

Casting

1.- Càsting de variables primitives

En ocasions és necessari **convertir** una variable (o una expressió en general) **d'un tipus a un d'altre**. Este procediment s'anomena **CASTING** o Conversió de Tipus primitius. També existeix Casting o Conversió d'objectes, que ja veurem en altres temes.

Recordem que els tipus de variables primitives o dades simples suportats per Java són:

- Per a números **enters**: **byte, short, int, long**.
- Per a números **reals**: **float, double**.
- Per a **caràcters**: **char**.
- Per valors **lògics**: **boolean**.

Per tant, fer un càsting és passar d'un tipus **int** a tipus **float**, o d'un **double** a **short**. Al'hora de reslitzar un càsting hem de tindre en compte que:

- **NO** podem fer un càsting entre una variable **boolean** i qualsevol altra variable primitiva
- **SI** podem fer un càsting entre una variable **char** i una variable que emmagatzeme números **enters** (**byte, short, int, long**).

Int int char boolean float double short

Dins del càsting de variables primitives trobem dos tipus:

- **Càsting implícit**: No necessita escriure codi, es tracta d'una conversió ampla (**widening casting**) , es a dir, quan col·loquem un valor **xicotet** en un contenidor **gran**.

```
int num1 = 100;

long num2 = num1; // Un int cap en un long
long num2 = 100; // 100 en un long

byte num4 = 0;
short num5 = num4; // Un byte cap en un short
```

- **Càsting explícit**: Sí és necessari escriure codi. Es tracta d'una conversió estreta (**narrowing casting**) , es a dir, quan col·loquem un valor **gran** en un contenidor **xicotet**. (OJO!! fer este càsting pot suposar la pèrdua de dades)

```
int num1 = 100;

short num2 = (short) num1; // Ací fa falta un casting explícit:
                           // short té menor capacitat que int

float num3 = 3.1415f;
double num4 = 3.141516;
int num5;

num5 = (int) num3; // int té menor capacitat que float
num5 = (int) num4; // int té menor capacitat que double
```

La forma general de realitzar un càsting és:

(tipus) valor_a_convertir

Exemple amb (char). Recorda que sols podem fer casting entre **char** i una variable que emmagatzeme números **enters** (**byte, short, int, long**).

```
public static void main(String[] args) {
    // 65 -> codi ascii de A
    // casting de int a char

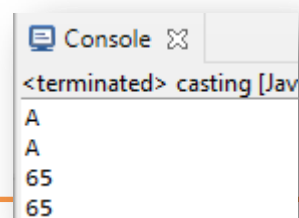
    int num1 = 65;
    char lletra = (char) num1;

    System.out.println(lletra);
    System.out.println((char) 65);

    // casting de char a int

    char lletra2='A';
    int num2 = (int)lletra2;

    System.out.println(num2);
    System.out.println((int)'A');
}
```



```
<terminated> casting [Java]
A
A
65
65
```

NOTA IMPORTANT (float):

A l'exemple anterior trobem l'expressió: **float** num3 = 3.1415f;

En **Java** sempre que escrivim un **número en decimal** (amb coma) l'**interpreta automàticament com un double** (**sempre**), si posem:

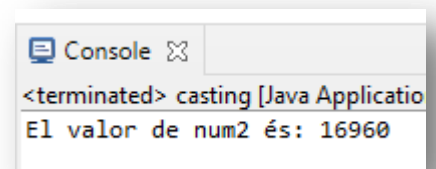
float num3 = 3.1415; el compilador avisa d'un error, per això hi ha que fer un càsting explícit per que compile, que es pot fer de dos maneres:

```
float num = (float)3.1415; // casting explícit
float num = 3.1415f;      // casting abreujat (sols en floats)
```

(OJO!! el càsting pot suposar la pèrdua de dades)

```
public static void main(String[] args) {
    int num1 = 1000000;
    short num2 = (short) num1;

    System.out.println("El valor de num2 és: " + num2 );
}
```



```
<terminated> casting [Java Applicatio]
El valor de num2 és: 16960
```

Si recordem un **short** té un rang entre -32768 i 32767, per tant, al realitzar el càsting amb un valor més gran del que pot representar (**int** num1 = 1000000;) el resultat és incongruent. (num = 16960)

(OJO!! el càsting pot suposar la pèrdua de dades)

```
public static void main(String[] args) {
    int x = 2;
    int y = 9;
    double divisio;

    divisio = (double) y / (double) x;

    System.out.println("El resultat de la di-
    visión és: " + divisio);
}
```

Console
<terminated> casting [Java Application] C:\Pro
El resultat de la división és 4.5

```
public static void main(String[] args) {
    int x = 2;
    int y = 9;
    double divisio;

    divisio = y / x;

    System.out.println("El resultat de la
    división és " + divisio);
}
```

Console
<terminated> casting [Java Application] C:\Progr
El resultat de la división és 4.0

Primer tenim:

divisio = (double) y / (double) x;

(double) y : converteix el valor d' y en un double, és a dir, en un número amb decimals i (double) x fa el mateix amb x, per tant, l'operació de divisió quedaria com 9.0 / 2.0. En tractar-se d'una divisió entre nombres decimals, Java entén que el resultat ha de tindre també decimals i usa tota la precisió disponible en el tipus. El resultat que es guarda en la variable divisio és 4.5 que és el valor correcte.

En el segon cas tenim:

divisio = y / x;

Ara la divisió és 9 / 2. Java veu una divisió entre dos nombres enters, així que el resultat que ofereix és un altre nombre enter després de menysprear els decimals si n'hi haguera. El resultat que dona Java és la divisió sencera, és a dir, 4. El següent pas és una assignació, tindríem divisio = 4;. Com la variable divisio és de tipus double, el valor 4 es guarda en realitat com 4.0;

!!!COSAS DE JAVA!!!

Recorda : Fes el casting per a conservar els decimals en les divisions amb enters

La divisió entre dos números enters es un altre número enter. Pera conservar els decimals de la divisió hi ha que fer un càsting a float o double.