UTT
UNIVERSIDAD TECNOLÓGICA DE TIJUANA
GOBIERNO DE BAJA CALIFORNIA

TEMA:

Design Patterns

PRESENTADO POR:

Valenzuela Esparza Javier Ivan

GRUPO:

10B

MATERIA:

Desarrollo Movil Integral

PROFESOR:

Ray Brunett Parra Galaviz

FECHA:

07/01/2025

SINGLETON DESIGN PATTERN

The Singleton Design Pattern is a creational design pattern that ensures a class has only one instance throughout the application, providing a global point of access to that instance. This pattern is commonly used when there is a need to control access to shared resources, such as database connections, logging services, or configuration settings. By restricting the instantiation of a class to a single object, it helps in managing resources efficiently and ensuring consistency across the application.

One of the main characteristics of the Singleton pattern is that it guarantees a single instance. This ensures that no matter how many times the class is accessed, only one instance of it will ever exist. The instance is typically created lazily, meaning it is only instantiated when it is first required, which optimizes memory usage. Additionally, this instance is made accessible globally throughout the application, often via a static method, which ensures that all parts of the application use the same instance.

The Singleton pattern also addresses issues related to thread safety. In multi-threaded environments, special mechanisms such as mutexes or double-check locking are used to ensure that the Singleton instance is created only once, even when multiple threads attempt to access it simultaneously. This prevents race conditions and ensures the application remains stable in concurrent environments.

One of the significant advantages of the Singleton pattern is its ability to provide controlled access to resources. For example, in an application that requires a shared configuration or logging service, the Singleton ensures that only one instance is created, reducing memory overhead and avoiding the creation of multiple unnecessary objects. Moreover, the global access to the Singleton instance simplifies access to shared functionality across various parts of the application.

However, the Singleton pattern does have some drawbacks. Global state can lead to hidden dependencies, making it harder to track where and how the Singleton is being used throughout the application. This can complicate debugging and testing. Additionally, overuse of the Singleton pattern may lead to tight coupling within the application, which can hinder maintainability and flexibility. It's important to use the Singleton pattern judiciously to avoid these issues.

The pattern is particularly useful in scenarios such as configuration management, where the application needs to access configuration values globally. It is also commonly used for logging services or database connections, ensuring that a single instance is used across the system to manage these shared resources efficiently.

Bibliographical Soource

Singleton. (s/f). Refactoring.guru. Recuperado el 8 de enero de 2025, de https://refactoring.guru/es/design-patterns/singleton