Algoritmos y Estructuras de Datos
Cristian Laynez - 201281
Elean Rivas - 19062

Patrón de diseño Singleton

En ingeniería de software, singleton o instancia única es un patrón de diseño que le permite limitar el valor del objeto o el tipo de la clase creada a un solo objeto. El objetivo es garantizar que una clase tenga solo una instancia y proporcionar un punto de acceso global para ella. El modo singleton se implementa creando un método en nuestra clase que solo crea una instancia de objeto cuando la instancia de objeto no existe. Para asegurar que la clase no pueda ser instanciada nuevamente, el alcance del constructor está restringido (usando modificadores de acceso como protected o private). La detección de patrones puede ser complicada en programas multiproceso. Si dos subprocesos de ejecución intentan crear una instancia al mismo tiempo, pero la instancia aún no existe, solo uno de ellos puede crear correctamente el objeto

Ventajas

- Reduce el espacio de nombres. El patrón es una mejora sobre las variables globales. Ya no se reservan nombres para las variables globales, ahora solo existen instancias

- Controla el acceso a la instancia única, porque la clase Singleton encapsula la única instancia. Así se obtiene control sobre cómo y cuándo se accede a ella.

- Permite el refinamiento de las operaciones y la representación.

- Permite un número variable de instancias. El patrón es fácilmente configurable para permitir más de una instancia

- Más flexible que las operaciones de clases

Desventajas

- Serializabilidad: Es muy común que los singletons sean con estado. Estos singletons generalmente no deben ser serializables

- Restricciones de codificación: Hay cosas que se pueden hacer en clases normales, pero que están prohibidas en las clases enum . Por ejemplo, accediendo a un campo estático en el constructor. El programador debe tener más cuidado ya que está trabajando en una clase especial.

Patrón de diseño Factory

El patrón de Factory es el que regresa una instancia de un severo posible cambio depende de la data que se brindó. Usualmente todas las clases retorna algo en común con las clases padres y con métodos en común, pero cada uno de ellos realiza una tarea diferente y está optimizado para diferentes tipos de datos. Esta suele ocultar los detalles de creación del objeto.

La intención del Factory Method es tener una clase a la cual delegar la responsabilidad de la creación de los objetos, para que no sea el mismo programador el que decida que clase instanciar, sino que delegó esta responsabilidad al Factor´confiando en que este le regresará la clase adecuada para trabajar (Blancarte, 2014).

Ventajas

- La factoría es reutilizable, solo hay que pasarla como dependencia.
- El testing del cliente es mucho más sencillo, podemos usar mocks para la factoría y simular cualquier tipo de respuesta por parte de la misma, permitiendo testear todos los casos de uso posibles.
- Si queremos añadir un nuevo objeto, solo hay que editar la factoría y añadirlo ahí, pasará a estar disponible para todos los usuarios de la factoría sin tener que replicar código

(Sánchez, 2017)

Desventajas

- Si dado caso la cantidad de opciones crece a montones, la factoría puede llegar a ser una clase difícil de mantener y habría que encontrar otras alternativas al switch.

(Sánchez, 2017)

Referencias:

Blancarte, O. (2014). Patrón de Diseño Factory. Extraído de: https://www.oscarblancarteblog.com/2014/07/18/patron-de-diseno-factory/

Sánchez, M. (2019). Patrón Factory. Extraído de: https://dev.to/mangelsnc/patron-factory-58gg