## CONVERSIÓN DE TIPOS

Ya sabes que existe un problema cada vez que usamos tipos de datos y tratamos de combinarlos. Habrás visto en entornos de desarrollo que existe una clasificación de lenguajes de programación en base a su tipado de datos.

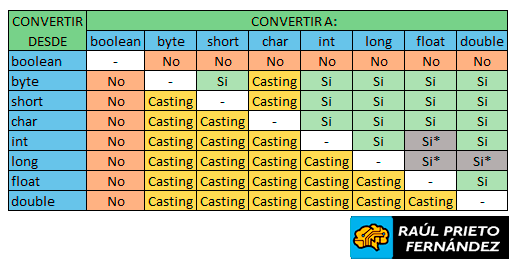
* En los **lenguajes de tipado estático** donde cada variable y tipo de expresión ya se conoce en tiempo de compilación. Una vez que se declara una variable es de un cierto tipo de datos, no puede contener valores de otros tipos de datos. Ejemplo: C, C ++, Java.
* En los l**enguajes de tipado dinámico**: estos lenguajes pueden recibir diferentes tipos de datos a lo largo del tiempo. Ejemplo: Ruby, Python.

**Java está tipado estáticamente** y es fuertemente tipado porque en Java, cada tipo de datos (como entero, carácter, hexadecimal, decimal empaquetado, etc.) está predefinido como parte del lenguaje de programación y todas las constantes o variables definidas para un programa dado debe describirse con uno de los tipos de datos.

Para comprender bien este proceso es necesario tener conocimientos básicos sobre clases y métodos.

## ¿QUÉ TIPOS DE VARIABLES EXISTEN EN JAVA?

En Java es posible transformar el tipo de una variable u objeto en otro diferente al original. Esta transformación se denomina **"conversión", "tipado"** o mas comúnmente **"casting".** En la siguiente tabla se pueden ver las distintas posibilidades de conversión con las variables de tipo primitivo:



En Java es posible transformar el tipo de una variable u objeto en otro diferente al original con el que fue declarado. Este proceso se denomina **"conversión", "moldeado" o "tipado"** y es algo que debemos manejar con cuidado pues un mal uso de la conversión de tipos es frecuente que dé lugar a errores.

De forma general trataremos de atenernos a la norma de que **"en las conversiones debe evitarse la pérdida de información".** En la siguiente tabla vemos conversiones que son seguras por no suponer pérdida de información.

|  |  |
| --- | --- |
| TIPO ORIGEN | TIPO DESTINO |
| byte | double, float, long, int, char, short |
| short | double, float, long, int |
| char | double, float, long, int |
| int | double, float, long |
| long | double, float |
| float | Double |

## Conversión automática de tipos o ampliación

La conversión de ampliación tiene lugar cuando dos tipos de datos se convierten automáticamente.

* Los dos tipos de datos son compatibles.
* Cuando asignamos el valor de un tipo de datos más pequeño a un tipo de datos más grande.



int i = 100;

//conversion automatica de tipo

long l = i;

//conversion automatica de tipo

float f = l;

## Conversión explícita de tipo o casting

Si queremos asignar un valor de tipo de dato más grande a un tipo de dato más pequeño, realizamos un **casteo/casting** o lo que se conoce como **conversión de tipo explícito**.

* Esto es útil para tipos de datos incompatibles donde la conversión automática no se puede realizar.
* **char** y **int** no son compatibles entre sí. Veamos luego cuando tratamos de convertir uno en otro.



double d = 100.04;

//casting de tipo

long l = (long)d;

//casting de tipo

int i = (int)l;

System.out.println("Valor Double "+d);

//parte fraccionaria perdida

System.out.println("Valor Long "+l);

//parte fraccionaria perdida

System.out.println("Valor Int "+i);

## Tipo de promoción en Expresiones

Al evaluar expresiones, el valor intermedio puede exceder el rango de operandos y, por lo tanto, se promoverá el valor de la expresión. Algunas condiciones para la promoción de tipo son:

* Java automáticamente promueve cada operando byte, short, o char al evaluar una expresión.
* Si un operando es long, float o double, la expresión se promueve a long, float o double, respectivamente.

byte b = 42;

char c = 'a';

short s = 1024;

int i = 50000;

float f = 5.67f;

double d = .1234;

// La expresion

double result = (f \* b) + (i / c) - (d \* s);

## Casting de Tipo explícito en Expresiones

Al evaluar expresiones, el resultado se actualiza automáticamente a un tipo de datos más grande del operando. Pero si almacenamos ese resultado en un tipo de datos más pequeño, genera un error de tiempo de compilación, por lo que debemos “castear” el resultado.

byte b = 50;

//casting de tipo int a byte

//mostraría error si no detallamos (byte)

b = (byte)(b \* 2);

## MÉTODO VALUEOF PARA CONVERSIÓN DE TIPOS

El método valueOf es un método sobrecargado aplicable a numerosas clases de Java y que permite realizar conversiones de tipos. Veamos algunos ejemplos de uso.

|  |  |
| --- | --- |
| **EXPRESIÓN** | **INTERPRETACIÓN aprenderaprogramar.com** |
| **miInteger = miInteger.valueOf (i)** | Con i entero primitivo que se transforma en Integer |
| **miInteger = miInteger.valueOf (miString)** | El valor del String se transforma en Integer |
| **miString = miString.valueOf (miBooleano)** | El booleano se transforma en String “true” o “false” |
| **miString = miString.valueOf (miChar)** | El carácter (char) se transforma en String |
| **miString = miString.valueOf (miDouble)** | El double se transforma en String. Igualmente aplicable a float, int, long. |

**No todas las conversiones son posibles.** Muchas veces por despiste los programadores escriben instrucciones de conversión incoherentes como miInteger = (int) miString;. El resultado en este caso es un error de tipo “Inconvertible types”. Un uso típico de valueOf es para convertir tipos primitivos en objetos.

## Parsear Java, Método Parse

El método "parse", entre sus múltiples funciones, nos permite convertir caracteres númericos a datos numéricos, es decir, convertir un número almacenado como String a un dato del tipo int, double u otro según se requiera.

Esto es muy útil a la hora de ingresar números desde el teclado, ya que Java toma los datos capturados por el teclado como cadenas de caracteres con las que no se pueden realizar operaciones matemáticas, y en muchas ocaciones necesitamos leer números para realizar operaciones con ellos.

■ Convertir un dato numérico almacenado en un String usando el método "parse" se hace de la siguiente forma:

• De String a int:

String cadena = "123";

int entero = Integer.parseInt(cadena);

• De String a double:

String cadena = "12.3";

double decimal = Double.parseDouble(cadena);

• De String a float:

String cadena = "1.23";

float flotante = Float.parseFloat(cadena);

• De String a byte:

String cadena = "123";

byte entero\_byte = Byte.parseByte(cadena);

• De String a short:

String cadena = "123";

short entero\_short = Short.parseShort(cadena);

• De String a Long:

String cadena = "123";

long entero\_long = Long.parseLong(cadena);

## Pasa de un tipo a otro:

**DE STRING A BOOLEAN**

boolean boolean1 = Boolean.valueOf("true");  
boolean boolean2 = Boolean.parseBoolean("false");

**DE STRING A CHAR**

String cadena= "E";  
 char caracter = cadena.charAt(0);

**DE STRING A DOUBLE**

double doble = Double.parseDouble("900.1");

**DE STRING A FLOAT**

float cantidad = Float.parseFloat("123.5");

**DE STRING A INTEGER**

Integer n1 = Integer.valueOf("623");  
 int n2 = Integer.parseInt("8237");

**DE BOOLEAN A STRING**

boolean a = true;  
 String cadena1 = String.valueOf(a);  
  
 boolean b = false;  
 String cadena2 = Boolean.toString(b);

**DE CHAR A STRING**

char caracter = 'A';  
 String cadena = Character.toString(caracter);

**DE DOUBLE A STRING**

double doble = 9876.7;  
 String cadena = String.valueOf(doble);

**DE FLOAT A STRING**

String total = Float.toString(87650.3f);

**DE INTEGER A STRING**

String importe = Integer.toString(73643)

Fuentes:

<https://docs.oracle.com/javase/specs/jls/se7/html/jls-5.html>

<https://javadesdecero.es/basico/conversion-tipo-ejemplos-casting/>

<https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=636:conversion-de-tipos-de-datos-en-java-tipado-ejemplos-metodo-valueof-error-inconvertible-types-cu00670b&catid=68&Itemid=188>

<https://www.raulprietofernandez.net/blog/programacion/conversion-de-tipos-de-datos-en-java>