



# Patrón de diseño Singleton

¿En que se  
basa este  
patrón de  
diseño?

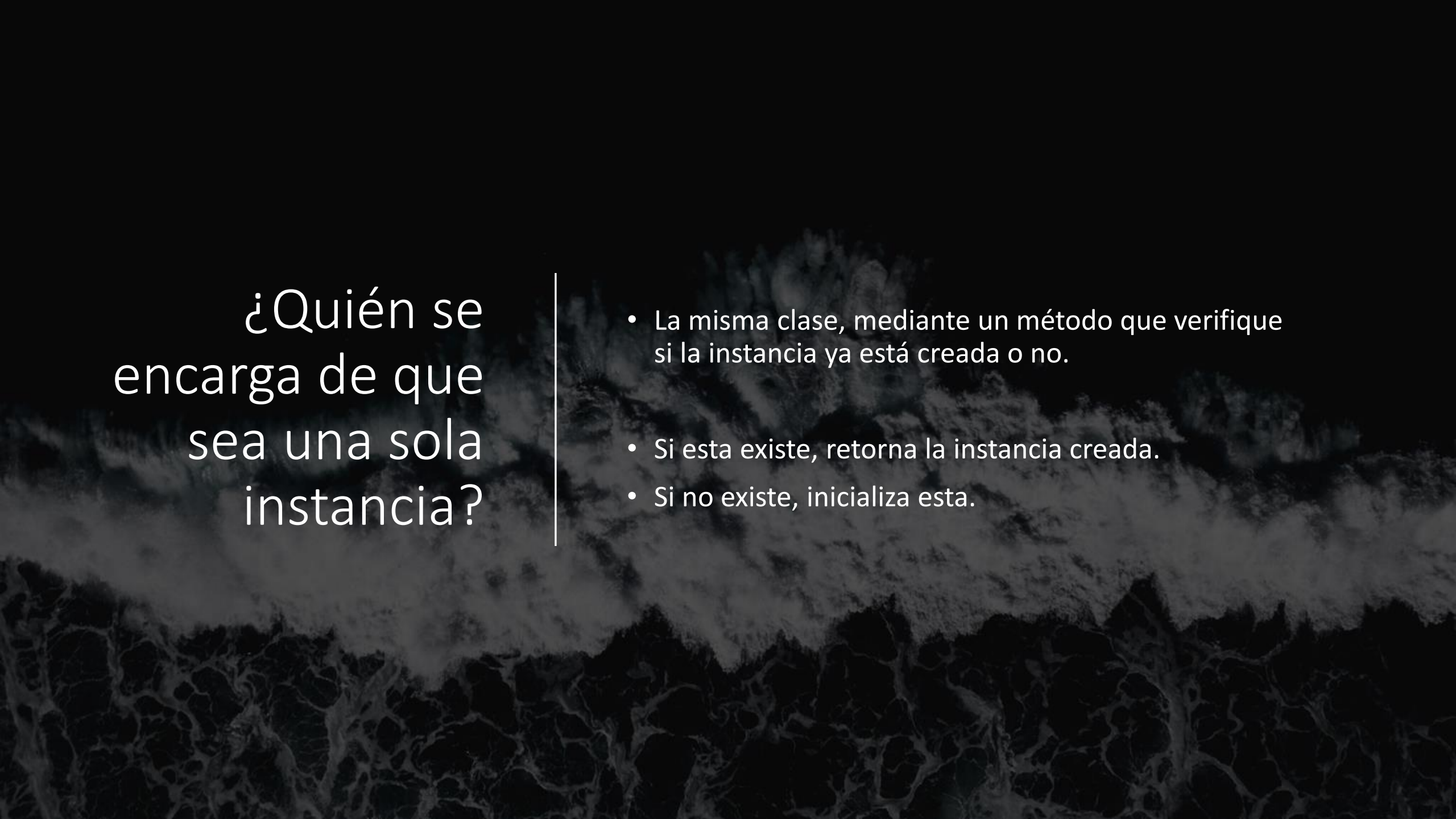
- El patrón singleton se basa en que exista una única instancia de una clase dentro del sistema, y que esta sea accesible desde un punto común para el resto de clases.

# ¿Porqué usarla?

- Cuando necesitamos que la clase posea una única instancia para todo, por ejemplo:
  - - Un administrador de cola de impresión.
  - - Un administrador de ventanas.

¿Qué podemos  
hacer con esta  
instancia?

- Extender esta clase a otras, sin modificar el código de la primera



¿Quién se  
encarga de que  
sea una sola  
instancia?

- La misma clase, mediante un método que verifique si la instancia ya está creada o no.
- Si esta existe, retorna la instancia creada.
- Si no existe, inicializa esta.



Detalles  
importantes  
con este  
patrón

La instancia debe  
ser **estática**.

El constructor  
debe ser **privado**.

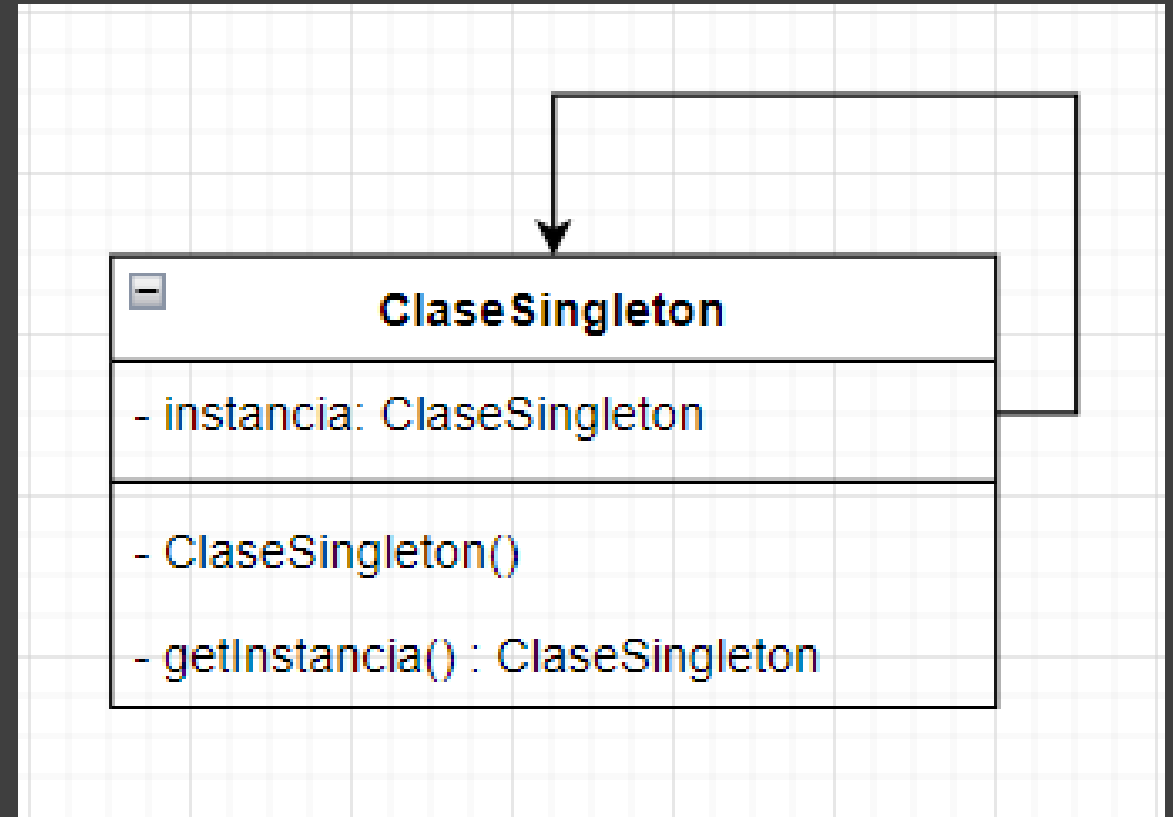
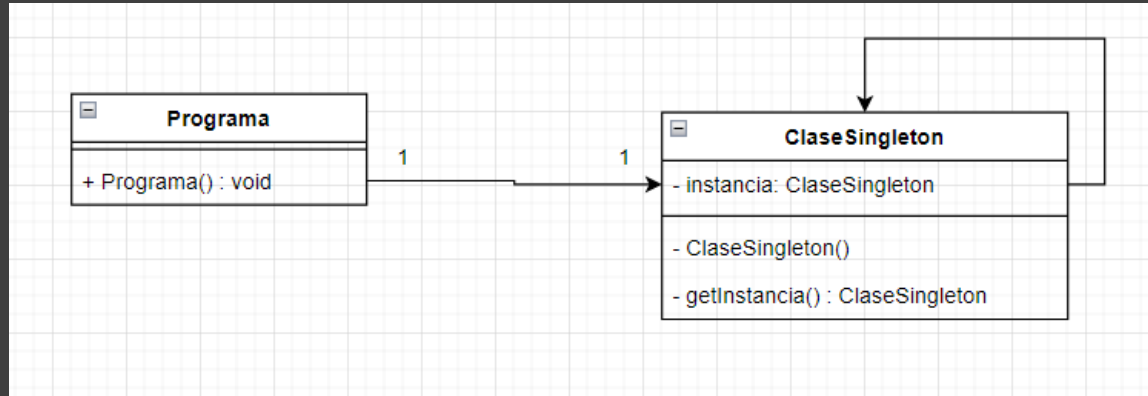


Diagrama UML  
Representación de una clase singleton

Ventajas	Desventajas
Acceso controlado a la instancia	Si no se controla el uso, puede llegar a parecer más una programación procedimental en vez de orientado a objetos.
Evita el exceso de variables globales *	Riesgos al manejar datos sensibles.
Se instancia el momento necesario	Mayor dificultad en el mantenimiento del software.

## Ventajas y desventajas

- \* Esto aplica para lenguajes como C++, los cuales poseen el concepto de variables globales. Java no lo posee.



# Implementación

- Repositorio GitHub:  
<https://github.com/Charlie00CR/DemonstracionSingleton>



# Referencias

- Gamma, E., Johnson, R. E., Helm, R., Johnson, R. E., Vlissides, J., Erich Gamma, R. H. R. J. J. V., & Booch, G. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- *Patron singleton: una clase propia*. (2021, 19 febrero). IONOS Digital Guide.  
<https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/patron-singleton/>