# Ejercicios PHP 9

## 1. Producto

Creamos la clase Producto con las propiedades privadas nombre, precio y cantidad, creamos un construct para inicializar los valores, los métodos getNombre, getPrecio y getCantidad para acceder a estos valores, creamos la clase hija Producto importado que añade la propiedad ImpuestoAdicional y el método calcularPrecioFinal, creamos instancias de ambas clases y probamos los métodos.

```php
<?php
class Producto{

    private $nombre; //Nombre del producto
    private $precio; //Precio del producto
    private $cantidad; //Cantidad del producto

    public function __construct($nombre, $precio, $cantidad){
//Inicializa los datos del producto

        $this->nombre = $nombre;
        $this->precio = $precio;
        $this->cantidad = $cantidad;

    }

    public function getNombre(){ //Devuelve el nombre del producto

        return $this->nombre."\n";

    }
    public function getPrecio(){ //Devuelve el precio del producto

        return $this->precio."\n";

    }
    public function getCantidad(){ //Devuelve la cantidad del producto

        return $this->cantidad."\n";


    }
}

class ProductoImportado extends Producto{
```

```php
    private $impuestoAdicional; //Impuesto extra del producto importado

    public function __construct($nombre, $precio, $cantidad,
$impuestoAdicional){ //Inicializa los datos del producto

        parent::__construct($nombre, $precio, $cantidad);
        $this->impuestoAdicional = $impuestoAdicional;

    }

    public function calcularPrecioFinal(){ //Calcula el precio total del
producto

        $precio = $this->getPrecio();
        return "Precio final: ".$precio + $this->impuestoAdicional;
    }
}

$producto = new Producto("pan", 3, 2);
echo $producto->getNombre();
echo $producto->getPrecio();
echo $producto->getCantidad();

$productoImportado = new ProductoImportado("azúcar", 6, 1, 5);
echo $productoImportado->getNombre();
echo $productoImportado->getPrecio();
echo $productoImportado->getCantidad();
echo $productoImportado->calcularPrecioFinal();
?>
```

```
TERMINAL
PS C:\xampp\htdocs\programacion\ejercicio9> php producto.php
pan
3
2
azúcar
6
1
Precio final: 11
PS C:\xampp\htdocs\programacion\ejercicio9>
```

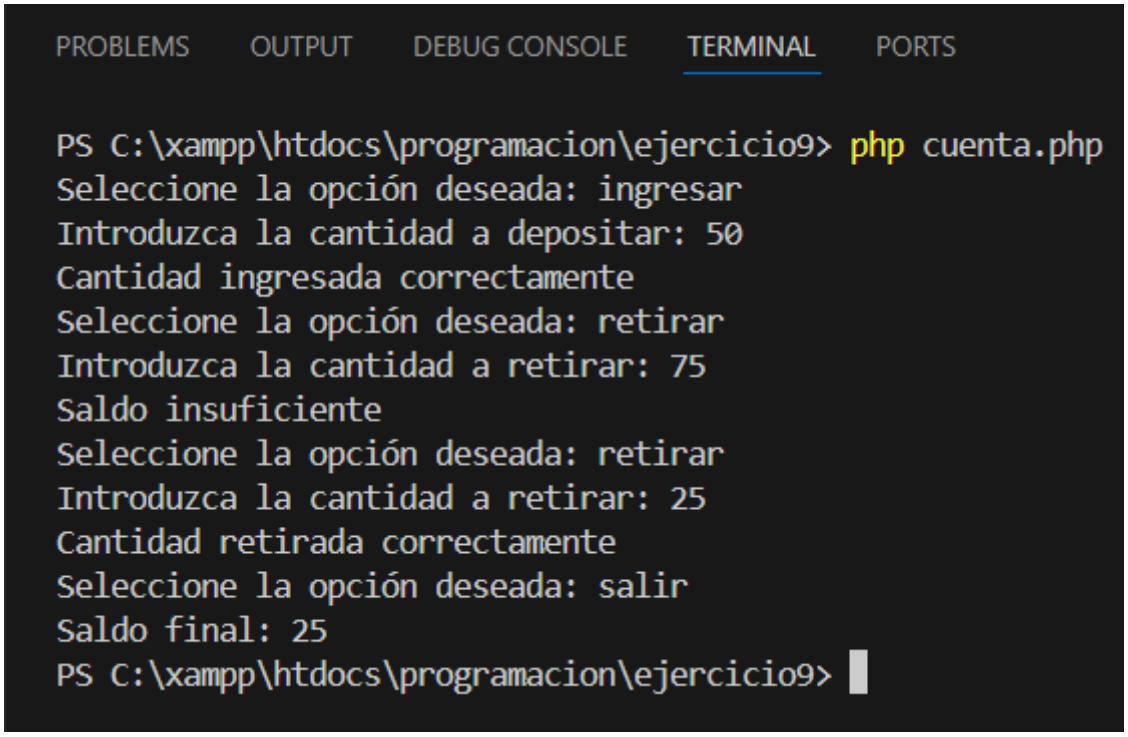
## 2. Cuenta bancaria

Creamos la clase cuentaBancaria con los atributos titular, saldo y tipoCuenta,

con los métodos depositar, retirar y verificarSaldoSuficiente para comprobar que hay dinero suficiente para retirar, añadimos el método getSaldo para mostrar el saldo al finalizar, creamos una instancia de la clase y probamos los métodos.

```php
<?php
class CuentaBancaria{

    private $titular; //Titular de la cuenta
    private $saldo; //Saldo de la cuenta
    private $tipoCuenta; //Tipo de cuenta

    public function __construct($titular, $tipoCuenta){ //Introduce los
datos iniciales de la cuenta

        $this->titular = $titular;
        $this->tipoCuenta = $tipoCuenta;
        $this->saldo = 0;

    }

    public function depositar($cantidad){ //Ingresa dinero en la cuenta

        $this->saldo = $this->saldo + $cantidad;
        return "Cantidad ingresada correctamente";

    }

    public function retirar($cantidad){ //Retira dinero de la cuenta

        if ($this->verificarSaldoSuficiente($cantidad) == false){
//Comprueba si hay saldo suficiente

            return "Saldo insuficiente";

        } else{

            $this->saldo = $this->saldo - $cantidad;
            return "Cantidad retirada correctamente";

        }
    }

    private function verificarSaldoSuficiente($cantidad){ //Comprueba si
hay saldo suficiente

        if ($this->saldo < $cantidad){

            return false;

        } else{

            return true;
```

```
        }
    }

    public function getSaldo(){ //Devuelve el saldo actual

        return  $this->saldo."\n";
    }
}

$cuenta = new CuentaBancaria("Antonio Perales", "corriente");

do{

    $opcion = readline("Seleccione la opción deseada: "); //Permite
escoger que hacer con la cuenta

    if ($opcion == "ingresar"){ //Ingresar dinero en la cuenta

        $cantidad = readline("Introduzca la cantidad a depositar: ");
        echo $cuenta->depositar($cantidad)."\n";

    } elseif ($opcion == "retirar"){ //Retirar dinero de la cuenta

        $cantidad = readline("Introduzca la cantidad a retirar: ");
        echo $cuenta->retirar($cantidad)."\n";

    } elseif ($opcion == "salir"){ //Salir

        break;

    } else{

        echo "Opción inválida"."\n"; //No se introduce una opción válida
    }
} while ($opcion != "salir"); //Repetir hasta que se introduzca salir

echo "Saldo final: ".$cuenta->getSaldo();

?>
```




## 3. Usuario

Creamos la clase Usuario con las propiedades privadas nombre e email y un construct para inicializarlos y añadimos el método mostrarInfo que muestre las propiedades.

Creamos la clase hija Administrador que añade la propiedad nivelAcceso y sobrescribimos mostrarInfo para añadir esta propiedad. Creamos instancias para ambas clases y probamos los métodos.

```php
<?php
class Usuario{

    private $nombre; //Nombre del usuario
    private $email; //Correo del usuario

    public function __construct($nombre, $email){ //Inicializa los datos del usuario

        $this->nombre = $nombre;
        $this->email = $email;

    }

    public function mostrarInfo(){ //Muestra los datos del usuario
```

```php
        return "Nombre: ".$this->nombre."\n".
        "E-mail: ".$this->email."\n";

    }
}

class Administrador extends Usuario{

    private $nivelAcceso; //Nivel de acceso del administrador

    public function __construct($nombre, $email, $nivelAcceso){
//Inicializa los datos del administrador

        parent::__construct($nombre, $email);
        $this->nivelAcceso = $nivelAcceso;

    }

    public function mostrarInfo(){ //Muestra los datos del administrador

        return parent::mostrarInfo().
        "Nivel de acceso: ".$this->nivelAcceso."\n";

    }
}

$usuario = new Usuario("Juan", "Juan@gmail.com");
echo $usuario->mostrarInfo();

$administrador = new Administrador("Manuel", "Manuel@gmail.com", 3);
echo $administrador->mostrarInfo();
?>
```

```
TERMINAL

PS C:\xampp\htdocs\programacion\ejercicio9> php usuario.php
Nombre: Juan
E-mail: Juan@gmail.com
Nombre: Manuel
E-mail: Manuel@gmail.com
Nivel de acceso: 3
PS C:\xampp\htdocs\programacion\ejercicio9>
```

## 4. Vehículo

Creamos la clase Vehículo con las propiedades privadas marca y modelo y un construct para inicializarlos y añadimos el método encender que indique que el vehículo esta encendido. Creamos la clase hija Coche que añade la propiedad combustible y el método mostrarDetalles que muestre los datos del coche. Creamos una instancia de coche y probamos los métodos.

```php
<?php
class Vehiculo{

    private $marca; //Marca del vehículo
    private $modelo; //Modelo del vehículo

    public function __construct($marca, $modelo){ //Inicializa los datos
del vehículo

        $this->marca = $marca;
        $this->modelo = $modelo;

    }

    public function encender(){ //Indica que el vehículo esta encendido

        return "El vehículo esta encendido"."\n";

    }

    public function getMarca(){ //Devuelve la marca del vehículo

        return $this->marca;

    }

    public function getModelo(){ //Devuelve el modelo del vehículo

        return $this->modelo;

    }
}

class Coche extends Vehiculo{

    private $combustible; //Tipo de combustible del coche

    public function __construct($marca, $modelo, $combustible){
//Inicializa los datos del coche
```

```php
        parent::__construct($marca, $modelo);
        $this->combustible = $combustible;

    }

    public function mostrarDetalles(){ //Muestra los datos del coche

        return "Marca: ".$this->getMarca()."\n".
        "Modelo: ".$this->getModelo()."\n".
        "Combustible: ".$this->combustible."\n";

    }
}

$coche = new Coche("Toyota", "Corola", "Gasolina");
echo $coche->encender();
echo $coche->mostrarDetalles();

?>
```

```
TERMINAL

 Nivel de acceso: 3
 PS C:\xampp\htdocs\programacion\ejercicio9> php vehiculo.php
 El vehículo esta encendido
 Marca: Toyota
 Modelo: Corola
 Combustible: Gasolina
 PS C:\xampp\htdocs\programacion\ejercicio9>
```

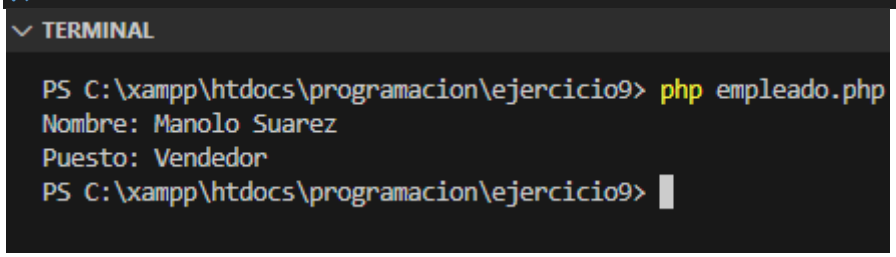
## 5. Empleado

Creamos la clase Empleado con las propiedades privadas nombre, sueldo y puesto y un construct para inicializarlos y añadimos los métodos setSueldo y getSueldo que . Creamos la clase hija Coche que añade la propiedad combustible y el método mostrarDetalles que muestre los datos del coche. Creamos una instancia de coche y probamos los métodos.

```php
<?php
class Empleado{

    private $nombre; //Nombre del empleado
    private $sueldo; //Sueldo del empleado
    private $puesto; //Puesto del empleado

    public function __construct($nombre, $sueldo, $puesto){ //Inicializa
los datos del empleado
```

```php
        $this->nombre = $nombre;
        $this->sueldo = $sueldo;
        $this->puesto = $puesto;

    }

    public function setSueldo($nuevoSueldo){ //Actualiza el sueldo del
empleado

        $this->sueldo = $nuevoSueldo;

    }

    public function getSueldo(){ //Devuelve el sueldo del empleado

        return "Sueldo: ".$this->sueldo."\n";

    }

    public function getNombre(){ //Devuelve el nombre del empleado

        return $this->nombre."\n";

    }

    public function getPuesto(){ //Devuelve el puesto del empleado

        return $this->puesto."\n";

    }
}

class Manager extends Empleado{

    private $departamento; //Departamento del manager

    public function __construct($nombre, $sueldo, $puesto,
$departamento){ //Inicializa los datos del manager

        parent::__construct($nombre, $sueldo, $puesto);
        $this->departamento = $departamento;

    }

    public function revisarEmpleado($empleado){ //Devuelve los datos del
empleado asignado

        return "Nombre: ".$empleado->getNombre().
        "Puesto: ".$empleado->getPuesto();
```

```
    }
}

$manager = new Manager("Juan Pérez", 1900, "Manager", "Ventas");
$empleado = new Empleado("Manolo Suarez", 1500, "Vendedor");
echo $manager->revisarEmpleado($empleado)

?>
```




https://github.com/Rodrigo-Garcia-Ortiz/Programacion.git