Patrón de Diseño Singleton

PROGRAMACIÓN DE DISPOSITIVOS MÓVILES

Carlos Rojas Sánchez

Licenciatura en Informática

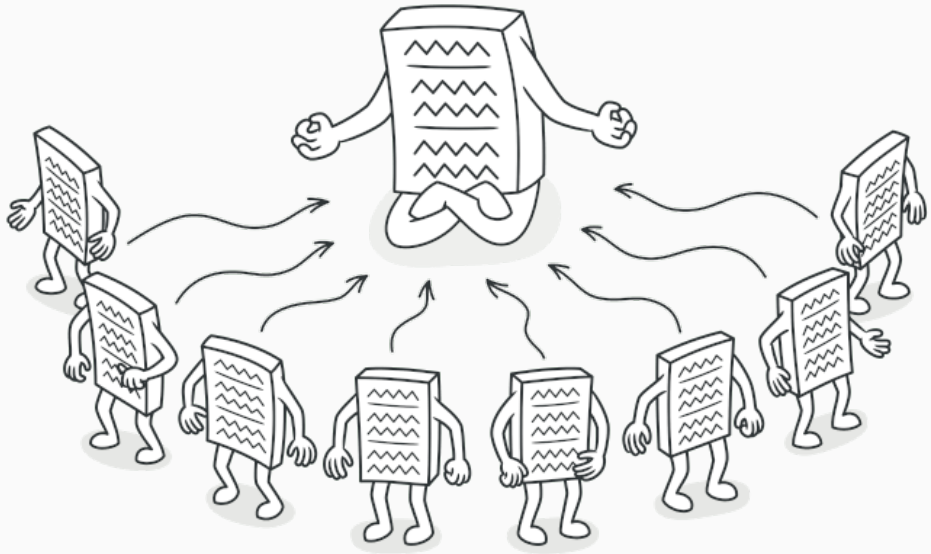
Universidad del Mar

Contenido

Introducción

Singleton

Es un patrón de diseño creacional que nos permite asegurarnos de que una clase tenga una única instancia, a la vez que proporciona un punto de acceso global a dicha instancia.



☹ Problema

El patrón Singleton resuelve dos problemas al mismo tiempo, vulnerando el Principio de responsabilidad única:

- Garantizar que una clase tenga una única instancia. ¿Por qué querría alguien controlar cuántas instancias tiene una clase? El motivo más habitual es controlar el acceso a algún recurso compartido, por ejemplo, una base de datos o un archivo. Funciona así: imagina que has creado un objeto y al cabo de un tiempo decides crear otro nuevo. En lugar de recibir un objeto nuevo, obtendrás el que ya habías creado.

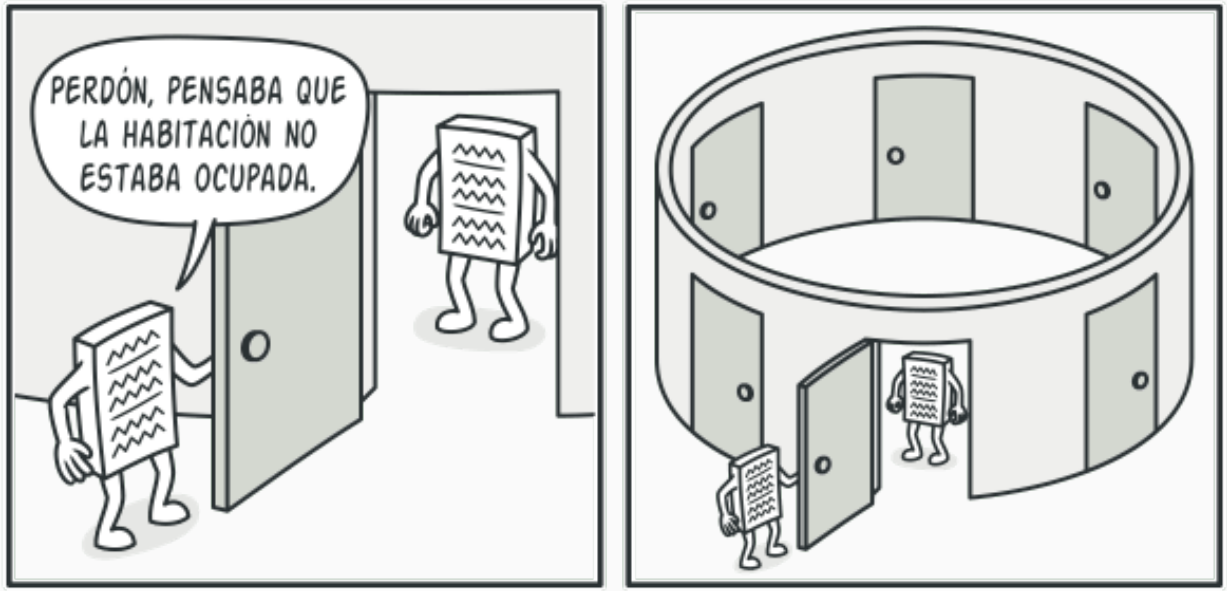
☹ Problema

- Proporcionar un punto de acceso global a dicha instancia. Al igual que una variable global, el patrón Singleton nos permite acceder a un objeto desde cualquier parte del programa. No obstante, también evita que otro código sobreescriba esa instancia.

Hoy en día el patrón Singleton se ha popularizado tanto que la gente suele llamar singleton a cualquier patrón, incluso si solo resuelve uno de los problemas antes mencionados.

☹ Problema

Puede ser que los clientes ni siquiera se den cuenta de que trabajan con el mismo objeto todo el tiempo.



☺ Solución

Todas las implementaciones del patrón Singleton tienen estos dos pasos en común:

- Hacer privado el constructor por defecto para evitar que otros objetos utilicen el operador new con la clase Singleton.
- Crear un método de creación estático que actúe como constructor. Tras bambalinas, este método invoca al constructor privado para crear un objeto y lo guarda en un campo estático. Las siguientes llamadas a este método devuelven el objeto almacenado en caché.