Proxy Intent

Provide a surrogate or placeholder for another object to control access to it.
(Structural)
Sometimes you just can not deal with the real object and need to use a **Proxy** for it.

Client deals with a proxy that looks exactly like the real subject.

Proxy deals with the real subject hiding its “flaws” from the client.

... but the real subject has problems such as:
- unwieldy size
- remote location
- needing access protection
- smart pointer.

Client

Proxy

Real Subject
The Client can not tell if the Subject is a RealSubject or a Proxy forwarding requests.

Here is a UML diagram showing the relationship:

- Client
- Subject
  - request()
- RealSubject
  - request()
- Proxy
  - request()
  - realSubject->request()

What is the nature of this relationship?
There are four categories of proxies to match the shortcomings identified earlier.

- Remote proxy - as above
  - Local representative for something in a different address space
  - Java RMI and Unix RPC are low-level tools to set up remote proxies automatically
  - Object brokers handle remote objects (CORBA or DCOM)

- Virtual proxy
  - Stand-in for an object that is expensive to implement or completely access
  - Example – image over the net
  - May be able to access some state (e.g., geometry) at low cost
  - Defer other high costs until it must be incurred
There are four categories of proxies to match the shortcomings identified earlier. (cont.)

- **Protection proxy**
  - *Control access to the "real" object*
  - *Different proxies provide different rights to different clients*
  - *For simple tasks, can do via multiple interfaces available to clients*
  - *For more dynamic checking, need a front-end such as a proxy*

- **Smart Pointers or Arrays (C++)**
  - *Perform additional actions when used as references*
  - *Example: Reference counting for simple memory management*
  - *Example: Ensure proper locking semantics on shared object*
  - *Example: Array bounds checking*