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", podrá desarrollar los ejercicios dispuestos en este taller.

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.

```
1   class Mesas{
2       public id;
3       private activo;
4       protected estado;
5       function Mesas(){
6           System.out.println("Inicia");
7       }
8   }
9
```

Character	Icon for field	Icon for method	Visibility
-	□	■	private
#	◇	◆	protected
~	△	▲	package private
+	○	●	public

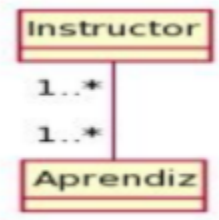
La visibilidad de una propiedad, un método o (a partir de PHP 7.1.0), una constante se puede definir anteponiendo a su declaración una de las palabras reservadas public, protected o private.

- 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)

Asociación

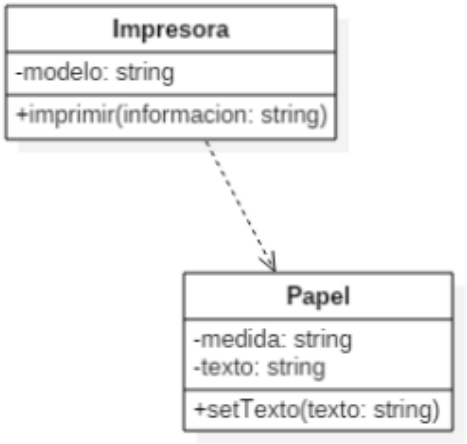
- La asociación es una relación donde los objetos tienen su propio ciclo de vida y no hay propietario.
- La frase para comprobar una relación de este tipo es A es una parte de B.



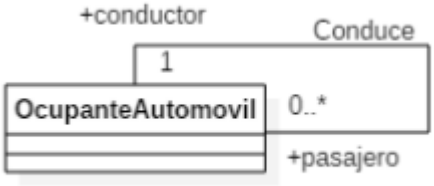
Tomemos un ejemplo de Instructor y Aprendiz.
Varios Aprendices pueden asociarse con un solo Instructor y un solo Aprendiz puede asociarse con varios Instructores. Ambos se pueden crear y eliminar de forma independiente.

Composición

Dependencia



Asociación reflexiva



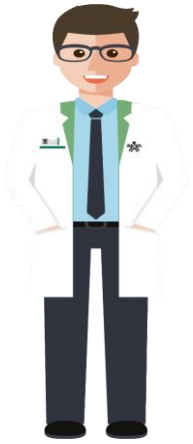
● La agregación es una forma especializada de asociación donde todos los objetos tienen su propia existencia, pero hay propiedad y los objetos secundarios no pueden pertenecer a otro objeto principal.

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.

```java
package facturas;
import java.util.*;
public class Facturas {
    static List<String> testList = new ArrayList<String>();
      void addDetalle(FacturaDetalle detalle){
          testList.add(detalle.nombre);
    }
     static public void MostrarDetalle(){
         System.out.println("->"+testList);
     }
    static public class FacturaDetalle{
        static String nombre;
        FacturaDetalle(String Que){
            this.nombre=Que;
        }
    }
    public static void main(String[] args) {
        // TODO code application logic here
        Facturas factura=new Facturas();
        factura.addDetalle(new FacturaDetalle("Juan"));
        factura.addDetalle(new FacturaDetalle("Maria"));
        factura.addDetalle(new FacturaDetalle("Carlos"));
        MostrarDetalle();
        System.out.println("Oprima una tecla para continuar....");
    }
}
```



```
run:
->[Juan, Maria, Carlos]
Oprima una tecla para continuar....
BUILD SUCCESSFUL (total time: 0 seconds)
```

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)

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)

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

```
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, constantes, y sobrescribir propiedades o métodos de una clase.", Fuente

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

```php
<?php
class OtherClass extends MyClass
{
    public static $my_static = 'variable estática';

    public static function doubleColon() {
        echo parent::CONST_VALUE . "\n";
        echo self::$my_static . "\n";
    }
}

$classname = 'OtherClass';
$classname::doubleColon(); // A partir de PHP 5.3.0

OtherClass::doubleColon();
?>
```

INVOCANDO A UN MÉTODO PARENT.

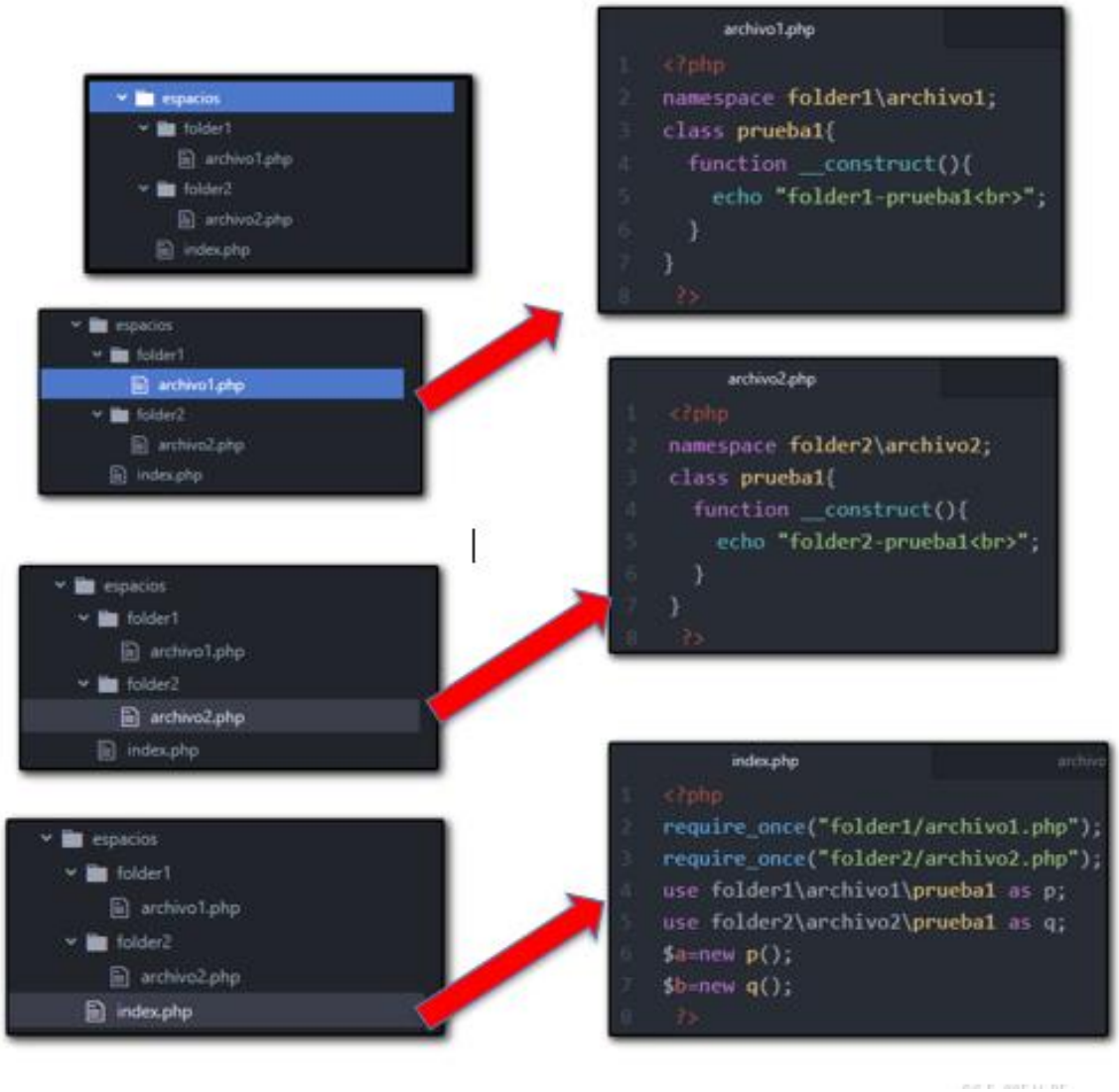
```php
<?php
class MyClass
{
    protected function myFunc() {
        echo "MyClass::myFunc()\n"; 1
    }
}

class OtherClass extends MyClass
{
    // Sobrescritura de definición parent
    public function myFunc() 2
    {
        // Pero todavía se puede llamar a la función parent
        parent::myFunc(); 3
        echo "OtherClass::myFunc()\n";
    }
}

$class = new OtherClass();
$class->myFunc();
?>
```

NAMESPACE (FUENTE)

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.

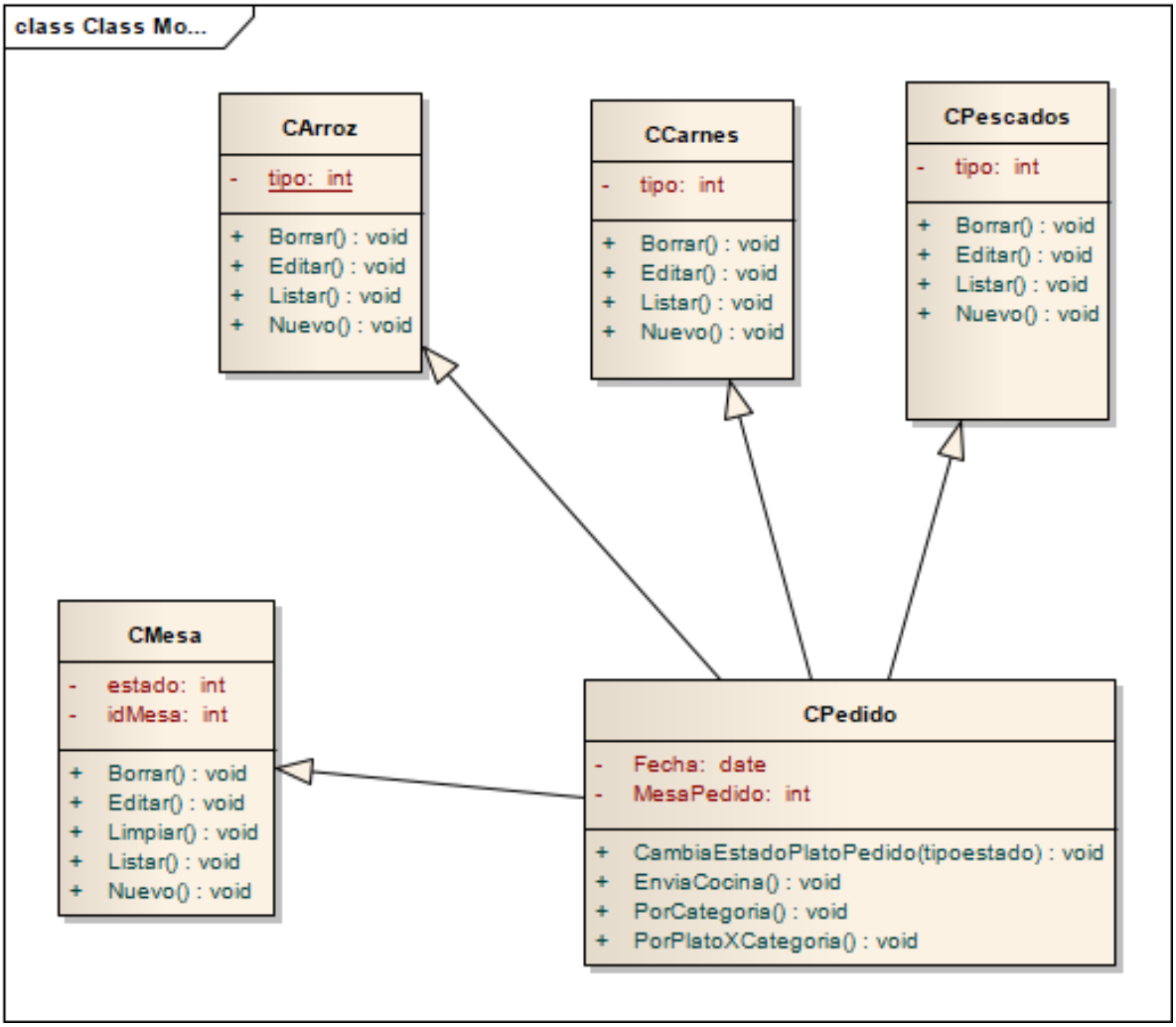
Para pode utilizar los namespace debemos tener en cuenta:

- 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"

Practica en clase:

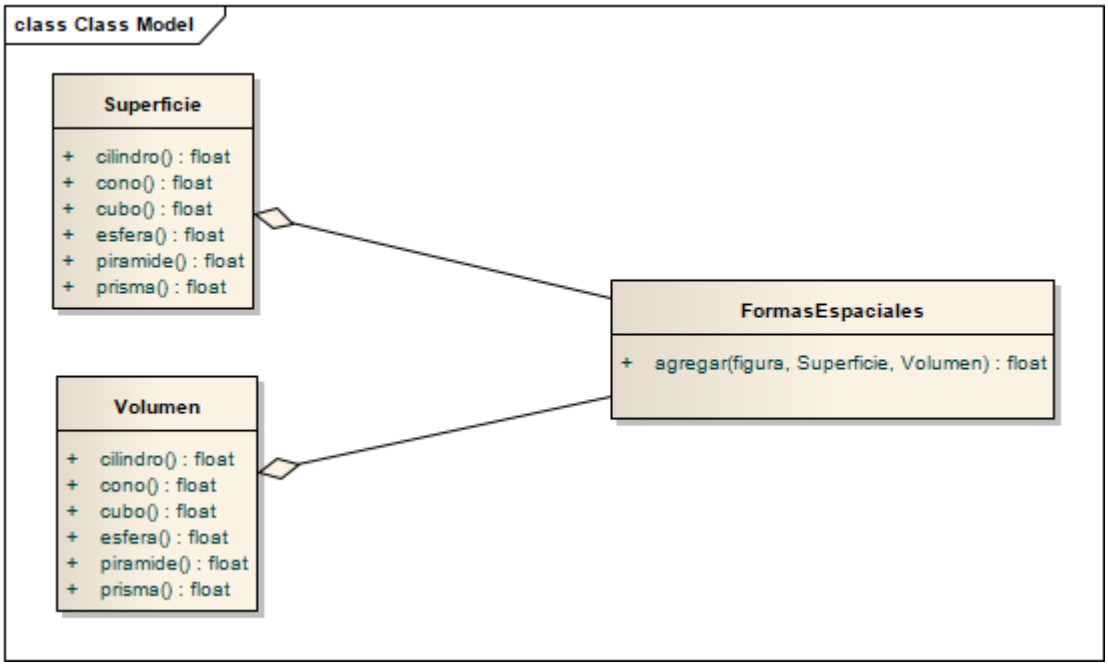


Desarrollar en PHP el anterior diagrama de clase, junto al instructor que dará las orientaciones para su construcción, no olvide guardar una copia en el portafolio de evidencias.

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


class Class Model
Superficie
+ cilindro() : float
+ cono() : float
+ cubo() : float
+ esfera() : float
+ piramide() : float
+ prisma() : float
Volumen
+ cilindro() : float
+ cono() : float
+ cubo() : float
+ esfera() : float
+ piramide() : float
+ prisma() : float
FormasEspaciales
+ agregar(figura, Superficie, Volumen) : float

Ejemplo de composición.

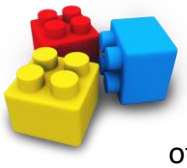
Salida en pantalla:

La suma entre 3 y 5 es 8

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..

CLASES ABSTRACTAS



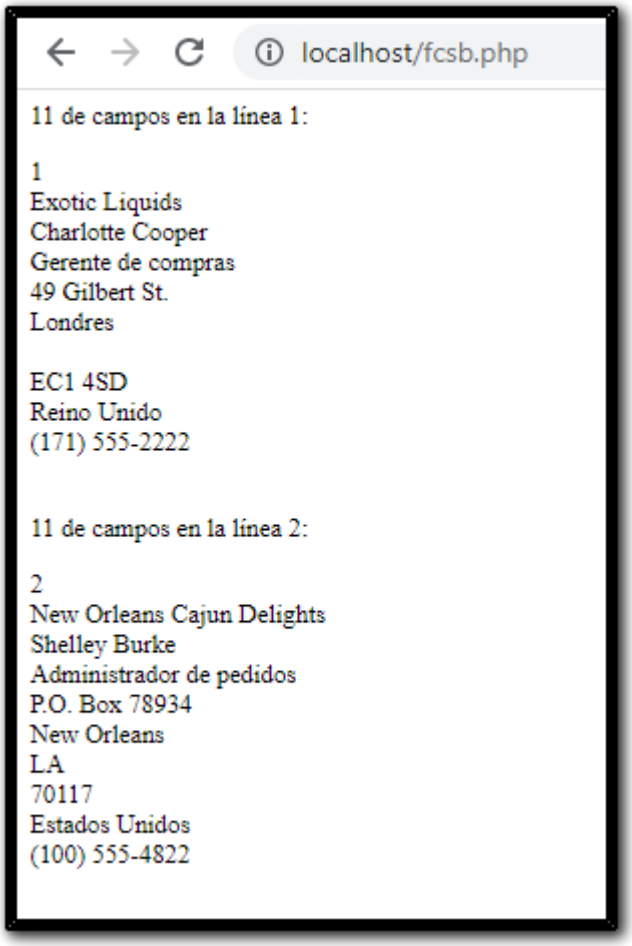
Las clases abstractas son aquellas que por sí mismas no se pueden identificar con algo 'concreto' (no existen como tal en el mundo real), pero sí poseen determinadas características que son comunes en otras clases que pueden ser creadas a partir de ellas. Leer más..

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..


localhost/fcsb.php
11 de campos en la línea 1:
1
Exotic Liquids
Charlotte Cooper
Gerente de compras
49 Gilbert St.
Londres
EC1 4SD
Reino Unido
(171) 555-2222
11 de campos en la línea 2:
2
New Orleans Cajun Delights
Shelley Burke
Administrador de pedidos
P.O. Box 78934
New Orleans
LA
70117
Estados Unidos
(100) 555-4822

ACTIVIDAD DE AFIANZAMIENTO

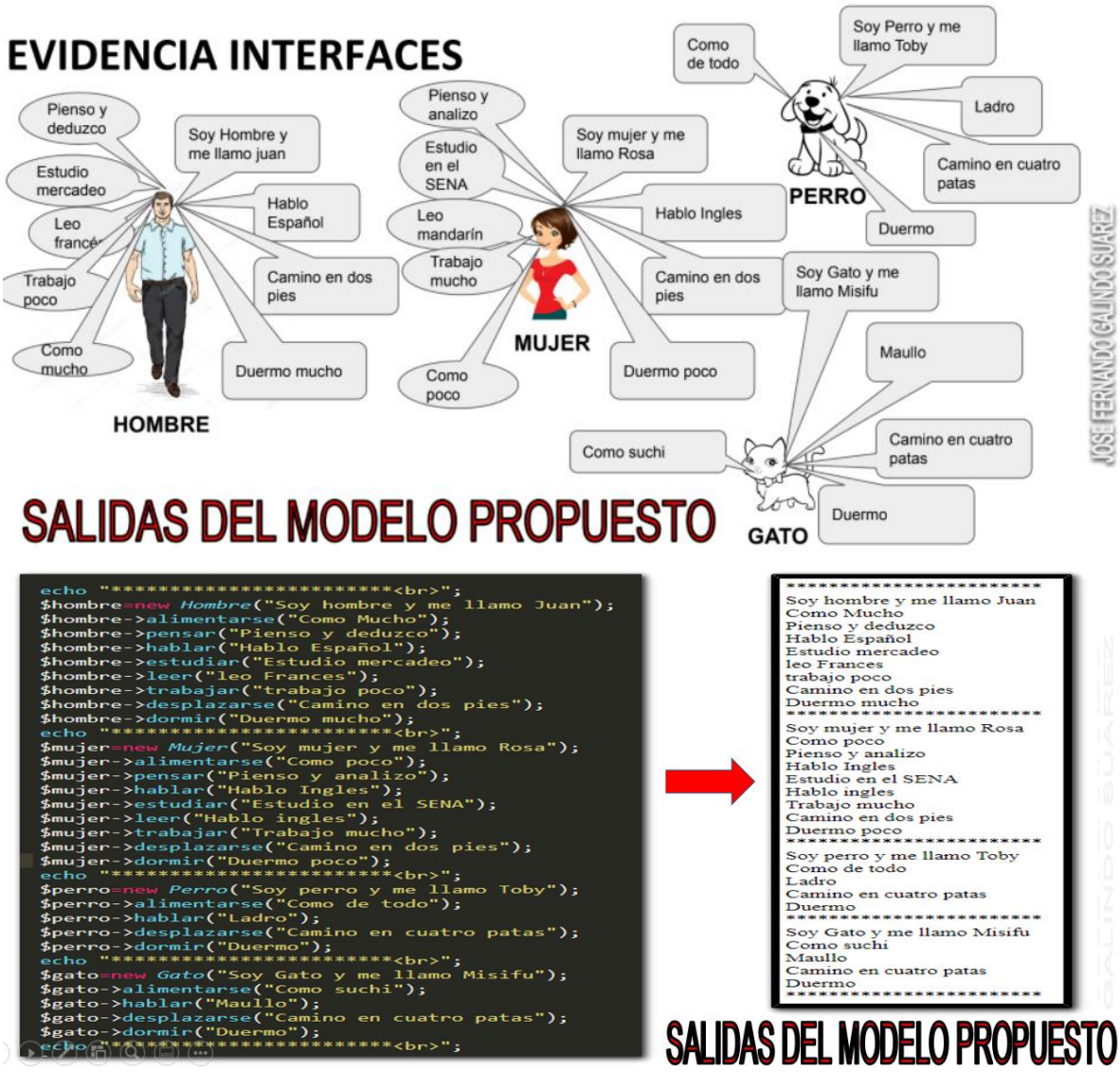
El instructor le dará las indicaciones necesarias para realizar la siguiente actividad de afianzamiento "A1. CRUCIGRAMA POO" y "A2. SOPA DE LETRA POO", que tratan sobre programación orientada a objetos, no olvide guardar una copia en su portafolio de evidencias.

PRACTICA No 1:

Desarrollar en JAVA de acuerdo a diseño propuesto por el instructor aplicando multinivel con los recursos disponibles 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:




```
echo "**********************<br>";
$hombre=new Hombre("Soy hombre y me llamo Juan");
$hombre->alimentarse("Como Mucho");
$hombre->pensar("Pienso y deduzco");
$hombre->hablar("Hablo Español");
$hombre->estudiar("Estudio mercadeo");
$hombre->leer("leo Frances");
$hombre->trabajar("trabajo poco");
$hombre->desplazarse("Camino en dos pies");
$hombre->dormir("Duermo mucho");
echo "**********************<br>";
$mujer=new Mujer("Soy mujer y me llamo Rosa");
$mujer->alimentarse("Como poco");
$mujer->pensar("Pienso y analizo");
$mujer->hablar("Hablo Ingles");
$mujer->estudiar("Estudio en el SENA");
$mujer->leer("Hablo ingles");
$mujer->trabajar("Trabajo mucho");
$mujer->desplazarse("Camino en dos pies");
$mujer->dormir("Duermo poco");
echo "**********************<br>";
$perro=new Perro("Soy perro y me llamo Toby");
$perro->alimentarse("Como de todo");
$perro->hablar("Ladro");
$perro->desplazarse("Camino en cuatro patas");
$perro->dormir("Duermo");
echo "**********************<br>";
$gato=new Gato("Soy Gato y me llamo Misifu");
$gato->alimentarse("Como suchi");
$gato->hablar("Maullo");
$gato->desplazarse("Camino en cuatro patas");
$gato->dormir("Duermo");
echo "**********************<br>";
```

```
**********
Soy hombre y me llamo Juan
Como Mucho
Pienso y deduzco
Hablo Español
Estudio mercadeo
leo Frances
trabajo poco
Camino en dos pies
Duermo mucho
**********
Soy mujer y me llamo Rosa
Como poco
Pienso y analizo
Hablo Ingles
Estudio en el SENA
Hablo ingles
Trabajo mucho
Camino en dos pies
Duermo poco
**********
Soy perro y me llamo Toby
Como de todo
Ladro
Camino en cuatro patas
Duermo
**********
Soy Gato y me llamo Misifu
Como suchi
Maullo
Camino en cuatro patas
Duermo
**********
```

SALIDAS DEL MODELO PROPUESTO

Desarrolle 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			


Atribución, no comercial, compartir igual
Este material puede ser distribuido, copiado y exhibido por terceros si se muestra en los créditos. No se puede obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de licencia que el trabajo original.
SENA
Creative Commons