¿ETIQUETAS DE DOC EN JAVA?

En Java, las etiquetas de documentación se utilizan para generar documentación API a partir del código fuente. Estas etiquetas se utilizan en comentarios de estilo Javadoc.

Las más comunes son:

1. **@author**: Indica el autor de la clase o método.

/\*\*

\* @author Juan Pérez

\*/

1. **@version**: Especifica la versión de la clase o método.

/\*\*

\* @version 1.0

\*/

1. **@param**: Describe un parámetro de un método.

/\*\*

\* @param x El valor de entrada

\*/

1. **@return**: Describe el valor devuelto por un método.

/\*\*

\* @return El resultado de la operación

\*/

1. **@throws** o **@exception**: Indica qué excepciones puede lanzar un método.

/\*\*

\* @throws IOException Si ocurre un error de entrada/salida

\*/

1. **@see**: Proporciona referencias a otras clases o métodos relacionados.

/\*\*

\* @see OtraClase

\*/

1. **@deprecated**: Indica que un método o clase ya no se recomienda su uso.

/\*\*

\* @deprecated Usar el método nuevo()

\*/

1. **@since**: Indica desde qué versión está disponible un método o clase.

/\*\*

\* @since 1.5

\*/

1. **@todo**: Marca un elemento que necesita trabajo futuro.

/\*\*

\* @todo Implementar la lógica de validación

\*/

Un ejemplo completo podría ser:

/\*\*

\* Clase para realizar operaciones matemáticas.

\*

\* @author Juan Pérez

\* @version 1.0

\* @since 2023

\*/

public class Calculadora {

/\*\*

\* Suma dos números.

\*

\* @param a El primer número

\* @param b El segundo número

\* @return La suma de a y b

\* @throws IllegalArgumentException Si a o b son negativos

\*/

public int sumar(int a, int b) {

if (a < 0 || b < 0) {

throw new IllegalArgumentException("Los números no pueden ser negativos");

}

return a + b;

}

}

Utilizando estas etiquetas, se pueden crear documentación clara y útil para otros desarrolladores.

¿CUAL ES EL USO DE JOPTIONPANE: LEA/ IMPRIMIR?

JOptionPane es una clase en Java que se utiliza para mostrar cuadros de diálogo en aplicaciones gráficas (GUI). Permite a los desarrolladores interactuar con el usuario a través de diálogos para leer entradas o mostrar mensajes.

**Uso de JOptionPane**

1. **Leer Entrada (Input)**

Para leer información del usuario, puedes usar JOptionPane.showInputDialog(), que muestra un cuadro de diálogo donde el usuario puede ingresar texto.

String nombre = JOptionPane.showInputDialog("Introduce tu nombre:");

1. **Imprimir Mensajes (Message)**

Para mostrar un mensaje al usuario, puedes usar JOptionPane.showMessageDialog(). Esto muestra un cuadro de diálogo con un mensaje informativo.

JOptionPane.showMessageDialog(null, "Hola, " + nombre + "!");

**Ejemplo**

import javax.swing.JOptionPane;

public class EjemploJOptionPane {

public static void main(String[] args) {

// Leer entrada

String nombre = JOptionPane.showInputDialog("Introduce tu nombre:");

// Mostrar mensaje

JOptionPane.showMessageDialog(null, "Hola, " + nombre + "!");

}

}

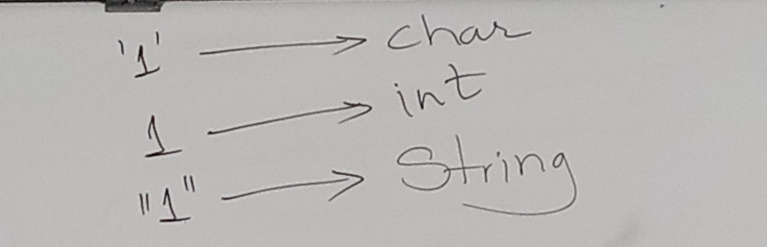
* **showInputDialog**: Muestra un cuadro de diálogo donde el usuario puede ingresar su nombre. El valor ingresado se almacena en la variable nombre.
* **showMessageDialog**: Muestra un mensaje que incluye el nombre ingresado por el usuario.

¿QUE SON CLASES ENVOLTORIOS?

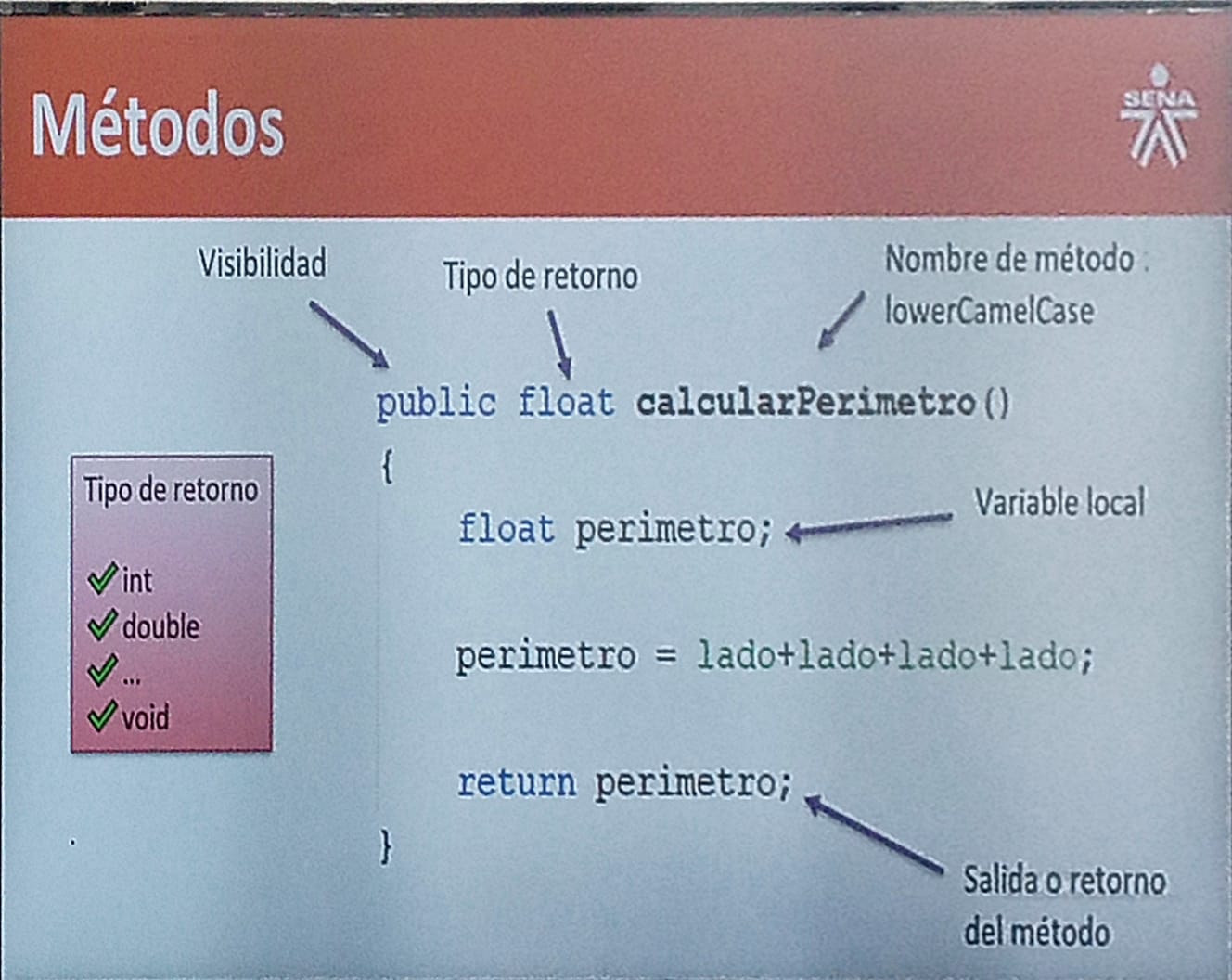
CLASES ENVOLTORIOS 🡪Se conocen porque son( clases) y (permiten trabajar tipos primitivos) (atreves de métodos).

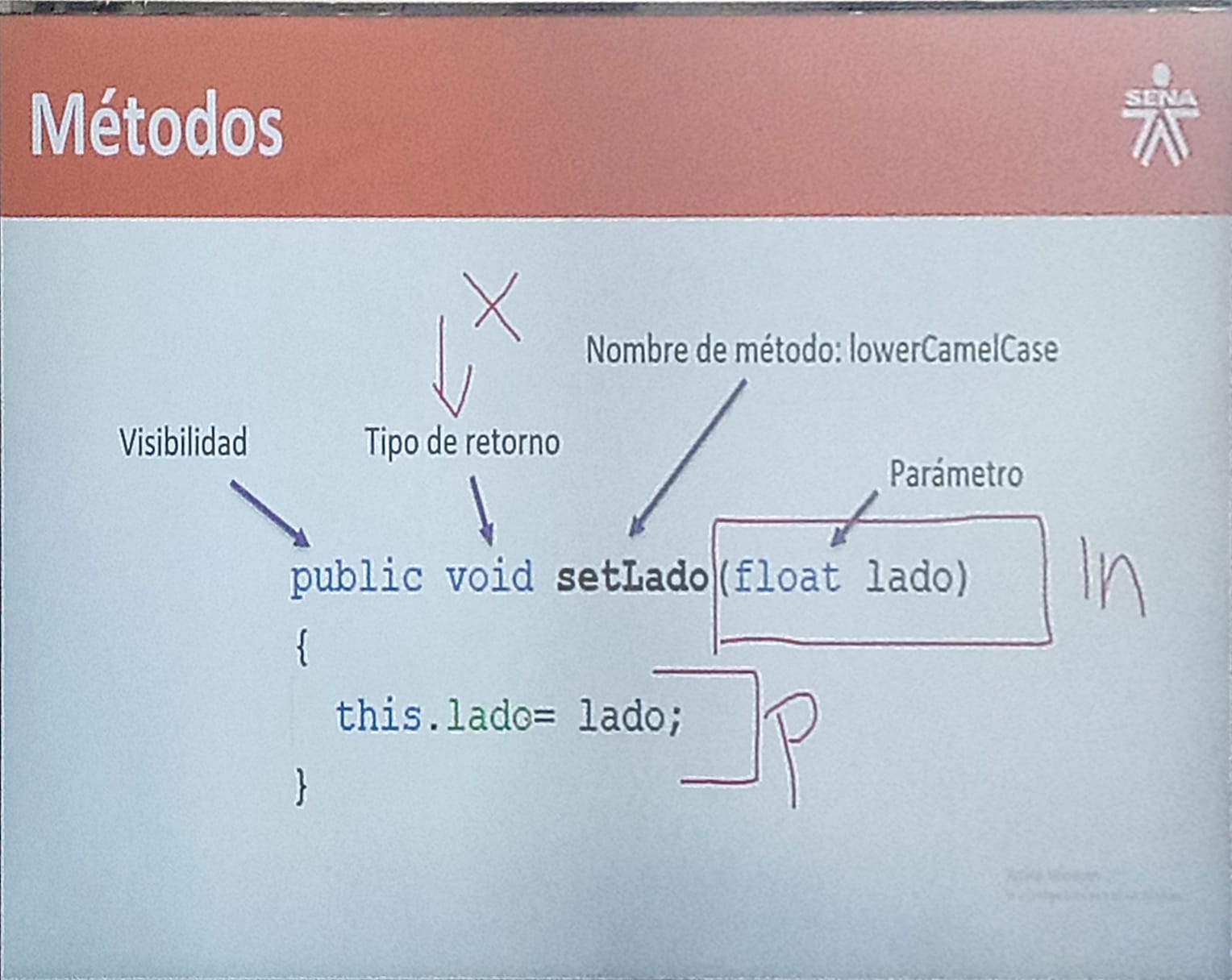
int🡪 es un tipo de datos primitivos porque está en minúscula. Int🡪 es una clase porque está en mayúscula.

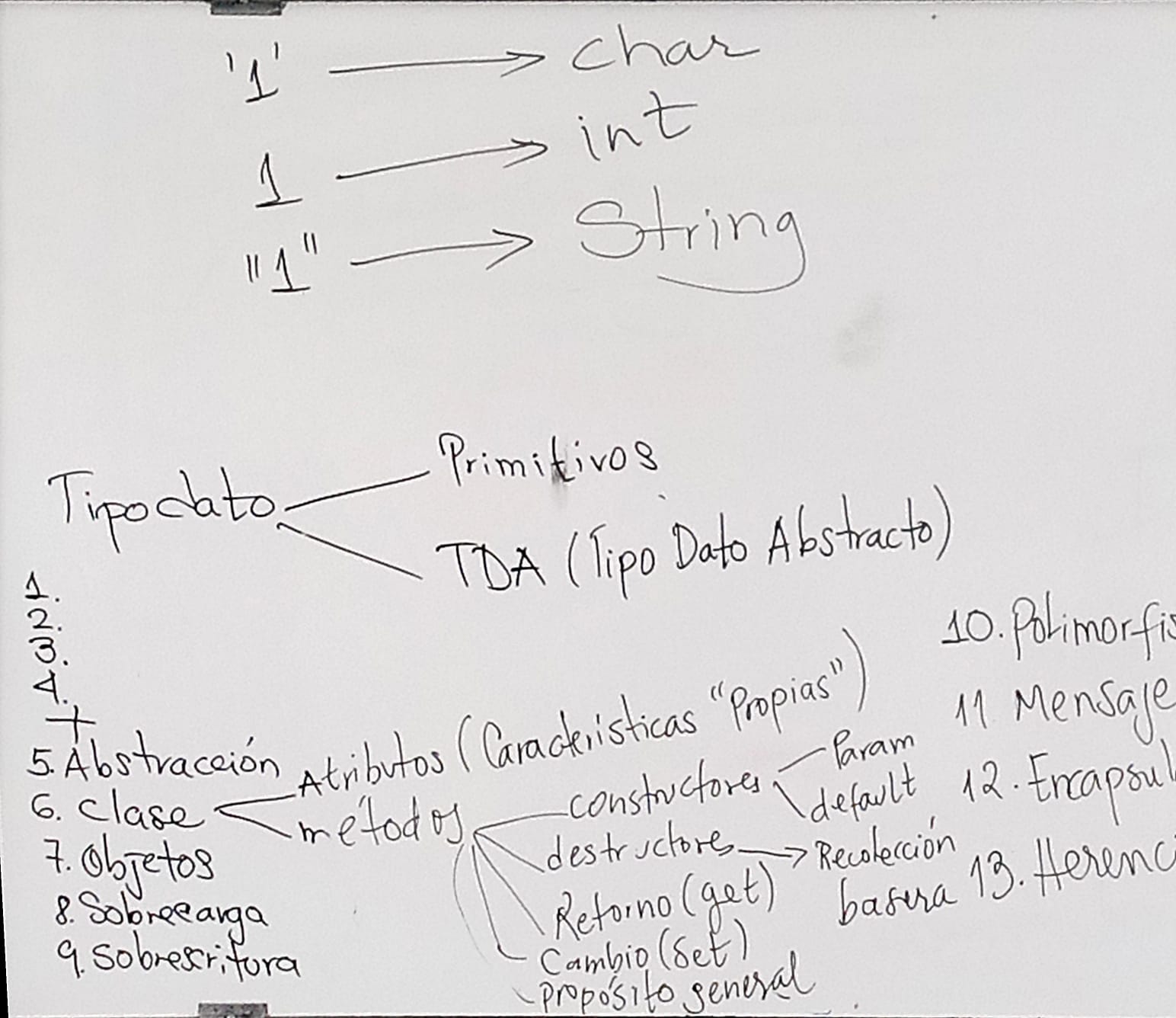
* **int** → Integer
* **boolean** → Boolean
* **char** → Character
* **byte** → Byte
* **short** → Short
* **long** → Long
* **float** → Float
* **double** → Double
* **Integer**: Se usa para envolver el tipo primitivo int, permitiendo almacenar y manipular números enteros como objetos.
* **Boolean**: Envolver el tipo primitivo boolean para trabajar con valores de verdad (true o false) como objetos.
* **Character**: Se usa para envolver el tipo primitivo char, que representa un solo carácter, como un objeto.
* **Byte**: Permite trabajar con valores de tipo byte (números pequeños) como objetos.
* Short: Envolver el tipo primitivo short, utilizado para trabajar con números enteros pequeños, como objetos.
* **Long**: Se usa para envolver el tipo primitivo long, permitiendo el manejo de números enteros largos como objetos.
* **Float**: Envolver el tipo primitivo float para trabajar con números de punto flotante de precisión simple como objetos.
* **Double**: Se usa para envolver el tipo primitivo double, permitiendo el manejo de números de punto flotante de doble precisión como objetos.



METODOS







### Ventajas de usar clases envoltorio

1. **Almacenamiento en colecciones**: Como se mencionó, las colecciones de Java (como ArrayList) solo pueden contener objetos, por lo que necesitas usar clases envoltorio para almacenar tipos primitivos.
2. **Métodos útiles**: Las clases envoltorio proporcionan métodos útiles para convertir entre tipos primitivos y cadenas, así como otros métodos relacionados.
3. **Autoboxing y Unboxing**: Java proporciona características que permiten la conversión automática entre tipos primitivos y sus clases envoltorio. Esto se llama autoboxing (de primitivo a objeto) y unboxing (de objeto a primitivo).

### Ejemplo:

import java.util.ArrayList;

public class EjemploWrapper {

public static void main(String[] args) {

// Creando una lista de enteros

ArrayList<Integer> lista = new ArrayList<>();

// Autoboxing: se convierte int a Integer automáticamente

lista.add(10);

lista.add(20);

// Unboxing: se convierte Integer a int automáticamente

int suma = lista.get(0) + lista.get(1);

System.out.println("La suma es: " + suma);

}

}

### Descripción del código

* **Autoboxing**: Al agregar un int a la lista, Java lo convierte automáticamente en un Integer.
* **Unboxing**: Al obtener un Integer de la lista, Java lo convierte automáticamente en un int.

Las clases envoltorio son una parte fundamental de Java que facilita el trabajo con tipos primitivos en situaciones donde se requieren objetos.

COMPLEMENTAR (codificar) SIGNATURA (titulo metodo)

PALABRAS RESERVADAS🡪