Hoja de Trabajo 4

Algoritmos y Estructuras de Datos – Sección 10

Julián Divas – 24687

Marcelo Detlefsen – 24554

Alejandro Jerez – 24678

Repositorio de GitHub: https://github.com/div468/HT4---Algoritmos-y-Estructuras-de-Datos.git

Patrón Singleton

Una ventaja inmediatamente notoria es el control de la instancia a trabajar, brindando el patrón de diseño un punto de acceso global a ella. Además, permite ahorrar recursos reduciendo el consumo de memoria gracias a la única instanciación trabajada, lo que evita la sobrecarga de crear muchos objetos de la misma clase. También permite cambiar la implementación de Singleton fácilmente puesto que afecta a un único lugar en el código.

Por otro lado, puede llevar a un alto grado de acoplamiento entre componentes del programa debido a que otros objetos utilizados dependen de la existencia de una única instancia, dificultando la reutilización de código. Además, al convertirse en un punto de estado global puede dificultar el seguimiento de flujo de datos, aumentando la complejidad del sistema.

Consideramos que el uso del patrón de diseño Singleton en nuestro programa es adecuado ya que necesitamos únicamente una instancia de la calculadora a utilizar. Además, el acceso global a ella es importante ya que los stacks a implementar pueden acceder a la misma calculadora en todo momento. Si bien Singleton puede llegar a dificultar las pruebas unitarias por la modificación de los estados del objeto, consideramos que la lógica del resto del programa se beneficia enormemente de asegurar una única instancia de la calculadora con Singleton.

Referencias

DeepSeek. (2023). Código de implementación de una lista doblemente enlazada en Java.

DeepSeek. (2023). Código de implementación de una lista simplemente enlazada en Java.

DeepSeek. (2023). Implementación de un Stack basado en listas enlazadas.

DeepSeek. (2023). Implementación de un Stack basado en ArrayList.

DeepSeek. (2023). Implementación de un Stack basado en Vector.