Introducción a los Patrones de Diseño

Martínez Coronel Brayan Yosafat

Ramírez Espinoza Luis Armando

# ¿Qué significa the Gang of Four (GoF)?

(Yosafat) Se trata de los autores del libro “Design Patterns: Elements of Reusable Object-Oriented Software”, que como era demasiado largo, fue reducido a “GoF book” en el argot de la comunidad; se trata de un grupo de (¡sorpresa!) cuatro personas: Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, quienes realizaron grandes aportes en los comienzos de los patrones de diseño. A menos de que se trate de the Gang of Four, el grupo comunista chino de 1967, (lo cual dudamos).

Tomado de <http://wiki.c2.com/?GangOfFour>.

# En la introducción (de la actividad) se marcan dos ideas, proporcione una interpretación de estas:

[...] ***capturan soluciones que se han desarrollado y evolucionado con el tiempo.***

[...] ***no son los diseños que las personas tienden a generar inicialmente.***

(Luis) En realidad, hablan de lo mismo con diferentes palabras, en la actividad, mencionan que como sin registrados por la propios desarrolladores que cuentan con la experiencia, y como la experiencia de ahora no es la misma que en 1994 (publicación del libro), es por lo que se menciona que son evolutivos, pues aunque en realidad se trata de una solución abstracta, los problemas de ayer, también evolucionan, y es necesario que nuestras soluciones lo hagan igual.

# Los patrones están clasificados de acuerdo a su propósito, indique a qué se refiere cada uno de estos propósitos:

(Luis) Bueno, antes de responder, pensemos en lo siguiente: en la forma más abstracta, todo se representa en clase y objeto, ahora pensemos, se sabe que la clase y el objeto tienen una estructura y un comportamiento, como analogía digamos la clase u objeto es el sujeto; y el comportamiento el verbo, en su forma más simple, la oración se conforma de esto, pero si vemos más allá de ello, recordemos que hay **estilos** para hacer las oraciones, su *creación*. Cada uno de ellos, tiene problemas comunes, y al igual que en las oraciones, se han clasificado, creacional: ¿cómo hago *la “*oración*”*?, estructural: ¿De qué se compone mi “oración”? y de comportamiento: ¿Qué hará mi sujeto?

# ¿En qué clasificación entra MVC, Singleton e Iterator?

MVC puede ser clasificado como creacional, así como Singleton. En cambio, Iterator lo clasificaremos con el patrón de comportamiento.

# ¿Pueden los patrones de diseño, usarse con otros patrones de diseño?

(Yosafat) Por supuesto que se puede, tomemos dos patrones: Singleton y Comando, uno hace que tengamos una instancia estática, mientras que el otro, usa las interfaces para cambiar de manera dinámica el comportamiento de un objeto, eso es algo que una persona que ha leído algo de patrones puede saber, pero, ¿cómo podemos ilustrar esto?

Bien, tomemos el siguiente ejemplo: por mi experiencia, conozco el uso de la API de Google Maps para Android, y, la usé para una aplicación de navegación, por supuesto, el usuario (en sentido figurado, ya que el que pasa es el mapa) debe pasar por la selección del destino, luego selección de ruta y al modo de navegación. Ahora, construir un mapa es realmente agotador en términos de recursos, ya que el mapa requiere una gran cantidad de datos (como los marcadores en el mapa, las líneas a dibujar, animaciones para el marcador del usuario, etc) por lo que, crear una instancia cada vez que se requiere un mapa es sumamente ineficiente, esto suena totalmente a Singleton.

Ahora, el botón de regresar no hace lo mismo siempre, bueno, regresa, pero eso es relativo, hace cosas bastante diferentes dependiendo de en qué **estado** se encuentra el mapa, por ejemplo, puede volver a mostrar los resultados de búsqueda, puede salir del modo de navegación, esto suena totalmente a otro patrón: Comando, podemos cambiar el objeto controlador del botón de regreso, mediante interfaces (en Java) o directamente el tipo de dato Function en Dart. Pero, tocamos otro patrón extra: Estado, el mapa desde un inicio tiene estados y si no se modela de esta forma se harán cosas demasiado complicadas (como un archivo de 6500 líneas, según recuerdo).

# Usando las tablas (proporcionadas en la actividad), rellene según la información de MVC, Singleton e Iterator:

## MVC

|  |  |
| --- | --- |
| Nombre del Patrón y clasificación | Modelo/Vistas/Controlador - Creador |
| El problema | Los diseños de Interfaz de Usuario juntan la aplicación, la presentación y el comportamiento de la IU, limitando severamente la flexibilidad y la reusabilidad.  En general, agrupar objetos, y tratar al conjunto como un objeto individual. |
| La resolución | MVC desacopla las vistas para que sean actualizadas cuando los modelos cambien |
| Las consecuencias | En ocasiones, entender esto e implementarlo significa reconstruir los componentes desde cero. |

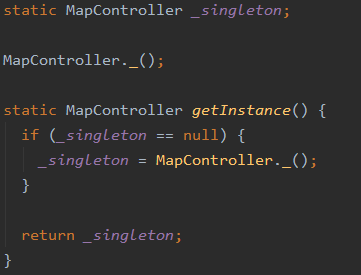
|  |  |
| --- | --- |
| Intención |  |
| Motivación |  |
| Aplicabilidad |  |
| Estructura |  |
| Colaboraciones |  |
| Implementación |  |
| Código ejemplo |  |

## Singleton - Yosafat

|  |  |
| --- | --- |
| Nombre y clasificación | Singleton - Creacional |
| El problema | Crear una instancia cuesta demasiado recursos / se requiere usar una instancia compartida en distintas partes. |
| La resolución | Impedimos la creación externa de instancias, manejamos una instancia estática, de esta forma se comparte. |
| Las consecuencias | Esto va en contra de los principios SOLID, ya que si no se usa bien, puede tener más de una responsabilidad, lo que lleva a complicar de manera exponencial las pruebas que se quieran hacer. Se considera incluso, un antipatrón. |

|  |  |
| --- | --- |
| Intención | Crear una instancia que se requiere compartir en distintas partes. |
| Motivación | Se requiere que la clase tenga un campo estático del tipo de la propia clase, hacer privado el constructor y un método estático que regresa la instancia. |
| Aplicabilidad | Cuando se requiere que se comparta la instancia: flujo de documentos de la impresora. Cuando se gastan muchos recursos al crear una instancia. |
| Estructura |  |
| Colaboración | Solamente es la clase que tiene el Singleton conectada a sí misma. |
| Implementación | Debe de verdad necesitar el patrón, en caso contrario, lo que puede provocar son grandes complicaciones para probar algo. |

Código de ejemplo (escrito en Dart)

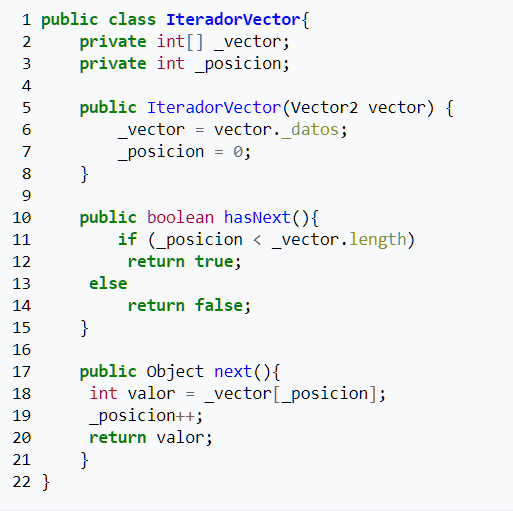


## Iterator - Luis

|  |  |
| --- | --- |
| Nombre del Patrón y clasificación | Iterador - Comportamiento |
| El problema | Unificación de la iteración de estructuras de datos. |
| La resolución | Creamos una interfaz Iterator, que cree un iterador, el iterador es un objeto tiene la responsabilidad de iterar una instancia en específico. |
| Las consecuencias | También se unifica el modo de agregar a la lista. |

|  |  |
| --- | --- |
| Intención | Ser un mecanismo de acceso en una estructura de datos, sin exponer la estructura interna. |
| Motivación | Surge de querer recorrer una estructura sin exponer su forma, unificando en un sólo bloque la forma de iterar para muchas estructuras. |
| Aplicabilidad | Prácticamente las estructuras que deseen iterar (y tengan forma de), en general, se puede ver en las propias estructuras de un lenguaje de programación. Como la interfaz Iterator o IterableCollection de Java. |
| Estructura |  |
| Colaboraciones | Un iterador concreto es encargado de guardar la posición actual de la estructura, determina si hay más elementos adelante, y regresa el siguiente. |
| Implementación | Se debe de tener en cuenta que se debe de crear el objeto especializado en iterar la estructura que queremos, no sólo la interfaz. |

Código de ejemplo - Java



# Realice el ejercicio de Tigger en Python usando Singleton (agregar código fuente y pantallas de ejecución):

# ¿Hemos detectado alguna oportunidad de usarlos en nuestro sistema de software?

(Yosafat y Luis) Sí, de hecho usamos MVC en el sistema, como usted lo menciona: resulta muy común usarlo en el desarrollo web. Aunque, hay alguna que otra oportunidad de agregar patrón Singleton para el Esquema de Mayor.

# ¿Han usado otro patrón de diseño? Si es así, indique cómo y si es posible añadir un fragmento de código:

(Yosafat) Sí, yo he usado patrón Comando para unificar el envío de peticiones a la API, y también me pongo a ver la estructura interna de Dart y la de Java, veo que Dart tiene el tipo de dato Function, que de hecho, se logra con el patrón Comando, ambos tienen las colecciones en Iterator, y la lista continúa bastante.

(Luis) Nel.

# ¿Es MVC un patrón, una arquitectura, un estilo arquitectónico, todas o ninguna? (Explique)

Es un patrón de diseño, que resuelve la problemática de juntar varios objetos en un sólo grupo, y tratar a ese conjunto como uno sólo, generalizando así el diseño.