# Overview:

* The Proxy design pattern provides a surrogate or placeholder for another object to control access to it.
  + Used when we want to provide controlled access for a functionality.
* The formal definition of a proxy is a person authorized to act for another person.
  + An agent or substitute.
  + The authority to act for another.
* There are situations in which a client does not or cannot reference an object directly, but wants to still interact with object.
  + Introduces a level of indirection when accessing an object.
* A proxy object can act as the intermediary between the client and the target object.
* Another common use case is to provide a wrapper implementation for better performance.

# Examples:

* A check or a credit card is a proxy for what is in a bank account
  + Can be used in place of cash.
    - Provides a means of accessing that cash when required.
  + Exactly what a proxy does, it controls and manage access to the object they are protecting.
* In a classroom, when one student is absent, during roll call; his best friend may try to mimic the student’s voice to try to keep his friend from being marked as absent.
* Let’s say we have a class that can run some command on a system:
  + If we are using it, it works fine.
  + If we want to give this program to a client application.
    - Can have severe issues because client program can issue commands to delete some system files or change some settings that you do not want to.
  + A proxy class can be created to provide controlled access of the program.
* One reason for controlling access to an object is to defer the full cost of its creation and initialization until we actually need to use it.
* For example, a document editor that can embed graphical objects in a document:
  + Large raster images can be expensive to create.
  + However, opening a document should be fast.
    - We should avoid creating all the expensive objects at once when the document is opened.
* These constraints would suggest creating each expensive object on demand.
  + Occurs when an image becomes visible.
* The solution is to use another object.
  + An image proxy that acts as a stand-in for the real image.
  + The proxy acts just like image and takes care of instantiating it when it is required.

# Types of proxies:

* **Remote proxy:**
  + Manages interaction between a client and a remote object.
  + Provides a reference to an object located in a different address space on the same or different machine.
* **Virtual proxy:**
  + Controls access to an object that is expensive to instantiate.
  + Allows or the creation of a memory intensive object on demand.
    - Object will not be created until it is really needed.
* **Copy-On-Write proxy:**
  + Defers copying (cloning) a target object until required by client actions.
  + A form of a virtual proxy.
* **Protection (Access) proxy:**
  + Provides different clients with different levels of access to a target object.
* **Cache proxy:**
  + Provides temporary storage of the results of expensive target operations so that multiple clients can share the results.
* **Firewall proxy:**
  + Protects targets from bad clients (or vice versa)
* **Synchronization proxy:**
  + Provides multiple accesses to a target object.
* **Smart reference proxy:**
  + Provides additional actions whenever a target object is referenced such as counting the number of references to the object.

# Implementation:

# Participants:

* **Subject:**
  + Defines the common interface for **RealSubject** and **Proxy**.
    - A Proxy can be used anywhere a RealSubject is expected.
* **Proxy:**
  + Clients interact with the RealSubject through the 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:
      * May be needed if the Subject is running on a remote machine.
      * May be needed if the Subject is expensive to create in some way or if access to the subject needs to be protected in some way.
    - Forwards requests to RealSubject when appropriate (delegation)
      * Depending on type of proxy.
  + Provides an interface identical to Subject’s so that a proxy can be substituted for the real subject.
  + May refer to a Subject if the RealSubject and Subject interfaces are the same.
  + Other responsibilities depend on the kind of proxy:
    - Remote proxies are responsible for encoding a request and its arguments and for sending the encoded request to the real subject in a different address space.
    - Virtual proxies may cache additional information about the real subject so that they can postpone accessing it.
    - Protection proxies check that the caller has the access permissions required to perform a request.
  + Has the authority the act on behalf of the client to interact with the target object.

# Summary:

* The proxy pattern provides a representative for another object in order to control the client’s access to it.
* Security is a big advantage:
  + Remote proxies ensures a more secure application by installing the local code proxy (stud) in the client machine and then accessing the server with help of the remote code.
* Avoids duplication of objects, which might be huge in size and memory intensive.
  + Increases the performance of the application.
* Proxy is structurally similar to decorator, but the two differ in their purpse.
  + The Decorator pattern adds behavior to an object.
  + The Proxy pattern controls access.
* Proxies will increase the number of classes and objects in your designs.