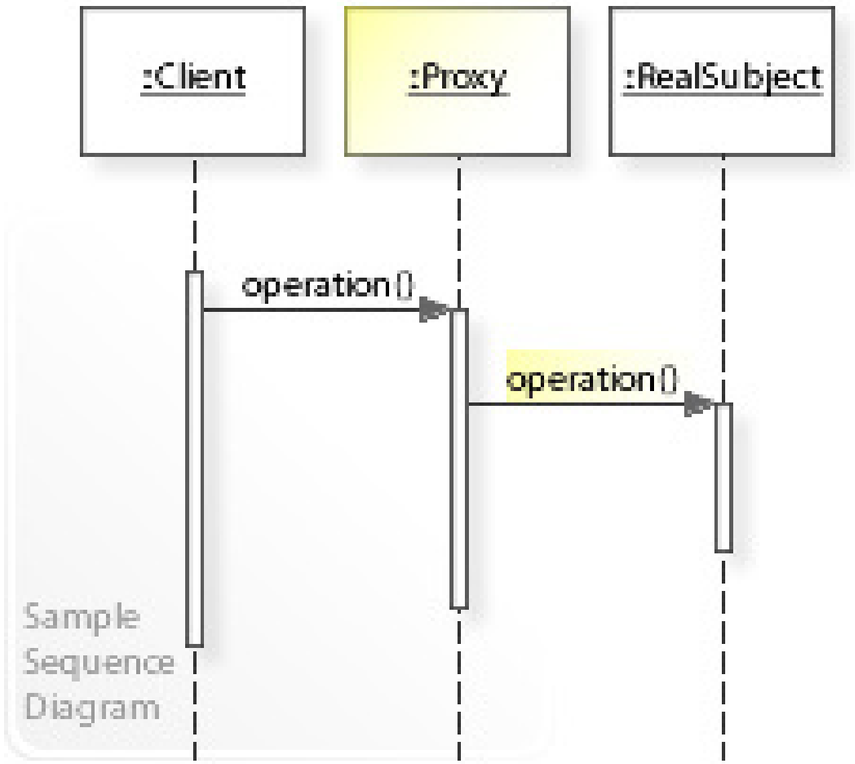
PATRÓN PROXY

Por Oscar Roni

TEC | Tecnológico de Costa Rica

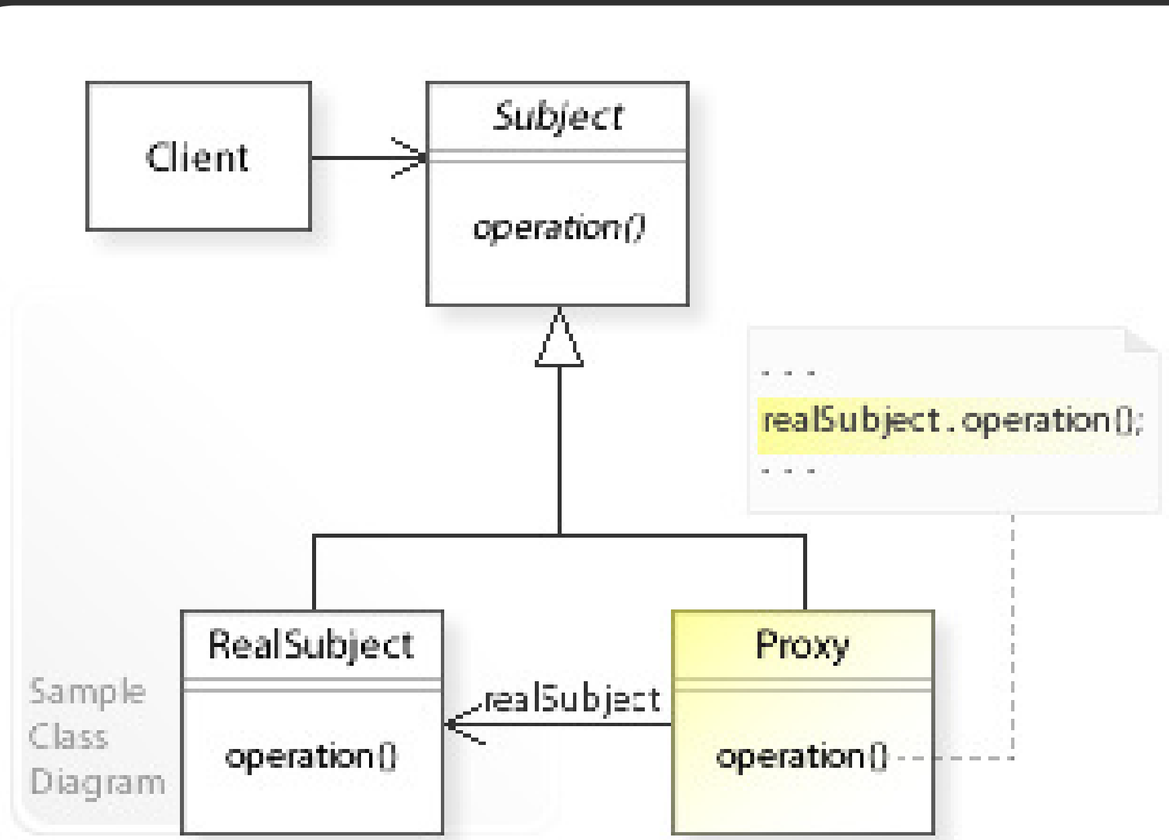
INDICE

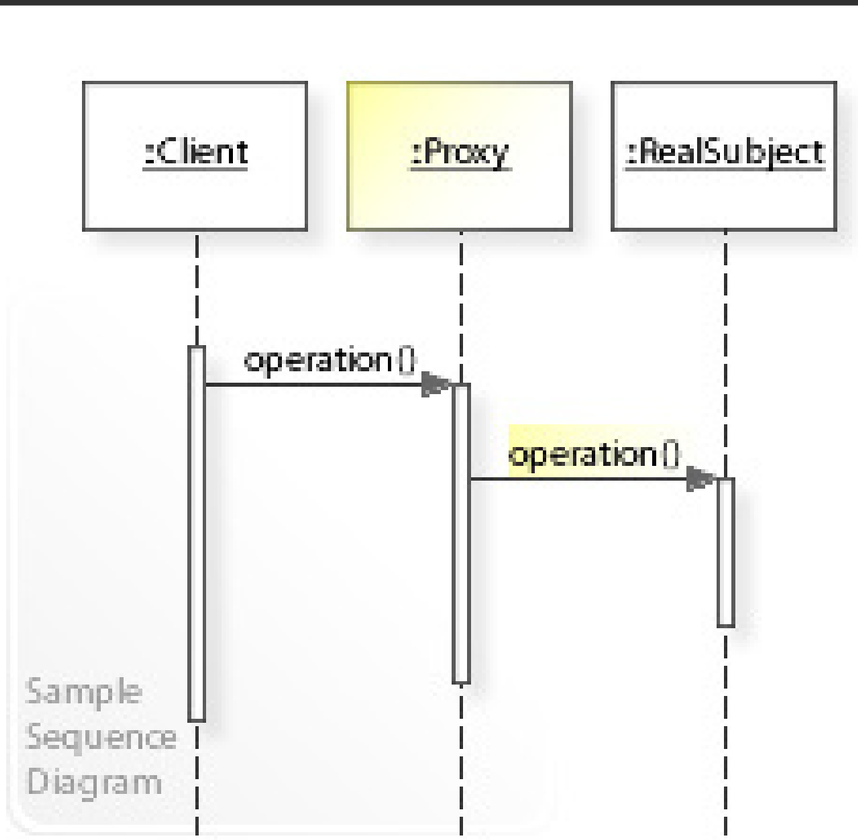


1 QUÉ ES?

El **patrón Proxy** es un patrón estructural que permite a un objeto actuar como sustituto o representante de otro objeto.

Su objetivo es **controlar el acceso al objeto real**, de manera que solo se interactúe con él cuando realmente sea necesario o bajo condiciones específicas.


Client
Subject
operation()
realSubject.operation();
RealSubject
operation()
realSubject
Proxy
operation()
Sample
Class
Diagram


:Client
:Proxy
:RealSubject
operation()
operation()
Sample
Sequence
Diagram

2

CUANDO USARLO?

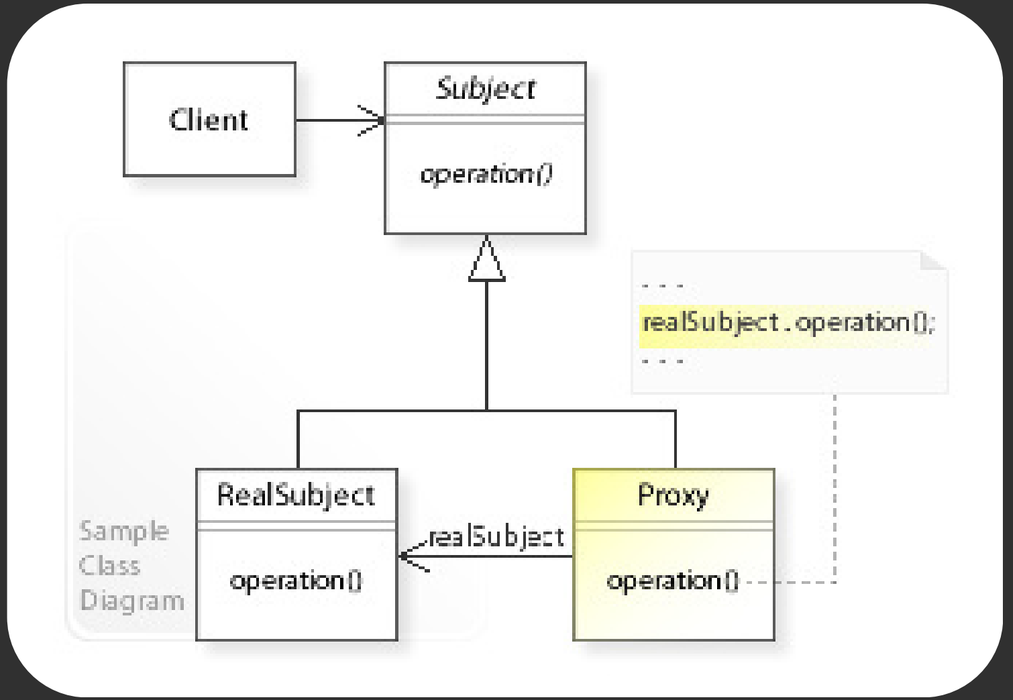
- Controlar quién puede acceder a un objeto específico.
- Retrasar la creación de un objeto pesado hasta el momento en que realmente se necesite.
- Representar un recurso remoto para que el cliente local lo trate como un objeto normal
- Para almacenar en memoria temporal resultados o estados de un objeto

3 COMPONENTES

Subject (Sujeto):

- Es una **interfaz** o **clase abstracta** que declara los métodos comunes que tanto el Proxy como el *RealSubject* implementarán.

- Define las operaciones que el cliente puede invocar, asegurando que el cliente no necesite conocer si está interactuando con el Proxy o con el RealSubject.

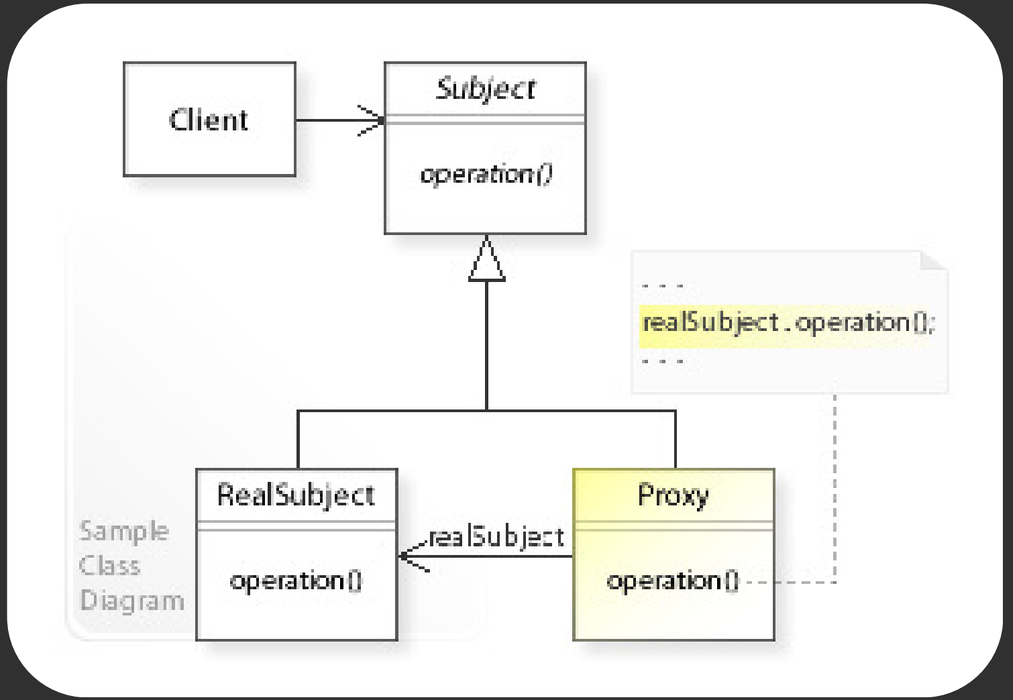


3 COMPONENTES

RealSubject (Sujeto Real):

- Es el **objeto real** que el Proxy representa. Este contiene la implementación completa del comportamiento y de la lógica de negocio que el cliente desea usar.

- El *RealSubject* implementa las operaciones declaradas por el *Subject*, y el Proxy controla el acceso a este objeto real.

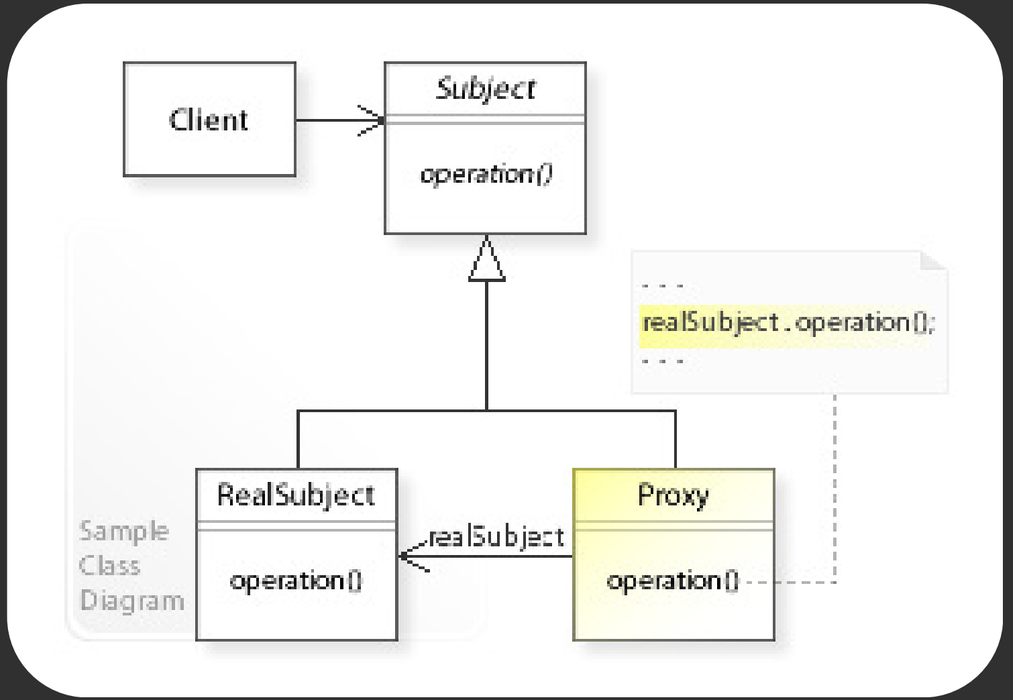


3 COMPONENTES

Proxy:

- Es el **intermediario** que controla el acceso al *RealSubject*. Implementa la misma interfaz que el *RealSubject*, por lo que el cliente puede tratarlo como si fuera el objeto real.

- El *Proxy* incluye lógica adicional para tareas del *RealSubject*.



4 IMPLEMENTACIÓN

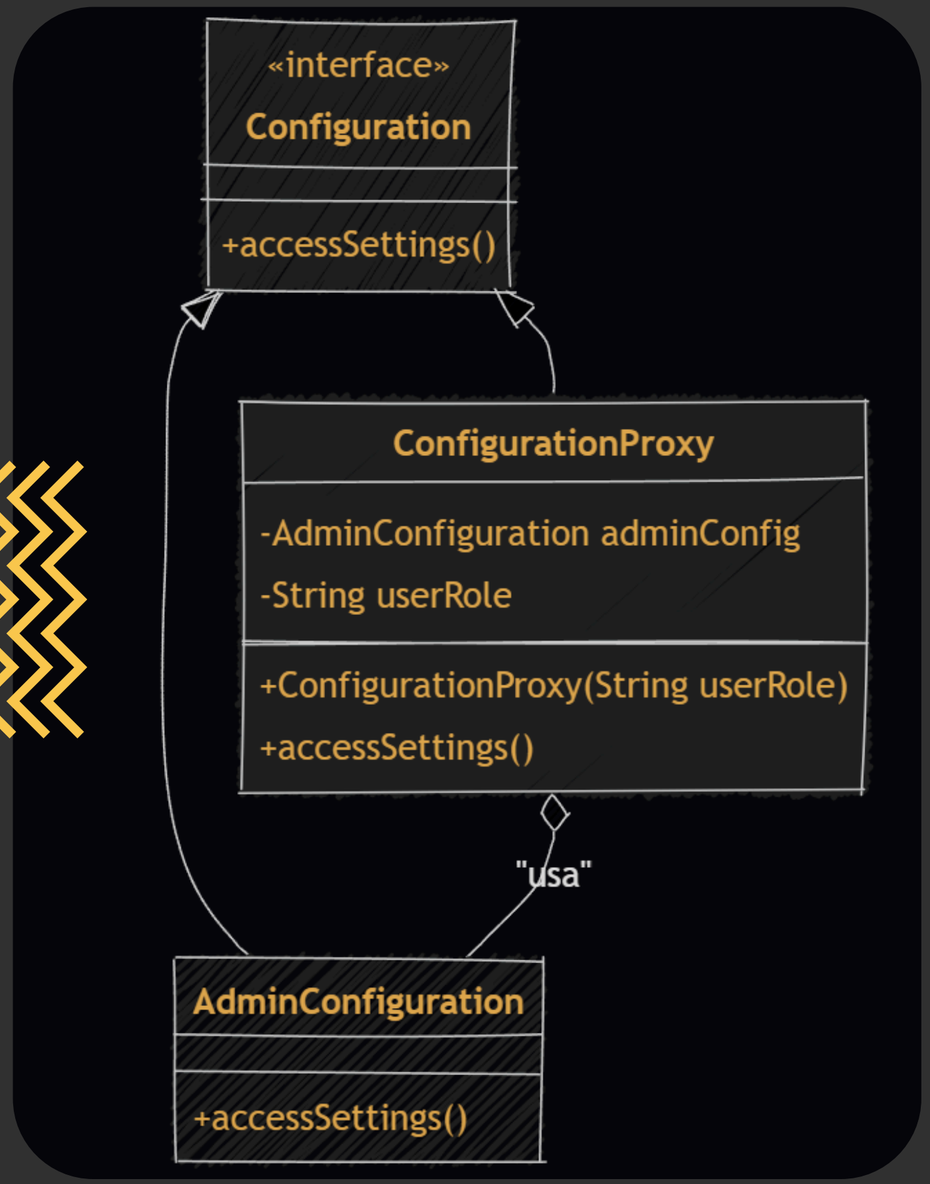
Ejemplo:

Tenemos un sistema donde solo usuarios con permisos de administrador pueden modificar configuraciones sensibles.

Usaremos un Proxy para controlar el acceso a estas configuraciones.

4 IMPLEMENTACIÓN

- **Subject:** Definimos una interfaz *Configuration* con un método *accessSettings()*.
- **RealSubject:** Implementamos una clase *AdminConfiguration* que permite acceder a las configuraciones.
- **Proxy:** Implementamos una clase *ConfigurationProxy* que verifica los permisos del usuario antes de acceder a las configuraciones.



4 IMPLEMENTACIÓN

```java
package com.logic;

public interface Configuration {
    void accessSettings();
}
```

4 IMPLEMENTACIÓN

```java
package com.logic;

public class AdminConfiguration implements Configuration {
    @Override
    public void accessSettings() {
        System.out.println("Acceso concedido!!");
    }
}
```

4 IMPLEMENTACIÓN

```java
package com.logic;

public class ConfigurationProxy implements Configuration {
    private AdminConfiguration adminConfig;
    private String userRole;

    public ConfigurationProxy(String userRole) {
        this.userRole = userRole;
        this.adminConfig = new AdminConfiguration();
    }

    @Override
    public void accessSettings() {
        if ("admin".equals(userRole)) {
            adminConfig.accessSettings();
        } else {
            System.out.println("Acceso denegado!!");
        }
    }
}
```

4 IMPLEMENTACIÓN

```java
package com.main;

import com.logic.Configuration;
import com.logic.ConfigurationProxy;

public class Main {
    Run | Debug
    public static void main(String[] args) {
        // Usuario sin permisos de administrador
        Configuration userConfig = new ConfigurationProxy(userRole:"user");
        userConfig.accessSettings();

        // Usuario con permisos de administrador
        Configuration adminConfig = new ConfigurationProxy(userRole:"admin");
        adminConfig.accessSettings();
    }
}
```

FIN!

Referencias

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995). Design patterns: elements of reusable object-oriented software. Pearson Education Inc.

Kumar, S. (2024). Proxy Design Pattern. https://www.geeksforgeeks.org/proxy-design-pattern/