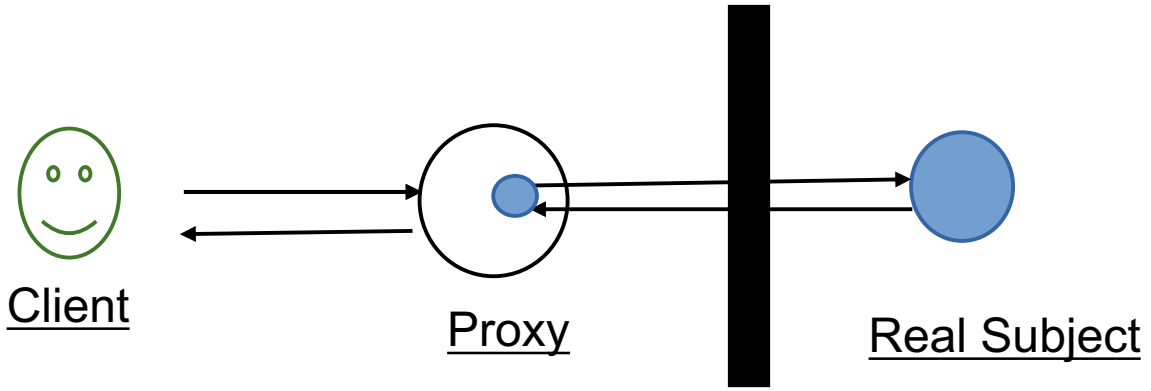
Proxy Pattern

Problem

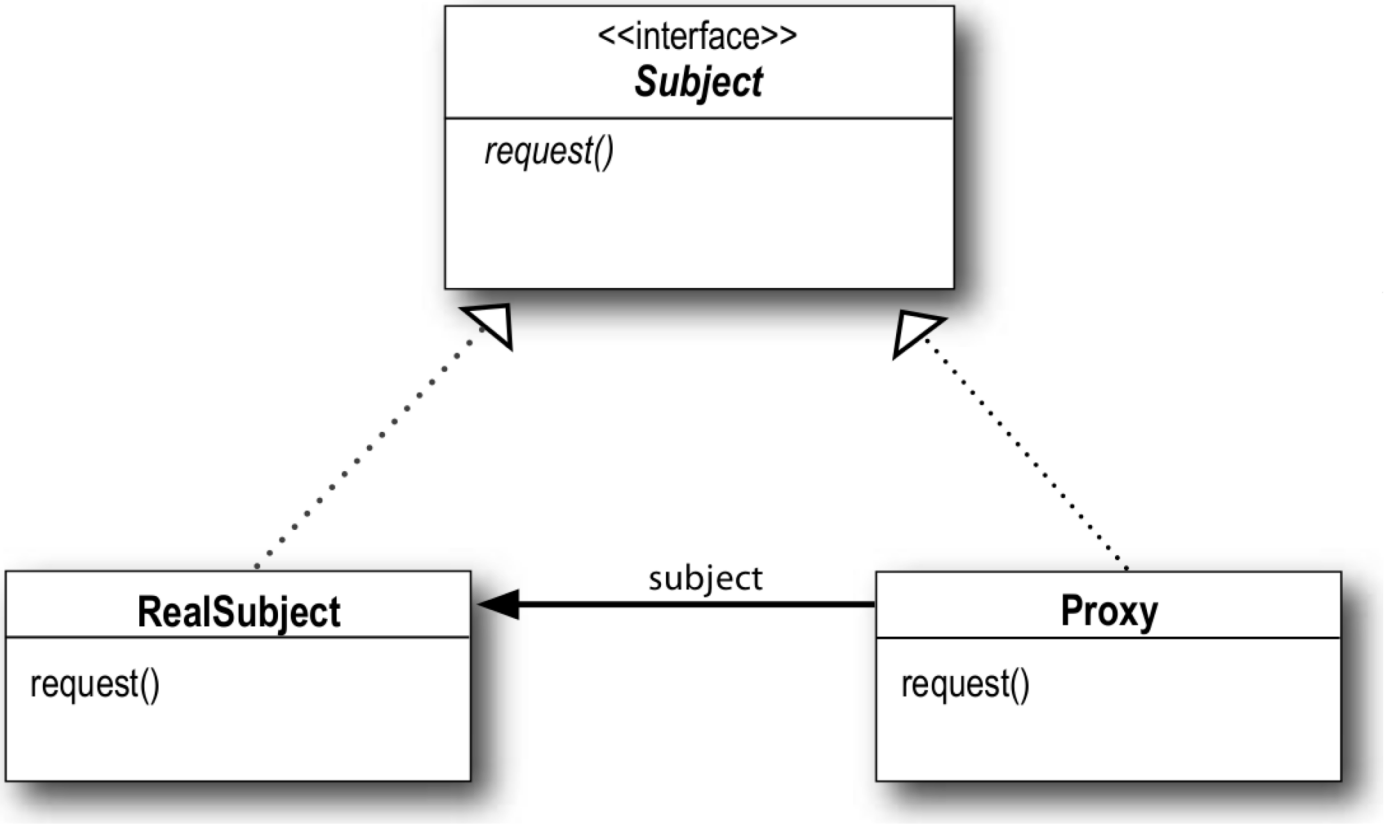
- Sometimes in your application you want to **control access to an object because of different reasons,** e.g., objects are remote, or object creation needs some computation.

- You have the situation that clients need to access the same object multiple times and control access can help, for example save costs (computation costs, or memory costs).

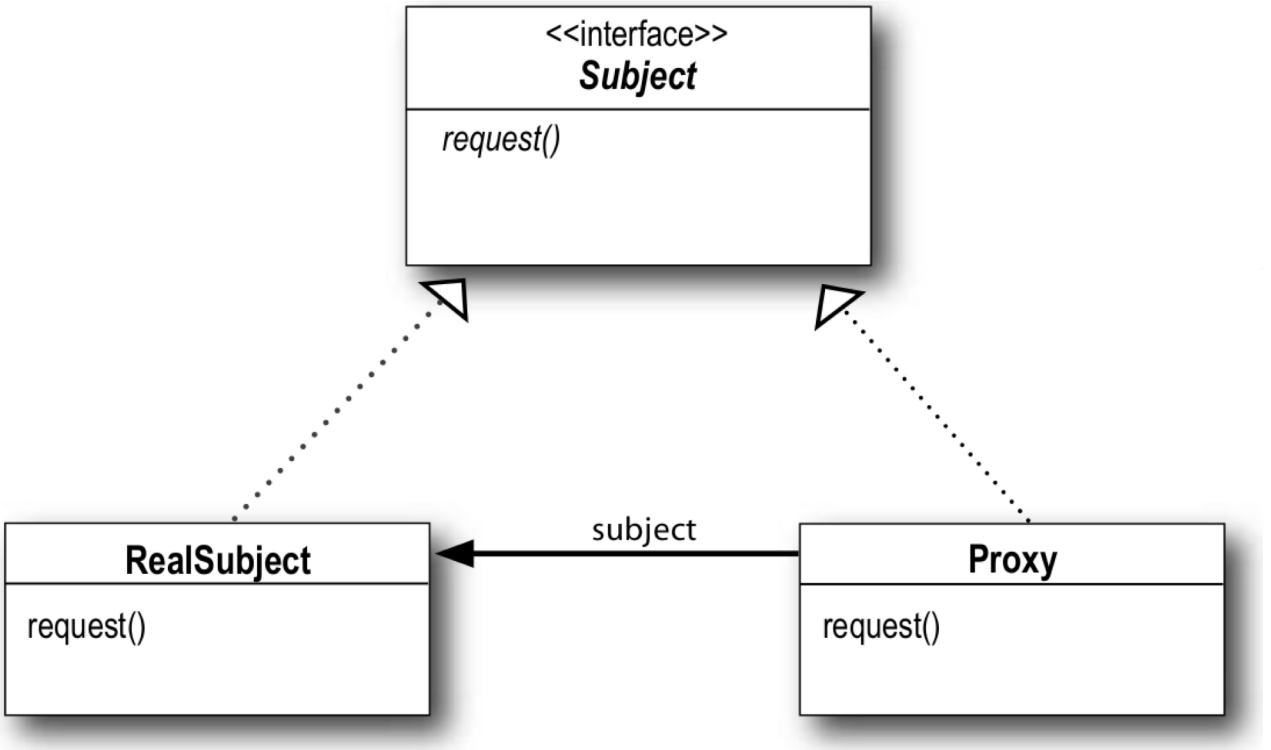


Proxy Pattern

- Definition: “The Proxy Pattern provides a surrogate or placeholder for another object to control access to it.”
- Use the Proxy Pattern to create a representative object that controls access to another object, which may be remote, expensive to create or in need of securing.



Participants



- **Subject**
  - defines the common interface for RealSubject and Proxy so that a Proxy can be used anywhere a RealSubject is expected.
- **Proxy**
  - maintains a reference that lets the proxy access the real subject.
  - controls access to the real subject and may be responsible for creating and deleting it.
- **RealSubject** (ConcreateSubject)
  - defines the real object that the proxy represents.

Types of Proxy Pattern

- A **Remote Proxy** manages interaction between a client and a remote object.
- A **Virtual Proxy** controls access to an object that is expensive to instantiate.
- A **Protection Proxy** controls access to the methods of an object based on the caller.
- **Caching Proxy** provides temporary storage for results of operations that are expensive. It can also allow multiple clients to share the results to reduce computation or network latency.
- **Synchronization Proxy** provides safe access to a subject from multiple threads.
- Many other variants including Complexity Hiding Proxy (Which pattern is this?) , Firewall proxies, copy-on-write proxies, and so on.

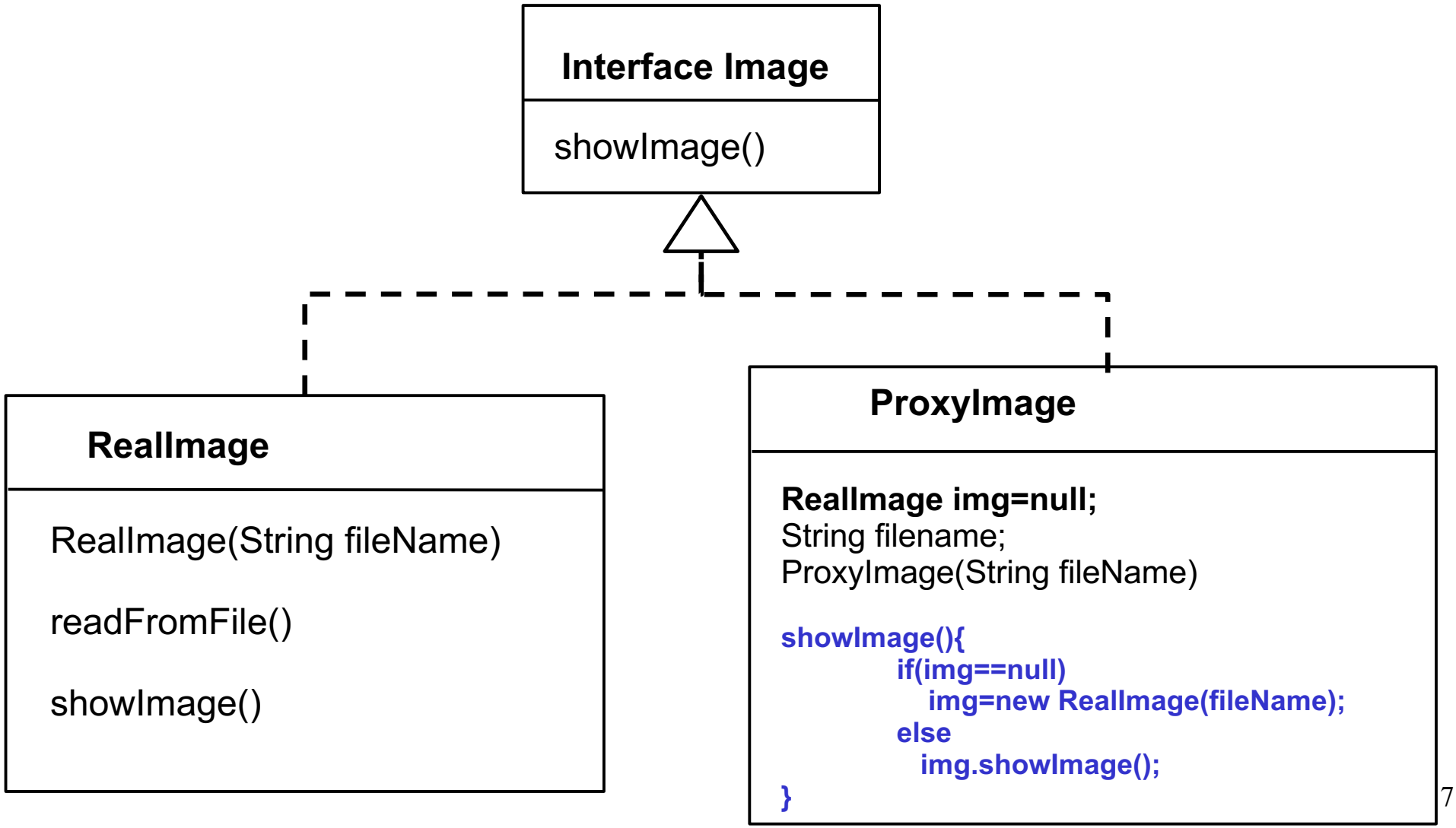
When use the Proxy Pattern

Common situations in which proxy pattern is applicable are the different types of proxy pattern …

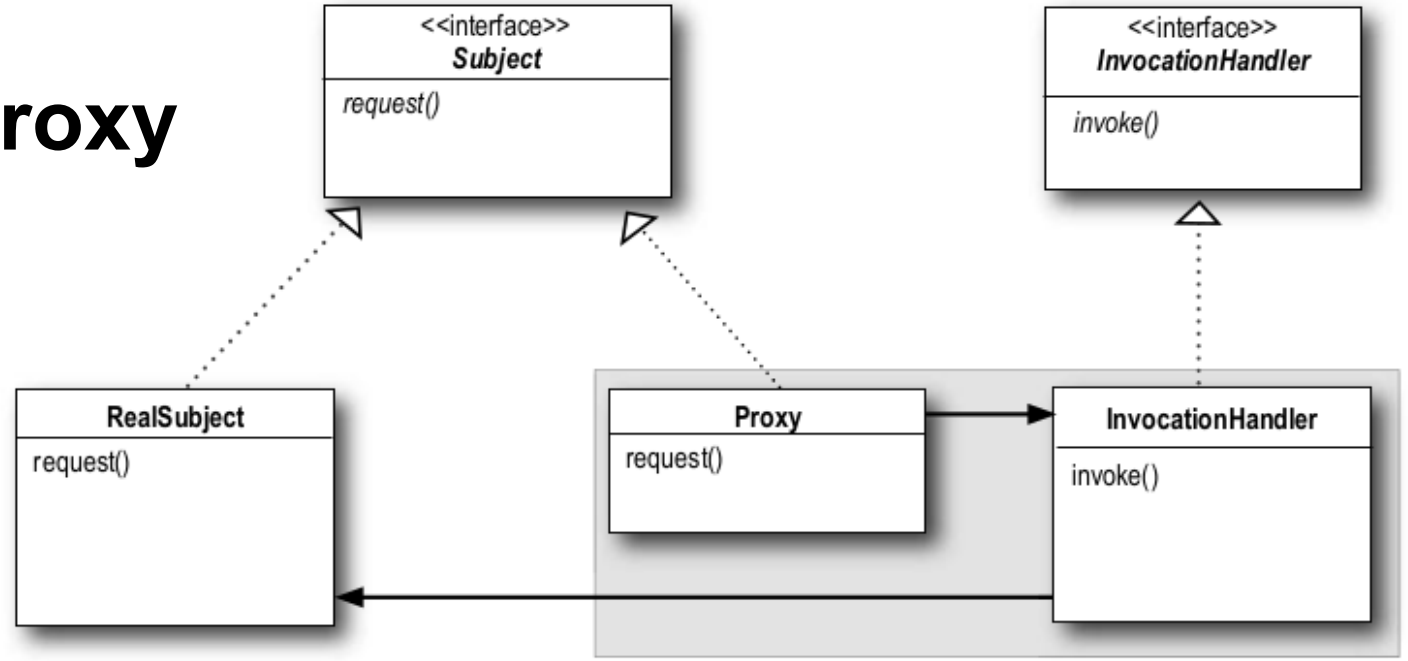
- A **remote proxy** provides a local representative for an object in a different address space.
- A **virtual proxy** creates expensive objects on demand.
- A **protection proxy** controls access to the original object. Protection proxies are useful when objects should have different access rights.
- For example in C++, **a smart reference pointer** is a replacement for a raw C++ pointer that performs additional actions when an object is accessed, like reference counting

Example : Implement this on your own

- Reading an image file is an expensive operation.
- Implement a Image Proxy for the showImage() operation



Java API's Proxy



- You supply the InvocationHandler, invoked on the passed all method calls that are controls access to **Proxy**. The InvocationHandler the methods of the RealSubject.
- The Proxy is generated by Java and implements the entire Subject interface.

Inteface java.lang.reflect.InvocationHandler

Class java.lang.reflect.Proxy

Class java.lang.reflect.Method

Consequences

+ A remote proxy can hide the fact that an object resides in a different address space or it is an actual remote object that we are accessing.

+ A virtual proxy can perform optimizations such as creating an object on demand (**lazy evaluation**)

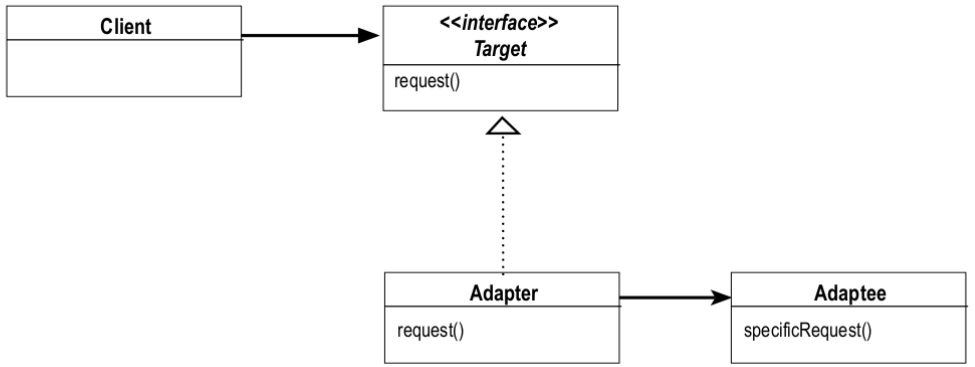
Both protection proxies and smart references allow additional housekeeping tasks when an object is accessed.

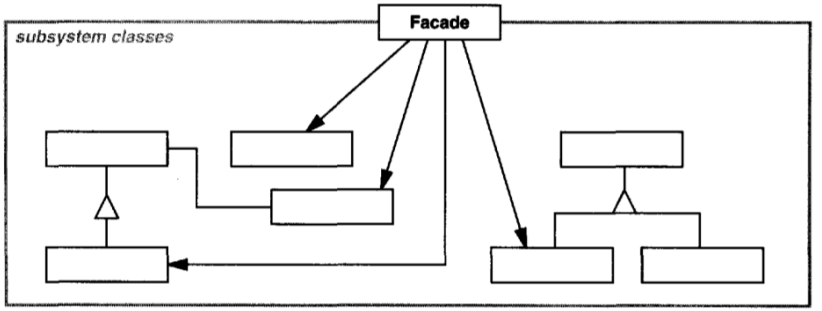
C++ (Smart references are doing object reference counting to delete objects from the memory when there is no reference to them)

Similar Patterns

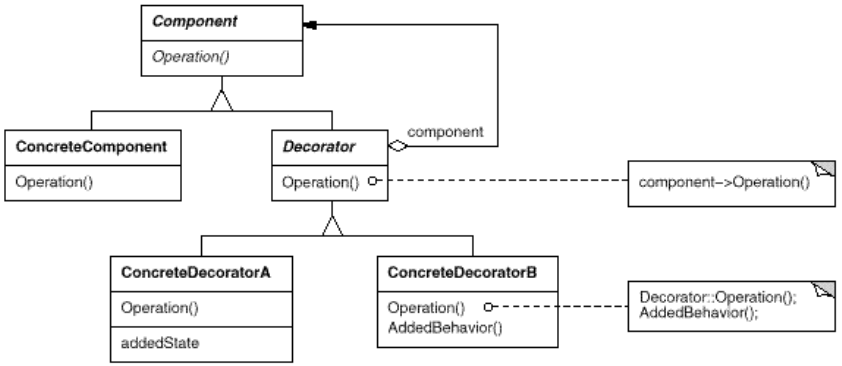
- Decorator, Adapter, Facade, Bridge

Object Adapter

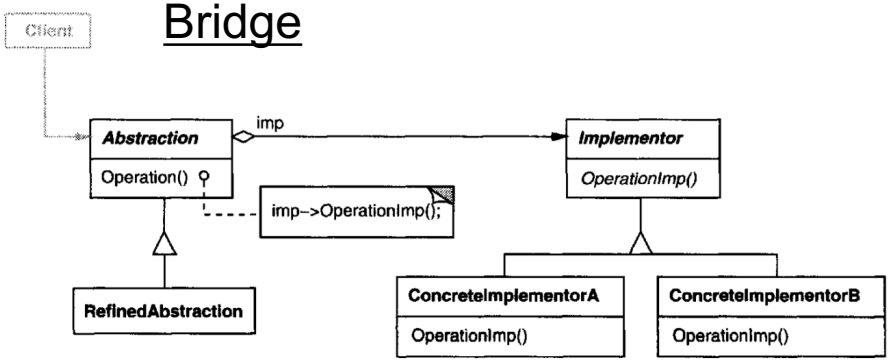




Decorator



Bridge



Similar Patterns

- Decorator, Adapter, Facade, Bridge

- Proxy is structurally similar to Decorator, but the two differ in their purpose.
- The Decorator Pattern adds behavior to an object, while a Proxy controls access.

- Like any wrapper, proxies will increase the number of classes and objects in your designs.

Summary

- The Proxy Pattern provides a placeholder for another object to control access to it because of different reasons.
- Proxy pattern has different applications and different types like remote proxy, virtual proxy, protection proxy, caching proxy, synchronization proxy, complexity hiding proxy (Facade pattern) , Firewall proxy, copy-on-write proxy.

- Proxy Pattern is of type **Structural Patterns**