Singleton

Nombre:
Singleton

Clasificación del patrón:
Creacional

Intención:
Asegurar que una clase tenga una única instancia, y proveer un punto de acceso global a la misma

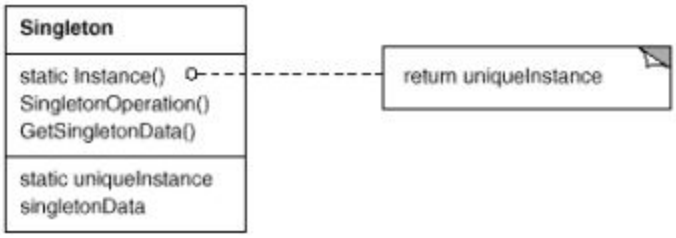
Otros nombres:
Singular o único

Motivación:
Observando sistemas como los encargados de realizar impresiones simultáneas se hizo necesaria la creación de un programa que gestiona de forma única los elementos que ingresaban en ella, para así garantizar la atomicidad de la operación. Éste patrón también fue inspirado en el control de ventanas.

Aplicabilidad:
Se debe usar cuando:

- Se requiere solo una instancia de una clase y ésta deba ser de acceso global
- Se hace una clase de solo lectura

Estructura:



Participantes:

- Singleton: Define una operación getInstance que permite a los clientes acceder a la única instancia que es creada por él también.

Colaboraciones:
Los clientes acceden a la única instancia solamente a través del getInstance

Ventajas:

- Control completo sobre la creación
    - Puede denegar creaciones basadas en reglas
    - Puede ser configurado para permitir cierto número de instancias

Desventajas:

- Se dificulta su implementación en ambientes multihilo
- Los cargadores de clases múltiples pueden dar varias instancias de la clase Singleton
- La implementación en ambientes multi hilos de C++ y la implementación superficial es difícil
    - ¿Quién libera la memoria retornada por el miembro de la función getInstance?
    - El chequeo de la doble comprobación no funciona sin código no portable

Implementación:
Por lo general los constructores usados en la instanciación de objetos son de tipo público. Singleton convierte el constructor de su clase en privado con el fin de ocultarlo de otras instanciaciones.

Para instanciar el objeto se hace a través de un método público y estático que verifica si el objeto ha sido creado antes, si encuentra una instancia devuelve la referencia a ella, si no, crea una nueva instancia.

Código de ejemplo:

- Main:

```
1  package Singleton;
2  public class Main
3  {
4      public static void main(String[] args)
5      {
6          for(int num=1; num<=5; num++)
7          {
8              Coche.getInstancia();
9          }
10     }
11 }
```

- Coche:

```
1  package Singleton;
2  public class Coche
3  {
4       private static Coche instancia;
5      // ----------------------------------------
6      private Coche() {
7      }
8      // ----------------------------------------
9       public static Coche getInstancia()
10     {
11          if (instancia == null) {
12              instancia = new Coche();
13              System.out.println("El objeto ha sido creado");
14          }
15          else {
16              System.out.println("Ya existe el objeto");
17          }
18          return instancia;
19     }
20 }
```

Usos conocidos:

- En java, la clase Runtime.
- Conexión a la base de datos.
- Login de clientes.

Patrones relacionados:

- Facade
- State

Bibliografía:

No específico. (No específico). GoF Design Patterns (Versión 2.1.0) [Aplicación móvil]. Descargado de: https://drive.google.com/file/d/0BywiVyFlIabXcVhGZlJBcnhWTkU/view.

Patrones de Diseño Software [Página web]. Sin fecha. Ubicación: https://informaticapc.com/patrones-de-diseno/singleton.php.

Gómez, V. (30 de diciembre de 2014). Patrones de Diseño. Patrón Singleton. Instinto Binario. https://instintobinario.com/patrones-de-diseno-patron-singleton/.

Junta de Andalucía. (s.f). Singleton. Marco de Desarrollo de la Junta de Andalucía. http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/202.