Singlenton

Para comenzar debemos decir que es el “Singlenton” por lo que es un patrón de diseño 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.

Este patrón de diseño es muy usado por sus ventajas que con lleva, pero también como sus ventajas tiene sus desventajas.

Ventajas

Control de Instancias: El patrón Singleton permite controlar el número de instancias (solo una) de una clase en todo el programa, lo que puede ser útil para controlar el acceso a recursos compartidos, como bases de datos o archivos.

Ahorro de Recursos: Dado que solo se crea una instancia, se ahorran recursos. No se necesita crear una nueva instancia cada vez que se necesita una.

Acceso Global: La instancia Singleton puede ser accesible globalmente, lo que puede ser útil para compartir datos entre diferentes partes de la aplicación.

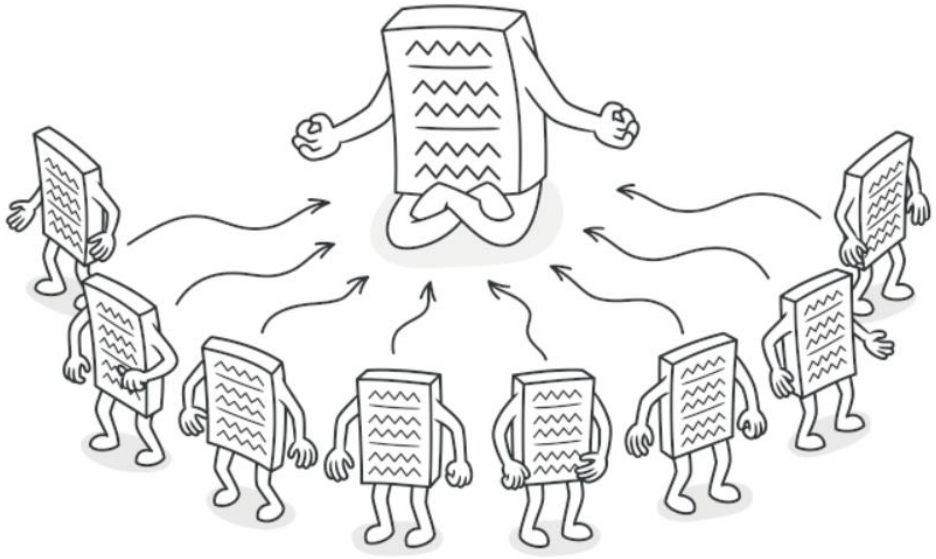
Desventajas

Acoplamiento Global: El patrón Singleton puede llevar a un acoplamiento global ya que permite el acceso a la instancia desde cualquier parte del código. Esto puede hacer que el código sea más difícil de depurar y mantener.

Dificultad para Pruebas Unitarias: Las clases Singleton pueden ser difíciles de probar debido a problemas con el estado global y la persistencia entre pruebas.

Violación del Principio de Responsabilidad Única: La clase Singleton puede terminar realizando tareas que podrían estar mejor encapsuladas en otras clases, violando el principio de responsabilidad única.

Problemas de Concurrencia: Si no se maneja correctamente, múltiples hilos pueden crear múltiples instancias en algunos casos. Esto puede ser evitado con la sincronización adecuada, pero eso puede llevar a una disminución del rendimiento.



En el contexto de este proyecto, el uso del patrón Singleton para la clase Calculadora puede ser adecuado. Aquí están las razones:

Control de Instancias: Como se mencionó en las instrucciones, se requiere que solo exista una instancia de la clase Calculadora. El patrón Singleton puede garantizar esto.

Acceso Global: La instancia de la clase Calculadora puede necesitar ser accesible desde diferentes partes del programa. El patrón Singleton permite un acceso global a su instancia.

Sin embargo, también hay que tener en cuenta las desventajas del patrón Singleton:

Dificultad para Pruebas Unitarias: Las clases Singleton pueden ser difíciles de probar debido a problemas con el estado global y la persistencia entre pruebas.

Acoplamiento Global: El patrón Singleton puede llevar a un acoplamiento global ya que permite el acceso a la instancia desde cualquier parte del código. Esto puede hacer que el código sea más difícil de depurar y mantener.

Referencias:

Singleton. (s/f). Refactoring.guru. Recuperado el 27 de febrero de 2024, de https://refactoring.guru/es/design-patterns/singleton