TIPOS DE DATOS (VARIABLES) EN JAVA.

Existen dos categorías de variables: las de tipo primitivo y las referenciadas. Una variable de tipo primitivo accede al valor asignado directamente. Las referenciadas acceden a través de un puntero, es decir, no almacenan un valor sino una dirección de memoria.

Los primeros lenguajes de programación no usaban objetos, solo variables. Una variable podríamos decir que es **un espacio de la memoria del ordenador a la que asignamos un contenido** que puede ser un valor numérico (sólo números, con su valor de cálculo) o de tipo carácter o cadena de caracteres (valor alfanumérico que constará sólo de texto o de texto mezclado con números).

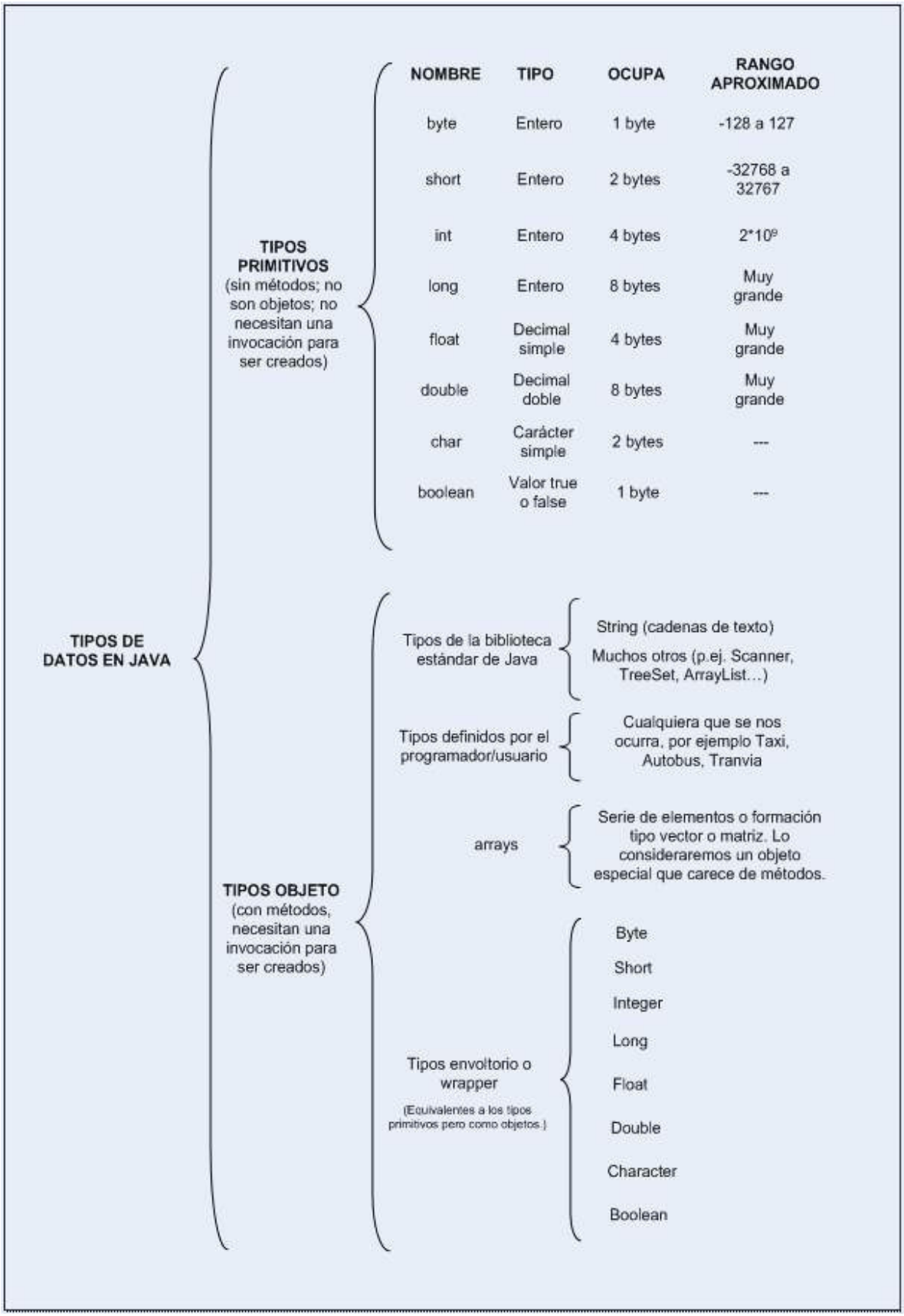
Como ejemplo podemos definir una variable a que contenga 32 y esto lo escribimos como a = 32. Posteriormente podemos cambiar el valor de a y hacer a = 78. O hacer "a" equivalente al valor de otra variable "b" así: a = b.

Dado que antes hemos dicho que un objeto también ocupa un espacio de memoria: **¿en qué se parecen y en qué se diferencia un objeto de una variable?** Consideraremos que las variables son entidades elementales: un número, un carácter, un valor verdadero o falso… mientras que los objetos son entidades complejas que pueden estar formadas por la agrupación de muchas variables y métodos. Pero ambas cosas ocupan lo mismo: un espacio de memoria (que puede ser más o menos grande).

En los programas en Java puede ser necesario tanto el uso de datos elementales como de datos complejos. Por eso en Java se usa el término "Tipos de datos" para englobar a cualquier cosa que ocupa un espacio de memoria y que puede ir tomando distintos valores o características durante la ejecución del programa. Es decir, en vez de hablar de tipos de variables o de tipos de objetos, hablaremos simplemente de tipos de datos. Sin embargo, a veces "coloquialmente" no se utiliza la terminología de forma estricta: puedes encontrarte textos o páginas web donde se habla de una variable en alusión a un objeto.

En Java diferenciamos dos tipos de datos: por un lado, los tipos primitivos, que se corresponden con los tipos de variables en lenguajes como C y que son los datos elementales que hemos citado. Por otro lado, los tipos objeto (que normalmente incluyen métodos).

Veamos los tipos de datos en Java sobre un esquema de síntesis:


TIPOS DE DATOS EN JAVA
TIPOS PRIMITIVOS
(sin métodos; no son objetos; no necesitan una invocación para ser creados)
NOMBRE
TIPO
OCUPA
RANGO APROXIMADO
byte
Entero
1 byte
-128 a 127
short
Entero
2 bytes
-32768 a 32767
int
Entero
4 bytes
2*10⁹
long
Entero
8 bytes
Muy grande
float
Decimal simple
4 bytes
Muy grande
double
Decimal doble
8 bytes
Muy grande
char
Carácter simple
2 bytes
---
boolean
Valor true o false
1 byte

TIPOS OBJETO
(con métodos, necesitan una invocación para ser creados)
Tipos de la biblioteca estándar de Java
String (cadenas de texto)
Muchos otros (p.ej. Scanner, TreeSet, ArrayList…)
Tipos definidos por el programador/usuario
Cualquiera que se nos ocurra, por ejemplo Taxi, Autobus, Tranvia
arrays
Serie de elementos o formación tipo vector o matriz. Lo consideraremos un objeto especial que carece de métodos.
Tipos envoltorio o wrapper
(Equivalentes a los tipos primitivos pero como objetos.)
Byte
Short
Integer
Long
Float
Double
Character
Boolean

Este esquema no es necesario aprendérselo de memoria en todos sus detalles, aunque sí lo iremos memorizando poco a poco a medida que lo utilicemos, por lo menos hasta tener en nuestra cabeza los nombres de todos los tipos primitivos y envoltorio y sus características (si son objetos o no y su rango aproximado). A continuación mostramos el mismo esquema en formato de tabla:

<table>
<tr><td rowspan="17">TIPOS DE DATOS EN JAVA</td><td></td><td>NOMBRE</td><td>TIPO</td><td>OCUPA</td><td>RANGO APROXIMADO</td></tr>
<tr><td rowspan="8">TIPOS PRIMITIVOS
(sin métodos; no son objetos; no necesitan una invocación para ser creados)</td><td>byte</td><td>Entero</td><td>1 byte</td><td>-128 a 127</td></tr>
<tr><td>short</td><td>Entero</td><td>2 bytes</td><td>-32768 a 32767</td></tr>
<tr><td>int</td><td>Entero</td><td>4 bytes</td><td>$2*10^9$</td></tr>
<tr><td>long</td><td>Entero</td><td>8 bytes</td><td>Muy grande</td></tr>
<tr><td>float</td><td>Decimal simple</td><td>4 bytes</td><td>Muy grande</td></tr>
<tr><td>double</td><td>Decimal doble</td><td>8 bytes</td><td>Muy grande</td></tr>
<tr><td>char</td><td>Carácter simple</td><td>2 bytes</td><td>---</td></tr>
<tr><td>boolean</td><td>Valor true o false</td><td>1 byte</td><td>---</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="9">TIPOS OBJETO
(con métodos, necesitan una invocación para ser creados)</td><td>Tipos de la biblioteca estándar de Java</td><td colspan="3">String (cadenas de texto)
Muchos otros (p.ej. Scanner, TreeSet, ArrayList...)</td></tr>
<tr><td>Tipos definidos por el programador / usuario</td><td colspan="3">Cualquiera que se nos ocurra, por ejemplo Taxi, Autobus, Tranvia</td></tr>
<tr><td>arrays</td><td colspan="3">Serie de elementos o formación tipo vector o matriz. Lo consideraremos un objeto especial que carece de métodos.</td></tr>
<tr><td rowspan="6">Tipos envoltorio o wrapper (Equivalentes a los tipos primitivos pero como objetos.)</td><td colspan="3">Byte</td></tr>
<tr><td colspan="3">Short</td></tr>
<tr><td colspan="3">Integer</td></tr>
<tr><td colspan="3">Long</td></tr>
<tr><td colspan="3">Float</td></tr>
<tr><td colspan="3">Double</td></tr>
<tr><td colspan="3">Character</td></tr>
<tr><td colspan="3">Boolean</td></tr>
</table>

Vamos a comentar distintas cuestiones:

1. Un objeto es una cosa distinta a un tipo primitivo, aunque "porten" la misma información. Tener siempre presente que los objetos en Java tienen un tipo de tratamiento y los tipos primitivos, otro. Que en un momento dado contengan la misma información no significa en ningún caso que sean lo mismo. Iremos viendo las diferencias entre ambos poco a poco. De momento, recuerda que el tipo primitivo es algo elemental y el objeto algo complejo. Supón una cesta de manzanas en la calle: algo elemental. Supón una cesta de manzanas dentro de una nave espacial (considerando el conjunto nave +

cesta): algo complejo. La información que portan puede ser la misma, pero no son lo mismo.

2. ¿Para qué tener esa aparente duplicidad entre tipos primitivos y tipos envoltorio? Esto es una cuestión que atañe a la concepción del lenguaje de programación. Tener en cuenta una cosa: un tipo primitivo es un dato elemental y carece de métodos, mientras que un objeto es una entidad compleja y dispone de métodos. Por otro lado, de acuerdo con la especificación de Java, es posible que necesitemos utilizar dentro de un programa un objeto que "porte" como contenido un número entero. Desde el momento en que sea necesario un objeto habremos de pensar en un envoltorio, por ejemplo Integer. Inicialmente nos puede costar un poco distinguir cuándo usar un tipo primitivo y cuándo un envoltorio en situaciones en las que ambos sean válidos. Seguiremos esta regla: usaremos por norma general tipos primitivos. Cuando para la estructura de datos o el proceso a realizar sea necesario un objeto, usaremos un envoltorio.

3. Los nombres de tipos primitivos y envoltorio se parecen mucho. En realidad, excepto entre int e Integer y char y Character, la diferencia se limita a que en un caso la inicial es minúscula (por ejemplo double) y en el otro es mayúscula (Double). Esa similitud puede confundirnos inicialmente, pero hemos de tener muy claro qué es cada tipo y cuándo utilizar cada tipo.

4. Una cadena de caracteres es un objeto. El tipo String en Java nos permite crear objetos que contienen texto (palabras, frases, etc.). El texto debe ir siempre entre comillas. Muchas veces se cree erróneamente que el tipo String es un tipo primitivo por analogía con otros lenguajes donde String funciona como una variable elemental. En Java no es así.

5. Hay distintos tipos primitivos enteros. ¿Cuál usar? Por norma general usaremos el tipo int. Para casos en los que el entero pueda ser muy grande usaremos el tipo long. Los tipos byte y short los usaremos cuando tengamos un mayor dominio del lenguaje.

6. ¿Cuántos tipos de la biblioteca estándar de Java hay? Cientos o miles. Es imposible conocerlos todos.

Tipos de variables JAVA según alcance de acceso.

Hay tres tipos de variables en Java:

- Variables locales
- Variables de clase o instancia
- Variables estáticas

Ahora aprendamos sobre cada una de estas variables en detalle.

Variables locales

Una variable definida dentro de un bloque, método o constructor se llama **variable local**.

- Estas variables se crean cuando el bloque ingresado o método se llama y destruye después de salir del bloque o cuando la llamada regresa del método.
- El alcance de estas variables solo existe dentro del bloque en el que se declara la variable, es decir, podemos acceder a estas variables solo dentro de ese bloque.

Variables de clase o instancia

Las variables de instancia son variables no estáticas y se declaran en una clase fuera de cualquier método, constructor o bloque.

- Como las variables de instancia se declaran en una clase, estas variables se crean cuando un objeto de la clase se crea y se destruye cuando se destruye el objeto.
- A diferencia de las variables locales, podemos usar especificadores de acceso para variables de instancia. Si no especificamos ningún especificador de acceso, se utilizará el especificador de acceso predeterminado.

Variables estáticas

Las variables estáticas también se conocen como **variables de clase.**

- Estas variables se declaran de forma similar a las variables de instancia, la diferencia es que las variables estáticas se declaran utilizando la palabra clave static dentro de una clase fuera de cualquier constructor o bloque de métodos.
- A diferencia de las variables de instancia, **solo podemos tener una copia de una variable estática por clase**, independientemente de cuántos objetos creemos.
- Las variables estáticas se crean al inicio de la ejecución del programa y se destruyen automáticamente cuando finaliza la ejecución.

Para acceder a variables estáticas, no necesitamos crear ningún objeto de esa clase, simplemente podemos acceder a la variable como:

nombre_clase.nombre_variable;

Variable de instancia vs Variable estática

- **Cada objeto tendrá su propia copia de la variable de instancia**, mientras que solo podemos tener una copia de una variable estática por clase, independientemente de cuántos objetos creemos.
- Los cambios realizados en una variable de instancia utilizando un objeto no se reflejarán en otros objetos, ya que cada objeto tiene su propia copia de la variable

de instancia. En caso de estática, los cambios se reflejarán en otros objetos ya que las variables estáticas son comunes a todos los objetos de una clase.

- Podemos acceder a variables de instancia a través de referencias de objetos y **se puede acceder directamente a las variables estáticas usando el nombre de clase**.

Creando constantes

La palabra clave "**final**" es un modificador que se puede usar al declarar variables para **evitar cualquier cambio posterior** en los valores que inicialmente se les asignaron.

Esto es útil cuando se almacena un valor fijo en un programa para evitar que se altere accidentalmente.

Las variables creadas para almacenar valores fijos de esta manera se conocen como "**constantes**", y es convencional nombrar constantes con **todos los caracteres en mayúsculas**, para distinguirlas de las variables regulares. Los programas que intentan cambiar un valor constante no se compilarán, y el compilador **javac** generará un mensaje de error.