CURSO DE PROGRAMACIÓN FULL STACK

APÉNDICE A

TIPOS DE DATOS Y OPERADORES EN JAVA

TIPOS DE DATOS PRIMITIVOS

Byte	Es un entero con signo de 8 bits, el mínimo valor que se puede almacenar es -128 y el máximo valor es de 127 (inclusive).
Short	Es un entero con signo de 16 bits. El valor mínimo es -32,768 y el valor máximo 32,767 (inclusive).
Integer	Es un entero con signo de 32 bits. El valor mínimo es -2,147,483,648 y el valor máximo es 2,147,483,64 (inclusive). Generalmente es la opción por defecto.
Long	Es un entero con signo de 64 bits, el valor mínimo que puede almacenar este tipo de dato es -9,223,372,036,854,775,808 y el máximo valor es 9,223,372,036,854,775,807 (inclusive).
Float	Es un número decimal de precisión simple de 32 bits (IEEE 754 Punto Flotante).
Double	Es un número decimal de precisión doble de 64 bits (IEEE 754 Punto Flotante).
Boolean	Este tipo de dato sólo soporta dos posibles valores: verdadero o falso y el dato es representado con tan solo un bit de información.
Character	El tipo de dato carácter es un simple carácter unicode de 16 bits. Su valor mínimo es de '\u0000' (En entero es 0) y su valor máximo es de '\uffff' (En entero es 65,535). Nota: un dato de tipo carácter se puede escribir entre comillas simples, por ejemplo 'a', o también indicando su valor Unicode, por ejemplo '\u0061'.
String	Además de los tipos de datos primitivos el lenguaje de programación Java provee también un soporte especial para cadena de caracteres a través de la clase String.

Encerrando la cadena de caracteres con comillas dobles se creará de manera automática una nueva instancia de un objeto tipo String.

String cadena = “Sebastián”;

Los objetos String son inmutables, esto significa que una vez creados, sus valores no pueden ser cambiados. Si bien esta clase no es técnicamente un tipo de dato primitivo, el lenguaje le da un soporte especial y hace parecer como si lo fuera.

VALORES POR DEFECTO

En Java no siempre es necesario asignar valores cuando nuevos atributos son declarados. Cuando los atributos son declarados, pero no inicializados, el compilador les asignará un valor por defecto. A grandes rasgos el valor por defecto será cero o null dependiendo del tipo de dato. La siguiente tabla resume los valores por defecto dependiendo del tipo de dato.

byte	0
short	0
int	0
long	0
float	0.0
double	0.0
boolean	false

char	'\u0000'
String	null
Objetos	null

Las variables locales son ligeramente diferentes; el compilador no asigna un valor predeterminado a una variable local no inicializada. Las variables locales son aquellas que se declaran dentro de un método. Si una variable local no se inicializa al momento de declararla, se debe asignar un valor antes de intentar usarla. El acceso a una variable local no inicializada dará lugar a un error en tiempo de compilación.

OPERADORES

Los operadores son símbolos especiales de la plataforma que permiten especificar operaciones en uno, dos o tres operandos y retornar un resultado. También aprenderemos qué operadores poseen mayor orden de precedencia. Los operadores con mayor orden de precedencia se evalúan siempre primero. Primeramente, proceden los operadores unarios, luego los aritméticos, después los de bits, posteriormente los relacionales, detrás vienen los booleanos y por último el operador de asignación. La regla de precedencia establece que los operadores de mayor nivel se ejecuten primero. Cuando dos operadores poseen el mismo nivel de prioridad los mismos se evalúan de izquierda a derecha.

OPERADOR DE ASIGNACIÓN

=	Operador de Asignación Simple

OPERADORES ARITMÉTICOS

+	Operador de Suma

-	Operador de Resta
*	Operador de Multiplicación
/	Operador de División
%	Operador de Módulo

OPERADORES UNARIOS

+	Operador Unario de Suma; indica que el valor es positivo.
-	Operador Unario de Resta; indica que el valor es negativo.
++	Operador de Incremento.
--	Operador de Decremento.
!	Operador Lógico de Negación.

Operadores de Igualdad y Relación

==	Igual
!=	Distinto
>	Mayor que
>=	Mayor o igual que

<	Menor que
<=	Menor o igual que

Operadores Condicionales

&&	AND
ll	OR
? :	Ternario

OPERADOR DE COMPARACIÓN DE TIPO

instanceof	Compara un objeto con un tipo específico

OPERADORES DE BITS

~	Complemento de bit a bit
<<	Desplazamiento a la izquierda con signo
>>	Desplazamiento a la derecha con signo
>>>	Desplazamiento a la derecha sin signo
&	Complemento AND
^	Complemento OR Exclusiva
|	Complemento OR Inclusiva

CONVERSIÓN DE TIPOS

Muchas veces es necesario realizar conversiones de tipos cuando se evalúa una expresión aritmética. Las conversiones de tipos pueden ser apropiadas para cambiar el tipo de una determinada expresión que no se corresponda con el tipo necesario.

```
TipoResultante variable = (TipoResultante) variableDeTipoAConvertir
```

Conversión por ampliación de tipos de datos simples

En estos casos no se pierde información sobre la magnitud de los valores numéricos.

Tipo a convertir	Tipo resultante
byte	short int long float double
short	int long float double
char	int long float double
int	long float double
long	float double
float	double

Los únicos casos en los que no se preserva necesariamente la exactitud entre los valores del tipo a convertir y el tipo resultante, ya que **puede perderse precisión** son:

Tipo a convertir	Tipo resultante
int	float
long	float double

CONVERSIÓN POR REDUCCIÓN DE TIPOS DE DATOS SIMPLES

En estos casos puede perderse información ya que los rangos de representación son distintos, no estando el del tipo de dato a convertir incluido en el rango del tipo de destino.

Tipo a convertir	Tipo resultante
byte	char
short	byte char
char	byte short
int	byte short char
long	byte short char int
float	byte short char int long
double	byte short char int long float