Clases y Objetos

1. Define una clase y un objeto en tus propias palabras.

- Clase: Una clase es un plano o plantilla que define un conjunto de propiedades y métodos que serán comunes a todos los objetos creados a partir de ella. Es como un molde que establece las características y comportamientos que tendrán los objetos.

- Objeto: Un objeto es una instancia de una clase. Es una entidad concreta creada a partir de una clase, que tiene un estado definido por las propiedades y puede realizar acciones mediante los métodos definidos en la clase.

2. ¿Qué propiedades y métodos tenía la clase Coche que creaste?

- La clase `Coche` puede tener propiedades como `color`, `marca`, `modelo`, `año`, y `velocidad`. Los métodos podrían incluir `acelerar()`, `frenar()`, `arrancar()`, y `detener()`. Estas propiedades y métodos permiten definir el estado y comportamiento del coche.

3. ¿Cómo se crea un objeto en PHP? Proporciona un ejemplo.

- Para crear un objeto en PHP, primero se define la clase y luego se usa el operador `new` para instanciar un objeto.

```
<?php
class Coche {
    public $marca;
    public $color;

    public function arrancar() {
        echo "El coche ha arrancado.";
    }
}

// Crear un objeto de la clase Coche
$miCoche = new Coche();
$miCoche->marca = "Toyota";
```

```
$miCoche->color = "Rojo";

$miCoche->arrancar();  // Salida: El coche ha arrancado.

?>
```

4. Describe un ejemplo de la vida real que podría ser representado por una clase y un objeto en programación.

- Un ejemplo de la vida real podría ser una biblioteca. La clase podría ser `Libro` y los objetos podrían ser diferentes libros como "El Quijote", "Cien Años de Soledad", etc. La clase `Libro` tendría propiedades como `título`, `autor`, `año de publicación` y métodos como `prestar ()`, `devolver()`, `consultarDatos()`.

Herencia

1. ¿Qué es la herencia en la programación orientada a objetos?

- La herencia es un principio de la POO que permite crear una nueva clase basada en una clase existente. La nueva clase, llamada clase derivada o subclase, hereda las propiedades y métodos de la clase base o superclase, y puede agregar o modificar funcionalidades adicionales.

2. ¿Qué clase creó la clase CocheDeportivo y qué propiedades y métodos heredó de la clase Coche?

- La clase `CocheDeportivo` se creó a partir de la clase `Coche`. Heredó propiedades como `marca`, `color`, y métodos como `arrancar () ` y `frenar () `. La clase `CocheDeportivo` podría añadir métodos adicionales como `activarTurbo () ` o sobreescribir el método `acelerar ()` para aumentar la velocidad de manera más rápida.

3. Proporciona un ejemplo de herencia en un contexto diferente al del coche (por ejemplo, animales, dispositivos electrónicos, etc.).

- En un contexto de animales, podríamos tener una clase base llamada `Animal` con propiedades como `especie`, `edad`, y métodos como `comer()` y `dormir()`. A partir de ella, podríamos crear una clase `Perro` que herede de `Animal`, añadiendo propiedades como `raza` y métodos como `ladrar()`.

Polimorfismo

1. Define polimorfismo en el contexto de la programación orientada a objetos.

- El polimorfismo es la capacidad de diferentes clases para ser tratadas como una misma clase base, generalmente a través de una interfaz o clase abstracta. Esto permite que una misma operación se comporte de manera diferente en distintas clases. Esencialmente, permite a los métodos con el mismo nombre actuar de manera diferente según el objeto que los esté llamando.

2. ¿Cómo demostraste el polimorfismo en la clase Coche y CocheDeportivo?

- Se puede demostrar el polimorfismo en las clases `Coche` y `CocheDeportivo` definiendo un método con el mismo nombre en ambas clases, pero con un comportamiento diferente. Por ejemplo, ambos pueden tener un método `acelerar()`, donde en `Coche` incrementa la velocidad de forma gradual, y en `CocheDeportivo`, la incrementa de manera más agresiva.

Encapsulamiento

1. Explica qué es el encapsulamiento y por qué es importante en la programación orientada a objetos.

- El encapsulamiento es un principio de la POO que restringe el acceso directo a las propiedades de un objeto y obliga a interactuar con ellas a través de métodos. Esto es importante porque protege la integridad del objeto, previene la modificación indebida de sus datos y permite controlar cómo se accede y modifica la información interna.

2. ¿Cómo protegiste las propiedades color y marca en la clase Coche?

- Las propiedades `color` y `marca` pueden ser protegidas declarando las propiedades como `private` o `protected`, y proporcionando métodos `getter` y `setter` para acceder y modificar estas propiedades de forma controlada.

```
php
class Coche {
    private $color;
    private $marca;

    public function getColor() {
        return $this->color;
    }

    public function setColor($color) {
        $this->color = $color;
    }

    public function getMarca() {
```

```
        return $this->marca;
    }

    public function setMarca($marca) {
        $this->marca = $marca;
    }
}
```

Abstracción

1. Define abstracción en tus propias palabras.

- La abstracción es un principio de la POO que consiste en simplificar la complejidad de un sistema modelando solo las características esenciales y relevantes de un objeto, ignorando los detalles innecesarios. Se centra en el "qué" hace un objeto, no en "cómo" lo hace.

2. ¿Qué es una clase abstracta y cómo se utilizó en el archivo 05_abstraccion.php?

- Una clase abstracta es una clase que no puede ser instanciada por sí misma y generalmente incluye métodos abstractos, los cuales deben ser implementados por las clases derivadas. En el archivo `05_abstraccion.php`, se podría haber definido una clase abstracta `Vehiculo` con métodos abstractos como `mover()` o `detener()`, obligando a las clases hijas a proporcionar una implementación específica de estos métodos.

Constructores y Destructores

1. ¿Qué es un constructor y cuándo se utiliza?

- Un constructor es un método especial que se ejecuta automáticamente al crear una instancia de una clase. Se utiliza para inicializar las propiedades del objeto o para ejecutar cualquier código que deba configurarse al inicio.

2. ¿Cómo inicializaste las propiedades color y marca en el constructor de la clase Coche?

- Las propiedades `color` y `marca` se pueden inicializar directamente dentro del constructor:

```
php
class Coche {
    private $color;
```

```
    private $marca;

    public function __construct ($color, $marca) {
        $this->color = $color;
        $this->marca = $marca;
    }
}

// Crear un objeto de la clase Coche
$miCoche = new Coche ("Rojo", "Toyota");
```

3. ¿Qué es un destructor y cuándo se ejecuta?

- Un destructor es un método especial que se ejecuta automáticamente cuando un objeto es destruido o cuando finaliza la ejecución del script. Se utiliza para liberar recursos o realizar tareas de limpieza.

Estas respuestas cubren los conceptos fundamentales relacionados con las preguntas planteadas en el laboratorio. Si necesitas más detalles o tienes preguntas adicionales, no dudes en pedirlas.