2.D. Casting de tipos.

1. Conversión de tipo.

Imagina que queremos dividir un número entre otro ¿tendrá decimales el resultado de esa división? Podemos pensar que siempre que el denominador no sea divisible entre el divisor, tendremos un resultado con decimales, pero no es así. Si el denominador y el divisor son variables de tipo entero, el resultado será entero y no tendrá decimales. Para que el resultado tenga decimales necesitaremos hacer una **conversión de tipo**.

Las conversiones de tipo se realizan para hacer que el resultado de una expresión sea del tipo que nosotros deseamos, en el ejemplo anterior para hacer que el resultado de la división sea de tipo real y, por ende, tenga decimales. Existen dos tipos de conversiones:

· **Conversiones automáticas**. Cuando a una variable de un tipo se le asigna un valor de otro tipo numérico con menos bits para su representación, se realiza una conversión automática. En ese caso, el valor se dice que es promocionado al tipo más grande (el de la variable), para poder hacer la asignación. También se realizan conversiones automáticas en las operaciones aritméticas, cuando estamos utilizando valores de distinto tipo, el valor más pequeño se promociona al valor más grande, ya que el tipo mayor siempre podrá representar cualquier valor del tipo menor (por ejemplo, de `int` a `long` o de `float` a `double`).

· **Conversiones explícitas**. Cuando hacemos una conversión de un tipo con más bits a un tipo con menos bits. En estos casos debemos indicar que queremos hacer la conversión de manera expresa, ya que se puede producir una pérdida de datos y hemos de ser conscientes de ello. Este tipo de conversiones se realiza con el **operador `cast`**. El operador `cast` es un operador unario que se forma colocando delante del valor a convertir el tipo de dato entre paréntesis. Tiene la misma precedencia que el resto de operadores unarios y se asocia de izquierda a derecha.

Debemos tener en cuenta que **un valor numérico nunca puede ser asignado a una variable de un tipo menor en rango, si no es con una conversión explícita**. Por ejemplo:

```
int a;

byte b;

a = 12;              // no se realiza conversión alguna

b = 12;              // se permite porque 12 está dentro

                     // del rango permitido de valores para b

b = a;               // error, no permitido (incluso aunque

                     // 12 podría almacenarse en un byte)

byte b = (byte) a;   // Correcto, forzamos conversión explícita
```

En el ejemplo anterior vemos un caso típico de error de tipos, ya que estamos intentando asignarle a `b` el valor de `a`, siendo `b` de un tipo más pequeño. Lo correcto es promocionar `a` al tipo de datos `byte`, y entonces asignarle su valor a la variable `b`.