



# TALLER 2

**EMANUEL TAMAYO  
JERONIMO TORO  
JUAN PABLO VELEZ LOPERA  
ESTEBAN PRESIGA POSADA**

```
background: url(../img/01.jpg)
background-size: 100vw 100vh
}
.box{
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  width: 400px;
  padding: 40px;
  background: rgba(0, 0, 0, 0.5);
  box-sizing: border-box;
  box-shadow: 0 15px 25px 0 #fff;
  border-radius: 10px;
}
.box h2{
  margin: 0 0 30px;
  padding: 0;
  color: #ffff;
  text-align: center;
}
.box h3{
  margin: 0 0 10px;
  padding: 0;
  color: #ffff;
  text-align: center;
}
.box .inputs{
  width: 100px;
  height: 30px;
  border: 1px solid #ffff;
  border-radius: 5px;
  font-size: 14px;
  font-family: sans-serif;
  margin-bottom: 10px;
}
```

# ¿QUE ES UNA VARIABLE?

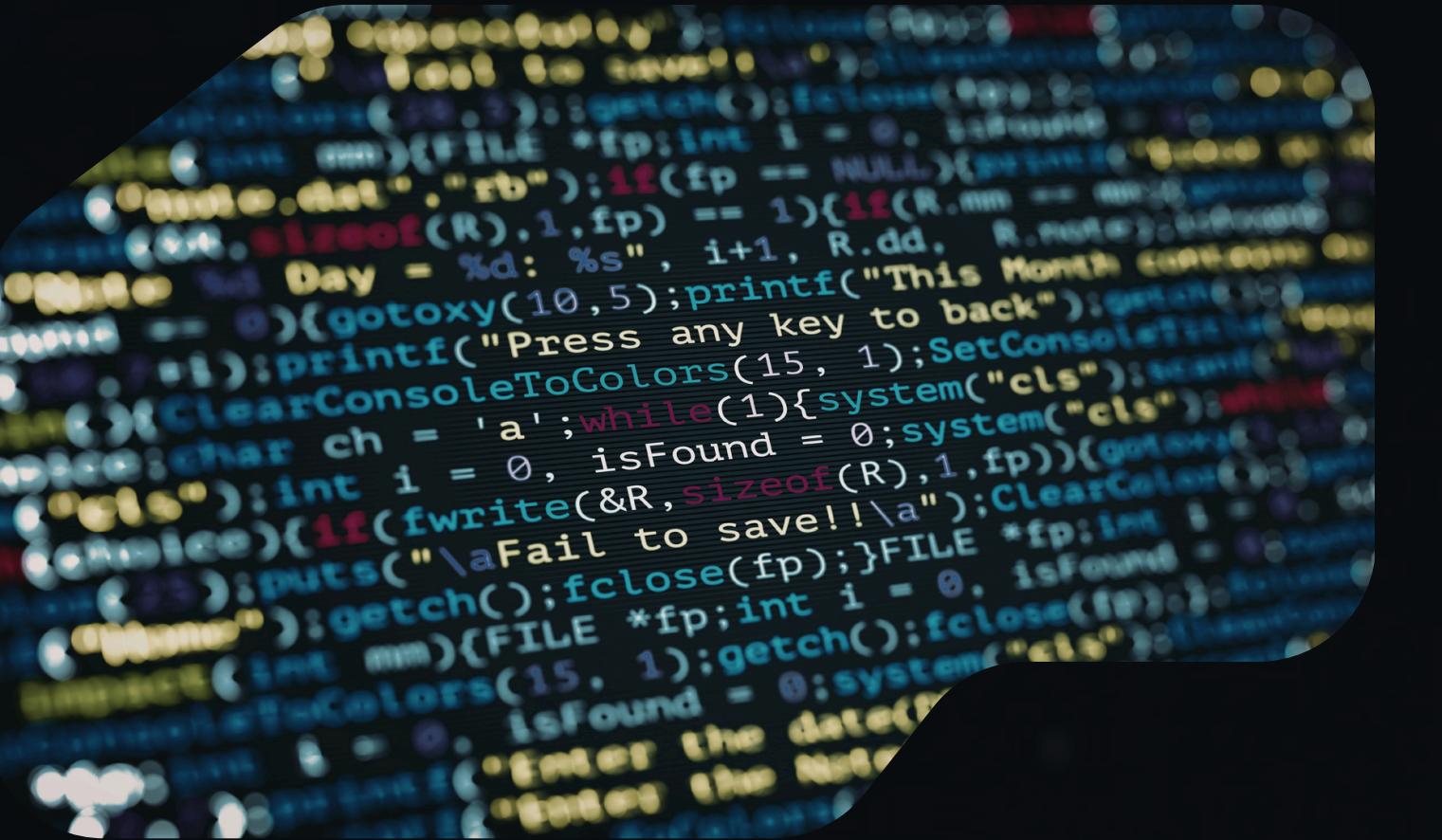
## DEFINICION

Una variable es un contenedor que almacena un valor o conjunto de valores en la memoria de un ordenador y les asigna un nombre único.

{x}



# TIPOS DE VARIABLES



## ALGUNOS TIPOS SON:

### Variables numéricas:

Almacenan valores numéricos, ya sean enteros o decimales. Por ejemplo, edad = 25 o precio = 10,99.



### Variables de texto:

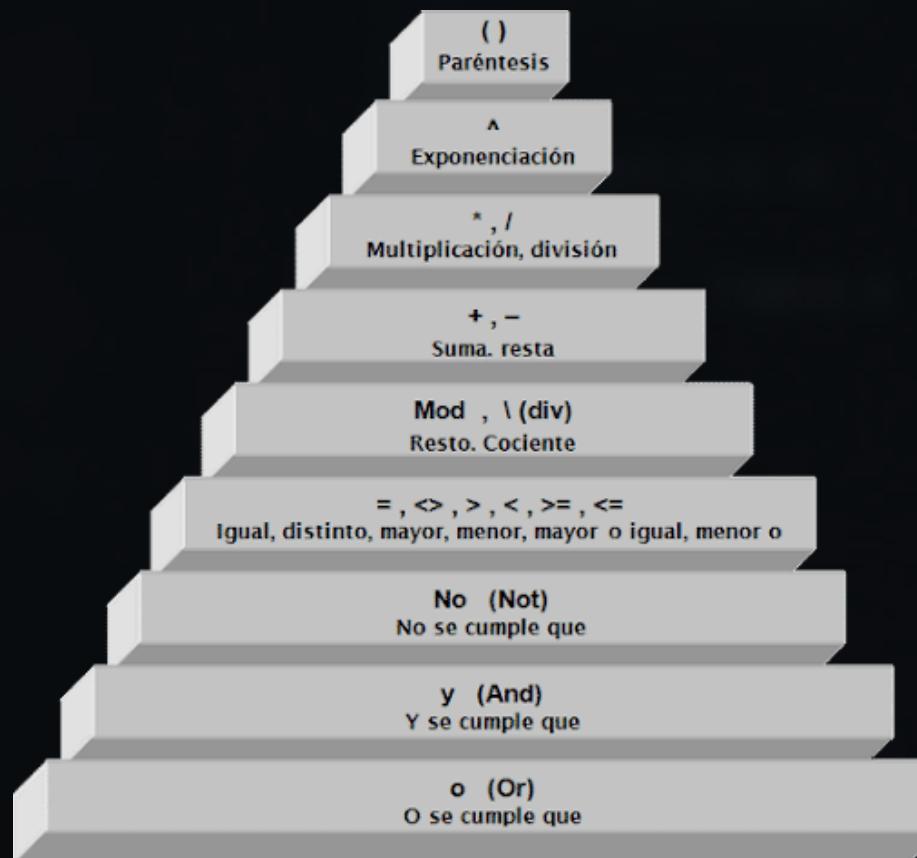
almacenan cadenas de caracteres. Por ejemplo, nombre = Juan o mensaje = Hola, mundo!.



### Variables booleanas:

Almacenan valores de verdadero (true) o falso (false). Por ejemplo, activo = true o validado = false.

# JAVA



# OPERADORES



Logica de  
Programacion



## 20 DECLARACIONES DE VARIABLES DE MULTIPLES TIPOS

```
public class taller3{  
    Run | Debug  
    public static void main(String[] args) {  
        try {  
  
            //Declaración e inicialización de 20 variables  
  
            byte edad = 18;  
            short año = 2026;  
            int poblacion = 1000000;  
            long distancia = 123456789L; //El "L" se debe poner a todas las variables de tipo Long porque si no, el codigo no compila  
            float estatura = 1.75f; //El "f" se debe poner a todas las variables de tipo float, si no, el codigo no compila  
            double peso = 68.5;  
            char inicial = 'E'; //char es un tipo de dato que almacena un solo caracter  
            boolean esEstudiante = true;  
            String nombre = "Ema";  
            String ciudad = "Medellin";  
            int numero1 = 10;  
            int numero2 = 20;  
            double promedio = 0.0;  
            float temperatura = 25.5f;  
            long sumalarga = 0L;  
            boolean mayorDeEdad = false;  
            char genero = 'M';  
            int codigo = 1234;  
            byte nivel = 5;  
            double salario = 1500.50;
```

# DECLARACIONES

# REASIGNACIONES

```
// Reasignaciones  
  
// Usando valores de otras variables  
promedio = (numero1 + numero2) / 2.0;  
sumaLarga = poblacion + distancia;  
mayorDeEdad = edad >= 18;  
salario = salario + promedio;  
codigo = codigo + nivel;
```



**REASIGNAMOS LOS VALORES DE AL MENOS 5 VARIABLES**

```
// Hard coded (datos quemados)
nombre = "Carlos";
ciudad = "Bogota";
nivel = 10;
codigo = 4321;
genero = 'F';
edad = 20;
año = 2027;
peso = 50.0;
inicial = 'J';
esEstudiante = false;
temperatura = 32.1f;
```



Hard Coded es una práctica en programación que consiste en incluir valores específicos directamente en el código en lugar de utilizar variables o funciones para obtenerlos de forma dinámica.

# HARD CODED





# CAMBIO NICK EN GIT

```
@@ -22,17 +22,17 @@
22  >El programa de java cuenta con 20 variables de diferente tipo iniciadas con datos compatibles con cada una y tambien reasigna los valores de varias
variables.

23
24  ** [!]Como renombrar commits:
25  - >Para cambiar el nombre solo para commits proximos, dejando los antiguos con el nombre que ya tenian, dentro del repositorio:
26  - >Comando: git config user.name "Nombre Deseado"
27  - >Comando : git config user.email "email deseado"
28  - >Ahora, para cambiar el commit directamente anterior, dentro del repositorio:
29  - >Comando: git commit --amend --reset-author

30  + > - Para cambiar el nombre solo para commits proximos, dejando los antiguos con el nombre que ya tenian, dentro del repositorio:
31  + > - Comando: git config user.name "Nombre Deseado"
32  + > - Comando : git config user.email "email deseado"
33  + > - Ahora, para cambiar el commit directamente anterior, dentro del repositorio:
34  + > - Comando: git commit --amend --reset-author

35
36  ** [*] Exposición:
37  >Explicación del proceso de realización del taller
```

## ¿COMO LO HACEMOS?

Para cambiar el nombre solo para commits proximos, dejando los antiguos con el nombre que ya tenian, dentro del repositorio:  
Comando: git config user.name "Nombre Deseado"  
Comando : git config user.email "email deseado"  
Ahora, para cambiar el commit directamente anterior, dentro del repositorio:  
Comando: git commit –amend –reset-author

# CODIGO CONTINUACION TALLER

```
em.out.println("Saliendo del sistema....");
};

em.out.println("Entrada no valida, porfavor vuelva
```

## ESTRUCTURA



```
    // Aquí el usuario podrá escoger 2 valores para dos variables X y Z, con
    // el fin de resolverlas
    System.out.println("El usuario podrá escoger 2 valores para dos variables X y Z, con
    // el fin de resolverlas");

    // ingrese: 1 Para escoger la ecuacion uno | 2 para escoger la ecuacion
    // de acuerdo a lo que elija el usuario
    Scanner scanner = new Scanner(System.in);
    System.out.print("Por favor ingrese el valor para la ecuación uno: ");
    int opcion = scanner.nextInt();

    if (opcion == 1) {
        System.out.print("Por favor ingrese el valor para la variable X: ");
        double x = scanner.nextDouble();
        System.out.print("Por favor ingrese el valor para la variable Z: ");
        double z = scanner.nextDouble();
    } else if (opcion == 2) {
        System.out.print("Por favor ingrese el valor para la variable X: ");
        double x = scanner.nextDouble();
        System.out.print("Por favor ingrese el valor para la variable Y: ");
        double y = scanner.nextDouble();
    } else {
        System.out.println("Entrada no valida, porfavor vuelva");
    }
```

```
    // Aquí se resuelve la ecuación 1
    resultado1 = ((3*x)/(1+(3*x)/(3*z*z+2)))/(1/((1/(1+z))+3*x*x+2*z+3));
    System.out.println("El resultado y de la ecuacion 1 es igual a: " + resultado1);
    break;

    case 2:
        a = ((x*x+3*z*z+2) / (2 + (1 / (1+2*z))));
        b = (2 / (1+(3*x*x*x)+3*z*z));
        c = 1 / (1 + 3 * x);
        d = ((3*x*x+1)/(2*x) + (3/1+(5/z)));
        resultado2 = a + (b / 1 / (c + d));
        System.out.println("El resultado y de la ecuacion 2 es igual a: " + resultado2);
        break;

    case 3:
        System.out.println("Saliendo del sistema....");
        break;

    default:
        System.out.println("Entrada no valida, porfavor vuelva");
    }
}
```

### Scanner

En esta ocasión usamos el Scanner para la solicitud de datos, en este caso, de la elección de ecuaciones y el valor de las variables X y Z.

### Switch

En este caso elegimos el Switch para tomar el valor antes ingresado con el Scanner como un caso posible y realizar multiples instrucciones, en este caso las ecuaciones.

## Ecuación 1

$$\Rightarrow \text{Ecuación (1)}$$

$$y = \frac{3x}{1+3x}$$

$$x=1 \quad z=2$$

$$y = \frac{3 \cdot (1)}{1+3 \cdot (1)}$$

$$y = \frac{3 \cdot (1)}{3 \cdot (2)^2 + 2}$$

$$y = \frac{1}{1+3 \cdot (1)^2 + 2 \cdot 2 + 3}$$

$$\frac{a}{c} = \frac{a \cdot d}{b \cdot c}$$

$$\Rightarrow y = \frac{3 \cdot 1}{\left(1 + \frac{3 \cdot 1}{3 \cdot 2^2 + 2}\right) \cdot 1}$$

$$\Rightarrow y = \frac{31}{14}$$

$$\Rightarrow y = \frac{31 \cdot 14}{14} = \frac{434}{14}$$

Decimal: 30.999999999999996

## Ecuación 2

$$\Rightarrow \text{Ecuación (2)}$$

$$y = \frac{x^2 + 3z + 2}{2 + \frac{1}{1+2z}}$$

$$x=1 \quad z=2$$

$$y = \frac{1^2 + 3(2) + 2}{2 + \frac{1}{1+2(2)}}$$

$$y = \frac{1^2 + 3(2) + 2}{2 + \frac{1}{1+4}}$$

$$y = \frac{1^2 + 3(2) + 2}{2 + \frac{1}{5}}$$

$$\Rightarrow y = \frac{1^2 + 3(2) + 2}{\frac{11}{5}} = \frac{45}{11}$$

$$\Rightarrow y = \frac{1^2 + 3(2) + 2}{1+3(1)} = \frac{33}{120}$$

$$\Rightarrow y = \frac{1^2 + 3(2) + 2}{1+3(1)} = \frac{1921}{440}$$

Decimal: 4.3659000000000005

## Implementación

$\Rightarrow$  Ecuaciones en variables de JAVA:

1.) Ecuación #1

$$((3^*x)/(1+(3^*x)/(3^*z^*z+2)))/(1/(1+z))+3^*x^*x+2^*z+3)$$

2.) Ecuación #2

$$a=((x^*x+3^*z+2)/(2+(1/(1+2^*))));$$

$$b=(2/(1+3^*x^*x)+3^*z+2);$$

$$c=1/(1+3^*x);$$

$$d=((3^*x+1)/(2^*z)+(3/1+(5/z)));$$

$$\text{Resultado: } a + (b/1/(c+d));$$



# EVIDENCIAS ECUACIONES



Logica de Programacion



# ECUACIONES →

## CASO 1 - ECUACION 1

```
switch (ecuacion) {  
    case 1:  
        resultado1 = ((3*x)/(1+(3*x)/(3*z*z+2)))/(1/((1/(1+z))+3*x*x+2*z+3));  
  
        System.out.println("El resultado y de la ecuacion 1 es igual a: " + resultado1);  
  
        break;
```

## CASO 2 - ECUACION 2

```
case 2:  
    a = ((x*x+3*z+2) / (2 + (1 / (1+2*z))));  
    b = (2 / (1+(3*x*x)+3*z+2));  
    c = 1 / (1 + 3 * x);  
    d = ((3*x+1)/(2*z) + (3/1+(5/z)));  
  
    resultado2 = a + (b / 1 / (c + d));  
    System.out.println("El resultado y de la ecuacion 2 es igual a: " + resultado2);  
  
    break;
```

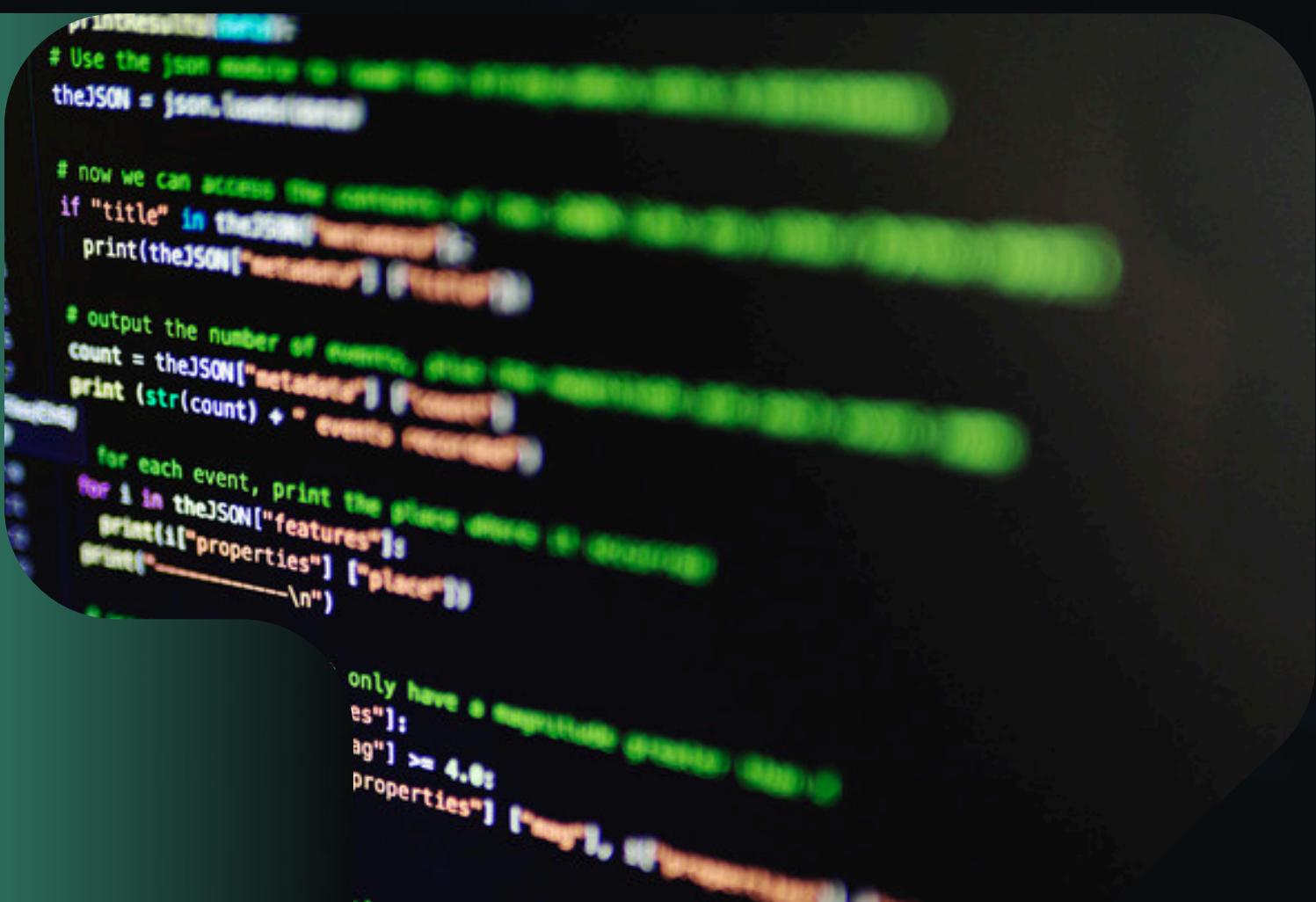
## CASO 3 - ECUACION 3

```
case 3:  
    System.out.println(x: "Saliendo del sistema....");  
    break;  
  
default:  
    System.out.println(x: "Entrada no valida, porfavor vuelva a intentarlo");
```

# REFERENCIAS



- Variables
- Hard Coding
- Repositorio



```
# Introducing JSON
# Use the json module to parse JSON
theJSON = json.loads(response)

# now we can access the contents of the JSON object
if "title" in theJSON["features"]:
    print(theJSON["features"]["title"])

# output the number of events after filtering
count = theJSON["metadata"]["eventCount"]
print(str(count) + " events recorded")

# for each event, print the place name & rating
for i in theJSON["features"]:
    print(i["properties"]["place"])
    print(i["properties"]["rating"])
    print("\n")

    # only have a negative rating
    if i["properties"]["rating"] < 0:
        print("This place has a negative rating!")

Properties = theJSON["features"]
```

# GRACIAS

