# 2. Declaración e inicialización

Antes de explicar cómo se pueden declarar e inicializar las variables, necesitamos saber cuáles son los tipos de datos básicos o primitivos que existen en Java.

## 2.1. Tipos de datos

### A. Tipos de datos básicos o primitivos

Hemos empezado hablando sobre el valor de los datos. Las empresas y las organizaciones públicas los utilizan, respectivamente, para aumentar sus beneficios o para mejorar la vida de las personas. Pero todos esos datos necesitan ser gestionados a partir de programas surgidos para resolver problemas del mundo real, traduciéndolos en algoritmos y datos básicos. En Java existen los 8 tipos de datos siguientes:

| DESCRIPCIÓN | NOMBRE TIPO EN JAVA | FORMATO DE VALORES ADMITIDOS | VALOR POR DEFECTO | TAMAÑO EN BITS | TAMAÑO EN BYTES | VALOR MÍNIMO RANGO (MIN_VALUE) | VALOR MÁXIMO RANGO (MAX_VALUE) | EJEMPLOS |
|---|---|---|---|---|---|---|---|---|
| Valor binario booleano | boolean | true o false | false | 1 Depende de la JVM | 1 Depende de la JVM | Igual a cero | distinto de cero | true = 1, false = 0 |
| Número entero pequeño | byte | entero complemento de dos | 0 | 8 | 1 | -2^7 -128 | 2^7-1 127 | 22, -9 |
| Número entero corto | short | entero complemento de dos | 0 | 16 | 2 | -2^15 32768 | 2^15 -1 32767 | -30303, 3210 |
| Número entero | int | entero complemento de dos | 0 | 32 | 4 | -2^31 | 2^31 -1 | -303030, -1000,-3, -1, 0, 1, 654321 |
| Número entero largo | long | entero complemento de dos | 0 | 64 | 8 | -2^63 | 2^63 -1 | -3L, -1L, 0L,1L,3L |
| Carácter | char | carácter Unicode | \u0000 | 16 | 2 | '\u0000' | '\uFFFF' | 'R', '\u0082', |
| Número científico | float | coma flotante IEEE 754 | 0.0 | 32 | 4 | 2^-149 | 2^-149 -1 | 1.2e100f, 3.21f |
| Número científico largo | double | coma flotante IEEE 754 | 0.0 | 64 | 8 | 2^-1074 | 2^-1074-1 | 1.2345e300d, -1.2345e-300d |

*Tabla 2.1. Los 8 tipos de datos primitivos en Java.*

El primer tipo se utiliza para representar valores de cierto o falso en el algebra booleana de índole binaria. Los siguientes 4 tipos son para números enteros. El siguiente se emplea para caracteres sueltos en Unicode. Y los dos últimos se destinan a los números en coma flotante (con decimales) para cálculos científicos. NOTA: los MIN_VALUE y MAX_VALUE de estos últimos se han simplificado para que nos hagamos una idea inicial. Para más información, conviene visitar la página web del fabricante.

También utilizaremos el tipo de dato String. Este empieza por mayúsculas, ya que no es un tipo de dato primitivo, sino un objeto especial. Nos servirá para trabajar con agrupaciones de datos char, para crear cadenas de caracteres —desde palabras a frases enteras—. Por tanto, lo habitual para almacenar y gestionar datos alfanuméricos será este tipo de dato String. Además, ese texto que va a almacenar se le asignará entre comillas dobles, a diferencia de la asignación de los char, que utilizan comillas simples.

### B. Variables vs constantes (variables finales)

Esos datos van a poder comportarse básicamente de dos modos: como **constantes** o como **variables**. En el caso de las primeras, introduciremos un valor inicial que ya no cambiará nunca. En las variables vamos a poder ir utilizando su valor actual, pudiendo almacenar cada resultado en la misma variable o en otra variable que nos interese, cuyo valor podremos alterar en cualquier momento.

## 2.2. Declaración

Por medio de la declaración, le estamos asociando a cada variable su identificador y su tipo de datos. Este limitará el conjunto de valores que puede almacenar, y según el tamaño que le ha asignado el fabricante decidimos la zona de memoria que va a ocupar —desde 1 bit, para los booleanos, hasta los 8 bytes para los tipos de datos más grandes—. Al mismo tiempo, también estaremos decidiendo el conjunto de operaciones permitidas sobre esa variable declarada. Las operaciones concretas las iremos viendo a lo largo de la presente unidad.

Se refieren tanto a objetos como a tipos de datos primitivos.

- Las variables tienen que declararse antes de usarse:

```
<tipo> <identificador>;
```

```
double perimetro;
```

Como puedes ver en la tabla 2.2., todos los tipos de datos tienen un valor por defecto al que se inicializan cuando se declaran sin asignarles ningún valor inicial concreto; en este caso, sería el cero.

**IMPORTANTE**

Las declaraciones de variables pueden incluirse en cualquier parte del programa, pero siempre antes de que la variable se utilice. Además, hay que tener cuidado con el rango de validez (scope) de la declaración.

Ejemplos:

```
int i;
int j;
char letra;
boolean estamosBien;
String holaMundo;
```

## 2.3. Inicialización

Vamos a ver cómo inicializar las variables en el mismo momento de su declaración. Estas se pueden inicializar mediante una asignación, que es una operación que permite cambiar el valor por defecto del tipo de datos por el valor que aparece a la derecha del igual. Así, por ejemplo, con el valor numérico entero positivo 22 se haría de esta manera:

```
<tipo> <identificador> = <valor>;
int perimetro = 22;
```

En este ejemplo hemos creado el identificador de la variable `perimetro`, de tipo entero y que ocupará 4 bytes en la memoria. Además, en dicha posición de la memoria se almacenará ese valor `22` para cuando necesitemos usarlo en nuestro programa. Como no hemos definido el dato como una constante, podremos cambiar su valor en cualquier momento —por el que decidamos o por el que calculemos—, pero siempre dentro del rango de las constantes MIN_VALUE y MAX_VALUE que ha definido el fabricante para ese tipo de dato.

Ejemplos:

```
int i = 0;
int j = 1;
char letra = 'R';
boolean estamosBien = true;
String holaMundo = "¡Hola Mundo!";
```

## 2.4. Tipos referenciados

A partir de los ocho tipos datos primitivos que acabamos de aprender, se pueden construir otros tipos de datos. Estos tipos de datos se llaman tipos referenciados o referencias, porque se utilizan para almacenar la dirección de los datos en la memoria del ordenador. Veamos un ejemplo:

```
int[ ] arrayDeEnteros; // Solo estamos declarando el tipo de datos del array.
Cuenta cuentaCliente; // Definimos una referencia a un objeto de la clase Cuenta.
```

En la primera instrucción, declaramos una lista de números del mismo tipo, en este caso, enteros. En la segunda instrucción, estamos declarando la variable u objeto `cuentaCliente` como una referencia de tipo Cuenta. Pronto comenzaremos a aprender a crear objetos declarados de este modo.

Cuando el conjunto de datos utilizado tiene características similares, los datos se suelen agrupar en estructuras para facilitar el acceso a los mismos. Hablamos entonces de **datos estructurados**.

- Tendremos datos de tipo **estático** como los arrays para crear vectores o matrices, entre otros.
- O también datos **dinámicos** como árboles, listas, etcétera.

Ambos pueden estar en la memoria del programa en ejecución, guardados en el disco como ficheros, o almacenados en una base de datos.

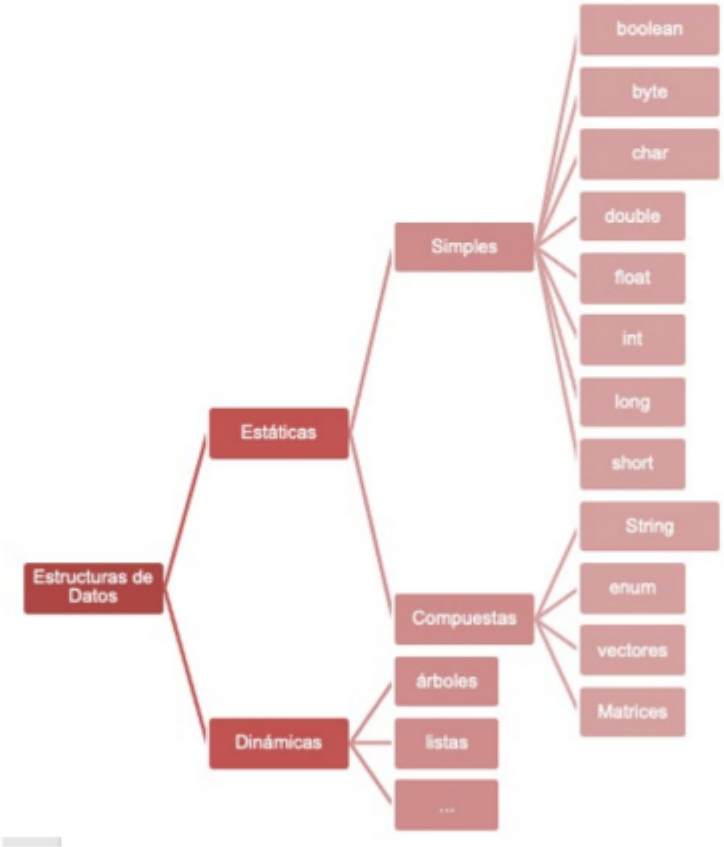


*Figura 2.1. Estructuras de datos estáticas y dinámicas*

## 2.5. Tipos enumerados

Los tipos de datos enumerados son una forma de declarar un nuevo tipo de variable con un conjunto restringido de valores. Por ejemplo, los días de la semana, las estaciones del año, los meses, etcétera. Es como si definiéramos nuestro propio tipo de datos.

La forma de declararlos es con la palabra reservada `enum`, seguida del nombre de la variable y de la lista de valores que puede adoptar, reflejada entre llaves. A los valores que se colocan dentro de las llaves se los considera constantes, van separados por comas y deben ser valores únicos.

La lista de valores se coloca entre llaves, porque un tipo de datos `enum` no es otra cosa que una suerte de clase especial en Java, y todas las clases llevan su contenido entre llaves.

Por ejemplo:

```
public enum Day {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY
}
```

O podríamos crear otra clase en nuestro propio idioma, si bien para nosotros el primer día de la semana —en un sentido cultural— sería el lunes, por lo que haríamos este cambio:

```
public enum Dia {
    LUNES, MARTES, MIERCOLES, JUEVES, VIERNES, SABADO, DOMINGO
}
```

Así podremos utilizar todas estas constantes en nuestros programas, aunque internamente para el tipo de datos enumerado cada constante recibirá el valor de su posición empezando por el cero.

Por tanto: `el LUNES valdrá 0, el MARTES un 1, el MIERCOLES un 2, el JUEVES un 3, el VIERNES un 4, el SABADO un 5 y el DOMINGO un 6.`

A lo largo del curso iremos creando y utilizando este tipo de clase especial. No obstante, si quieres recabar más información al respecto, puedes consultar la página web del fabricante.