EJERCICIO 1

Crear una clase Persona con tres propiedades (nombre, apellidos y sexo) que no deben ser accesibles directamente desde el exterior.

- Define un constructor que puede recibir como parámetro las 3 propiedades o únicamente las dos primeras. En este caso el valor que debe tomar el atributo sexo es 'H'.
- Al mostrar el sexo debe devolver 'hombre' si el valor de sexo es 'H', 'mujer' si el valor de sexo es 'M', y desconocido para cualquier otro valor.
- De una persona se podrán consultar cualquiera de las tres propiedades pero sólo se podrá modificar el sexo y el nombre.
- La clase debe incluir un método de *informacionPersona* que devuelva un String con los datos de la persona.
- Crea varios objetos para comprobar que funciona correctamente y verifica el funcionamiento de *informacionPersona*.

Un ejemplo de la salida podría ser:

```
Su nombre es: Pedro
Sus apellidos son: Pan Bueno
Su sexo es: hombre
```

EJERCICIO 2

class Persona2 {

$nombre;

$profesion;

$edad;

}

- Crea la clase Persona2, de forma que cumpla con la ocultación y el encapsulamiento teniendo en cuenta que sólo se podrá modificar la profesión y la edad. El nombre de la persona no puede variar.
- Implementa un constructor de forma que el nombre y la edad siempre sean obligatorios, la profesión si la tenemos se pasará al constructor y de lo contrario aparecerá como “Sin especificar”.
- Modifica el constructor para que sólo se creen los objetos si el nombre y la profesión son de tipo string y la edad de tipo entero.

- Implementa el método mágico toString para que devuelva las propiedades y el valor de cada una de ellas. Crea una clase PruebasPersona, en la que se creen dos objetos (uno para cada posibilidad del constructor) y se visualice el contenido de cada uno de ellos.

EJERCICIO 3

```
class Miscelanea {

    static function autores() {
      return "Pepe y Paco";
    }

  }
```

- Dada la clase anterior, implementa la llamada al método autores de las dos formas posibles.

EJERCICIO 4

```
class Estatica {

  static $uno = '1';
  var $dos = '2';

}
```

Dada la clase anterior:

- ¿Qué imprimen las siguientes líneas? ¿Qué tipo de visibilidad tiene $uno y $dos?

```
echo "Estatica::\$uno: (" . Estatica::$uno . ")<br />";
echo "(" . Estatica::$dos . ")<br />";
```

- ¿Qué imprimen las siguientes líneas?

```
$o = new Estatica();
echo "\$o->uno: (" . $o->uno . ")<br />";
echo "\$o->dos: (" . $o->dos . ")<br />";
```

EJERCICIO 5

```
class cuenta_objs {

   static private $contador = 0;

   static function cuantosObjetos() {
      return cuenta_objs::$contador;
   }

   function __construct() {
      cuenta_objs::$contador++;
   }


}


$o1 = new cuenta_objs();
$o2 = new cuenta_objs();
$o3 = new cuenta_objs();
```

//Visualiza el número de objetos existentes

```
unset($o2);
```

//Visualiza el número de objetos existentes

Dada la clase anterior:

- Implementan el código de las líneas comentadas. ¿Cuál es el resultado?

- Implementa el destructor de forma que cuando eliminamos un objeto se reste del contador. Comprueba el correcto funcionamiento del número de objetos existentes.