Programación Orientada a Objetos en Java

Curso 2006 - 2007

Tema 2 – Sintaxis Básica

Gonzalo Méndez Pozo
Dpto. de Ingeniería de Software e Inteligencia Artificial
Universidad Complutense de Madrid

Comentarios

- Existen 3 tipos de comentarios
  - De línea: //comentario de línea
  - De bloque: /* comentario
    de bloque */
  - De javadoc: /** @author Gonzalo Méndez
    @version 1.0
    comentario de javadoc */

Identificadores

- Permiten nombrar paquetes, clases, interfaces, variables, objetos
- Comienzan con una letra, _ o $
- Seguidos por letras o dígitos, y pueden incluir ñ y vocales acentuadas
- Cualquier longitud
- Se distinguen mayúsculas y minúsculas
- x, _var1, año, $nuevo_carácter

Identificadores

- Por convenio:
  - Las clases comienzan con mayúscula: Hola
  - Los métodos y atributos comienzan con minúscula: leer, altura
    - los atributos pueden comenzar con _: _altura
  - Las constantes van en mayúsculas: PI
  - En identificadores formados por varias palabras, la primera letra de cada palabra en mayúsculas: HolaMundo, getWidth

Palabras Reservadas

abstract	continue	for	new	switch	volatile
boolean	default	goto	null	synchronized	while
break	do	if	package	this	
byte	double	implements	private	threadsafe	
byvalue	else	import	protected	throw	
case	extends	instanceof	public	throws	
catch	false	int	return	transient	
char	final	interface	short	true	
class	finally	long	static	try	
const	float	native	super	void	

Otras palabras reservadas (sin uso actual):

cast	future	generic	inner	operator
outer	rest	var		

Separadores

- ()
  - Para delimitar listas de parámetros o de argumentos.
  - Para modificar la precedencia en una expresión.
  - Para delimitar condiciones.
  - Para realizar conversiones de tipo (moldes o casting).
- {}
  - Para definir bloques de código.
  - Para delimitar listas de valores iniciales de los vectores (arrays).
- []
  - Para declarar vectores y referenciar elementos de los mismos.
- ;
  - Para terminar instrucciones.
- ,
  - Para separar identificadores en declaraciones.
  - Para encadenar expresiones (operador coma).
- .
  - Para acceder a atributos de objetos u objetos globales de clases.
  - Para pasar mensajes a objetos.
  - Para acceder a un subpaquete de un paquete.

Tipos de Datos Numéricos

Tipo	Tamaño	Valor mínimo	Valor máximo
byte	8 bits	-128	127
short	16 bits	-32768	32767
int	32 bits	-2147483648	2147483647
long	64 bits	$< -9 \times 10^{18}$	$> 9 \times 10^{18}$
float	32 bits	+/- 3.4×10^{38} con 7 dígitos significativos	
double	64 bits	+/- 1.7×10^{38} con 15 dígitos significativos	

- Un tipo A es de mayor rango que un tipo B si A es superconjunto de B.
- Las variables de tipo B siempre se pueden asignar a las de tipo A
- **double > float > long > int > short > byte**

Tipos de Datos no Numéricos

- boolean
  - true y false
- char: caracteres (16 bits)
  - ‘a’, ‘7’
  - Caracteres de escape: ‘\n’
  - OJO: 7 ≠ ‘7’


Variables

- Espacios de memoria donde se almacena un número, un carácter, un objeto…
- En los lenguajes tipados como Java, a las variables se les asigna un tipo
- El tipo determina los valores que puede tener la variable, además de las operaciones que se le pueden aplicar


Declaración de Variables

- La declaración de una variable es
  - tipoVariable nombreVariable [=valorInicial];
    - int ancho;
    - int x = 3;
- Otras posibilidades
  - Juntar la declaración de varias variables del mismo tipo:
    - int altura,x;
    - int a, x=3;
    - int edad=25, altura=173;

Constantes

- Son variables cuyo valor no se puede modificar
- Se declaran como una variable, añadiéndoles el modificador final
  - final float PI = 3.14159;
  - final char SALTO_LINEA = '\n';
  - final short MAX_SHORT = 32767;

Ámbito de los Identificadores

- { … } define un bloque de código
- Los bloques pueden anidarse
- Un identificador puede usarse dentro del bloque de código en el que está definido, y en los bloques anidados

Operador de Asignacion

- Para darle valor a una variable
  - a = 2;
  - b = 3+5;
  - c = a+b;
- Asignación múltiple
  - A = b = 5;

Operadores Aritméticos

- Unarios: + - ++ --
  - Preincremento: se incrementa y y se asigna el valor a x → x = ++y;
  - Postincremento: se asigna el valor de y a x y luego se incrementa y → x = y++;
- Binarios: + - * / %
- Abreviados: += -= *= /= %=
  - x += 3; equivale a x = x+3;


Otros Operadores

- Relacionales: == != < <= > >=
- Lógicos (evaluación perezosa):
  - AND: && o &para evaluación impaciente
  - OR: || o | para evaluación impaciente
  - NOT: !
  - XOR: ^


Conversión de Tipos

- La conversión a tipos mayores es automática
- La conversión a tipos menores hay que hacerla explícita, indicando entre paréntesis el tipo deseado → cast
  - int x; float f = 5.3; x = (int)f;

Precedencia de Operadores

Mayor cast
++ -- (prefijos)
!
*** / %**
+ -
< <= > >=
== !=
&
^
|
&&
||
Menor **= += -= *= /= %=**

- Se puede modificar la precedencia de los operadores agrupando expresiones con paréntesis
  - x = 25 + 4 * 3 // x = 37
  - x = (25 + 4) * 3 // x = 87

Matemáticas con Math

- Math es una clase que proporciona operaciones matemáticas más complejas

Math.abs(*num*)	**Math.max(*a*,*b*)**	**Math.min(*a*,*b*)**
Math.asin(num)	**Math.acos(*num*)**	**Math.atan(*num*)**
Math.sin(num)	**Math.cos(*num*)**	**Math.tan(*num*)**
Math.exp(*num*)	**Math.log(*num*)**	**Math.pow(*a*,*b*)**
Math.sqrt(*num*)	**Math.random()**	**Math.round(*num*)**
Math.toDegrees(*num*)	**Math.toRadians(*num*)**	


Tipos Enumerados

- Disponibles desde la versión 5 de java

```
public class EjEnumerados
{
    public enum Estación {PRIMAVERA, VERANO, OTOÑO,
    INVIERNO};

    public static void main(String[] args)
    {
            Estación mia;
            mia = Estación.PRIMAVERA;
            System.out.println("Mi estación preferida es: "+mia);
    }
}
```



Arrays

- Se pueden declarar de varias formas:
  - int[] primeros_primos;
  - int primeros_primos[];
- Hay que asignarles un tamaño:
  - primeros_primos = new int[10];
- Se pueden inicializar al declararlos:
  - int[] primeros_primos={1,2,3,5,7,11,13,17,19,23};
- Se les puede preguntar su longitud
  - int longitud = primeros_primos.length;


Arrays

- Acceder a un elemento del array:

```
int primer_primo=primeros_primos[0];
```

- Otra forma de inicializar un array:

```
import java.util.Arrays;
Arrays.fill(primeros_primos,0);
```

- Asignación de arrays:
  - Se copia una referencia, no el array entero

```
otros_primos=primeros_primos;
```



Arrays

- Arrays de arrays:
  - Funcionan igual que los anteriores:

```
int matriz[][] = new int[4][4];
```

  - También la inicialización:

```
int matriz[][] = {{1,2,3},{4,5,6},{7,8,9}};
```

  - Pueden tener distinta longitud:

```
float ejemplo[][] = new float[2][];
ejemplo[0] = new float[9];
ejemplo[1] = new float[373];
```

  - Pueden tener más de dos dimensiones

Sentencias de Control de Flujo

- Sirven para variar la ejecución del programa en función de una determinada condición
- Selección condicional
  - if-else, switch
- Bucles
  - for, while, do-while
- Paso de control
  - break, continue

Bifurcación Condicional: IF

- **if(*condición*) *instrucción*;**
- **if(*condición*) *instrucción-if*;**
  else *instrucción-else*;
- **if(*condición*)**
  {
  **	*secuencia-de-instrucciones*;**
  }
- **if(*condición*)**
  {
  **	*secuencia-de-instrucciones-if*;**
  }
  else
  {
  **	*secuencia-de-instrucciones-else*;**
  }
- CUIDADO
  - a==5 //comparacion, true o false
  - a=5 //asignacion, true
  - 5==a //comparacion, true o false
  - 5=a //ERROR

- if (a>b) max=a;
- if (a>b) max=a;
  else max=b;
- if (a>b)
  {
  max=a;
  System.out.println (“Max = “ +max);
  }
- if (a>b)
  {
  max=a;
  System.out.println (“Max = “ +max);
  }
  else
  {
  max=b;
  System.out.println (“Max = “ +max);
  }

if-else Anidados

```
if (condicion)
{
   if (otra_condicion)
          instrucción;
}
else
{
   if (otra_mas)
          instrucción;
   else
   {
          instrucciones;
   }
}
```

```
if (es_redondo==true)
{
   if (radio>7)
       radio--;
}
else // es cuadrado
{
   if (ladoX != ladoY)
       es_rectangulo=true;
   else
   {
        es_rectangulo=false;
        es_cuadrado=true;
   }
}
```


if con Varias Ramas

- En realidad son if anidados
- Cada sentencia if es una única instrucción

```
if (condicion1)
    instrucción1;
else if (condicion2)
    instrucción2;
 else if (condicion3)
    instrucción3;
else
    instrucción4;
```

El Operador Condicional

- Es un sustituto del if para elecciones simples

```
Variable=condicion?expresion_si_true:expresion_si_false;

max=(a>b)?a:b;
```


Selección Condicional con Varias Ramas

- Como se ha visto antes, la sentencia if se puede usar para elegir la ejecución entre más de dos opciones
- Si la elección se puede realizar en función del valor que tome una variable o del resultado de evaluar una expresión, entonces es más adecuado utilizar la sentencia switch


switch

```
switch (expresion)
{
case a: instrucciones;
    break;
case b: instrucciones;
    break;
default: instrucciones;
}
```

```
switch (num_mes)
{
case 1: mes="Enero";
    break
case 2: mes="Febrero";
    break;
…
default:
    System.out.println("error");
    System.exit(-1);
}
```



switch

- La expresión que se evalúa tiene que ser un tipo simple
  - boolean, char, int
- Si no se pone la sentencia break al final de cada case también se ejecutan las instrucciones del siguiente case
  - Útil para agrupar opciones en las que se hace lo mismo


Bucle for

- Su uso está indicado cuando se quiere repetir una acción un número fijo de veces

```
for (inicialización;condición;incremento)
    instrucción;

for (inicialización;condición;incremento)
{
    instrucciones;
}
```



Bucle for

```
for (int i=0;i<10;i++)
    System.out.println(i);
```

- La condición del bucle es la condición de permanencia en el bucle, no la de finalización del bucle.
- Se puede poner más de una expresión en cada una de las tres partes del for, separadas por comas.


Bucle for

- Existe una variante para recorrer arrays y colecciones:

```
for (tipo iterador : array)
{
    //expresiones
}

int primeros_primos[]={1,2,3,5,7,11,13,17,19,23};
for (int elemento : primeros_primos)
{
    System.out.println(elemento);
}
```



Bucle while

- Este bucle no tiene predeterminado un número fijo de iteraciones
- La duración del bucle viene determinada por la evaluación de una condición. Mientras la condición sea cierta se sigue ejecutando el bucle.
- Cuidado con los bucles infinitos (la condición nunca se hace falsa)

Bucle while

```
while (condición)
    instrucción;

i=0;
while (i<257)
    i++;

i=0;
while ((i++)<257)
     System.out.println(i);
```

```
while (condición)
{
    instrucciones;
}

i=0;
while (i<257)
{
    i++;
    System.out.println(i);
}
```


Bucle do-while

- Es igual que while, pero la condición se comprueba al final, por lo que las instrucciones del bucle se ejecutan al menos una vez

```
do
{
    instrucciones;
}
while (condición);  // no olvidar el punto y coma
```


break

- Sirve para finalizar las ramas case dentro de un switch
- También se usa para salir de un bucle while o do-while
- Este último uso no es muy limpio. Hay formas más elegantes de salir de un bucle (y que funcionan en todos los lenguajes de programación)


continue

- Sirve para terminar la iteración actual de un bucle sin ejecutar todas las instrucciones
- No fuerza la salida del bucle, sino que se ejecuta la siguiente iteración
- Su uso tampoco es demasiado elegante