# Variables y constantes

Los algoritmos son un conjunto de instrucciones sistemáticas y finitas que realizan cálculos y resuelven un problema en particular.

Al realizar cálculos y operaciones sobre datos necesitaremos que el lenguaje de programación pueda trabajar con más de un tipo de dato y a su vez almacene de manera temporaria los resultados intermedios de las operaciones antes de dar con el resultado que resuelve el problema para el cual fue diseñado.

# ¿Que es una variable y que es una constante?

Las variables no son más que etiquetas que representan un valor almacenado en la memoria de la pc. Con el correr de las instrucciones de nuestro programa ese valor en la mayoría de los casos irá sufriendo modificaciones. Por ejemplo: si nuestro cliente nos pide que hagamos un programa de control o gestión de stock, es necesario que nuestro programa tenga una variable que vaya modificando su valor a medida que los productos se vendan o ingresen productos para la reposición cuando el proveedor realice las entregas.

Las constantes en cambio son variables que no tienen permitido cambiar su valor en memoria mientras el programa ejecuta su ciclo de vida. Por ejemplo: a medida que nuestro cliente venda sus productos en la facturación deberá incluir al precio final un 21% de recargo correspondiente con el impuesto al IVA, independientemente de la venta el 21% será siempre constante, por ello para realizar los cálculos necesitaremos que la pc almacene de manera temporal 21% y estar seguros que ese valor no cambiará hasta que el programa se termine de ejecutar.

# Tipos de datos en primitivos en JAVA

Siguiendo con el ejemplo del sistema de gestión de stock y facturación  pensemos en las siguientes situaciones.. si deseamos almacenar el stock de distintos productos, no solo debemos tener en cuenta la cantidad existente si no el nombre del producto, el código de identificación que puede ser alfanumérico, el precio que es un dato decimal referente a la moneda, etc . Para ellos JAVA pone a disposición distintos tipos de datos para poder llevar a cabo operaciones de distinto tipo.

Veamos los tipos de datos primitivos existentes en JAVA

# Tipos de datos Primitivos en JAVA

| Tipo de datos | Valor por defecto | Tamaño por defecto |
|---|---|---|
| boolean | false | 1 bit |
| char | '\u0000' | 2 byte |
| byte | 0 | 1 byte |
| short | 0 | 2 byte |
| int | 0 | 4 byte |
| long | 0L | 8 byte |
| float | 0.0f | 4 byte |
| double | 0.0d | 8 byte |

# ¿Cómo utilizar una variable?

Dentro de nuestro bloque de código deberemos declarar e inicializar las variables y constantes.
La declaración es el hecho de asignar un nombre y un tipo de dato a la variable que utilizaremos.
Y la inicialización es acción de asignar por primera vez un valor a dicha constante, para asignar un valor deberemos utilizar el operador de asignación acompañado de una literal.

Estructura basica: [tipo] [identificador] [operador de asignación] [literal];

```
int identificador;

identificador = 12;
```

podremos realizar la declaración y la asignación por separado

```
int identificador = 12;
```

o realizar las dos acciones en la misma línea de código

# ¿Cómo utilizar una constante?

La utilización de constante es la misma que la de las variables solo que se agrega la palabra reservada “final” para indicar que esa variable solo se le asigna un valor por única vez

Estructura básica:  final [tipo] [identificador] [operador de asignación] [literal];

```
final int identificador;

identificador = 12;
```

```
final int identificador = 12;
```

# ¿Qué identificadores podemos utilizar?

En java los identificadores (nombres de las variables y constantes) siguen ciertas reglas y convenciones.

-En Primer lugar los identificadores siempre deben empezar con minúscula.
-No pueden empezar con un número pero si contenerlo.
-No pueden comenzar o contener un carácter especial o de puntuación a excepción de dos '$' ó '_' (pueden ser utilizados más de una vez por identificador)
-Pueden contener acentos, aunque no es recomendable ya que según el sistema o la codificación del lenguaje humano que esté representando nos puede traer problemas
-En el caso de las constantes, si bien podemos definir el identificador con letras minusculas, por convención siempre van con mayúsculas en toda su extensión. Por ejemplo "MI_CONSTANTE"
- Los identificadores no pueden repetirse en el mismo bloque de código.
- En java se suele utilizar por convención la notación CamelCase para nombrar las variables por ejemplo: "variableEntera", "variableFlotante", etc. La primera letra de cada palabra va con mayúscula a excepción de la primera letra del identificador para respetar la primera regla.

```
//IDENTIFICADORES PERMITIDOS
int identificador = 2;
int identificador2 = 2;
int identificadór = 2;
int identi2FICADOR = 2;
int $identificador = 2;
int _identificador = 2;
int identificador$ = 2;
int identificador_ = 2;
int identi_ficador = 2;
int identi$ficador = 2;

final int CONSTANTE = 2;
final int CONS_TANTE = 2;
final int CONS$TANTE = 2;

//IDENTIFICADORES NO PERMITIDOS
int int = 2;
int 2variable = 2;
int iden?tificador = 2;
```

# ¿Qué son las literales y cuales podemos utilizar en JAVA?

Las literales son los valores hardcodeados (escritos explícitamente en el código) que se le asignan a nuestras variables o constantes. Y dependen del tipo de dato asociado a la variable por ejemplo en el caso de los booleanos sólo hay dos valores factibles true o false,  en el caso de los char podemos usar literales explícitas como **'L'** o en unicode **'\u0000'** o una secuencia de escape **'\n'** (algo que veremos más adelante)

```
boolean booleano_verdadero = true;
boolean booleano_falso = false;

char caracter = 'a';
char unicode = '\u0065';
char escapado = '\n';
```

# ¿Qué sucede con la literales para datos numéricos?

En cuanto a las literales numérica suceden cosas particulares. Por defecto las literales enteras (números sin coma) son tratados como enteros en todos los casos, y las literales decimales (números con coma) serán tratados por defecto como un double.
Ahora bien si por ejemplo declaramos una variable long y la literal entera supera el valor máximo admitido por una literal entera deberemos agregar al final una “L” mayuscula o minuscula, aunque se recomienda usar mayusculas para evitar confusiones con el carácter “1”( “l”).

En el caso de variables flotantes, siempre será necesario agregar una F al final, ya que el IDE interpretará que es un double y puede haber pérdida de precisión en la conversión. Y en el caso de las variables double podremos agregar una D al final para ser más claros aunque no hará falta ya que el IDE trata las literales decimales como double por defecto para todos los casos.

Veamos unos ejemplos:

```
//Literales enteras
int entero = 5;
long largo = 4560000000L;

//Literales decimales
float flotante = 6F;
double decimal_largo = 6.71;
```

# Sistemas de base no decimal para declarar literales

En JAVA está permitido utilizar distintas bases para representar las literales. Aunque lo común es declarar las literales como vimos anteriormente, en algunos casos muy particulares los problemas requieren que sus variables estén inicializadas en otros sistemas de numeración como el Hexadecimal, Binario y Octal

Si bien es importante saber que esto es posible en JAVA, no entraremos en detalle sobre su uso, ya que no es imprescindible para aprender a programar. Quedara a consideracion de cada estudiante si desea profundizar o no en el tema.

| Base de numeración | valor | valor decimal |
|---|---|---|
| Decimal | 100 | 100 |
| Octal | 0100 | 64 |
| Hexadecimal | 0x100 | 256 |
| Binario | 0b100 | 4 |

```
//Literales y representaciones
int decimal = 100;
int octal = 0100;
int hexadecimal = 0x100;
int binario = 0b100;
```

# Literales y notación científica

La notación científica es una forma de escribir números muy grandes o muy pequeños. Un número está escrito en notación científica cuando un número entre 1 y 10 se multiplica por una potencia de 10. Por ejemplo, 650,000,000 puede escribirse en notación científica como 6.5 × 10^8.

JAVA incorpora una manera sencilla de utilizar notación científica para representar números muy pequeños o muy grandes

```
//Literales en notacion cientifica
double notacion = 2.15E2;          // -> + 2.15  * 10 ^2
float cientifica = -50.445e-10F; // -> -50.445 * 10 ^10
```

# Facilitando la lectura de enteros

Los literales de tipo entero, en cualquier base, pueden contener el carácter "_" para facilitar la lectura del número.

Este carácter sólo puede aparecer entre dígitos.
El carácter _ no puede aparecer en los siguientes lugares:

- Al principio o final de un número.
- Junto a un punto decimal en un número de tipo real.
- Antes de los sufijos F ó L.
- Entre el 0X hexadecimal o el 0B binario

```
//Utilizando separador en las literales
long separadorLong = 2_014_120;
float separador_float = 3.14_15F;
int separador_hexa = 0xFF_EC_DE_5E;
int separador_binario = 0b11010010_01101001_10010100_10010010;
int separador_octal = 04_34;
```