**Name:** Proxy

**Category:** Structural

**Description:**

The idea behind the Proxy design pattern is to control access to an object by using a proxy object. The user can only access the obfuscated object via the proxy. This provides a layer of security to the object.

In my example implementation, the proxy is a "CreditCard" object that has access to the proxied "BankAccount". The user cannot directly access the bank account's money. But the credit card can spend money via the bank account and checks the credit card pin before executing the spend.

**When to Use:**

When you have an object whose access needs to be controlled for security reasons/monitoring. Proxies allow caching and can access remote resources like authentication services without affecting the requested data. Proxies can also hide the complicated interfaces of the objects they access.

**Advantages:**

* Security - A proxy can allow you to control access to an object by checking permissions of the requesting user before allowing access.
* Resource Management - Proxies can delay creation of real objects until needed also known as lazy loading.
* Caching - Proxies can store data from previous requests allowing the system to save time when hit with repeated requests.

**Disadvantages:**

* Complexity - Introducing a proxy can add complexity to a system, especially when multiple layers are being used.
* Increased response times - Navigating through multiple proxy layers for a request can add latency.
* Coupling - Using a proxy can lead to tight coupling between the proxy and the object it protects meaning the proxy might need changes when the object is changed.

**General UML:**



**Specific UML:**

