

Curso de Java Standard



Ing. Octavio Robleto



octavio.robleto@gmail.com



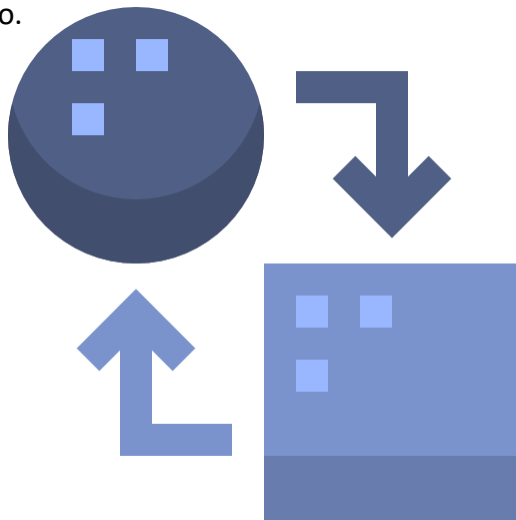
<https://octaviorobleto.com>



Introducción

Muchas veces nos encontraremos con la necesidad de convertir un tipo de dato a otro tipo; Java al ser un lenguaje de tipado fuerte (cada variable u objeto del programa se definen como poseedores de un tipo de dato específico) nos da las herramientas para poder lograr lo que también se conoce como conversión de tipos.

Existen dos tipos de Casteo: Implícito y Explícito.



Casteo Implícito

Esto ocurre cuando necesitamos un tipo de dato mas pequeño en otro mas grande, lo podemos ver con los números; sabemos que un numero entero pertenece a los números reales (con decimales) cuando nos encontremos con la necesidad de hacer esto no tendremos que escribir ningún tipo de código o sentencia.

```
byte miByte = 37;

short miShort = miByte;

int miInt = miShort;

long miLong = miInt;

float miFloat = miLong;

double miDouble = miFloat;
```

Algo particular que pasa con los **char** es que podemos asignarlos directamente a un **int** o a un numérico mas grande, y esto ¿Por qué? Java devolverá el valor ASCII del carácter dado

```
char miChar = 'P';

int miInt = miChar;
```

Casteo Explícito

Este caso sucede cuando queremos asignar un tipo de dato mas grande a un tipo de dato mas pequeño, por ejemplo, un numérico decimal a numérico entero, en estos caso necesariamente tenemos que escribir entre paréntesis el tipo de dato al que queremos convertir.

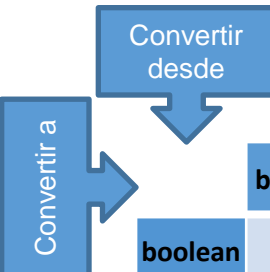
Hay que tomar en cuenta que lo que pasara es que el numero será truncado cuando hagamos la conversión de un decimal a un entero.

```
double miDouble2 = 2.9856;  
  
float miFloat2 = (float) miDouble2;  
  
long miLong2 = (long) miFloat2;  
  
int miInt2 = (int) miLong2;  
  
short miShort2 = (short) miInt2;  
  
byte miByte2 = (byte) miShort2;
```

Ahora bien al contrario de lo que pasa de un **char** a un **int**, la conversión de **int** a **char** no es compatible por si sola, por lo que tendremos que hacerla de forma explicita.

```
int miInt2 = (int) miLong2;  
  
char miChar2 = (char) miInt2;
```

Cuadro de Casteo



	boolean	byte	short	char	int	long	float	double
boolean		no	no	no	no	no	no	no
byte	no		si	cast	si	si	si	si
short	no	cast		cast	si	si	si	si
char	no	cast	cast		si	si	si	si
int	no	cast	cast	cast		si	si*	si*
long	no	cast	cast	cast	cast		si*	si*
float	no	cast	cast	cast	cast	cast		si
double	no	cast	cast	cast	cast	cast	cast	

Valor	Descripción
no	no hay posibilidad de conversión
si	casting es implícito
si*	casting implícito pero se puede producir pérdida de precisión
cast	casting explícito

Clases Envoltorio (Wrapper)

Muchas veces nos vamos a encontrar con otro tipo de problemas a la hora de querer convertir un dato, por ejemplo un valor numérico dentro de una cadena de caracteres cuando le pedimos desde una caja de texto valores al usuario.

También nos vamos a encontrar mas adelante con objetos en Java que no permiten el uso de los valores primitivos.

Java nos provee una forma de envolver los primitivos en un objeto (wrapper class) por cada uno de los 8 tipos que existe.

Además nos dan un variedad de métodos de utilidad que los primitivos no poseen, la mayoría de estos están relacionados con varias conversiones.

Otro punto importante es que nos permite usar el **null** a diferencia de los primitivos que si no los asignamos nos da un valor por defecto en los objetos donde se utilicen.



Como se utilizan

Cada una de estas clases envoltorio (menos Character) tienen dos constructores: uno que admite el tipo primitivo como parámetro y otro que admite un String.

Primitivo	Envoltorio	Constructor
byte	Byte	byte o String
short	Short	short o String
int	Integer	int o String
long	Long	long o String
boolean	Boolean	boolean o String
float	Float	float o String
double	Double	double o String
char	Character	char

Para el constructor del **Boolean** cuando el String es true (sin importar mayúsculas o minúsculas) será true, cualquier otro valor será falso.

```
// enteros
Byte miByte1 = new Byte((byte) 1);
Byte miByte2 = new Byte("127");

Short miShort1 = new Short((short) 2000);
Short miShort2 = new Short("2000");

Integer miInteger1 = new Integer(1000);
Integer miInteger2 = new Integer("100");

Long miLong1 = new Long(1000);
Long miLong2 = new Long("100");

// decimales
Float miFloat1 = new Float(3216.33);
Float miFloat2 = new Float("65491.33");

Double miDouble1 = new Double(3216.33);
Double miDouble2 = new Double("65491.33");

//booleanos
Boolean miBoolean1 = new Boolean(true);
Boolean miBoolean2 = new Boolean("TrUe");
Boolean miBoolean3 = new Boolean("false");
Boolean miBoolean4 = new Boolean("8");

//Caracter
Character miCharacter = new Character('P');
```

Miembros de Instancia

Objeto.xxxvalue(): Devuelve el valor envuelto por el objeto en el tipo especificado, xxx representa el primitivo a obtener.

```
// enteros
byte miByte = miByte1.byteValue();

short miShort = miShort1.shortValue();

int miInt = miInteger1.intValue();

// decimales
long miLong = miLong1.longValue();

float miFloat = miFloat1.floatValue();

double miDouble = miDouble1.doubleValue();

// booleanos
boolean miBoolean = miBoolean1.booleanValue();

// Caracter
char miChar = miCharacter.charValue();
```


Miembros de clase

Envoltorio.valueOf(String) o Envoltorio.valueOf(primitivo) : Devuelve un tipo de objeto (de las clases envoltorio) a partir de un String o dato primitivo, este método es mucho mas eficiente de usar que los constructores, además, dichos constructores están obsoletos a partir de la versión 9 de Java.

Para todas las clases es igual, para Short sería (Short.valueOf(primitivo) o Short.valueOf(cadena), etc), excepto para "Character" que solo posee el Character.valueOf(primitivo).

```
// enteros
Byte miByte3 = Byte.valueOf((byte) 1);
Byte miByte4 = Byte.valueOf("127");

Short miShort3 = Short.valueOf((short) 2000);
Short miShor42 = Short.valueOf("2000");

Integer miInteger3 = Integer.valueOf(1000);
Integer miInteger4 = Integer.valueOf("100");

Long miLong3 = Long.valueOf(1000);
Long miLong4 = Long.valueOf("100");

// decimales
Float miFloat3 = Float.valueOf(3216.33f);
Float miFloat4 = Float.valueOf("65491.33");

Double miDouble3 = Double.valueOf(3216.33);
Double miDouble4 = Double.valueOf("65491.33");

//booleanos
Boolean miBoolean5 = Boolean.valueOf(true);
Boolean miBoolean6 = Boolean.valueOf("TrUe");
Boolean miBoolean7 = Boolean.valueOf("false");
Boolean miBoolean8 = Boolean.valueOf("8");

//Caracter
Character miCharacter2 = Character.valueOf('P');
```

Miembros de clase

parseXXX(String): Es un método que permite convertir la cadena de caracteres en el valor que corresponda donde XXX representa el envoltorio consecuente a ese valor.

No esta disponible en el Character, recordemos que poseemos un método de la clase String que nos devuelve un carácter de la posición deseada (**charAt(posicion)**)

Como pudimos observar al castear un carácter a un entero lo que nos devolvió fue el numero ASCII que representa este numero pero que pasa si necesitamos es el valor numérico literal de ese char?

Tenemos el método **Character.getNumericValue(char)**

```
// enteros
miByte = Byte.parseByte("63");

miShort = Short.parseShort("89");

miInt = Integer.parseInt("896633");

// decimales
miLong = Long.parseLong("191513333");

miFloat = Float.parseFloat("321654.366");

miDouble = Double.parseDouble("25698.36985");

// booleanos
miBoolean = Boolean.getBoolean("true");

// caracteres
miInt = Character.getNumericValue('3');
```

Boxing

En Java se puede asignar directamente un dato primitivo a su respectivo envoltorio sin instanciar el objeto, a esto se le conoce como **boxing**.

Lo que sucede es que Java se da cuenta de lo que uno quiere hacer y convierte el tipo básico en un objeto antes de terminar de asignarlo.

Lo mismo pasa cuando queremos asignar un envoltorio a su respectivo primitivo, por lo que no es necesario utilizar el **xxxvalue()**, a esto se le conoce como **unboxing**.

```
Byte miByte5 = 126;

Short miShort5 = 200;

Integer miInteger5 = 2000;

Long miLong5 = 3985622222L;

Float miFloat5 = 36985.366f;

Double miDouble5 = 369744.36;

Boolean miBoolean9 = true;

Character miCharacter5 = '*';
```

```
miByte = miByte1;

miShort = miShort1;

miInt = miInteger1;

miLong = miLong1;

miFloat = miFloat1;

miDouble = miDouble1;

miBoolean = miBoolean1;

miChar = miCharacter;
```