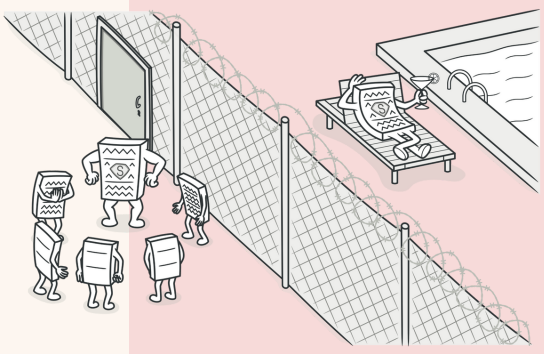
PROXY

Structural Design Pattern

PURPOSE & DEFINITION



A proxy pattern is a structural design pattern that allows you to provide a Surrogate or placeholder to an existing object, and controls access to the original object, enabling you to perform actions before or after the request gets through it, and it is used to interface its functionality to the outer world.

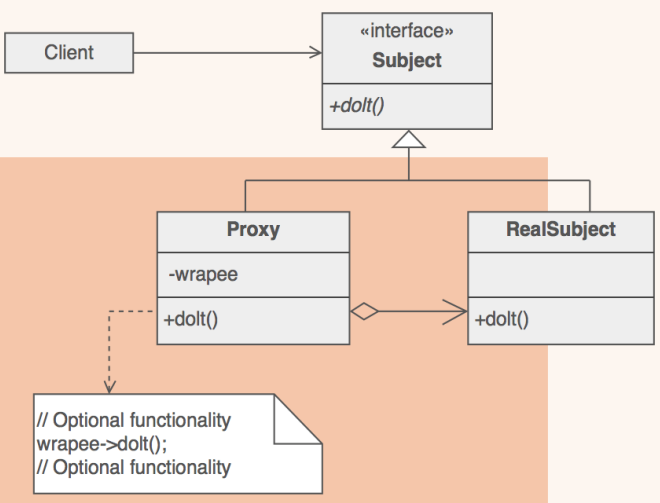
PROBLEM

- Implementation of lazy initialization causes a lot of code duplication, and we do not want to create a full object until it is actually needed.
- The desire to use a class that can perform as an interface to something else.
- The desire to add an additional security layer around the original object as well.

INTENT

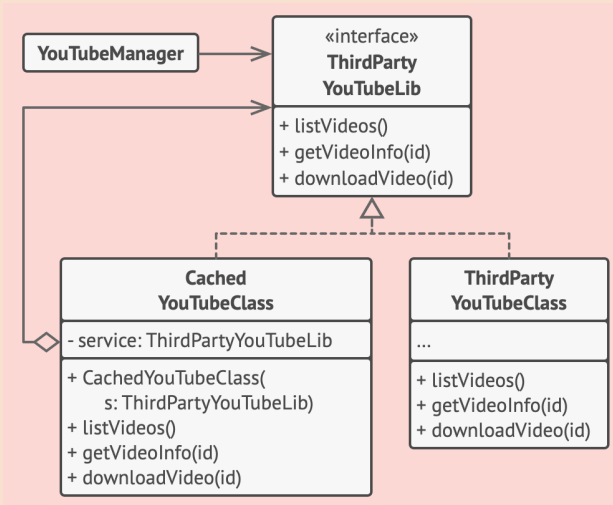
- To provide a substitute for another object to control access to it.
- To use an extra level of indirection to support distributed, controlled, or intelligent access.
- To add a wrapper and delegation to protect the real component from undue complexity.

SOLUTION



Designing a proxy object that instantiates the real object the first time the user makes a request of the proxy, remembers the identity of this real object, and forwards the instigating request to this real object.

Then all subsequent requests are simply forwarded directly to the encapsulated real object.



EXAMPLE

Using the Proxy pattern, the example illustrates how this pattern could help to implement lazy initialization and caching for a third-party YouTube integration library. The library provides the video downloading class.

It's inefficient in this case. If the client application requests the same video multiple times, the library downloads the same video over and over instead of caching and reusing the first download.

The proxy class implements the same interface as the original downloader and delegates all the work. However, it keeps track of the downloaded files and whenever the app repeatedly requests the same video, it returns the cached results based on those files.

REFERENCES

- https://www.javatpoint.com/proxy-pattern
- https://refactoring.guru/design-patterns/proxy
- https://www.geeksforgeeks.org/proxy-design-pattern/

Prepared for: Dr. Fatmah Bamashmoos

Prepared by: Elaf Yousef Aloufi - 1911265 - CAR - CPIT252