# Enunciados Ejercicios Patrones de Diseño

## N1-PATRON SINGLETON

Tienes la siguiente definición de clase que pretende modelar al famoso personaje de Tigger de los libros "Winnie The Pooh" de A A. Milne:

Archivo: tigger.php

class Tigger {  
  
       private function \_\_construct() {  
               echo "Building character..." . PHP\_EOL;  
       }  
  
       public function roar() {  
               echo "Grrr!" . PHP\_EOL;  
       }  
  
}

Parece razonable esperar que sólo haya un objeto Tigger (después de todo, ¡él es el único!), pero la implementación actual permite tener múltiples objetos Tigger diferentes:

Refactorizar la clase Tigger en un Singleton considerando los siguientes puntos:

* Define el método getInstance ()que carezca de argumentos. Esta función es responsable de crear una instancia de la clase Tigger sólo una vez y devolver esta única instancia cada vez que se llama.
* Imprime en pantalla múltiples veces el rugido de Tigger.
* Añade un método getCounter ()que devuelva el número de veces que Tigger ha realizado rugidos.

## N2-PATRON ADAPTER

El siguiente problema fue adaptado de [FREEMAN] págs. 238-240.

Supone que tienes las siguientes dos clases de PHP.

Archivo: poultry.php

class Duck {  
  
       public function quack() {  
              echo "Quack \n";  
       }  
  
       public function fly() {  
              echo "I'm flying \n";  
       }  
}  
  
class Turkey {  
  
       public function gobble() {  
              echo "Gobble gobble \n";  
       }  
  
       public function fly() {  
       echo "I'm flying a short distance \n";  
       }  
}

¿Sam Quiere que sus pavos se comporten como patos, por lo que tienes que aplicar el adaptador pattern. En el mismo archivo, escribe una clase llamada TurkeyAdapter y asegúrate de que tenga en cuenta lo siguiente:

La traducción del quack entre clases es fácil: simplemente llama al método Gobble cuando sea apropiado.

Aunque ambas clases tienen un método fly, los pavos sólo pueden volar a rachas cortas, no pueden volar largas distancias como los patos. Para mapear entre el método fly de un pato y el método del pavo, debe llamarse al método fly del pavo cinco veces para compensarlo.

Prueba tu clase con el siguiente código:

Archivo: index.php

function duck\_interaction($duck) {  
       $duck->quack();  
       $duck->fly();  
}  
  
$duck = new Duck;  
$turkey = new Turkey;  
$turkey\_adapter = new TurkeyAdapter($turkey);  
echo "The Turkey says...\n";  
$turkey->gobble();  
$turkey->fly();  
echo "The Duck says...\n";  
duck\_interaction($duck);  
echo "The TurkeyAdapter says...\n";  
duck\_interaction($turkey\_adapter);  
  
Output  
The expected output is as follows:  
The Turkey says...  
Gobble gobble  
I'm flying a short distance  
The Duck says...  
Quack  
I'm flying  
The TurkeyAdapter says...  
Gobble gobble  
I'm flying a short distance  
I'm flying a short distance  
I'm flying a short distance  
I'm flying a short distance  
I'm flying a short distance