Design Pattern - Structural Patterns

CSCI-630 - Software Design & Maintenance

-Sandesh Sobarad and Abhilash Sreenivasa

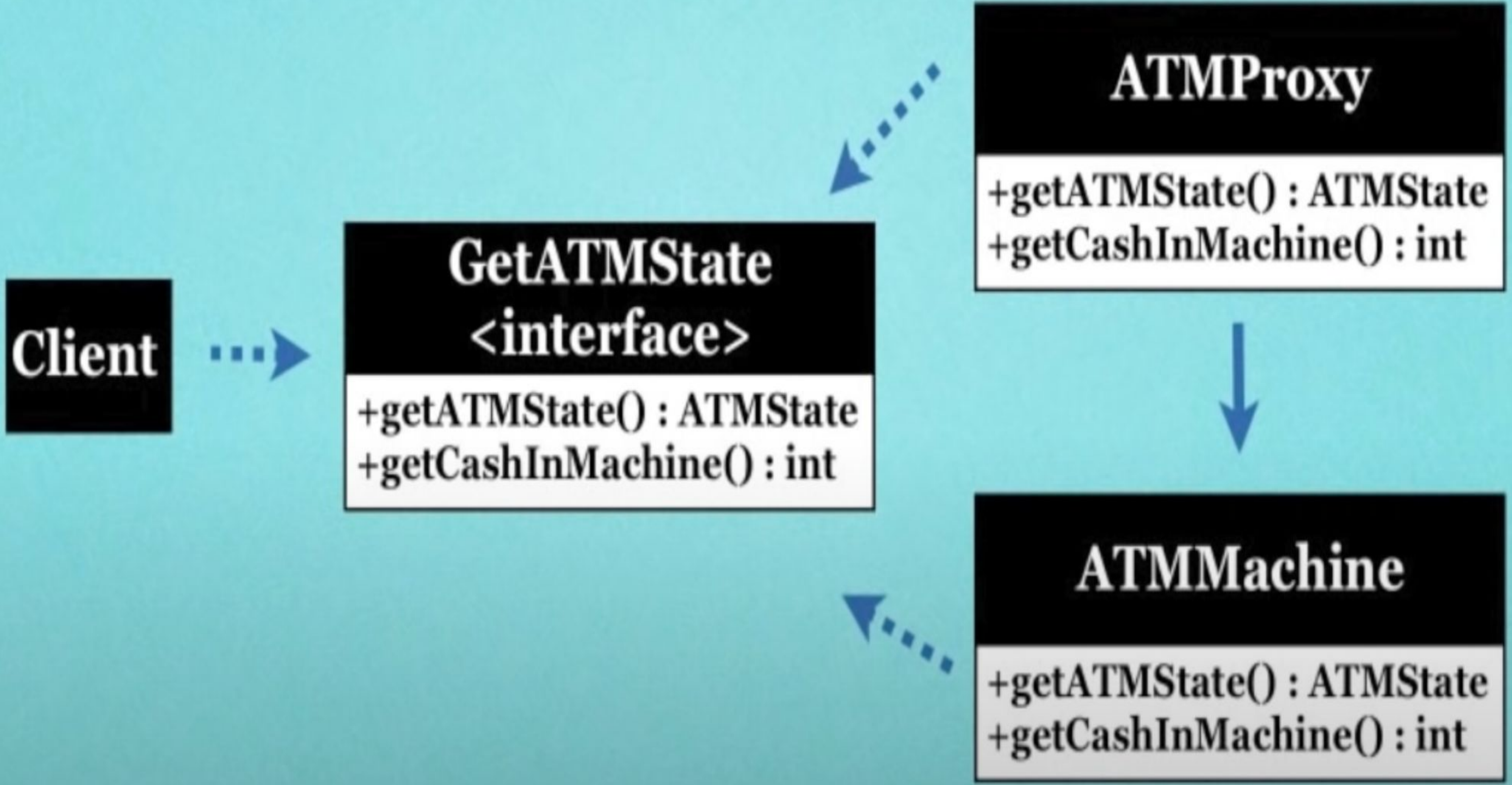
Proxy Design Pattern

Proxy Design Pattern and its Purpose

- The Proxy design pattern is a structural design pattern that provides a surrogate or placeholder object that acts as a representative of another object or adds additional functionality to it.

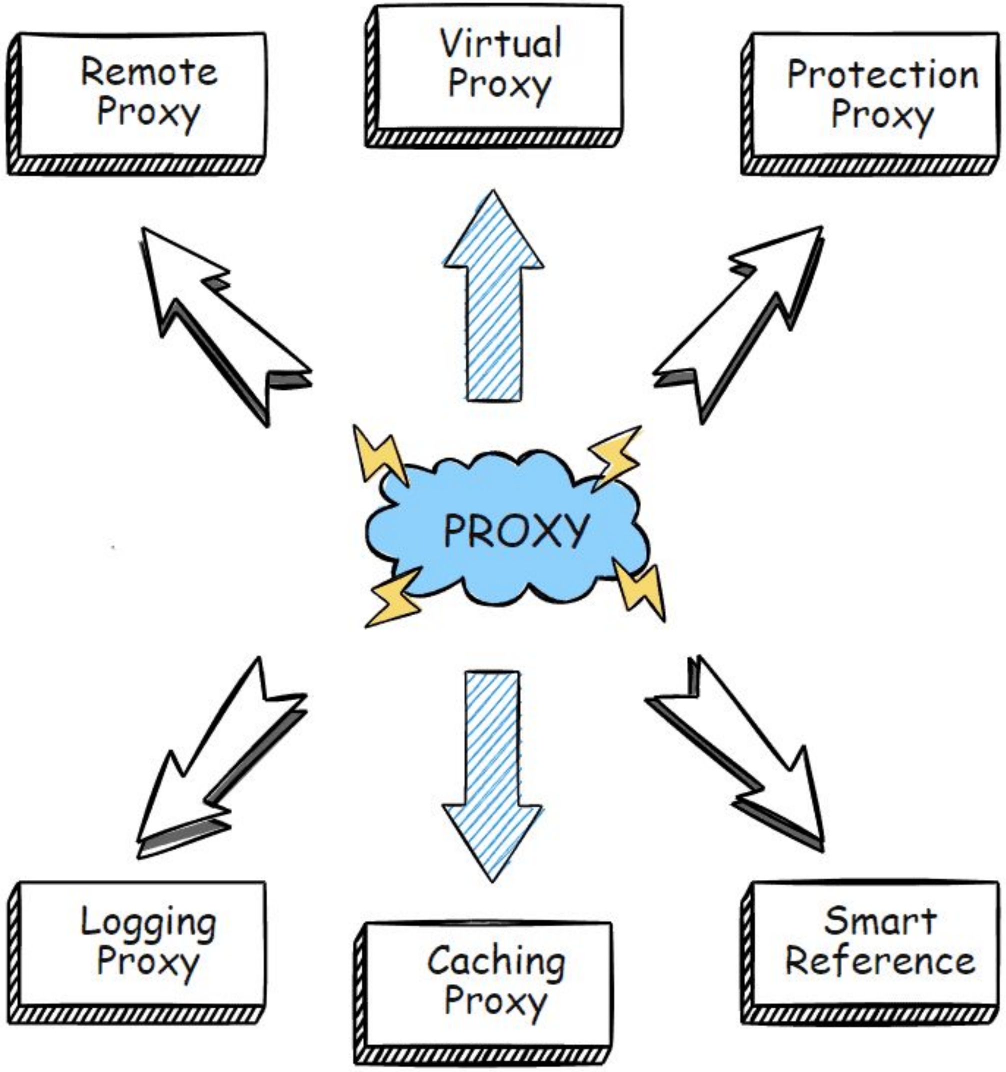
- The purpose of the Proxy pattern is to control access to an object or add additional behavior without modifying its implementation.

- The Proxy pattern is commonly used in scenarios where direct access to the real object may not be desirable or possible, or when additional behavior needs to be added to the real object without modifying its implementation


Client
GetATMState
<interface>
+getATMState() : ATMState
+getCashInMachine() : int
ATMProxy
+getATMState() : ATMState
+getCashInMachine() : int
ATMMachine
+getATMState() : ATMState
+getCashInMachine() : int

The Proxy pattern applications

- Controlling access: The proxy can enforce access control policies by checking permissions or authentication before allowing the client to access the real object.
- Adding functionality: The proxy can provide additional functionality or services around the real object.
- Optimizing performance: The proxy can cache results of expensive operations and return cached results for subsequent requests.
- Simplifying interfaces: The proxy can provide a simplified or specialized interface to the real object, tailored to the needs of the client.


Remote Proxy
Virtual Proxy
Protection Proxy
PROXY
Logging Proxy
Caching Proxy
Smart Reference

Protection Proxy

A protection proxy is a design pattern in computer programming that acts as a surrogate or intermediary for an object or resource, controlling access to it and providing additional protection or security features. It is a structural design pattern that allows an object (the proxy) to control access to another object (the real subject) by intercepting and handling requests from clients.

Real-time Examples:

- File System Access
- Remote Service Authentication
- Payment Processing.

Remote Proxy

In modern distributed systems, remote proxies are commonly used to communicate with remote services, APIs, or databases. They provide a local representation of the remote object, allowing the client to interact with it as if it were a local object, abstracting the complexities of network communication and protocols. Remote proxies are important in enabling seamless integration with remote resources, facilitating remote method invocations, and managing remote resource access.

Real-time Examples:

- Remote API calls
- Cloud Storage
- Database Connectivity

Pros and Cons of Proxy Design Pattern

Pros	Cons
Improved Performance	Increased Complexity
Enhanced Security	Reduced Transparency
Simplified Client Code	Overhead
Lazy Initialization	increased code duplication

References:

- https://www.youtube.com/watch?v=cHg5bWW4nUI&t=258
- https://www.google.com/search?q=proxy+design+pattern
- https://en.wikipedia.org/wiki/Proxy_pattern