4. Dada la siguiente clase
class Cliente { public $id; private $nombre; private $apellidos; protected $usuario; public $password;
public function __construct($id = null, $nombre = null, $apellidos = null, $usuario = null, $password = null) { $this->id = $id; $this->nombre = $nombre; $this->apellidos = $apellidos; $this->usuario = $usuario; $this->password = $password; } }
Realizo class Registrado extends Cliente;
a. La clase Registrado hereda los atributos $usuario, $id y $password

5. ¿Qué función tiene el constructor?
   a. Inicializar las propiedades del objeto.
6. Deseo crear un objeto de la clase cliente
   b. $cliente = new cliente();
7. ¿Cuántos objetos puedo definir a partir de una clase?
   a. Todos los que necesite
8. ¿Qué es una superclase?
   b. Es una clase de la que desciende o deriva otra clase
9. Propiedades y métodos heredados por las subclase.
   c. Los Protected y Public
10. Creo la subclase usuarioRegistrado a partir de la superclase usuario.
    d. class usuarioRegistrado extends usuario
11. Quiero que se ejecute el constructor de una clase padre
    a. parent::__construct();
12. class cliente { $nombre, $apellidos}
    c. Atributos de tipo public
13. Declaro en una clase un atributo estático llamado gravedad
    a. static $gravedad
14. Deseo usar desde el objeto $obj1 el atributo estático $gravedad declarado en la clase classFisica como public
    c. classFisica::$gravedad
15. Deseo usar desde un método de la clase classFisica el atributo estático $gravedad declarado como private
    a. self::$gravedad
16. Métodos básicos en las clases que no reciben ningún parámetro y devuelven siempre un valor
    b. getters
17. Método setters para el atributo $apellido en una clase
    d. public function setApellido($value) { $this->apellido = $value; }
18. Constructor con parámetros opcionales
    b. public function __construct($p1=null, $p2=null)

19. ¿Qué es la Programación Orientada a Objetos?
    d. Estilo de programación avanzado y extendido

20. ¿Qué son las clases?
    b. Son definiciones a partir de las cuales se crean objetos

21. Concepto de Encapsulamiento
    a. Mantienen ocultas las propiedades de una clase

22. Concepto de abstracción
    a. Conjunto de propiedades y métodos de una clase

23. Variable $this
    b. Permite asignar un valor a una de las propiedades de la clase desde un método

24. $user1=new usuario()
    c. Creo un objeto a partir de la clase usuario

25. $per1→imprimir('$archivo')
    a. Uso el método imprimir del objeto $per1 pasando como parámetro $archivo

26. cláusula extends
    c. Permite establecer el concepto de herencia

27. Se puede acceder a $matricula desde exterior
    d. public $matricula

28. Permite el concepto de encapsulamiento
    a. setters y los getters

29. Permite el concepto de herencia y encapsulación
    d. setter y getters

30. Permite usar atributo definido como static $valorIVA en un método
    a. self::$valorIVA

31. Alumno::saludo()
    a. Saludo es un método declarado como tipo static

32. Propiedad del método __construct
    d. Es un método opcional

33. Propiedad del método __destruct
    a. Las tres opciones dadas son correctas

Sin número

- Dada la siguiente clase
  class Cliente { public $id; public $nombre; public $apellidos; public $usuario; public $password;
  public function __construct($id = null, $nombre = null, $apellidos = null, $usuario = null, $password = null) { $this->id = $id; $this->nombre = $nombre; $this->apellidos = $apellidos; $this->usuario = $usuario; $this->password = $password; } }
  Realizo $usuario = new Cliente(); Ahora deseo obtener el valor $id desde un método de la

clase Cliente
d. echo self::$id

- Dada la siguiente clase
  class Cliente { static public $id = null; protected $nombre; protected $apellidos; protected $usuario; protected $password;
  public function __construct($id = null, $nombre = null, $apellidos = null, $usuario = null, $password = null) { $this->id = $id; $this->nombre = $nombre; $this->apellidos = $apellidos; $this->usuario = $usuario; $this->password = $password; } }
  Realizo $usuario = new Cliente(); Ahora deseo mostrar el valor $id desde fuera de la clase
  b. echo Cliente::id;

- Dada la siguiente clase
  class Cliente { public $id; public $nombre; public $apellidos; public $usuario; public $password;
  public function __construct($id = null, $nombre = null, $apellidos = null, $usuario = null, $password = null) { $this->id = $id; $this->nombre = $nombre; $this->apellidos = $apellidos; $this->usuario = $usuario; $this->password = $password; } }
  Realizo $usuario = new Cliente(); $usuario->nombre = "Pedro";
  a. Se asigna valor a la propiedad $nombre del objeto $usuario