**Ventajas del patrón Singleton:**

* Control sobre instancias: Garantiza que una clase tenga una única instancia, proporcionando un punto de acceso global a ella.
* Conservación de recursos: Al limitar la instancia a una sola, se ahorran recursos, especialmente si la creación de la instancia es costosa.
* Espacio de nombres: Evita contaminar el espacio de nombres global con variables que tienen un alcance amplio.
* Compartir estado: Facilita la compartición de un estado común entre diferentes partes de la aplicación.

**Desventajas del patrón Singleton:**

* Acoplamiento: El uso excesivo puede llevar a un alto grado de acoplamiento entre clases, lo que dificulta las pruebas y el mantenimiento.
* Pruebas unitarias: Puede ser más difícil realizar pruebas unitarias debido a que el estado global puede persistir entre pruebas.
* Concurrencia: En entornos multihilo, puede ser complicado manejar correctamente la instancia única sin incurrir en problemas de concurrencia.
* Flexibilidad: El patrón puede hacer más difícil modificar la aplicación, ya que el uso de la instancia global puede estar muy extendido.

En el programa actual, el uso del patrón Singleton se considera adecuado para la clase Calculadora, ya que la aplicación requiere un único punto de cálculo y un estado compartido. Este enfoque simplifica la gestión del estado y la ejecución de las operaciones matemáticas en un contexto de un solo hilo. Sin embargo, si el programa evoluciona hacia un entorno multihilo o necesita múltiples calculadoras independientes, el patrón Singleton podría no ser el más indicado debido a las restricciones que impone en la concurrencia y el estado del objeto.