Respuesta a la pregunta:

El patrón Singleton es un diseño que permite crear una sola instancia de una clase en toda la aplicación, y suele usarse para administrar recursos compartidos como conexiones a bases de datos, registros o configuraciones globales. Sus ventajas incluyen garantizar una única instancia para evitar inconsistencias, optimizar el uso de memoria al reducir la creación innecesaria de objetos, facilitar el acceso desde cualquier parte del código y mantener el estado de la instancia mientras el programa se ejecuta. Sin embargo, también tiene desventajas, ya que se comporta como una variable global, lo que puede generar dependencias ocultas y hacer el código más difícil de entender y mantener. Además, dificulta las pruebas unitarias porque afecta el estado global, crea un fuerte acoplamiento que limita la flexibilidad del código y, si no se maneja bien, puede causar problemas en entornos con múltiples hilos de ejecución. En este caso, su uso dependerá del propósito de la clase. Si se necesita para manejar recursos globales, como la configuración de la calculadora o una única pila de operaciones, podría ser útil. Sin embargo, no parece la mejor opción, ya que la calculadora podría necesitar manejar varias pilas de forma independiente. Usar Singleton aquí podría causar problemas de concurrencia y hacer el código innecesariamente rígido. En conclusión, no es recomendable aplicarlo en este programa, ya que cada instancia de la calculadora debe funcionar de manera independiente para evitar errores.