Proxy Design Pattern

Proxy is something that acts as an intermediary for requests from clients. In the similar way, for design patterns, we use proxy to access certain functions or objects.
A real world example can be a cheque or credit card or debit card is a proxy for what is in our bank account. It can be used in place of cash, and provides a means of accessing that cash when required. And that's exactly what the Proxy pattern does – "Controls and manage access to the object they are protecting". Also, we can look at the DAO classes in spring as proxy for services to access the database.

There are 4 types of proxies:

1. **Remote Proxy**: They are responsible for representing the object located remotely. Talking to the real object might involve marshalling and unmarshalling of data and talking to the remote object.

2. **Virtual Proxy**: These proxies will provide some default and instant results if the real object is supposed to take some time to produce results. These proxies initiate the operation on real objects and provide a default result to the application.

3. **Protection proxy**: If an application does not have access to some resource then such proxies will talk to the objects in applications that have access to that resource and then get the result back.

4. **Smart Proxy**: A smart proxy provides additional layer of security by interposing specific actions when the object is accessed. An example can be to check if the real object is locked before it is accessed to ensure that no other object can change it.

[All these were explained using some diagrams]