POLITÉCNICA

"Ingeniamos el futuro"

CAMPUS
DE EXCELENCIA
INTERNACIONAL

MiW

Patrones de Diseño

23. *Singleton*

Luis Fernández Muñoz

https://www.linkedin.com/in/luisfernandezmunyoz

setillofm@gmail.com

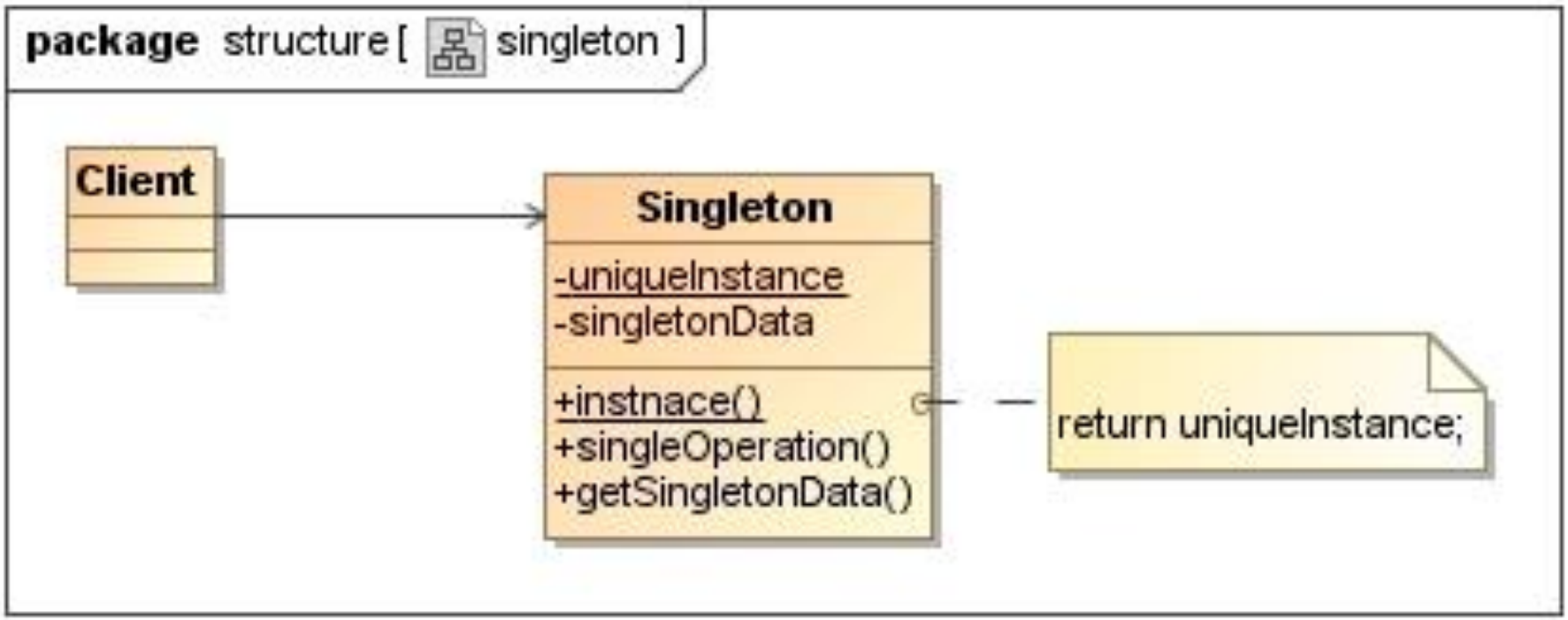
INDICE

1. Problema

- *Si en una organización que disfruta de un personal de seguridad no ofrece un número de teléfono o cualquier mecanismo global, éste debería presentarse, si es nuevo en la organización, y avisar constantemente de su ruta para que cualquier pudiera dar a aviso por alguna incontingencia.*
- *Es el número de teléfono global que permite la flexibilidad de localizar e incluso cambiar de personal sin afectar a nadie.*

- ***Asegurar que una clase tenga una sola instancia y proveer un punto de acceso global para ésta***
  - Debe ser exactamente una única instancia de la clase y debe ser accessible a los clientes desde un punto de acceso bien conocido.
  - Cuando la única instancia debería ser extensible por subclasificación y los clientes deberían ser capaces de usar la instancia extendida sin modificar su código

2. Solución


package structure [singleton]
Client
Singleton
-uniqueInstance
-singletonData
+instnace()
+singleOperation()
+getSingletonData()
return uniqueInstance;

2. Solución

- Define una operación *Instance* que permite a los clientes acceder a su única instancia. *Instance* es una operación de clase
- Responsable de crear su propia instancia única.
- Se puede subclasificar el *Singleton* para devolver una instancia polimórfica:
  - con variables de entorno y sentencias condicionales para determinar la subclase es más flexible, pero con fuertes conexiones al conjunto de posibles clases *Singleton;*
  - poner la implementación de *Instance* fuera de la clase padre (p.e. Factoría) o ponerla en la subclase. Esto permite al programador de C++ decidir la clase de *Singleton* en tiempo de compilación

3. Consecuencias

- Permite tener un control estricto sobre cómo y cuándo los clientes acceden a ésta
- Es una mejora sobre las variables globales ya que se evita contaminar el espacio de nombres con las variables globales que almacenen las instancias únicas.
- Permite tener subclases y es fácil configurar una aplicación con una instancia de esta clase extendida que se necesite en tiempo de ejecución.
- Facilita cambiar de opinión y permitir más de una instancia de la clase. Además, se puede utilizar el mismo enfoque para controlar el número de instancias que utiliza la aplicación.
- Difícil de cambiar el diseño para permitir más de una instancia de una clase. Y las subclases no pueden redefinirlas polimórficamente