EJERCICIOS REPASO POO

1. Crea una clase Empleado. Sus atributos son nombre, apellidos y sueldo. Encapsula las propiedades mediante *getters/setters* y añade métodos para:
   - Obtener su nombre completo → getNombreCompleto(): string
   - Que devuelva un booleano indicando si debe o no pagar impuestos (se pagan cuando el sueldo es superior a 3333€) → debePagarImpuestos(): bool

Nota: No olvides que cada clase se pondrá en un archivo. En este caso el archivo se llamará Empleado.php

2. Copia la clase del ejercicio anterior y modifícala (nombra el nuevo archivo como EmpleadoTelefonos.php). Añade una propiedad privada que almacene un array de números de teléfonos. Añade los siguientes métodos:
   - public function anyadirTelefono(int $telefono) : void → Añade un teléfono al array
   - public function listarTelefonos(): string → Muestra los teléfonos separados por comas
   - public function vaciarTelefonos(): void → Elimina todos los teléfonos

3. Copia la clase del ejercicio anterior y modifícala(archivo EmpleadoConstructor.php). Elimina los *setters* de nombre y apellidos, de manera que dichos datos se asignan mediante el constructor (utiliza la sintaxis de PHP7). Si el constructor recibe un tercer parámetro, será el sueldo del Empleado. Si no, se le asignará 1000€ como sueldo inicial.

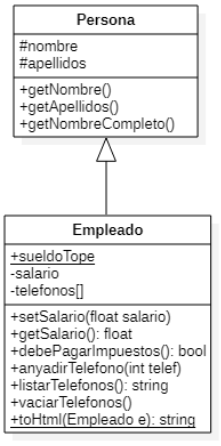
4. Modifica la clase y utiliza la sintaxis de PHP 8 de promoción de las propiedades del constructor.(EmpleadoConstructor8.php)

5. Copia la clase del ejercicio anterior y modifícala (EmpleadoConstante.php). Añade una constante SUELDO_TOPE con el valor del sueldo que debe pagar impuestos, y modifica el código para utilizar la constante.

6. Copia la clase del ejercicio anterior y modifícala(EmpleadoSueldo.php). Cambia la constante por una variable estática sueldoTope, de manera que mediante *getter/setter* puedas modificar su valor.

7. Copia la clase del ejercicio anterior y modifícala (EmpleadoStatic.php). Completa el siguiente método con una cadena HTML que muestre los datos de un empleado dentro de un párrafo y todos los teléfonos mediante una lista ordenada (para ello, deberás crear un *getter* para los teléfonos):
   - public static function toHtml(Empleado $emp): string

8. Copia la clase del ejercicio anterior y modifícala.
   Crea una clase Persona (Persona.php) que sea padre de Empleado, de manera que Persona contenga el nombre y los apellidos, y en Empleado quede el salario y los teléfonos.

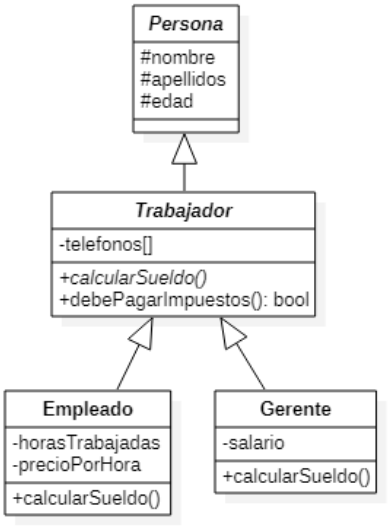


9. Copia las clases del ejercicio anterior y modifícalas.
   Añade en Persona un atributo edad. Tenga en cuenta que ahora un empleado pagará impuestos si tiene más de 21 años y su sueldo supera el tope. Modifica todo el código necesario para mostrar y/o editar la edad cuando sea necesario.

10. Copia las clases del ejercicio anterior y modifícalas.
    Añade nuevos métodos que hagan una representación de todas las propiedades de las clases Persona y Empleado, de forma similar a los realizados en HTML, pero sin que sean estáticos, de manera que obtenga los datos mediante $this.
    function public __toString(): string
    El método __toString() es un método mágico que se invoca automáticamente cuando queremos obtener la representación en cadena de un objeto.

11. Transforma Persona a una clase abstracta donde su método estático toHtml(Persona $p) tenga que ser redefinido en todos sus hijos.

12. Cambia la estructura de clases conforme al gráfico respetando todos los métodos que ya están hechos.
    Trabajador es una clase abstracta que ahora almacena los teléfonos y donde calcularSueldo es un método abstracto de manera que:
    - El sueldo de un Empleado se calcula a partir de las horas trabajadas y lo que cobra por hora.
    - Para los Gerentes, su sueldo se incrementa porcentualmente en base a su edad:
      salario + salario*edad/100



13. Utilizando las clases de los ejercicios anteriores:
    a. Crea una clase Empresa que además del nombre y la dirección, contenga una propiedad con un array de Trabajadores, ya sean Empleados o Gerentes.
    b. Añade *getters/setters* para el nombre y dirección.
    c. Añade métodos para añadir y listar los trabajadores.
       - public function anyadirTrabajador(Trabajador $t)
       - public function listarTrabajadoresHtml() : string .
         Utiliza Trabajador::toHtml(Persona $p)
    d. Añade un método para obtener el coste total en nóminas.
       - public function getCosteNominas(): float
         Recorre los trabajadores e invoca al método calcularSueldo().

14. Copia las clases del ejercicio anterior y modifícalas.
    a. Crea un interfaz JSerializable, de manera que ofrezca los métodos:

- toJSON(): string . Utiliza la función json_encode(mixed). Ten en cuenta que como tenemos las propiedades de los objetos privados, debes recorrer las propiedades y colocarlas en un mapa. Por ejemplo:

```
<?php
public function toJSON(): string {
   foreach ($this as $clave => $valor) {
      $mapa->$clave = $valor;
   }
   return json_encode($mapa);
}
?>
```

- toSerialize(): string .Utiliza la función serialize(mixed)

b. Modifica todas las clases que no son abstractas para que implementen el interfaz creado.