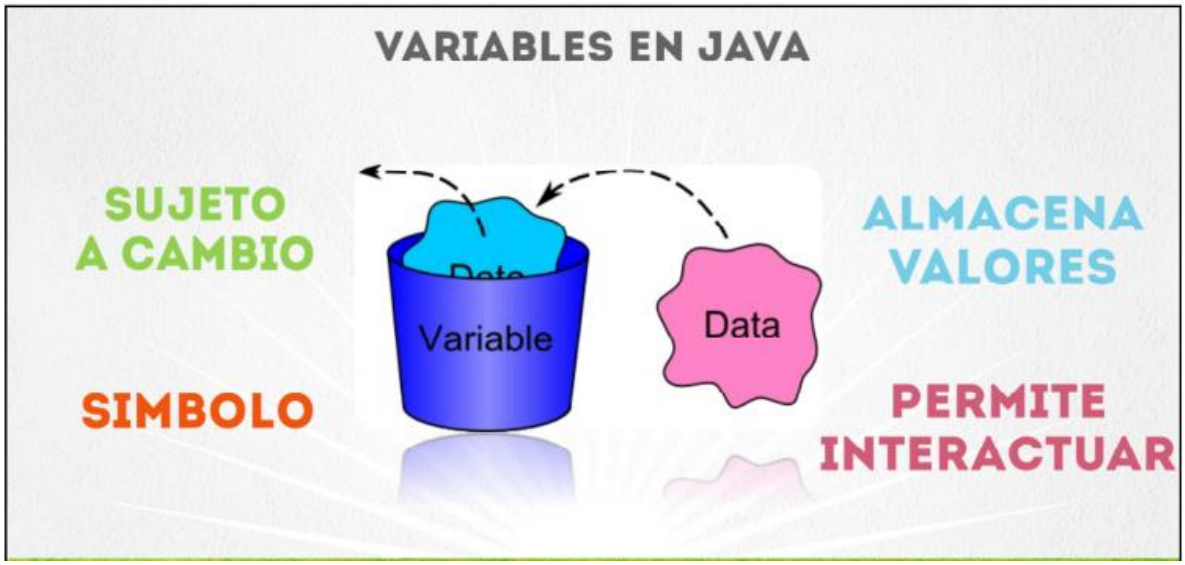
# VARIABLES EN JAVA



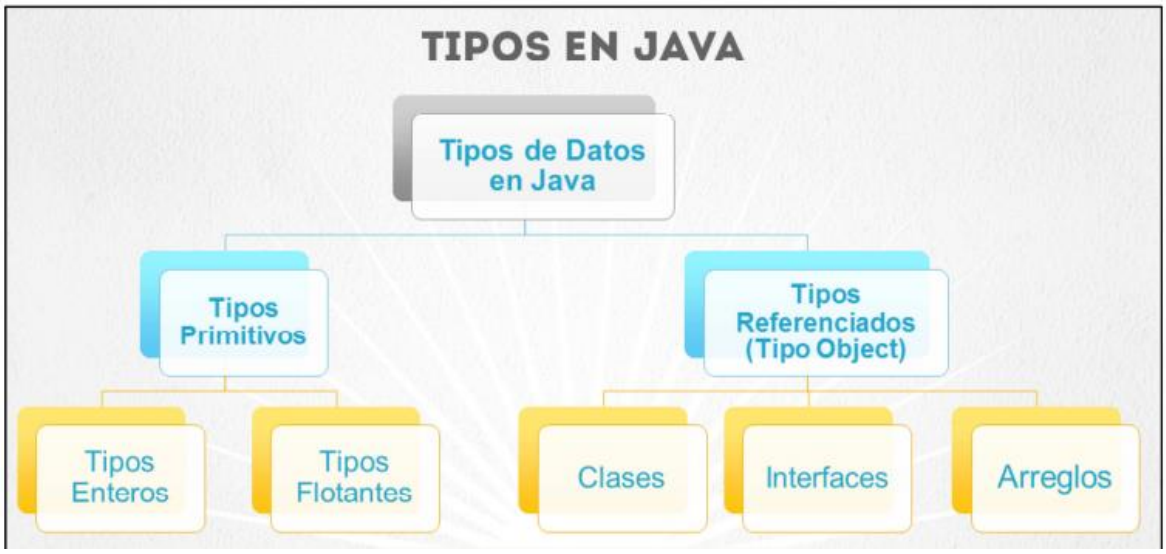

En Java, como en cualquier lenguaje de programación, para almacenar información, es necesario la creación de variables, las cuales nos permitirán almacenar datos de nuestro programa de manera temporal.

El objetivo de declarar una variable es reservar espacio de memoria dependiendo del tipo que vayamos a utilizar.

Estas variables nos permiten también hacer programas dinámicos, por lo que en la mayoría de los casos los valores cambiarán durante la interacción con el usuario y el programa.

Por ejemplo, como podemos ver en la figura tenemos datos, los cuales pueden ir cambiando a lo largo de la ejecución de nuestro programa, y tenemos variables, que son las que permiten almacenar nuestros datos durante la ejecución del mismo.

En Java existen distintos tipos de datos, por lo que estudiaremos a más detalle los tipos de datos y por lo tanto los tipos de variables que necesitamos crear para almacenar un tipo de dato específico. Así que continuemos.




Existe una clasificación amplia respecto a los tipos que se manejan en Java, sin embargo podemos resumirla en la figura mostrada.

Por un lado tenemos los tipos primitivos y por otro lado tenemos los tipos que se consideran como extensiones de la clase Object, también conocidos como referencia a objetos.

Respecto a los tipos primitivos podemos clasificarlos en tipos Enteros y Flotantes, sin embargo tenemos también el tipo boolean, cada uno de estos tipos los veremos a más detalle a continuación.

Por el contrario, tenemos los tipos Object, los cuales pueden ser Clases, Interfaces o Arreglos (Arrays) en Java. Estos tipos los estudiaremos en el tema de Clases y Objetos en Java, así que de momento nos enfocaremos en los tipos primitivos y la forma en cómo declararlos y usar este tipo de datos en Java.


Como comentamos anteriormente, tenemos distintos tipos de datos en Java, en total son 8.

Por un lado tenemos los tipos enteros, entre los cuales tenemos el tipo byte el cual ocupa 8 bits. Posteriormente tenemos el tipo short, el cual ocupa 16 bits. También tenemos el tipo char, el cual ocupa 16 bits pero maneja el código UNICODE para almacenar valores tipo char. A su vez tenemos el tipo int el cual ocupa 32 bits, y finalmente el tipo long el cual ocupa 64 bits.

Por otro lado tenemos los tipos flotantes, por un lado el tipo float el cual ocupa 32 bits, y el tipo double que ocupa 64 bits. El tipo boolean en Java también es un tipo primitivo y puede almacenar sólo el valor de true o false. Su valor por default es false.

Como comentamos anteriormente, tenemos distintos tipos de datos en Java, en total son 8.

Por un lado tenemos los tipos enteros, entre los cuales tenemos el tipo byte el cual ocupa 8 bits. Posteriormente tenemos el tipo short, el cual ocupa 16 bits. También tenemos el tipo char, el cual ocupa 16 bits pero maneja el código UNICODE para almacenar valores tipo char. A su vez tenemos el tipo int el cual ocupa 32 bits, y finalmente el tipo long el cual ocupa 64 bits.

Por otro lado tenemos los tipos flotantes, por un lado el tipo float el cual ocupa 32 bits, y el tipo double que ocupa 64 bits. El tipo boolean en Java también es un tipo primitivo y puede almacenar sólo el valor de true o false. Su valor por default es false.


En Java el manejo de cadenas es un tipo Object, pero que tiene varias características en particular. Aunque aún no detallaremos aún el manejo de objetos, pero cabe mencionar que en Java es necesario el uso del operador new para crear un nuevo objeto.

Sin embargo cuando hablamos de cadenas esto no es necesario. Podemos simplemente declarar un tipo String y asignar un valor cadena directamente a esta variable, es por ello que pareciera que el tipo String es un tipo primitivo pero no es así, sin embargo debido al uso tan frecuente de este tipo al momento de estar programando, es que se decidió simplificar el proceso de creación y asignación de valores en este tipo String en particular. Ejemplo:

```
String saludo = "Hola Mundo";
```

Podemos observar que para declarar una cadena en Java, únicamente basta con utilizar el tipo String, posteriormente definir su nombre (identificador), y finalmente asignar un valor, sin la necesidad de utilizar el operador new. Así, podemos asignar directamente el valor de una cadena, simplemente utilizando comillas dobles para envolver el valor deseado.

En Java, a diferencia de otros lenguajes, no se permite la sobrecarga de operadores, lo que significa que no podemos alterar la función de los operadores para hacer lo que nosotros deseemos, sin embargo existe una excepción a esta regla cuando manejamos tipos cadena. Cuando usamos el operador + y se detecta un tipo String en la operación, se dice que tenemos el contexto String, y por lo tanto en lugar de sumar valores, lo que hace Java es concatenar los valores que se encuentren en la operación. Esto lo veremos más adelante con un ejercicio, pero es importante notar este punto ya que es otra simplificación importante al momento de usar cadenas en Java.

Una de los detalles más importantes que estudiaremos más adelante, es la forma en que se comparan objetos en Java, sin embargo en esta sección sólo aclararemos que al utilizar objetos no se utiliza el símbolo == como con los tipos primitivos, sino el método equals(). En el caso de cadenas esto nos permitirá comparar el contenido de la cadena. En el tema de objetos estudiaremos esto a más detalle, pero de momento es importante que conozcan este detalle para que sepan cómo comparar cadenas.

## CARACTERES DE ESCAPE AL UTILIZAR CADENAS

| Secuencia de Escape | Descripción |
|---|---|
| \t | Inserta un tabulador |
| \b | Inserta un retroceso (backspace) |
| \n | Inserta una nueva línea |
| \r | Inserta un retorno de carro |
| \f | Se mueve a la siguiente pagina (Form feed). Se utiliza para impresoras, no en consolas. |
| \' | Inserta una comilla simple |
| \" | Inserta una comilla doble |
| \\ | Inserta una barra invertida |

En la tabla mostrada, podemos observar los caracteres de escape que son utilizados al momento de desplegar mensajes en la salida estándar. Estos caracteres los pondremos en práctica en los ejercicios que vamos a realizar a continuación.