

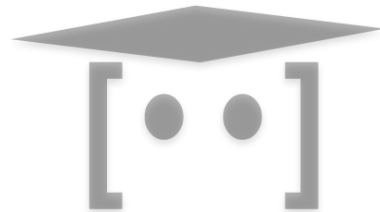
**JAVA**

Profesor: Nahuel Meza

**Cod.Ar[••]**

# ¿Qué vamos a ver hoy?

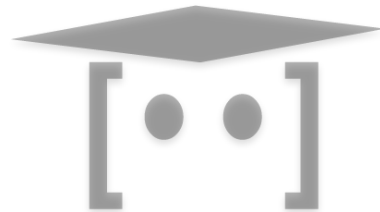
- Tipo de datos primitivos
- Variables
- Operadores
- Casting y conversión de tipos



# Cultura general

Como se sabe, en la computación, todo termina siendo una representación más abstracta de 1s y 0s.

Esto no escapa de los tipos de datos, y es muy importante por que suele marcar un rango de lo que puede representar u ocupar el dato.

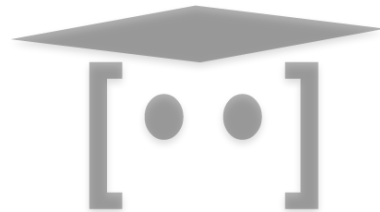


# Cultura general

Un bit es 1 o 0. Una cadena de bits, suele referenciar principalmente a un número.

$$1000110001 = 561$$

Donde el número se lee de derecha a izquierda. Y cada posición donde haya un 1, es una potencia de dos de esa posición.

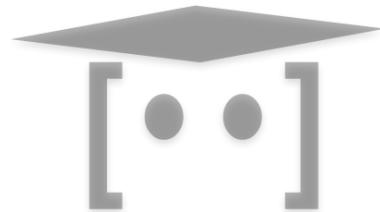


# Cultura general

$$1000110001 = 2^9 + 2^5 + 2^4 + 2^0 = 561$$

Una forma más fácil de ver el tamaño máximo de una cadena de bits, es pensar en su cantidad de elementos menos 1.

$$1111111111 = 2^{10} - 1 = 1023$$

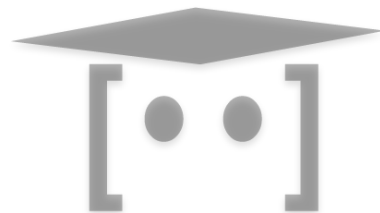


# Datos numéricos

Los más comunes:

***int*** | enteros 32 bits | (-2.147.483.648 a 2.147.483.647)

***long*** | entero 64 bits | ( $\pm 9$  cuatrillones)

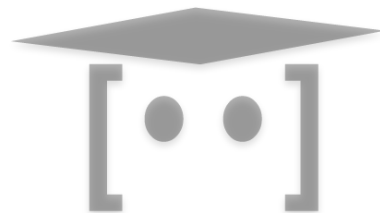


# Datos numéricos

Para decimales:

***float*** | decimal 32 bits | (hasta 6 decimales)

***double*** | decimal 64 bits | (hasta 15 cifras decimales)

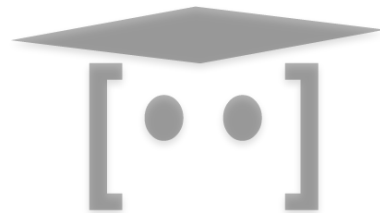


# Datos numéricos

Otros tipos:

***byte*** | entero 8 bits | (-128 a 127)

***short*** | entero 16 bits | (-32.768 a 32.767)

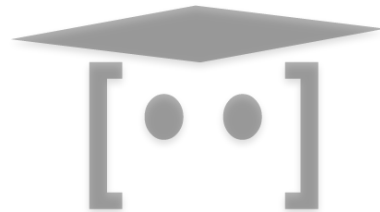




# Caracteres

***char*** | 16 bits | 'A', 'a', 'ñ', etc.

A diferencia de otros lenguajes, el carácter de java pesa 16 bits, ya que soporta unicode, lo que nos permite tener caracteres especiales (como la ñ).

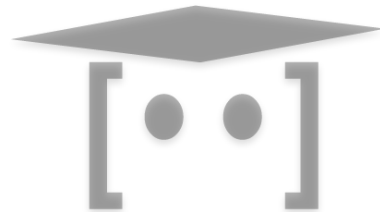


# Cadenas de texto

En java, no hay un tipo primitivo para cadenas de texto, lo que hay es una **clase primitiva**.

**String** | “Hello World”

(su peso en bits es la cantidad de caracteres x 16, más un peso extra correspondiente a ser una clase y no un tipo)

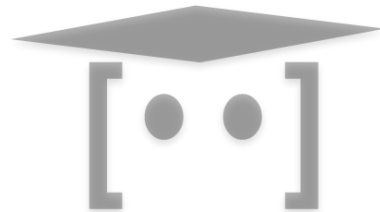


# Cadenas de texto

En java, no hay un tipo primitivo para cadenas de texto, lo que hay es una **clase primitiva**.

**String** | “Hello World”

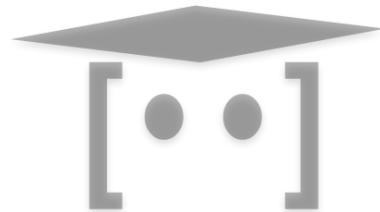
(su peso en bits es la cantidad de caracteres x 16, más un peso extra correspondiente a ser una clase y no un tipo)



# Booleans

**Bool** | 1 bit | true, false

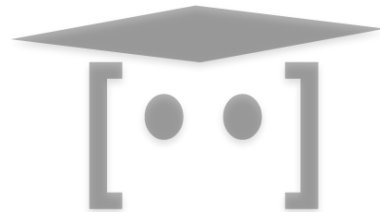
Su peso es tan poco pues, un bit en 0 es falso, y un bit en 1 es verdadero.



# Variables

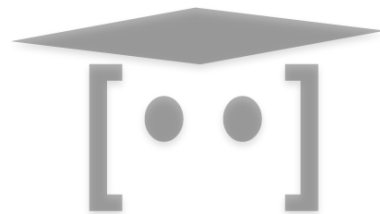
Una **variable** es un espacio en memoria, donde se guardan datos de un tipo asignado. Con un **nombre fijo** y cuyo valor puede **variar** a lo largo del programa  
Se escriben de la forma:

```
tipo variable = dato;
```



# Variables

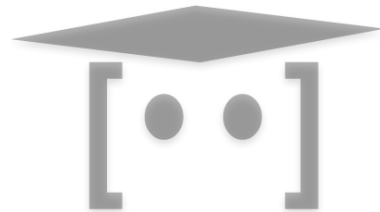
```
// Tipos enteros
int entero = 12;
byte bits8 = 1;
long enteroGrande = 100000L;
// Tipos decimales
float pi = 3.14f;
double piMas = 3.14159;
// Caracteres
char caracter = 'a';
String palabra = "Hola Mundo";
// Booleanos
boolean esVerdadero = true;
```



# Variables

Algunas reglas para la creación de variables:

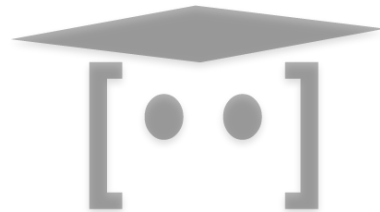
- No deben **empezar** con números
- No pueden ser palabras **reservadas**
- Son **case sensitive** (número != Número)
- Es buena práctica empezar la variable con **minúscula**
- Es buena práctica usar **camelCase** (esUnaVariable)



# Operadores

Los **operadores** son símbolos especiales que realizan una operación sobre dos o más variables/datos

Veremos los distintos tipos de operadores

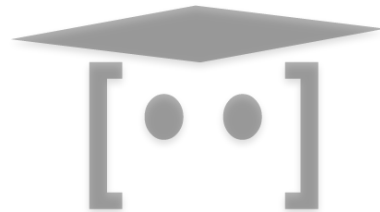




# Operadores Aritméticos

Estos son los típicos utilizados para operaciones aritméticas, los cuales devuelve un int o double:

- Suma: +
- Resta: -
- Multiplicación: \*
- División: /
- Módulo: %



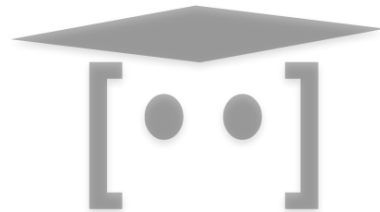
# Operadores Aritméticos

Estos se aplican de la forma: `variable1 + variable2`

Los únicos que pueden tener una implicación notable son la división y el módulo.

El operador `/` sirve para la **división entera y decimal**.

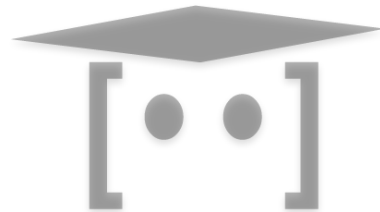
El operador `%` devuelve el **resto** de dividir dos variables.



# Operadores Relacionales

Estos son los típicos utilizados para operaciones en las que se precisa comparar elementos, devuelve un booleano:

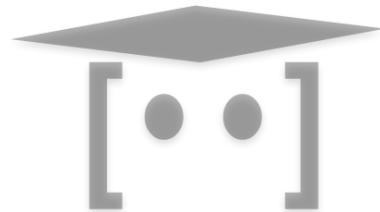
- Mayor que:  $>$
- Menor que:  $<$
- Mayor o igual que:  $>=$
- Mayor o menor que:  $<=$
- Iguales:  $==$
- Distintos:  $!=$



# Operadores Lógicos

Estos son los típicos utilizados para operaciones en las que se precisa comparar sentencias booleanas:

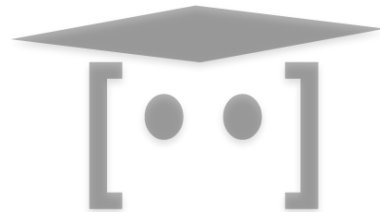
- **&&** : Da verdadero si en ' $a \ \&\& \ b$ ', tanto  $a$  como  $b$ , son verdaderas.
- **||** : Da verdadero si en ' $a \ || \ b$ ', ya sea  $a$  o  $b$ , alguno de los dos verdadero
- **!** : Da el valor contrario al actual, si  $a$  es verdadero  $!a$  es falso, y viceversa.



# Operadores de Asignación

Estos son los utilizados para asignar algún dato a una variable:

- `=` : Asignamiento común
- `+=`, `-=`, `*=`, `/=` : Asigna la operación de si mismo con lo asignado. Ej: `a += b`, es lo mismo que `a = a + b`



# Casting y conversión de Tipos

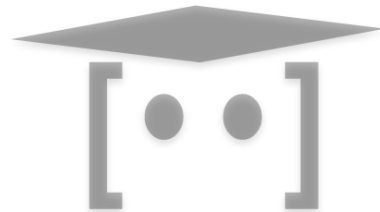
Al ser un lenguaje tipado, nos limita en cierto aspectos a la hora de codear. Sin embargo existen métodos para poder cambiar un tipo.

String -> int : Integer.parseInt(s)

String -> double: Double.parseDouble(s)

String -> boolean: Boolean.parseBoolean(s)

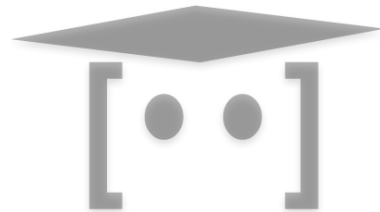
String -> long: Long.parseLong(s)



# Casting y conversión de Tipos

Existe el **casteo**, el cual es una herramienta que usaremos más adelante en POO. Pero se refiere a decirle al compilador “**Confía en mi que es este tipo**”.

Por ejemplo, no podríamos hacer la división entre un double y un int, pero estamos de acuerdo que  $1 == 1.0$



# Casting y conversión de Tipos

Por lo que podríamos hacer el cambio casteando el int a double.

```
double division = valorDouble / (double) valorInt
```

