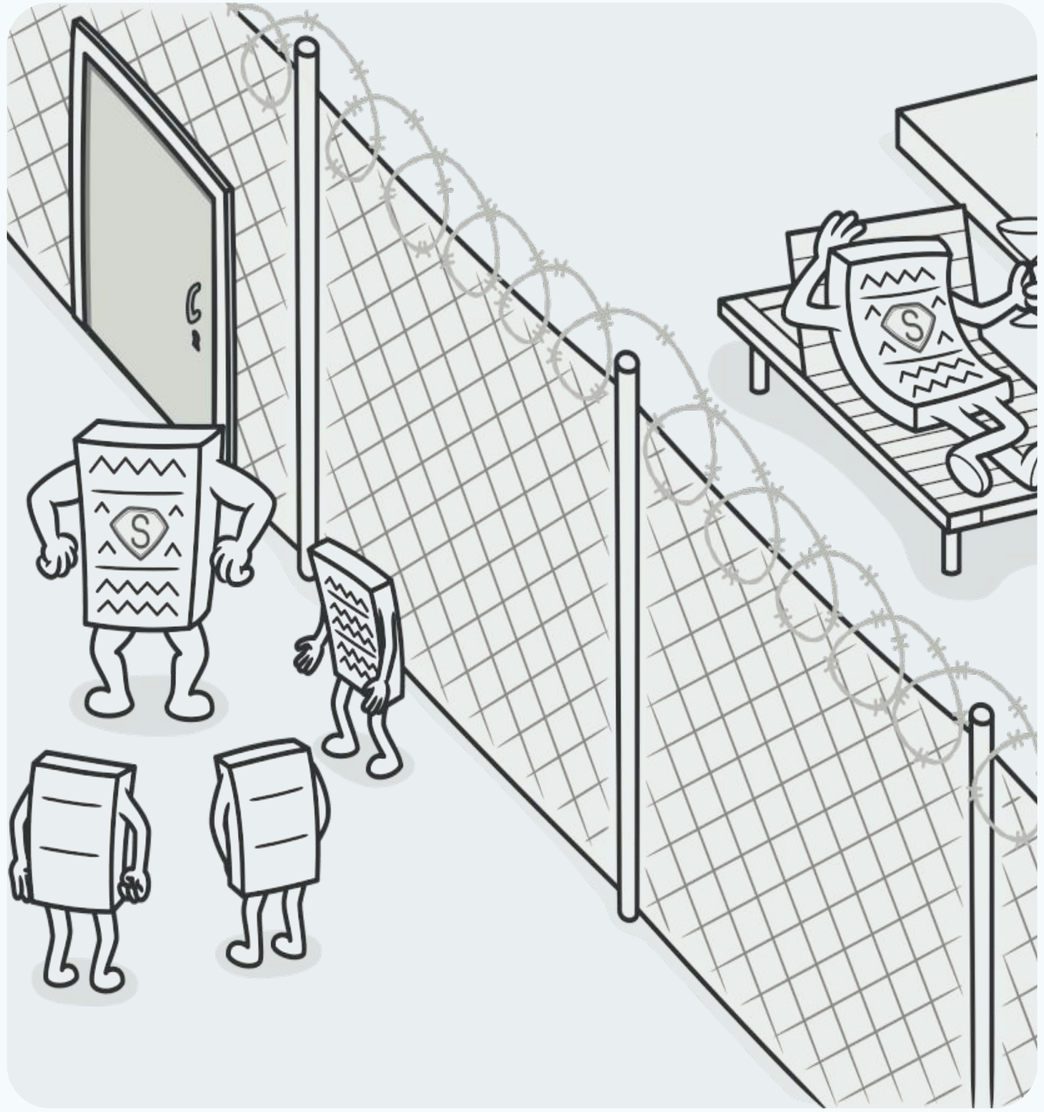
universidade de aveiro
departamento de electrónica,
telecomunicações e informática

Proxy
Design Pattern

Rúben Garrido, 107297

PADRÕES E DESENHO DE SOFTWARE



When to use it?

It provides a placeholder
for another object
to control access to it.

It addresses various concerns in software development, providing a way to control access to objects, add functionality, or optimize performance.

Access Control

Enforces access control policies by acting as a gatekeeper to the real object, restricting access based on certain conditions, providing security or permission checks.

Caching

Store results or resources, optimizing repeated operations on a real object by caching previous results, avoiding redundant computations or data fetching.

Lazy Loading

Delays the creation of the real object until it is actually needed, improving performance by avoiding unnecessary resource allocation.

Logging

Provides a convenient point to add logging information, tracking usage, or measuring performance without modifying the real object.

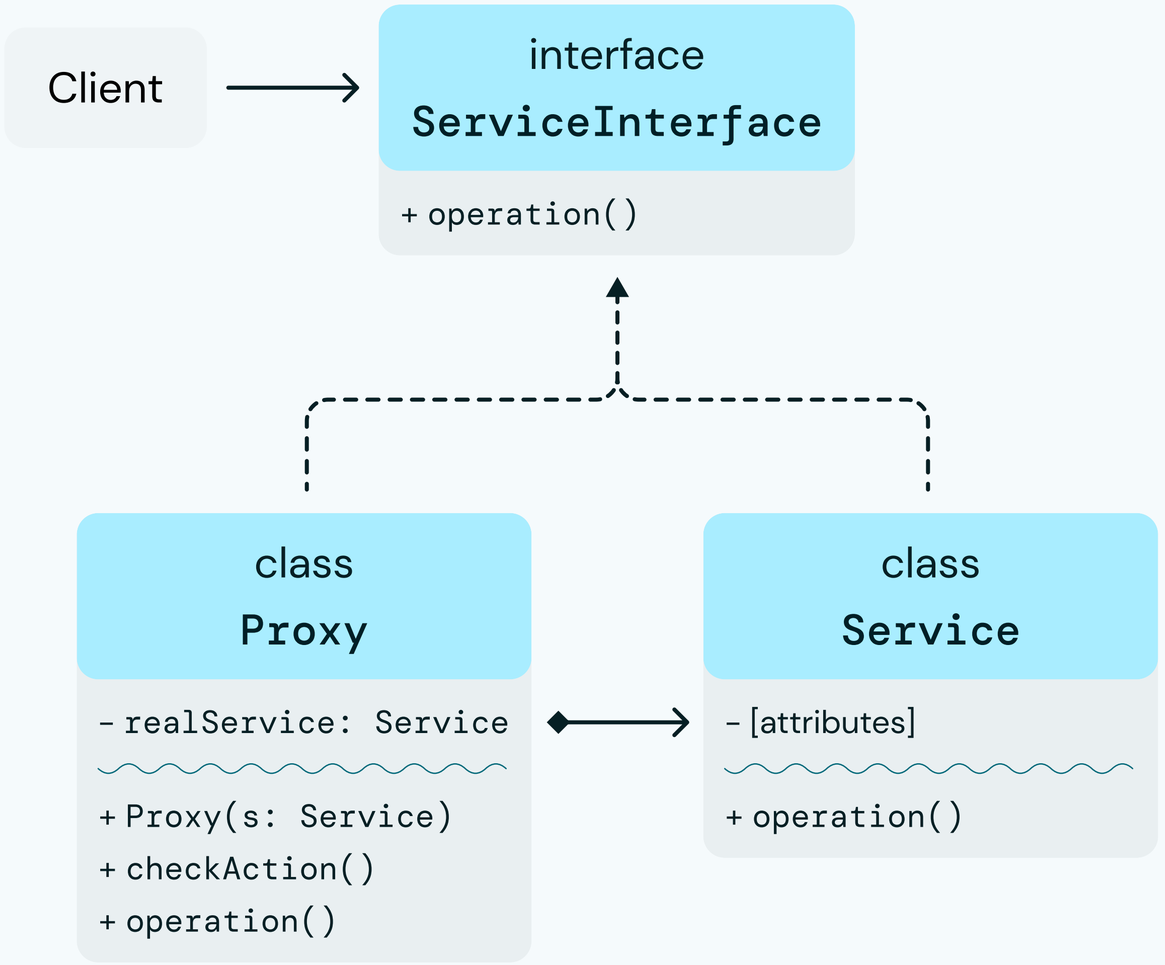
How to implement?

This pattern is pretty simple, it only requires three entities.

1. **Define an interface**
   that both the real object and the proxy class implement
2. **Create the proxy class**
   that wraps the real service
3. **Use the client to call the interface**
   and the proper implementation (aka the proxy)

Base Class Structure

Learn how to implement this pattern with this beautiful UML diagram.



Code Example

Check a Java code example for a simpler learning. Slay.

```
interface ServiceInterface {
    void operation();
}

class Service implements ServiceInterface {
    @Override
    public void operation() {
        // Do something
    }
}

class Proxy implements ServiceInterface {
    private final ServiceInterface realService;

    public Proxy(ServiceInterface realService) {
        this.realService = realService;
    }

    public boolean checkAccess() {
        // Implement access logic
        return true;
    }

    @Override
    public void operation() {
        if (checkAccess()) {
            realService.operation();
        }
    }
}
```

```
public class Client {
    public static void main(String[] args) {
        ServiceInterface service = new Service();
        ServiceInterface proxy = new Proxy(service);
        proxy.operation();
    }
}
```