**CONVERSIONES DE TIPOS DE DATOS (casting)**

### Conversión entre tipos primitivos (casting)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| El casting es un procedimiento para transformar una variable primitiva de un tipo a otro. También se utiliza para transformar  un objeto de una clase a otra clase siempre y cuando haya una relación de herencia entre ambas. En este caso nos centraremos en el primer ripo de casting.  Dentro de este casting de variables primitivas se distinguen dos clases: Casting implícito En este caso no se necesita escribir código para que la conversión se lleve a cabo. Ocurre cuando se realiza lo que se llama una conversión ancha (*widening casting*), es decir, cuando se coloca un valor pequeño en un contenedor grande.  Ejemplo de casting explícitos serían:  int num1 = 100;  long num2 = num1;       // Un int cabe en un long  long num2 = 100;        // 100 en un int Casting explícito En el casting explícito sí es necesario escribir código. Ocurre cuando se realiza una conversión estrecha (*narrowing casting*),  es decir, cuando se coloca un valor grande en un contenedor pequeño. Son susceptibles de pérdida de datos.  int num1 = 100;  short num2 = (short) num1;       // Aquí hace falta un casting explícito: short  tiene menor rango que int  Como se ve, el formato general para indicar que queremos realizar la conversión es:  (tipo) valor\_a\_convertir  En este ejemplo, si se sustituye la primera línea int num1=100 por int num1=1000000, el código compilaría bien, pero habría pérdida de datos, pues el 1000000 se sale del rango de short, que comprende desde  -32768 a 32767. Al mostrar por consola el valor se obtendría un resultado incongruente.  double num1 = 25.5;  float num2 = (float) num1;  float num3 = 17.25;    En este ejemplo se hace un casting explícito en la segunda línea porque float es de menor rango que double. En la tercera  línea el compilador avisaría de que hay un error: cuando escribimos directamente un número en decimal  (como aquí el 17.25), Java lo interpreta siempre como un double; por eso sería necesario un casting explícito para que compilase:  float num3 = (float) num3;  Para finalizar con el casting entre primitivas, conviene tener en cuenta lo siguiente:   * No es posible realizar casting entre una variable primitiva  booleana y cualquier otra variable primitiva. * Sí es posible realizar casting entre una variable primitiva char y una variable primitiva que almacene enteros:   int num1 = 164; char letra = (char) num1; System.out.println(letra); System.out.println((char) 164);  Siendo la salida del programa:  ñ ñ  La siguiente tabla resume las posibilidades de casting existentes:  **Convertir desde                    Convertir a...**   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | **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 |  |   Donde:   * no: indica que no hay posibilidad de conversión. * si: indica que el casting es implícito. * si\*: indica que el casting es implícito pero se puede producir pérdida de precisión. * cast: indica que hay que hacer casting explícito. |

**CONVERSIONES DE TIPOS DE DATOS (Métodos parse, toString, valueOf)**

public class PruebaApp {

    public static void main(String[] args) {

        String numString="1234";

        System.out.println("String: "+numString);

        long numLong=Long.parseLong(numString);

        System.out.println("long: "+numLong);

        int numInt=Integer.parseInt(numString);

        System.out.println("int: "+numInt);

        short numShort=Short.parseShort(numString);

        System.out.println("short: "+numShort);

        byte numByte=Byte.parseByte(numString);

        System.out.println("byte: "+numByte);

        double numDouble=Double.parseDouble(numString);

        System.out.println("double: "+numDouble);

        float numFloat=Float.parseFloat(numString);

        System.out.println("Float: "+numFloat);

    }

}

public class PruebaApp {

    public static void main(String[] args) {

        long numLong=1;

        String cad1=Long.toString(numLong);

        int numInt=2;

        String cad2=Integer.toString(numInt);

        short numShort=3;

        String cad3=Short.toString(numShort);

        byte numByte=4;

        String cad4=Byte.toString(numByte);

        double numDouble=5;

        String cad5=Double.toString(numDouble);

        float numFloat=6;

        String cad6=Float.toString(numFloat);

        System.out.println(cad1+cad2+cad3+cad4+cad5+cad6);

    }

}

Conversión de Tipos de Datos

Veamos fugazmente lo que es la conversión de tipos.

Empezando con el tema, la conversión de tipos consiste en pasar un tipo de dato a otro para poder realizar cierto uso de este según lo necesitemos.

Veamos la conversión de una cadena a un entero y flotante

String Letras="10";  
Float.parseFloat(Letras);  
Integer.parseInt(Letras);

Como pueden ver para pasar a tipo float se usa la clase Float con el método parseFloat y luego le pasamos el dato. Para el tipo entero sería similar con la clase Integer

**Ojo:** Demos tener cuidado al convertir tipos ya que en este caso se puede convertir a tipos numéricos porque la cadena tiene las características de un número, si tendríamos “Palabra x” como cadena en lugar de “10” surgiría un error ya que no se puede convertir a número “Palabra x” porque tal número no existe.

Ahora veamos de un tipo entero a una cadena y real.

int Numero=7;  
String auxNumero;  
auxNumero=String.valueOf(Numero);  
Float.parseFloat(auxNumero);

Para la conversión a cadena sólo usamos la clase String con su método valueOf y luego le pasamos el dato.

Ahora veamos la diferencia en la conversión del tipo entero al float. Podemos ver que hacemos uso de la cadena y no directamente del numero por qué? Simplemente porque las convesiones en las clases Integer y float entre otros sólo aceptan tipos String por lo cual es necesario para convertir un entero a un real o viceversa pasarlo primero a una cadena

Ahora qué de lo float. veamos

float NumeroReal=5.2f;  
String auxNumeroReal;  
int auxN;  
auxNumeroReal=String.valueOf(NumeroReal);  
auxN=(int) NumeroReal;

Como vemos para convertir a tipo String no presenta ningún problema, pero si quisiéramos pasar a entero surgiría erro así lo pasemos primero a cadena porque el valor es 5.2 y ese dato no se puede pasar a entero, entonces que es lo que hacemos, realizamos una conversión directa sacándole sólo la parte entera en este caso obteniendo sólo el 5 del valor real que es 5.2

■ 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);

**■ De dato numérico a String**  
  
Esta conversión puede hacerse de dos formas:  
  
• Usando "**valueOf**"  
int numero = 123;   
String cadena\_numero = String.valueOf(numero);  
  
• Usando "**toString**"   
int numero = 123;  
String cadena\_numero = Integer.toString(numero);