

PALABRAS RESERVADAS

JAVA

PALABRAS RESERVADAS

PALABRAS UTILIZADAS PARA PODER PROGRAMAR EN JAVA.

TITULO

PALABRA ETIMOLOGICA.

Concepto de que es y donde debe de aplicarse.

EJEMPLO

Ejemplo con algun tipo de programa.

SINTAXIS

```
variable_1 := 3;  
(* grupo de instrucciones 1 *)  
  
si (variable_1 = 5) entonces  
hacer  
(* grupo de instrucciones 2 *)  
fin_hacer  
  
en_otro_caso  
hacer  
(* grupo de instrucciones 3 *)  
fin_hacer
```

SWITCH

PROVIENE DEL ANTIGUO INGLES, "SWICCE" SIGNIFICA, CAMBIAR O ALTERNAR.

Es utilizado para ejecutar diferentes bloques de Código segun el valor de alguna Variable, switch debe aplicarse en proyectos donde el usuario quiera realizar cierta accion que lleva a la funcionalidad de cierto de bloque de codigo por medio del valor de una variable.

EJEMPLO

Una calculadora la cual necesite un menu para saber que operacion necesito realizar, como

1. Suma
2. Resta
3. Division
4. Multiplicacion.

SINTAXIS

```
switch (variable)
{
    case valor1:
        //instrucciones
        break;
    case valor2:
        //instrucciones
        break;
    case valor3:
        //instrucciones
        break;
    default:
        //instrucciones
        break;
}
```

CASE

PROVIENE DEL LATIN, “CASUS” SIGNIFICA OCASIÓN O EVENTO.

Case es utilizado para definir opciones específicas dentro de una estructura switch. Case debe aplicarse en proyectos los cuales necesiten diferentes opciones para ejecutar cierta acción por medio del switch y la el dato que se ingresara.

EJEMPLO

Podemos tomar de ejemplo un menu de meses del año:

Case Enero:

imprimir("Elegiste enero.")

Case Febrero:

Imprimir("Elegiste Febrero.")

SINTAXIS

```
int roll = 3 ;  
switch( roll )  
{  
    case 1 :  
        printf("I am Pankaj");  
        break;  
    case 2 :  
        printf("I am Nikhil");  
        break;  
    case 3 :  
        printf("I am John");  
        break;  
    default :  
        printf("No student found");  
        break;  
}
```



DEFAULT

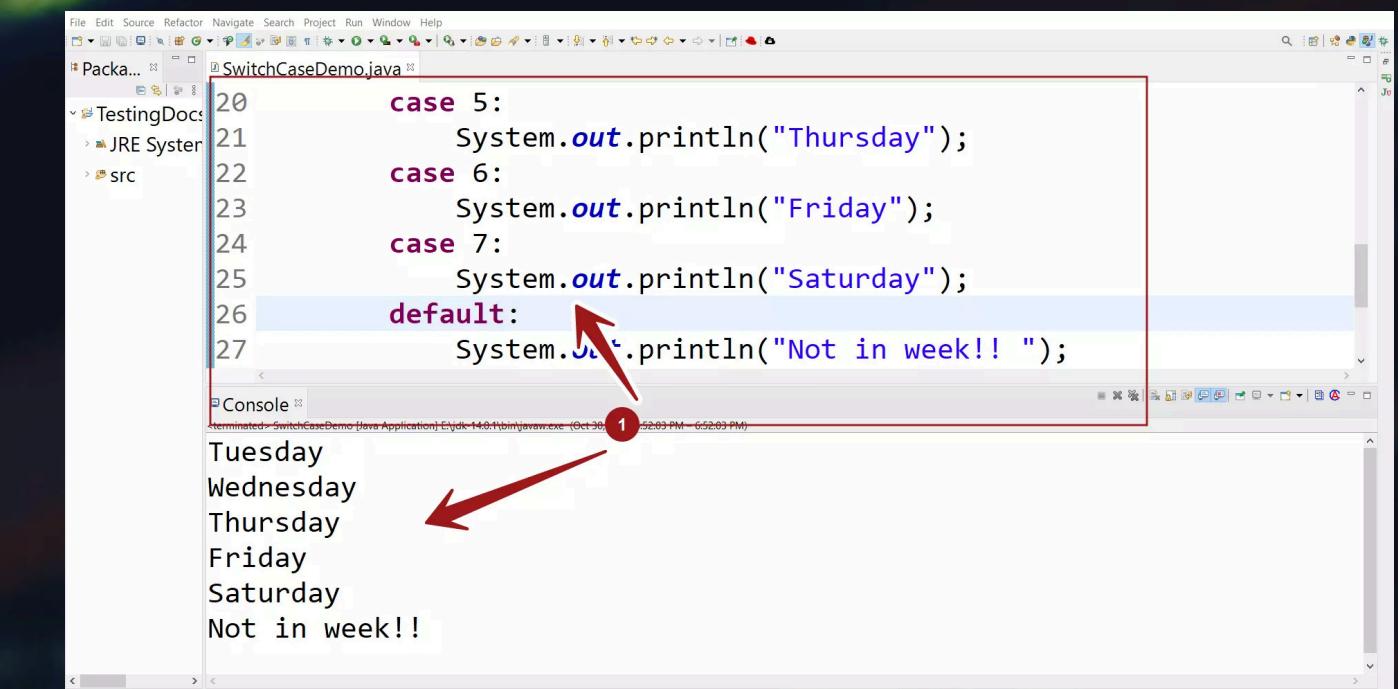
PROVIENE DEL LATIN “DEFECTUS”, SIGNIFICA FALTA O PREDETERMINADO.

Defatult es utilizado en un switch, funcionando de manera de que se ejecutara si en un switch ninguna de las opciones de los case coincide con los datos ingresados. Este debe aplicarse en proyectos los cuales cuenten con un switch, para asi evitar que el programa falle al no ingresarse datos que coincidan con los case.

EJEMPLO

Usaremos de ejemplo una calculadora con un menu, en el cual si no se especifica que operacion quiero realizar, mostrara un mensaje que diga que eliga una opcion valida.

SINTAXIS



```
case 5:  
    System.out.println("Thursday");  
case 6:  
    System.out.println("Friday");  
case 7:  
    System.out.println("Saturday");  
default:  
    System.out.println("Not in week!! ")
```

Tuesday
Wednesday
Thursday
Friday
Saturday
Not in week!!

IF

PROVIENE DEL INGLES ANTIGUO “GIF”, SIGNIFICA SI.

If es utilizado para evaluar una condición booleana la cual, si es verdadera ejecutara una accion, pero si es falsa ejecutara otra. If debe aplicarse en proyectos donde nosotros necesitemos que evalúe cierta información para ver si esta es correcta o falsa, para así poder ya sea ejecutar o no ejecutar un bloque de código.

EJEMPLO

podemos tomar de ejemplo un login sencillo, en el cual la variable esta definida como 123, si el dato ingresado es diferente a 123, no lo dejara ingresar mostrando un mensaje, pero si el dato ingresado si es 123, si se le dejara entrar.

SINTAXIS

```
if (expresión lógica) {  
    sentencias  
}  
else {  
    sentencias  
}
```

ELSE

PROVIENE DEL INGLES ANTIGUO “ELLES”, SIGNIFICA OTRO O DE OTRO MODO.

Else es utilizado cuando la condición if resulta ser falsa, ejecutando un bloque de código específico para esa situación, el else debe cuando un proyecto necesite validar la información que es ingresada, para que esta cuando sea falsa, no detenga el programa y muestre alguna acción de un bloque de código.

EJEMPLO

De ejemplo podemos utilizar una división, la cual si se divide por 1, continue normal, pero si se divide por 0, muestre un mensaje de error.

SINTAXIS

```
if (promedio>=10.5) {  
    situacion="Aprobado";  
}  
else {  
    situacion="Desaprobado";  
}
```

Condición
Instrucción
Variable
Comando

TRY

PROVIENE DEL LATIN “TRACTARE”, SIGNIFICA INTENTAR.

Try es utilizado para manejar errores los cuales se colocan dentro de este , evitando asi que el programa se detenga cuando cierta accion salga mal, try debe aplicarse cuando necesitemos evitar errores en un proyecto, try nos ayudara a tener previstos estos errores.

EJEMPLO

Podemos tomar de ejemplo un programa el cual necesite que ingresemos un dato, en el cual si no se ingresa nada, muestre un error, en el cual un catch se encargara de eso.

SINTAXIS

```
try {  
    //bloque codigo  
}  
catch (Exception e) {  
    //Captura error  
}  
finally {  
    //limpieza y liberación memoria  
}
```

CATCH

PROVIENE DEL LATIN “CAPTARE”, SIGNIFICA ATRAPAR.

El catch se encarga de capturar el error el cual fue lanzado en un bloque try, evitando que se detenga el programa, catch debe aplicarse en proyectos los cuales deben evitar errores sin importar lo minimo que sean, catch nos ayudara a evitar estos errores capturándolos.

EJEMPLO

En catch podemos usar de ejemplo un login, en el cual despues de ingresar los datos, verificará si es correcto o no, si no es correcto catch se encargara de atrapar el error con un mensaje que detalle el error, sin detener el programa.

SINTAXIS

```
// ...
try
{
    // código fuente que lanza una excepción
}
catch (excepcion 1)
{
    // se produjo la excepción 1, lo atrapa y se maneja
}
catch (excepcion 2)
{
    // se produjo la excepcion 2, lo atrapa y se maneja
}
```

LONG

PROVIENE DEL LATIN “LONGUS”, SIGNIFICA LARGO.

Long es un tipo de dato entero , este tipo de dato deja ingresar un valor maximo de $2^{63} - 1$. Long debe de aplicarse cuando necesitemos ingresar una cantidad grande de numeros en un proyecto.

EJEMPLO

Podemos tomar de ejemplo un programa de un banco que maneje una cantidad grande de transacciones de dinero.

SINTAXIS

```
1  public class Demo {  
2  
3  
4  public static void main(String[] args) {  
5  
6      long h = 9L;  
7      System.out.println(h);  
8  }  
9  
10 }  
11 }
```

WHILE

PROVIENE DEL INGLES ANTIGUO “HWILE”, SIGNIFICA MIENTRAS.

While se encarga de iniciar un bucle que se ejecuta cuando una cierta condicion sea verdadera. While debe aplicarse en proyectos donde necesitemos mostrar cierto dato o datos mientras la condicion sea verdadera, esto para evitar mostrar datos de forma individual, para asi automatizar mas el codigo.

EJEMPLO

Podemos utilizar de ejemplo una cuenta regresiva de 10 segundos , para que mientras 10 sea mayor a 0, muestre la cuenta regresiva hasta llegar a 0.

SINTAXIS

```
int contador = 1;

//ciclo while
while( contador <= 10 )
{
    System.out.println( contador );
    contador++; //incremento 1 a contador
    //puede usarse tambien contador = contador + 1;
}
```

DO

PROVIENE DEL LATIN “FACERE”, SIGNIFICA HACER U EJECUTAR.

El Do acompaña al while, iniciando un ciclo do-while, el cuale ejecuta el bloque de codigo al menos una vez antes de evaluar la condicion, Do debe aplicarse cuando se necesite una informacion especifica, la cual si no es esa, necesite volver a pedirla sin volver a iniciar el programa.

EJEMPLO

Podemos tomar de ejemplo un programa el cual al escribir cualquier numero entre al bucle y mientras se presionen numeros diferentes a 0 siga en el bucle, saldra del programa cuando se escriba 0.

SINTAXIS

```
package estrepeticion;
public class CicloDoWhile {
    public static void main(String[] args) {
        int contador = 0;
        do
        {
            System.out.println(contador);
            contador++;
        }
        while (contador<=10);
    }
}
```

BREAK

PROVIENE DEL LATIN “BRACCHIUM”, SIGNIFICA ROMPER O INTERRUMPIR.

Break es una palabra reservada eutilizada en los bucles para finalizarlos, saliendo de la estructura. Break debe aplicarse en los proyectos que lleven bucles, para que cuando se cumpla una accion dentro de este salga inmediatamente de este, finalizando el proceso.

EJEMPLO

Podemos tomar de ejemplo el programa donde al ingresar un mes del año te muestre un mensaje de que lo elegiste, en este caso break se encargara de salir del programa.

SINTAXIS

```
while (testExpression) {  
    // codes  
    if (condition to break) {  
        break;  
    }  
    // codes  
}  
  
do {  
    // codes  
    if (condition to break) {  
        break;  
    }  
    // codes  
}  
while (testExpression);  
  
for (init; testExpression; update) {  
    // codes  
    if (condition to break) {  
        break;  
    }  
    // codes  
}
```

FOR

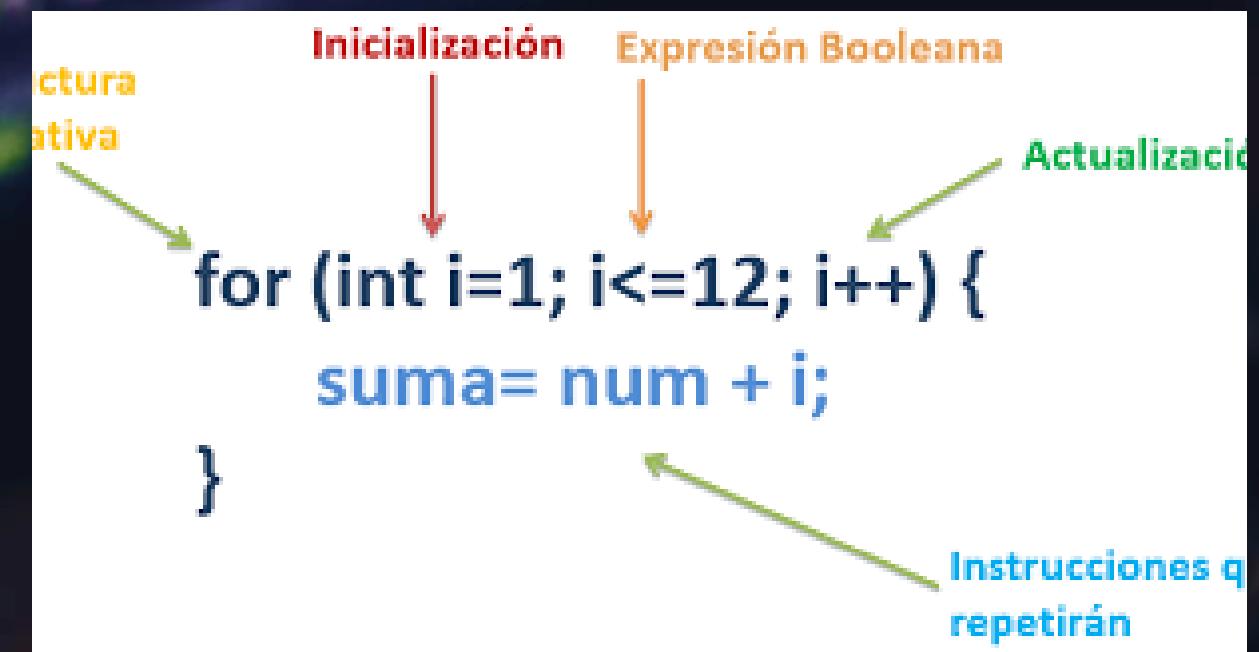
PROVIENE DEL LATIN “PRO”, SIGNIFICA PARA.

for es utilizado para realizar un bucle el cual se repite una cierta cantidad de veces, este cuenta con la inicializacion, condicion y actualizacion. For debe aplicarse cuando en un proyecto necesitemos que imprima una cierta cantidad de veces la informacion ingresada.

EJEMPLO

Podemos utilizar de ejemplo un programa el cual imprima la tabla de un numero ingresado, for se encargara de que el el numero se repita hasta 12 veces con los resultados, esto gracias al inicializador que es 1, el cual si es menor a 12, debera ir incrementando hasta 12, para que asi en cada bucle se multiplique con el numero ingresado.

SINTAXIS



BOOLEAN

PROVIENE DEL MATEMATICO GEORGE BOOLE, QUIEN CREO LA ALGEBRA BOOLEANA.

Este es un tipo de dato que solo admite los valores de falso y verdadero, este es muy utilizado en las estructuras condicionales. Este debe de aplicarse en programas en los cuales deba de especificarse si un valor es verdadero o falso, para asi mejorar su uso en el condicional, haciendolo mas sencillo.

EJEMPLO

Tomemos de ejemplo un acceso al programa usando if, en el cual solo entrara si es verdadero, el booleano “acceso” es verdadero, por lo cual la condicion if lo dejara entrar, pero si se cambia el booleano a falso, mostrara un else el cual le dira que no puede acceder.

SINTAXIS

```
public static void main(String[] args) {  
    boolean b1 = true;  
    boolean b2 = false;  
    boolean b3 = 2 < 3; // less than operator  
    boolean b4 = 2 > 3; // greater than operator  
  
    System.out.println(b1); // true  
    System.out.println(b2); // false  
    System.out.println(b3); // true  
    System.out.println(b4); // false  
}
```

THROW

PROVIENE DEL INGLES ANTIGUO “THROWAN”, SIGNIFICA LANZAR O ARROJAR.

Throw es una palabra clave la cual lanzara una excepcion, es decir un error. Throw debe de utilizarse cuando trabajemos con errores personalizados, para asi disminuir el riesgo de tener errores en nuestro codigo.

EJEMPLO

Tomemos de Ejemplo un programa que nos pedira ingresar un numero, si no ingresamos nada, utilizaremos un throw para decir que debemos ingresar un dato,para que asi no crashee el programa.

SINTAXIS

```
1 public void readfile(String fileName) throws FileNotFoundException
2     File file = new File(fileName);
3     if (!file.exists()) {
4         throw new FileNotFoundException("File not found");
5     }
6     // Rest of the code to read the file
7 }
```

PUBLIC

PROVIENE DEL LATIN “PUBLICUS”, SIGNIFICA DEL PUEBLO O COMUN,

Public sirve para permitir que un miembro de la clase sea accesible a cualquier parte del programa. Public debe de aplicarse cuando queremos que una funcion o metodo sea publico y pueda usarse en cualquier parte del programa.

EJEMPLO

Es como por ejemplo crear la funcion de suma, la cual sumara dos numeros, si queremos que esta funcione en todo el codigo debemos declararla como public int sumar (int a, int b) para que funcione en todo el codigo.

SINTAXIS

```
// simplest Java class
class Student { }

// in file Student.java
public class Student {
    String name;
    public String getName() {
        return name;
    }
    public void setName(String theName) {
        name = theName;
    }
}
```

return type

signature

IMPORT

PROVIENE DEL LATÍN “IMPORTĀRE”, SIGNIFICA TRAER.

Import se utiliza para acceder a clases y paquetes externos, para así facilitar el reutilizar código. Import lo podemos aplicar para cuando tengamos alguna función de un proyecto y la queremos pasar a otro proyecto, para así poder importar una función de un proyecto a otro por medio de un package.

EJEMPLO

Podemos tomar de ejemplo la operación suma de proyecto1, para pasársela proyecto2, simplemente ponemos proyecto1.sumar, porque sumar es la clase suma, al cual tiene todo lo necesario para poder sumar.

SINTAXIS

```
package paquete.mipaquete;  
  
import paquete3.otropaque2.MiClase;  
  
class Ejemplo2{}
```

PACKAGE

PROVIENE DEL ANGLOLATÍN “PACCAGIUM”, SIGNIFICA PAQUETE

Son un conjunto de clases, los cuales ayudan a la gestión de grandes proyectos. Estos deben de aplicarse cuando en un proyecto grande debamos pasar alguna función a otro proyecto, para así poder organizar el código.

EJEMPLO

Podemos tomar como ejemplo sumar, donde lo volvemos un package para pasarlo a tareas mate, donde simplemente pondremos sumar.suma, porque suma es la clase que tendrá todo lo necesario para poder sumar.

SINTAXIS

```
package com.mcnz.example;

import java.util.*; // wildcard import

public class JavaScannerImportExample {

    public static void main(String args[]) {

        System.out.println("What is your favorite color?");
        Scanner stringScanner = new Scanner(System.in);
        String color = stringScanner.next();
        System.out.println(color + " is my favorite color too!");
        stringScanner.close();
    }
}
```

FINAL

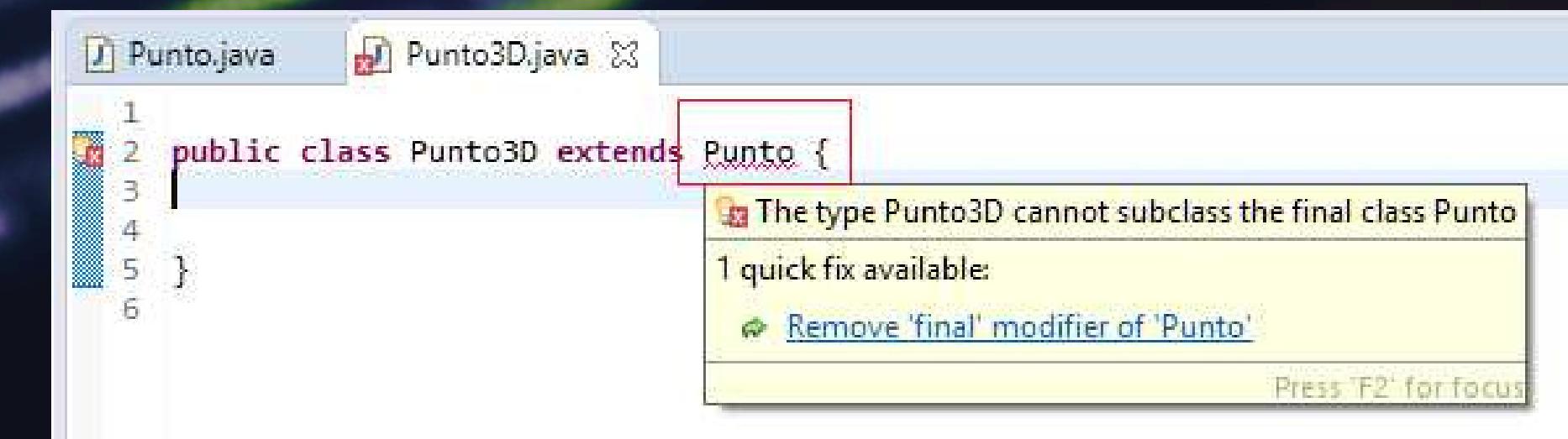
PROVIENE DEL LATIN “FINALIS”, SIGNIFICA FIN O LIMITE.

Final es utilizado para declarar en un metodo, clase o variable, cuando se aplica el final no puede ser editado. Aplicamos el final cuando necesitemos que algun metodo, clase o variable no tenga cambios.

EJEMPLO

Tomemos de ejemplo una variable edad=30, como la variable ya esta definida en 30, no puede ser cambiada.

SINTAXIS



BYTE

PROVIENE DE LA PALABRA BITE, LA CUAL SIGNIFICA MORDISCO.

Es un tipo de dato de cantidad de almacenamiento de -128 a 127, este un tipo de dato entero, Byte se aplica cuando en algun proyecto trabajemos con numeros de poca cantidad.

EJEMPLO

Podemos tomar de ejemplo un programa que imprima los numeros hasta 127 con un for, llegara hasta 127 porque la cantidad que numeros que aceptara byte.

SINTAXIS

```
public class Ejemplo {  
    byte miByteValorMin=-128;  
    byte miByteValorMax=127;
```

INT

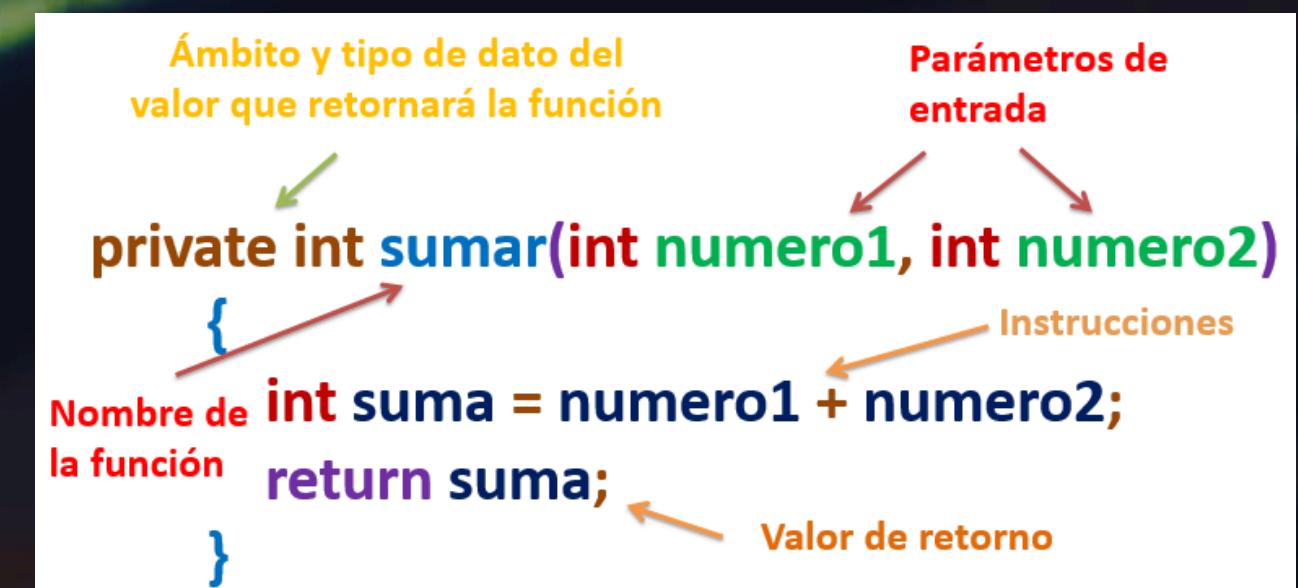
PROVIENE DE LA PALABRA “INTEGER”, LA CUAL PROVIENE DEL LATINO “INTEGRARE”, LA CUAL SIGNIFICA COMPLETO.

Int es un tipo de dato el cual tiene la capacidad de almacenar -2,147,483,648 hasta 2,147,483,647 numeros. Int debe aplicarse cuando nosotros vayamos a trabajar en un proyecto normal, el cual no pida una cantidad grande de numeros.

EJEMPLO

Tomemos de ejemplo una app de mensajeria, la cual contara el numero de correos enviados, esta app de mensajeria es en base a un correo pequeño, para no tener una cantidad grande de correos.

SINTAXIS



```
import './index.css';
import { ReactComponent as ArrowIcon } from './Assets/Icons/arrow.svg';
import { ReactComponent as BottomIcon } from './Assets/Icons/bottom.svg';
import { ReactComponent as RightArrowIcon } from './Assets/Icons/right-arrow.svg';
import React, { useState, useEffect, useRef } from 'react';
import { CSSTransition } from 'react-transition-group';
```