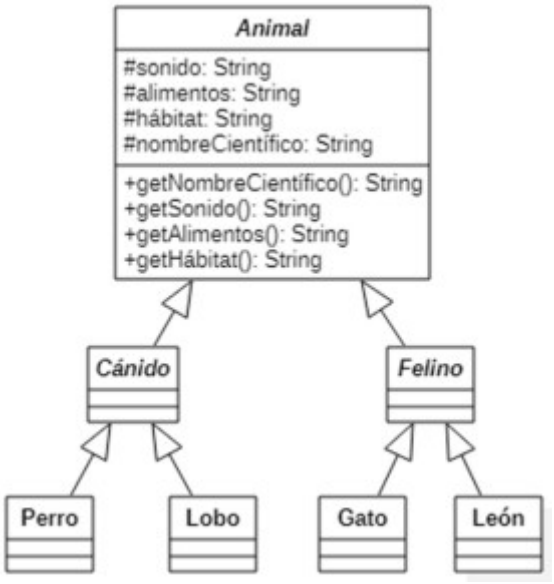
Practica Nro. 2

Apellidos y Nombres: VARGAS PAUCARA BORIS

Realice la codificación de los siguientes problemas:

1. Implementa el siguiente diagrama de clases:



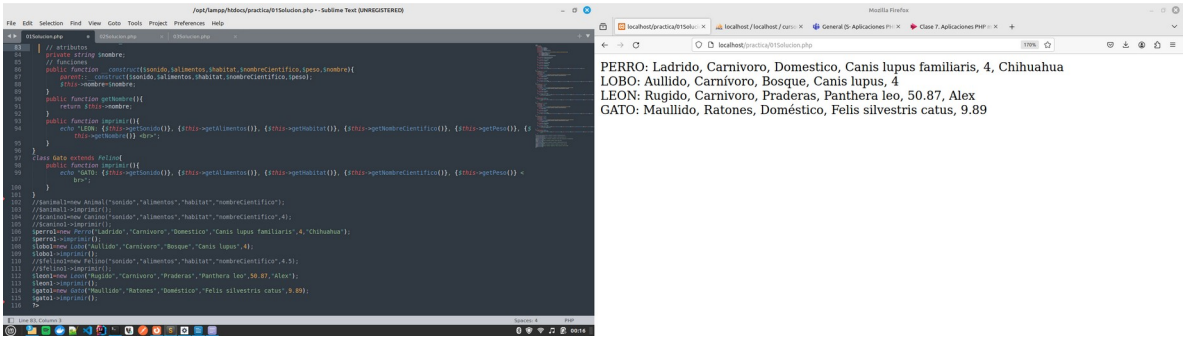
Debe agregar atributos y métodos a las clases heredadas.
SOLUCIÓN:

```
<?php
class Animal{
   // atributos
   private string $sonido;
   private string $alimentos;
   private string $habitat;
   private string $nombreCientifico;
   // functiones
   public function __construct($sonido,$alimentos,$habitat,
$nombreCientifico){
      $this->sonido=$sonido;
      $this->alimentos=$alimentos;
      $this->habitat=$habitat;
      $this->nombreCientifico=$nombreCientifico;
   }
   public function getNombreCientifico(){
      return $this->nombreCientifico;
   }
   public function getSonido(){
      return $this->sonido;
   }
```

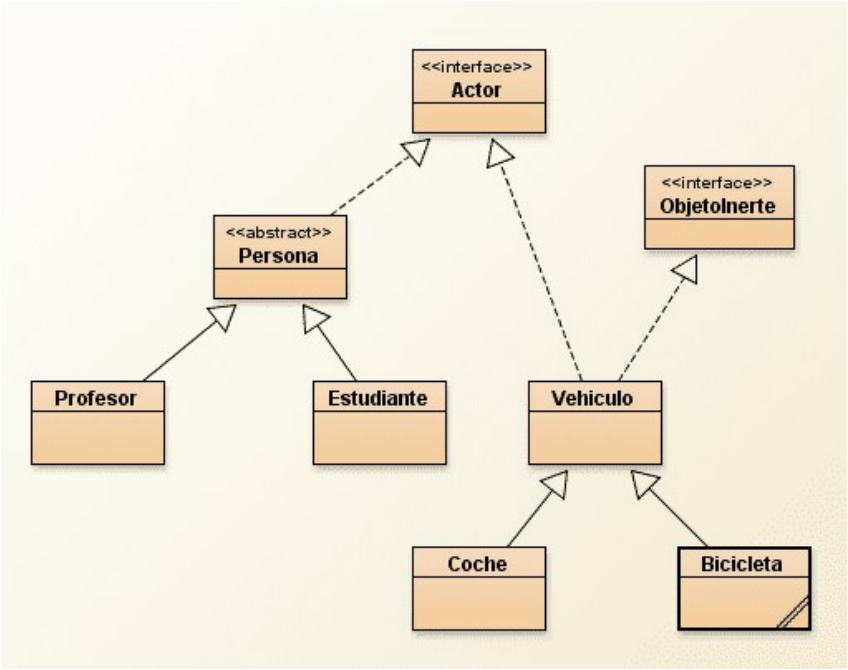
```
    public function getAlimentos(){
        return $this->alimentos;
    }
    public function getHabitat(){
        return $this->habitat;
    }
    public function imprimir(){
        echo "{$this->getSonido()}, {$this->getAlimentos()}, {$this-
>getHabitat()}, {$this->getNombreCientifico()} <br>";
    }
}
class Canino extends Animal{
    // atributos
    private int $patas;
    // funciones
    public function __construct($sonido,$alimentos,$habitat,
$nombreCientifico,$patas){
        parent::__construct($sonido,$alimentos,$habitat,
$nombreCientifico);
        $this->patas=$patas;
    }
    public function getPatas(){
        return $this->patas;
    }
    public function imprimir(){
        echo "{$this->getSonido()}, {$this->getAlimentos()}, {$this-
>getHabitat()}, {$this->getNombreCientifico()}, {$this->getPatas()}
<br>";
    }
}
class Perro extends Canino{
    // atributos
    private string $raza;
    // funciones
    public function __construct($sonido,$alimentos,$habitat,
$nombreCientifico,$patas,$raza){
        parent::__construct($sonido,$alimentos,$habitat,$nombreCientifico,
$patas);
        $this->raza=$raza;
    }
    public function getRaza(){
        return $this->raza;
    }
    public function imprimir(){
```

```
        echo "PERRO: {$this->getSonido()}, {$this->getAlimentos()},
{$this->getHabitat()}, {$this->getNombreCientifico()}, {$this-
>getPatas()}, {$this->getRaza()} <br>";
    }
}
class Lobo extends Canino{
    public function imprimir(){
        echo "LOBO: {$this->getSonido()}, {$this->getAlimentos()},
{$this->getHabitat()}, {$this->getNombreCientifico()}, {$this-
>getPatas()} <br>";
    }
}
class Felino extends Animal{
    // atributos
    private float $peso;
    // funciones
    public function __construct($sonido,$alimentos,$habitat,
$nombreCientifico,$peso){
        parent::__construct($sonido,$alimentos,$habitat,
$nombreCientifico);
        $this->peso=$peso;
    }
    public function getPeso(){
        return $this->peso;
    }
    public function imprimir(){
        echo "{$this->getSonido()}, {$this->getAlimentos()}, {$this-
>getHabitat()}, {$this->getNombreCientifico()}, {$this->getPeso()}
<br>";
    }
}
class Leon extends Felino{
    // atributos
    private string $nombre;
    // funciones
    public function __construct($sonido,$alimentos,$habitat,
$nombreCientifico,$peso,$nombre){
        parent::__construct($sonido,$alimentos,$habitat,$nombreCientifico,
$peso);
        $this->nombre=$nombre;
    }
    public function getNombre(){
        return $this->nombre;
    }
```

```
    public function imprimir(){
        echo "LEON: {$this->getSonido()}, {$this->getAlimentos()},
{$this->getHabitat()}, {$this->getNombreCientifico()}, {$this-
>getPeso()}, {$this->getNombre()} <br>";
    }
}
class Gato extends Felino{
    public function imprimir(){
        echo "GATO: {$this->getSonido()}, {$this->getAlimentos()},
{$this->getHabitat()}, {$this->getNombreCientifico()}, {$this-
>getPeso()} <br>";
    }
}

//$animal1=new
Animal("sonido","alimentos","habitat","nombreCientifico");
//$animal1->imprimir();
//$canino1=new
Canino("sonido","alimentos","habitat","nombreCientifico",4);
//$canino1->imprimir();
$perro1=new Perro("Ladrido","Carnivoro","Domestico","Canis lupus
familiaris",4,"Chihuahua");
$perro1->imprimir();
$lobo1=new Lobo("Aullido","Carnívoro","Bosque","Canis lupus",4);
$lobo1->imprimir();
//$felino1=new
Felino("sonido","alimentos","habitat","nombreCientifico",4.5);
//$felino1->imprimir();
$leon1=new Leon("Rugido","Carnivoro","Praderas","Panthera
leo",50.87,"Alex");
$leon1->imprimir();
$gato1=new Gato("Maullido","Ratones","Doméstico","Felis silvestris
catus",9.89);
$gato1->imprimir();

?>
```



2. Implemente el siguiente diagrama de clases:



Debe crear sus atributos y métodos que vea conveniente.

SOLUCIÓN:

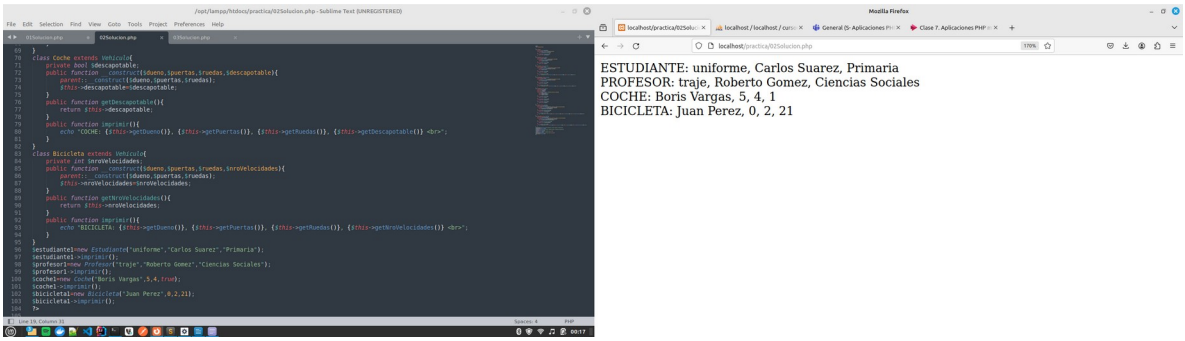
```
<?php
class ObjetoInerte{


}
class Actor{
    private string $vestimenta;
    public function __construct($vestimenta){
        $this->vestimenta=$vestimenta;
```

```
    }
    public function getVestimenta(){
        return $this->vestimenta;
    }
}
class Persona extends Actor{
    private string $nombre;
    public function __construct($vestimenta,$nombre){
        parent::__construct($vestimenta);
        $this->nombre=$nombre;
    }
    public function getNombre(){
        return $this->nombre;
    }
}
class Profesor extends Persona{
    private string $cargo;
    public function __construct($vestimenta,$nombre,$cargo){
        parent::__construct($vestimenta,$nombre);
        $this->cargo=$cargo;
    }
    public function getCargo(){
        return $this->cargo;
    }
    public function imprimir(){
        echo "PROFESOR: {$this->getVestimenta()}, {$this->getNombre()},
{$this->getCargo()} <br>";
    }
}
```

```
class Estudiante extends Persona{
    private string $curso;
    public function __construct($vestimenta,$nombre,$curso){
        parent::__construct($vestimenta,$nombre);
        $this->curso=$curso;
    }
    public function getCurso(){
        return $this->curso;
    }
    public function imprimir(){
        echo "ESTUDIANTE: {$this->getVestimenta()}, {$this->getNombre()},
{$this->getCurso()} <br>";
    }
}
class Vehiculo extends Actor{
    private string $dueno;
    private int $puertas;
    private int $ruedas;
    public function __construct($dueno,$puertas,$ruedas){
        $this->dueno=$dueno;
        $this->puertas=$puertas;
        $this->ruedas=$ruedas;
    }
    public function getDueno(){
        return $this->dueno;
    }
    public function getPuertas(){
        return $this->puertas;
    }
```

```
    public function getRuedas(){
        return $this->ruedas;
    }
}
class Coche extends Vehiculo{
    private bool $descapotable;
    public function __construct($dueno,$puertas,$ruedas,$descapotable){
        parent::__construct($dueno,$puertas,$ruedas);
        $this->descapotable=$descapotable;
    }
    public function getDescapotable(){
        return $this->descapotable;
    }
    public function imprimir(){
        echo "COCHE: {$this->getDueno()}, {$this->getPuertas()}, {$this-
>getRuedas()}, {$this->getDescapotable()} <br>";
    }
}
class Bicicleta extends Vehiculo{
    private int $nroVelocidades;
    public function __construct($dueno,$puertas,$ruedas,$nroVelocidades){
        parent::__construct($dueno,$puertas,$ruedas);
        $this->nroVelocidades=$nroVelocidades;
    }
    public function getNroVelocidades(){
        return $this->nroVelocidades;
    }
    public function imprimir(){
```

```
        echo "BICICLETA: {$this->getDueno()}, {$this->getPuertas()}, {$this->getRuedas()}, {$this->getNroVelocidades()} <br>";
    }
}
$estudiante1=new Estudiante("uniforme","Carlos Suarez","Primaria");
$estudiante1->imprimir();
$profesor1=new Profesor("traje","Roberto Gomez","Ciencias Sociales");
$profesor1->imprimir();
$coche1=new Coche("Boris Vargas",5,4,true);
$coche1->imprimir();
$bicicleta1=new Bicicleta("Juan Perez",0,2,21);
$bicicleta1->imprimir();
?>
```

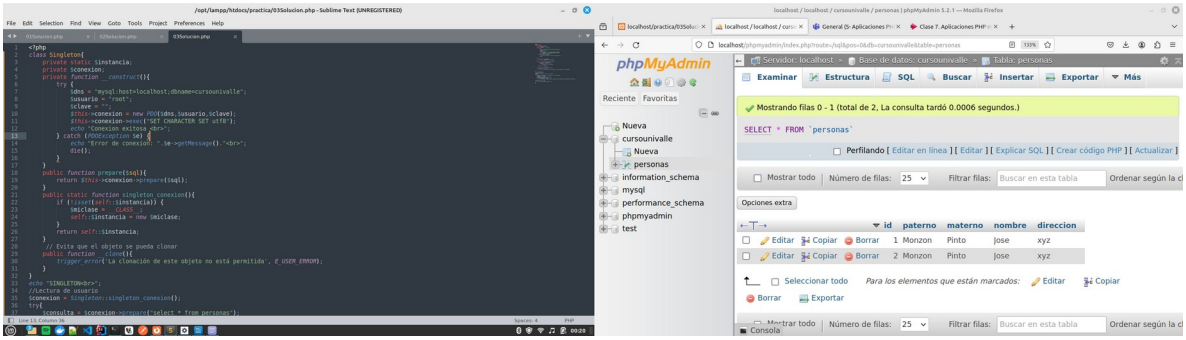


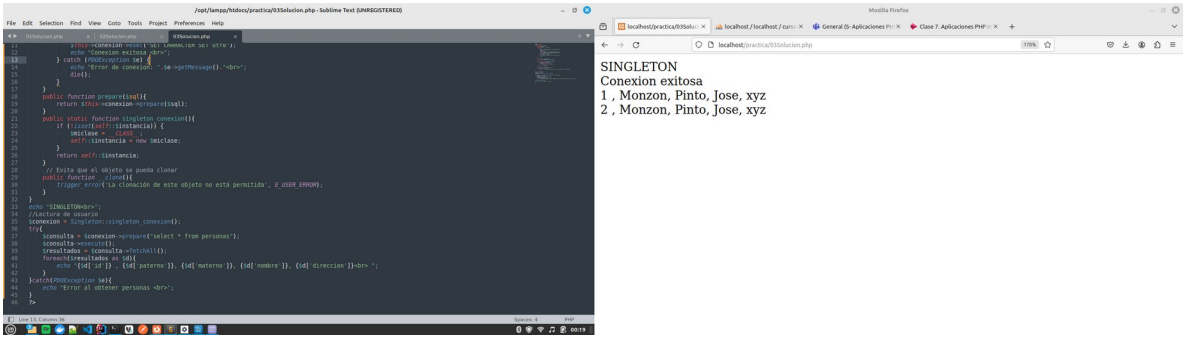
3. Cree una clase Singleton para conectar una Base de datos usando PDO y Mysql.

SOLUCIÓN:

```
<?php
class Singleton{
    private static $instancia;
    private $conexion;
    private function __construct(){
        try {
            $dns = "mysql:host=localhost;dbname=cursounivalle";
            $usuario = "root";
            $clave = "";
```

```
            $this->conexion = new PDO($dns,$usuario,$clave);
            $this->conexion->exec("SET CHARACTER SET utf8");
            echo "Conexion exitosa <br>";
        } catch (PDOException $e) {
            echo "Error de conexion: ".$e->getMessage()."<br>";
            die();
        }
    }
    public function prepare($sql){
        return $this->conexion->prepare($sql);
    }
    public static function singleton_conexion(){
        if (!isset(self::$instancia)) {
            $miclase = __CLASS__;
            self::$instancia = new $miclase;
        }
        return self::$instancia;
    }
     // Evita que el objeto se pueda clonar
    public function __clone(){
        trigger_error('La clonación de este objeto no está permitida',
E_USER_ERROR);
    }
}
echo "SINGLETON<br>";
//Lectura de usuario
$conexion = Singleton::singleton_conexion();
try{
    $consulta = $conexion->prepare("select * from personas");
    $consulta->execute();
    $resultados = $consulta->fetchAll();
    foreach($resultados as $d){
        echo "{$d['id']} , {$d['paterno']}, {$d['materno']}, {$d['nombre']},
{$d['direccion']}<br> ";
    }
}catch(PDOException $e){
    echo "Error al obtener personas <br>";
}
?>
```




SINGLETON
Conexion exitosa
1 , Monzon, Pinto, Jose, xyz
2 , Monzon, Pinto, Jose, xyz