**TALLER No 3 PROGRAMACIÓN ORIENTADA A OBJETOS CON JAVA**

**ACTIVIDADES POR DESARROLLAR:**

* Comprender Programación Orientada a Objetos en JAVA
* Aplicar Programación Orientada a Objetos en JAVA

**EVIDENCIA(S) A ENTREGAR:**

**EV1 Desarrollar la actividad a desarrollar propuesta en el taller**

**CONTROL DEL DOCUMENTO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Nombre** | **Cargo** | **Dependencia** | **Fecha** |
| **Autor (es)** | JOSE FERNANDO GALINDO SUAREZ | INSTRUCTOR | CGMLTI | 13/02/2024 |

**CONTROL DE CAMBIOS** (diligenciar únicamente si realizan ajustes al taller)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Nombre** | **Cargo** | **Dependencia** | **Fecha** | **Razón del Cambio** |
| **Autor (es)** |  |  |  |  |  |

**INTRODUCCIÓN**

Después de realizar la lectura “[programación orientada a objetos](https://diego.com.es/programacion-orientada-a-objetos-en-php)”, podrá desarrollar los ejercicios dispuestos en este taller. Interfaz de usuario gráfica, Aplicación

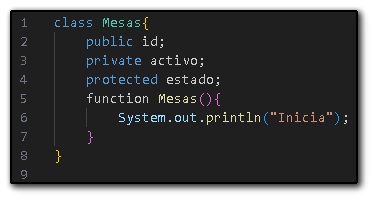
Descripción generada automáticamente

La programación orientada a objetos es un paradigma de la programación y es el más actualizado en la actualidad, aumentando así la modularidad y la reutilización de código en la programación, rompiendo el paradigma de la programación estructurada, aunque se puede combinar las dos al desarrollar aplicaciones.

Se acerca a la manera como se tratan los objetos en la realidad empezando con la fase de análisis, luego su implementación será la forma más adecuada para su desarrollo.

**Clases y objetos.**

Define los datos y el comportamiento, llamado atributos y métodos.





La visibilidad de una propiedad, un método o (a partir de PHP 7.1.0), una constante se puede definir anteponiendo a su declaración una de las palabras reservadas public, protected o private.

● public ​la variable/función se accede desde cualquier lugar y otras instancias de esa misma clase.

● private ​la variable/función solamente se accede desde la misma clase que la define.

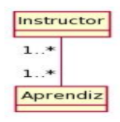
● protected ​la variable/función se accede desde la clase que las define y las clases que se herede de ella.

**RELACIONES** [**(Fuente)**](https://objetosweb.wordpress.com/category/uml/)



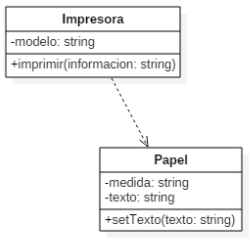
● La asociación​ es una relación donde los objetos tienen su propio ciclo de vida y no hay propietario.

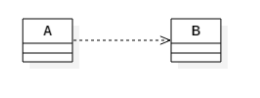
● La frase para comprobar una relación de este tipo es A es una parte de B.

Tomemos un ejemplo de Instructor y Aprendiz.

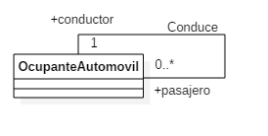
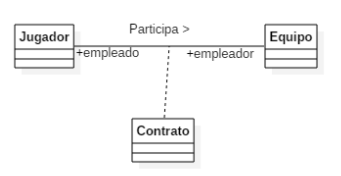
Varios Aprendices pueden asociarse con un solo Instructor y un solo Aprendiz puede asociarse con varios Instructores. Ambos se pueden crear y eliminar de forma independiente.



**Dependencia**



**Asociación reflexiva**



● La agregación​ es una forma especializada de asociación donde todos los objetos tienen su propia existencia, pero hay propiedad y los objetos secundarios no pueden pertenecer a

otro objeto principal.

Un ejemplo: Área e Instructor. Un solo Instructor no puede pertenecer a múltiples Áreas, pero si eliminamos el Área, el objeto Instructor ​no​ se destruirá. Podemos pensarlo como una relación

“tiene una”.

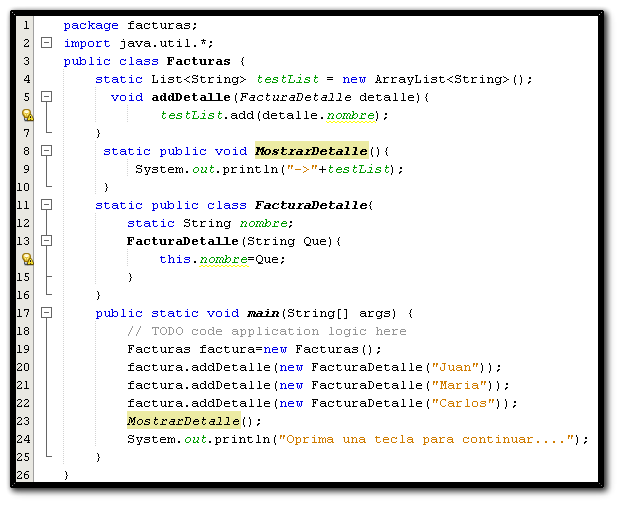
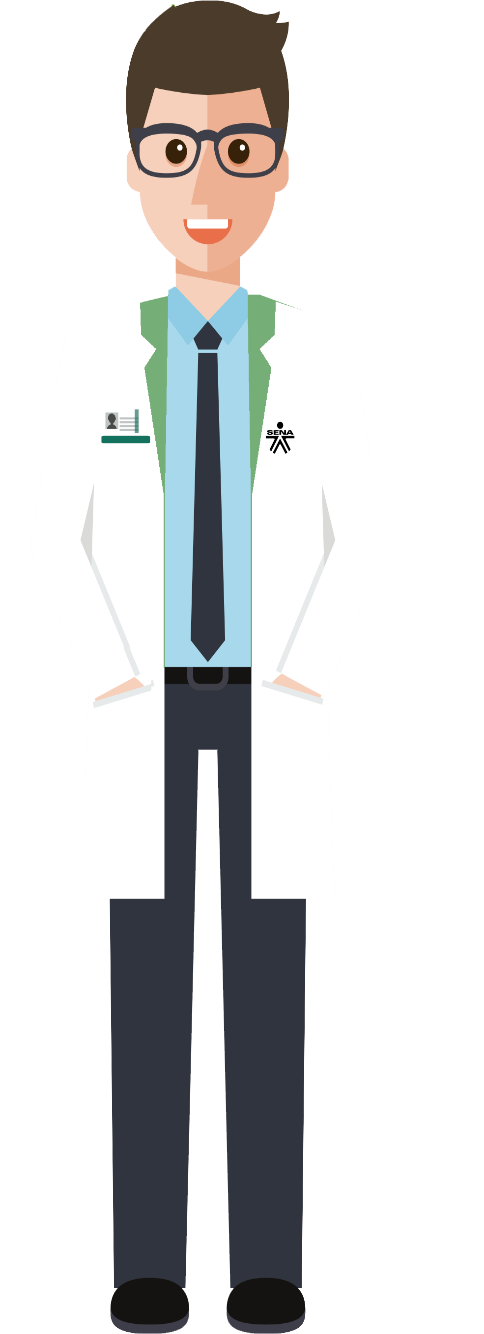
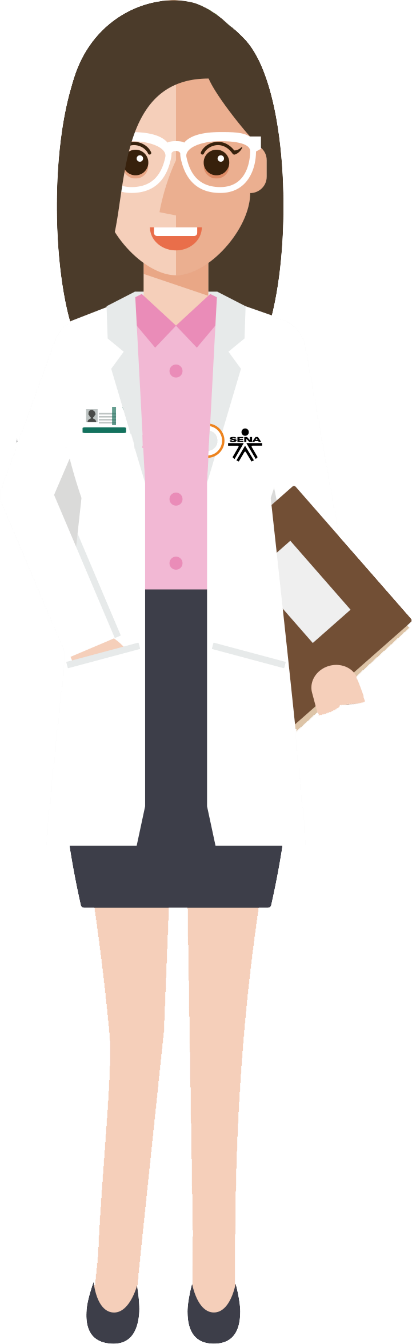


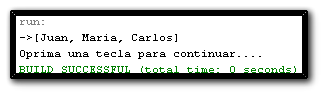
● La composición​ es una forma especializada de Agregación donde el objeto secundario no tiene su ciclo de vida propio y si se elimina el objeto principal, también se eliminarán todos los objetos secundarios.

La relación entre Preguntas y Opciones. Las

preguntas individuales pueden tener múltiples

opciones y la opción no puede pertenecer a otra pregunta.

****

****

**Diferencia entre Agregación y Composición**

* Las relaciones en una composición son requeridas, en la agregación son opcionales.
* En la composición una clase partícula no puede ser compartida por otras clases compuestas, en la agregación esto es posible.
* La relación de vida de la clase partícula y la clase contenedora, es muy fuerte, de hecho es la relación más fuerte; tanto así que si un objeto de la clase contenedora es destruido la clase partícula también lo será. Esto en la agregación no ocurre.

**INSTANCIAR UNA CLASE (**[**Fuente**](https://jairogarciarincon.com/clase/programacion-orientada-a-objetos-en-php/objetos-clases-atributos-y-metodos)**)**

A la acción de crear objetos, se denomina **instanciar una clase**; una **Clase** se puede considerar como un nuevo tipo de datos, personalizado, creado por el programador, y la variable de ese nuevo tipo de dato, es el objeto.

**$c=new NombreClase();**

**Visibilidad** ([Fuente](https://www.php.net/manual/es/language.oop5.visibility.php))

La visibilidad de una propiedad, un método o (a partir de PHP 7.1.0) una constante se puede definir anteponiendo a su declaración una de las palabras reservadas *public*, *protected* o *private*. A los miembros de clase declarados como 'public' se puede acceder desde donde sea; a los miembros declarados como 'protected', solo desde la misma clase, mediante clases heredadas o desde la clase padre. A los miembros declarados como 'private' únicamente se puede acceder desde la clase que los definió.

Visibilidad de propiedades

Las propiedades de clases deben ser definidas como 'public', 'private' o 'protected'. Si se declaran usando *var*, serán definidas como 'public'.

**HERENCIA DE OBJETOS**

Esto es útil para la definición y abstracción de la funcionalidad y permite la implementación de funcionalidad adicional en objetos similares sin la necesidad de Re implementar toda la funcionalidad compartida.

**CLASES TRAIT.**

Los TRAIT son clases que agrupan funcionalidades diseñadas y reutilizadas en otras clases para evitar la herencia simple. Estas no se instancian, sólo permiten utilizar sus funciones específicas en otras clases. [Fuente](https://www.php.net/manual/es/language.oop5.traits.php)

**public** **interface** Encendible {

**void** encender();

**void** apagar();

**boolean** estaEncendido();

}

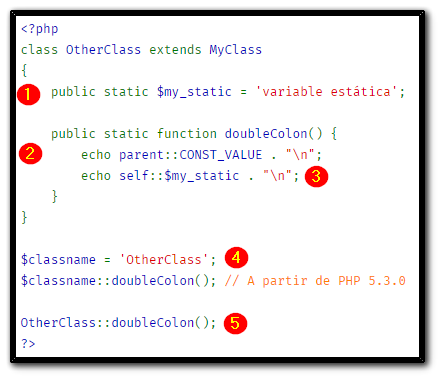
**OPERADOR DE RESOLUCIÓN DE ÁMBITO (::)**

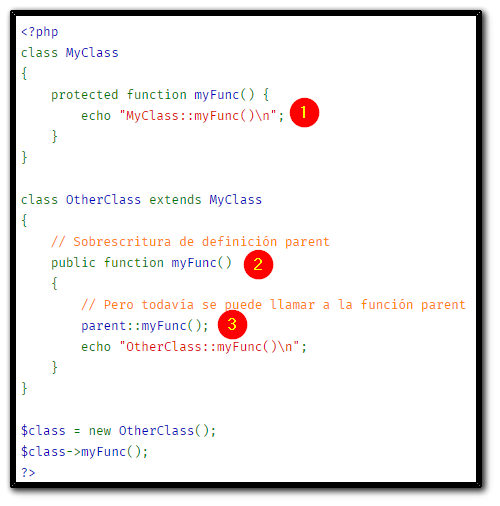
“El Operador de Resolución de Ámbito (también denominado Paamayim Nekudotayim) o en términos simples, el doble dos-puntos, es un token que permite acceder a elementos [estáticos](https://www.php.net/manual/es/language.oop5.static.php), [constantes](https://www.php.net/manual/es/language.oop5.constants.php), y sobrescribir propiedades o métodos de una clase.”, [Fuente](https://es.stackoverflow.com/questions/107152/que-significa-en-java)

**DESDE EL INTERIOR DE LA DEFINICIÓN DE LA CLASE**

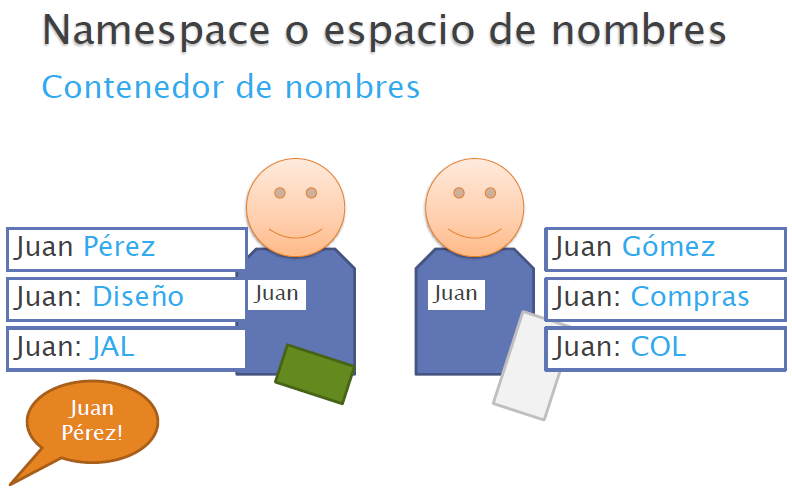
Las tres palabras claves especiales *self*, *parent* y *static* son utilizadas para acceder a propiedades y métodos desde el interior de la definición de la clase.

Declarar static a un método o atributo permite hacerlos accesibles sin la necesidad de instanciar la clase contenedora. Cuando se declara static no se puede acceder mediante una clase instanciada. La seudo variable $this no está disponible dentro de los métodos y atributos declarados static, se debe utilizar self::. [Fuente](https://www.php.net/manual/es/language.oop5.static.php)



**INVOCANDO A UN MÉTODO PARENT.**

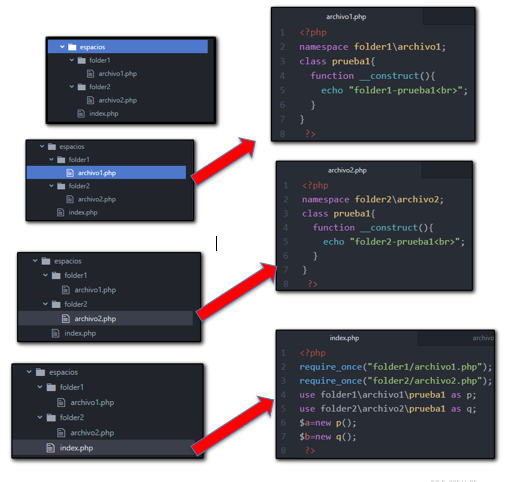
**NAMESPACE (**[**FUENTE**](https://jairogarciarincon.com/clase/programacion-orientada-a-objetos-en-php/namespaces)**)**



Soluciona conflictos entre nombres de clases y da una manera resumida a los nombre de clases cuyo nombre son demasiados largo con la utilización de apodos.

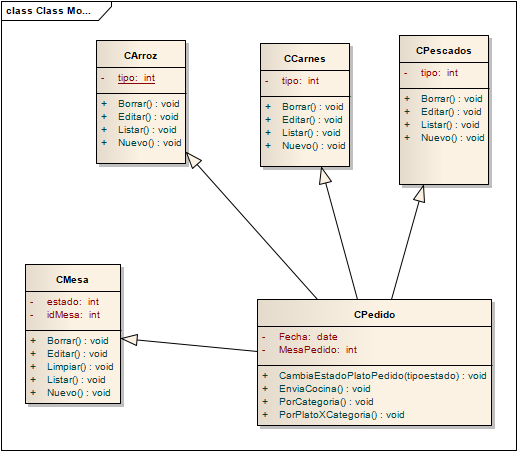
Para pode utilizar los namespace debemos tener en cuenta:

* A cada clase debemos añadir su namespace correspondiente en la primera línea del script.
* Debe incluir el archivo donde se encuentra el archivo de clases.
* Usar la cláusula use debe contener la ruta completa



Se recomienda ver el video en YouTube “[PHP namespaces](https://www.youtube.com/watch?v=s0LAJbTuquM)”

**Practica en clase:**



Desarrollar en PHP el anterior diagrama de clase, junto al instructor que dará las orientaciones para su construcción, no olvide guardar una copia en el portafolio de evidencias.

**Condiciones de la práctica:**

* No se permite cambiar el nombre a ningún método ni atributo de la práctica.
* Todos los métodos deben escribir el origen y el método por ejemplo “Desde la clase CArroz y el método Nuevo”
* No es permitido utilizar NameSpace

**Ejemplo de Agregación**

**Diagrama

Descripción generada automáticamente**

**Ejemplo de composición.** ****

**Salida en pantalla:**

****

**CLASE GENÉRICA**

Es un recurso que podemos encontrar en la mayoría de los lenguajes de programación. Esta clase genérica es una clase que no tiene ninguna propiedad ni método como solemos ver en otras clases predefinidas. Puede parecer poco practico, pero en realidad es de gran ayuda cuando se quiere crear un objeto e ir añadiendo las propiedades que querramos para, por ejemplo, añadirlo a un archivo JSON. [Leer más..](https://drive.google.com/file/d/1pKV4kz5ZW5gp9-K7LGlEPRTi1TX0RrVe/view?usp=sharing)

**CLASES ABSTRACTAS**

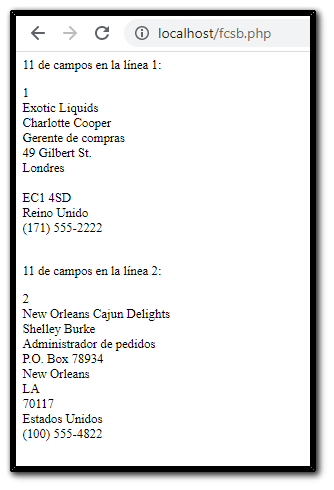
Las clases abstractas son aquellas que por sí mismas no se pueden identificar con algo 'concreto' (no existen como tal en el mundo real), pero sí poseen determinadas características que son comunes en otras clases que pueden ser creadas a partir de ellas. [Leer más..](https://drive.google.com/file/d/1iK4pegsBhoa-ff8t2_SoI9z2t20bATxj/view?usp=sharing)

**INTERFACES**

Una interfaz (interface) es sintácticamente similar a una clase abstracta, en la que puede especificar uno o más métodos que no tienen cuerpo ({}). Esos métodos deben ser implementados por una clase para que se definan sus acciones.

Por lo tanto, una interfaz especifica qué se debe hacer, pero no cómo hacerlo. Una vez que se define una interfaz, cualquier cantidad de clases pueden implementarla. Además,

una clase puede implementar cualquier cantidad de interfaces. [Leer más..](https://drive.google.com/file/d/1KoQCFdVOalzftyojmItwDhgA3eSPNd2c/view?usp=sharing)

****

**ACTIVIDAD DE AFIANZAMIENTO**

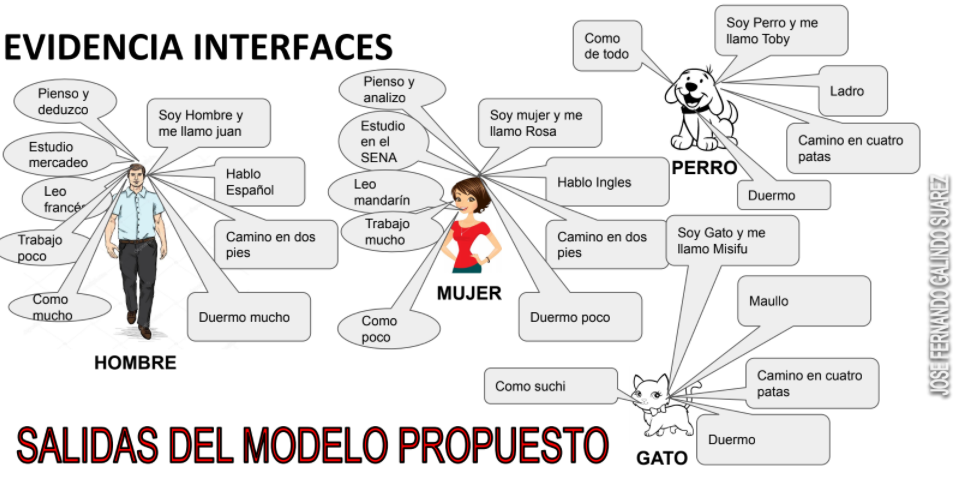
El instructor le dará las indicaciones necesarias para realizar la siguiente actividad de afianzamiento “[A1. CRUCIGRAMA POO](https://up9bp0pfq2ue4kizq9zseg-on.drv.tw/1CGMLTI/1-FORMACION/BASICO/GUIAS/PHP(BASICO)/MATERIALES%20DE%20APOYO/CrucigramaPOO.htm)” y “[A2. SOPA DE LETRA POO](https://up9bp0pfq2ue4kizq9zseg-on.drv.tw/1CGMLTI/1-FORMACION/BASICO/GUIAS/PHP(BASICO)/TALLERES/sopapop/sopapop.htm)”, que tratan sobre programación orientada a objetos, no olvide guardar una copia en su portafolio de evidencias.

**PRACTICA No 1:**

Desarrollar en JAVA de acuerdo a [diseño propuesto por el instructor](https://drive.google.com/file/d/1TflzJ0UdDiQ8Gcsq2fVBz8-yFi_g6owW/view?usp=sharing) aplicando multinivel con los [recursos disponibles](https://drive.google.com/file/d/1DnxfM9ESFz58pVidD-SMOOdamlYaUNg_/view?usp=sharing) para su desarrollo, se debe desarrollar en grupo de acuerdo a los integrantes de los anteriores talleres.

**PRACTICA No 2:**

Desarrollar la evidencia de interfaces de acuerdo a la salida de la siguiente grafica:

**** ****

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamenteDesarrolle esta evidencia en archivo , de acuerdo con la gráfica anterior y teniendo en cuenta las salidas del modelo propuesto, en la herramienta de su preferencia y envíela al instructor, guarde una copia en el portafolio del aprendiz; este taller se debe realizar con los integrantes del grupo de proyecto de formación.

Recuerde enviarlo en un archivo ZIP no RAR ni otra extensión de comprimido y colocar como nombre “**T3\_poo.zip**” y subirlo al LMS individualmente.

**INSTRUMENTO DE EVALUACIÓN**

|  |  |
| --- | --- |
| FECHA |  |
| APRENDIZ |  |
| |  |  |  |  | | --- | --- | --- | --- | | CRITERIO | SI | NO | OBSERVACIÓN | | Realiza la evidencia solicitada |  |  |  | | Utiliza clases abstracta |  |  |  | | Utiliza interfaces y las implementa correctamente. |  |  |  | | La salidas al ejecutar el programa corresponde a la salida de la grafica |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | | |

Texto

Descripción generada automáticamente