APÉNDICE

B Tipos de dato en Java

Java reconoce dos categorías de tipos: tipos primitivos y tipos objeto. Los tipos primitivos se almacenan directamente en las variables y tienen valores semánticos (se copian los valores cuando se asignan a otra variable). Los tipos objeto se almacenan mediante referencias al objeto (no se almacena el objeto propiamente dicho); cuando se asignan a otra variable sólo se copia la referencia, no el objeto.

B.1 Tipos primitivos

En la siguiente tabla se listan todos los tipos primitivos del lenguaje Java:

Nombre del tipo	Descripción	Ejemplos de literales
Números enteros		
`byte`	entero de 1 byte de tamaño (8 bit)	`24` `-2`
`short`	entero corto (16 bit)	`137` `-119`
`int`	entero (32 bit)	`5409` `-2003`
`long`	entero largo (64 bit)	`423266353L` `55L`
Números reales		
`float`	punto flotante de simple precisión	`43.889F`
`double`	punto flotante de doble precisión	`45.63` `2.4e5`
Otros tipos		
`char`	un solo carácter (16 bit)	`'m'` `'?'` `'\u00F6'`
`boolean`	un valor lógico (verdadero o falso)	`true` `false`

Notas:

- Un número que no contiene un punto decimal se interpreta generalmente como un `int`, pero se convierte automáticamente a los tipos `short`, `byte` o `long` cuando se le asigna (si el valor encaja). Se puede declarar un literal como `long` añadiendo una 'L' al final del número (también se puede utilizar la letra 'l' (L minúscula) pero debería evitarse ya que se puede confundir fácilmente con el uno).
- Un número con un punto decimal se considera de tipo `double`. Se puede especificar un literal como un `float` añadiendo una 'F' o 'f' al final del número.
- Un carácter se puede escribir como un carácter Unicode encerrándolo entre comillas simples o como un valor Unicode de cuatro dígitos precedidos por '`\u`'.
- Los dos literales booleanos son `true` y `false`.

Debido a que las variables de tipos primitivos no hacen referencia a objetos, no existen métodos asociados con los tipos primitivos. Sin embargo, cuando se usa un tipo primitivo en un contexto que requiere un tipo objeto se puede usar el proceso de *autoboxing* para convertir un valor primitivo en su correspondiente objeto. Para más detalles, recurra a la Sección B.3.

La siguiente tabla detalla los valores mínimo y máximo disponibles para los tipos numéricos.

Tipo	Mínimo	Máximo
byte	-128	127
short	-32768	32767
int	-2147483648	2147483647
long	-9223372036854775808	9223372036854775807
	Mínimo positivo	**Máximo positivo**
float	1.4e-45	3.4028235e38
double	4.9e-324	1.7976931348623157e308

B.2 Tipos objeto

Todos los tipos que no aparecen en la sección *Tipos primitivos* son tipos objeto. Esto incluye los tipos clase e interface de la biblioteca estándar de Java (como por ejemplo, `String`) y los tipos definidos por el usuario.

Una variable de tipo objeto contiene una referencia (o un «puntero») a un objeto. Las asignaciones y los pasajes de parámetros utilizan referencias semánticas (es decir, se copia la referencia, no el objeto). Después de asignar una variable a otra, ambas variables hacen referencia al mismo objeto. Se dice que las dos variables son alias del mismo objeto.

Las clases son las plantillas de los objetos: definen los campos y los métodos que poseerá cada instancia.

Los arreglos (*arrays*) se comportan como tipos objeto; también utilizan referencias semánticas.

B.3 Clases «envoltorio»

En Java, cada tipo primitivo tiene su correspondiente clase «envoltorio» que representa el mismo tipo pero que en realidad, es un tipo objeto. Estas clases hacen posible que se usen valores de tipos primitivos en los lugares en que se requieren tipos objeto mediante un proceso conocido como *autoboxing*. La siguiente tabla enumera los tipos primitivos y sus correspondientes clases envoltorio del paquete `java.lang`. Excepto `Integer` y `Character`, los nombres de las clases envoltorio coinciden con los nombres de los tipos primitivos, pero con su primera letra en mayúscula.

Tipo primitivo	Tipo envoltorio
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Siempre que se use un valor de un tipo primitivo en un contexto que requiera un tipo objeto, el compilador utiliza la propiedad de *autoboxing* para encapsular automáticamente al valor de tipo primitivo en un objeto envoltorio equivalente. Esto quiere decir, por ejemplo, que los valores de tipos primitivos se pueden agregar directamente en una colección. La operación inversa (*autounboxing*) también se lleva a cabo automáticamente cuando se utiliza un objeto envoltorio en un contexto que requiere un valor del tipo primitivo correspondiente.