Universidad del Valle de Guatemala
Algoritmos y estructuras de datos

Esteban Sánchez 25312
Javier Sánchez 25341

Patron Singleton

El patrón de diseño Singleton tiene la función que una clase tenga una única instancia y así poder proporcionar un punto de acceso global a la instancia. Este patrón se utiliza cuando resulta un problema crear múltiples objetos de la misma clase. En el código del proyecto, la clase Calculator implementa este patrón en un atributo estático privado que almacena la única instancia, un constructor privado que impide crear objetos con new, y un método público estático getInstance() que devuelve siempre la misma instancia. De esta forma, durante toda la ejecución del programa solo existe una calculadora responsable de realizar las operaciones.

Entre sus principales ventajas se encuentra el control sobre la creación de instancias, ya que evita la duplicación de objetos. Centraliza la lógica en un solo punto, facilitando la organización del código cuando la clase no necesita mantener múltiples estados independientes. En este proyecto, toda la funcionalidad relacionada con la conversión de expresiones infix a postfix y la evaluación de expresiones postfix se encuentra concentrada en la clase Calculator, lo cual mejora la claridad del diseño y evita que otras clases implementen lógica repetida.

El patrón Singleton también presenta desventajas importantes. Su comportamiento es similar al de una variable global, lo que puede aumentar el acoplamiento entre clases, ya que cualquier parte del sistema puede acceder directamente a la misma instancia. Asimismo, dificulta las pruebas unitarias si la clase mantiene estado interno, ya que todas las pruebas compartirían la misma instancia. En proyectos grandes, esto puede afectar la mantenibilidad y escalabilidad del sistema, especialmente si la clase comienza a almacenar información que cambie durante la ejecución.

En esta hoja de trabajo, usar el patrón Singleton en la clase Calculator es adecuado porque, según el código implementado, la clase no mantiene estado interno persistente entre llamadas. Sus métodos dependen únicamente de los parámetros que reciben (la expresión a convertir o evaluar) y no almacenan resultados previos ni información global. Bajo esta condición, el Singleton no genera problemas de consistencia y permite centralizar la funcionalidad de conversión y evaluación de expresiones de manera clara, controlada y coherente con la arquitectura del programa.