**PATRON SINGLETON:**

Llamado también, instancia única.

Empecemos explicando que es una instancia En Java, una **instancia** se refiere a un **objeto** creado a partir de una **clase**. Es decir, cuando creas una nueva copia de una clase en la memoria, estás creando una instancia de esa clase.



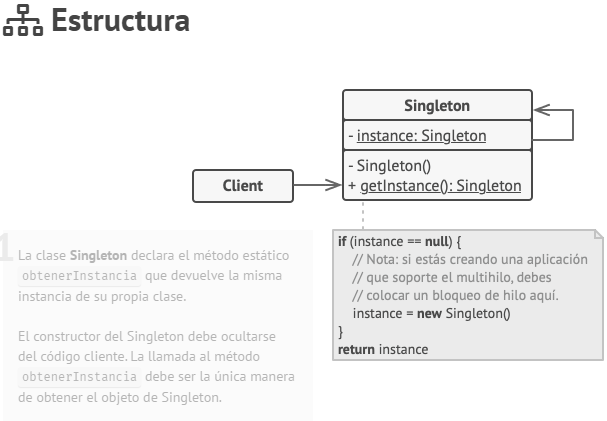
1. Persona es una clase que tiene un atributo nombre y un método saludar ().
2. persona1 es una instancia de la clase Persona, creada con new Persona("Juan").
3. Al crear persona1, se reserva espacio en la memoria para un objeto de tipo Persona, con su propio valor de nombre.

**Diferencia entre Clase e Instancia**

* Clase: Es un molde o plano para crear objetos.
* Instancia: Es un objeto específico basado en la clase.

En términos simples, una clase es la receta y una instancia es el pastel que creas a partir de esa receta.

**Estructura del patrón singleton:**



**Características del singleton:**

*Garantizar una única instancia de una clase:*

* Algunas clases, como las que manejan bases de datos, registros o configuraciones, deben tener una sola instancia para evitar problemas de inconsistencias o acceso simultáneo.
* Por ejemplo, si cada parte de un programa creara su propia conexión a la base de datos, se podrían generar problemas de rendimiento y errores.

*Controlar el acceso a un recurso compartido:*

* En lugar de permitir múltiples objetos que acceden a un recurso compartido (como un archivo o una base de datos), el Singleton asegura que todos los componentes del sistema usen la misma instancia.

**¿Como funciona?**

El Singleton se asegura de que, si intentas crear una nueva instancia de una clase, en lugar de devolver un objeto nuevo, devuelve el que ya fue creado.

El problema es que, en Java, el constructor de una clase siempre devuelve un objeto nuevo. Para evitarlo, el Singleton oculta su constructor y proporciona un método estático que devuelve siempre la misma instancia.

**¿Porque viola el SRP?**

El SRP dice que una clase debe tener una sola responsabilidad.  
El Singleton mezcla dos responsabilidades:

1. Gestiona la cantidad de instancias (control de creación).
2. Define la lógica del negocio de la clase.

Al hacer ambas cosas a la vez, se rompe el SRP. Es por esto que algunos desarrolladores consideran que el Singleton no siempre es una buena práctica.



**Ejemplo:**





**EXPLICACION CODIGO:**

private static conexion parametro;

* Se declara una variable estática parametro para almacenar la única instancia de la clase.

private conexion() {}

* Se declara un constructor privado para evitar que otras clases creen nuevas instancias con new conexion().

public static conexion getparametro()

* Este método controla la creación de instancias:
  + Si parametro == null, se crea un nuevo objeto conexion.
  + Si ya existe una instancia, se devuelve la misma sin crear otra.

public void conectar()

* Método que imprime "Conectado a la base de datos.".

public void desconectar()

* Método que imprime "Desconectado a la base de datos.".

**COMO FUNCIONAN JUNTOS :**

1. App1.java llama a conexion.getparametro(), obteniendo la única instancia de conexion.
2. Luego ejecuta c.conectar();, lo que imprime "Conectado a la base de datos.".
3. Se verifica si c es realmente una instancia de conexion con instanceof.
4. Finalmente, se imprime true, confirmando que el Singleton funciona correctamente.