www.youtube.com/c/ProgramacionDesdeCero

Java • *hoja de referencia: lo básico*

TIPOS DE DATOS

boolean = true / false
int = 10
float = 4.75
double = 1.0002
char = 'D'
String = "¡hola!"

OPERADORES NUMÉRICOS

+	suma
-	resta
*	multiplicación
/	división
%	módulo
++	incremento en 1
--	decremento en 1

OPERADORES COMPARACIÓN

==	igual
!=	distinto
>	mayor
<	menor
>=	mayor o igual
<=	menor o igual

OPERADORES BOOLEANOS

&&	"y" lógico
||	"o" lógico
!	negación lógica

PROGRAMA: ESTRUCTURA BÁSICA

```
class Main {
   public static void main(String[] args)
   {  //código  }
}
```

VARIABLES

DECLARACIÓN

```
int radio;
```

ASIGNACIÓN

```
radio=20;
```

USO

```
radio*2;
```

STRINGS

CONCATENAR

```
"Hola " + "mundo"
```

OBTENER LONGITUD

```
"día".length()
```

CARÁCTER POSICIÓN 0

```
"Música"[0]
```

COMPARAR

```
strA.equals(strB)
```

BUCLES

FIJOS

```
for (int i=0; i<=10; i++) {
  System.out.println("Número: "+i); }
```

CONDICIONALES (0 O MÁS REPETICIONES)

```
String nombre = entrada.next();
while (!nombre.equals("Luis")) {
  System.out.println("Este no es Luis");
  nombre = entrada.next(); }
```

CONDICIONALES (1 O MÁS REPETICIONES)

```
int opcion;
do {
  opcion = entrada.nextInt();
} while (opcion < 1 || opcion > 5);
```

COMENTARIOS

```
//una línea
/*varias líneas*/
```

Programación Desde Cero

ENTRADA / SALIDA DE DATOS

OBJETO NECESARIO PARA LEER DATOS DE TECLADO

```
import java.util.Scanner;
```

```
Scanner entrada=new Scanner(System.in);
```

LEER Y GUARDAR UN STRING INGRESADO POR EL USUARIO

```
String nombre=entrada.next();
```

LEER Y GUARDAR UN NÚMERO INGRESADO POR EL USUARIO

```
int edad=entrada.nextInt();
```

IMPRIMIR (MOSTRAR) DATOS

```
System.out.println("Hola, mundo");
```

IMPRIMIR MÁS DE UN VALOR

```
System.out.println("Tu edad es "+edad);
```

DECISIONES

SIMPLES (DOS POSIBILIDADES)

```
int n=entrada.nextInt();
if (n==9) {
  System.out.println("¡Ganaste!"); }
else {
  System.out.println("No adivinaste"); }
```

MÚLTIPLES (MÁS DE DOS POSIBILIDADES)

```
int opcion=entrada.nextInt();
switch (opcion) {
  case 1: System.out.println("1"); break;
  case 2: System.out.println("2"); break;
  default: System.out.println("Ni 1 ni 2"); }
```

👆 else y default son opcionales

ARREGLOS

DECLARAR E INICIALIZAR

```
int A[]={1,2,3};
int B[]=new int[100];
int[][] C={ {1,2,3,4}, {5,6,7} };
```

ACCEDER AL ELEMENTO EN LA POSICIÓN 5

```
B[5]=100;
```

Java • *hoja de referencia: POO*

CLASE CON ATRIBUTOS Y CONSTRUCTOR

```
class Estudiante {
   private String nombre;
   private int legajo;

   public Persona(String nombre,int nLeg)
   {  //código  }
}
```

INSTANCIAR OBJETO

DECLARACIÓN

```
Estudiante juan;
```

INSTANCIACIÓN

```
juan=new Estudiante("juan","52245");
```

MODIFICADOR STATIC

ATRIBUTOS DE CLASE

```
static String escuela;
```

MÉTODOS DE CLASE

```
static void darNombreAEscuela(String n) {
   Estudiante.escuela=n;   }
```

INVOCAR MÉTODO DE CLASE

```
Estudiante.darNombreAEscuela("ABC");
```

👉no se requiere una instancia del objeto para utilizarlos.
👉un método static no puede usar atributos no static.

MODIFICADOR ABSTRACT

CLASE ABSTRACT

```
public abstract class Personal { }
```

👉no pueden ser instanciadas (sólo heredadas).

MÉTODOS ABSTRACT

```
public abstract double calcularSalario();
```

👉no pueden ser implementados (lo hace la clase hija).

INTERFACES

```
public interface Empleado {
   public long calcularAntiguedad(); }
```

👉sólo permiten el modificador public.

MODIFICADORES DE ACCESO

public	clase, paquete, subclase, resto del programa
protected	clase, paquete, subclase
sin modificador	clase, paquete
private	clase

GETTERS Y SETTERS PARA LOS ATRIBUTOS

GETTER

```
public int getLegajo(){
     return this.legajo;
}
```

SETTER

```
public void setLegajo(int nLeg){
     this.legajo=nLeg;
}
```

MODIFICADOR FINAL

ATRIBUTOS FINAL

```
private final long dni = 39243612;
```

👉su contenido no puede variar. Si es un objeto, no puede cambiar la referencia, pero sí el estado del objeto referenciado.

MÉTODOS FINAL

```
private final calcularPromedio() { }
```

👉no puede ser sobreescrito por las subclases.

CLASES FINAL

```
public final class Asignatura { }
```

👉no pueden ser heredadas (pero sí instanciadas).

COLECCIONES

Contenedor	Orden de los elementos	Acceso aleatorio	Pares clave-valor	Permite duplicados	Permite valores nulos
HashSet	✗	✓	✗	✗	✓
TreeSet	✓	✓	✗	✗	✗
LinkedHashSet	✓	✗	✗	✗	✓
ArrayList	✓	✓	✗	✓	✓
LinkedList	✓	✗	✗	✓	✓
Vector	✓	✓	✗	✓	✓
LinkedBlockingQueue	✓	✗	✗	✓	✗
ArrayBlockingQueue	✓	✗	✗	✓	✗
PriorityBlockingQueue	✓	✗	✗	✓	✗
DelayQueue	✓	✗	✗	✓	✗
SynchronousQueue	✓	✗	✗	✓	✗
HashMap	✗	✓	✓	✗	✓
TreeMap	✓	✓	✓	✗	✗
LinkedHashMap	✓	✗	✓	✗	✓

Programación Desde Cero