran el tipus de retorn **void** en la signatura.

```
void printHeadings( )    //sense paràmetres, el tipus retornat és void
{ <cos del mètode> }

void printDetailLine( String nom, int nombre, double gpa )   //3 paràmetres, el tipus retornat és void
{ < cos del mètode> }

int getCount( )    // sense paràmetres, el tipus retornat és int
{ < cos del mètode> }

double max( double x, double y )   // paràmetres, el tipus retornat és double
{ < cos del mètode> }
```

Quan es crida a un mètode, les dades es passen als paràmetres (si n'hi ha) mitjançant arguments

//Arguments: "Jack Wilson", 100, 3.50 pasats com a Paràmetres: nom, nombre, gpa al Mètode: printDetailLine (mireu la capçalera més amunt) : **`printDetailLine( "Jack Wilson", 100, 3.50);`**

Un mètode pot ser declarat amb un paràmetre de longitud variable. Ha de ser l'últim paràmetre declarat. La sintaxi de la declaració és <tipus> **`...`** <nom-paràmetre>.

Exemples: `int... nombres,  double ... valors, String ...noms`