there are no Dumb Questions

Q: If the Facade encapsulates the subsystem classes, how does a client that needs lower-level functionality gain access to them?

A: Facades don't "encapsulate" the subsystem classes; they merely provide a simplified interface to their functionality. The subsystem classes still remain available for direct use by clients that need to use more specific interfaces. This is a nice property of the Facade Pattern: it provides a simplified interface while still exposing the full functionality of the system to those who may need it.

Q: Does the facade add any functionality or does it just pass through each request to the subsystem?

A: A facade is free to add its own "smarts" in addition to making use of the subsystem. For instance, while our home theater facade doesn't implement any new behavior, it is smart enough to know that the popcorn popper has to be turned on before it can pop (as well as the details of how to turn on and stage a movie showing).

Q: Does each subsystem have only one facade?

A: Not necessarily. The pattern certainly allows for any number of facades to be created for a given subsystem.

Q: What is the benefit of the facade other than the fact that I now have a simpler interface?

A: The Facade Pattern also allows you to decouple your client implementation from any one subsystem. Let's say for instance that you get a big raise and decide to upgrade your home theater to all new components that have different interfaces. Well, if you coded your client to the facade rather than the subsystem, your client code doesn't need to change, just the facade (and hopefully the manufacturer is supplying that!).

Q: So the way to tell the difference between the Adapter Pattern and the Facade Pattern is that the adapter wraps one class and the facade may represent many classes?

A: No! Remember, the Adapter Pattern changes the interface of one or more classes into one interface that a client is expecting. While most textbook examples show the adapter adapting one class, you may need to adapt many classes to provide the interface a client is coded to. Likewise, a Facade may provide a simplified interface to a single class with a very complex interface.

The difference between the two is not in terms of how many classes they "wrap," it is in their **intent.** The intent of the Adapter Pattern is to **alter** an interface so that it matches one a client is expecting. The intent of the Facade Pattern is to provide a **simplified** interface to a subsystem.

A facade not only simplifies an interface, it decouples a client from a subsystem of components.

Facades and adapters may wrap multiple classes, but a facade's intent is to simplify, while an adapter's is to convert the interface to something different.

Head First Design Patterns By Eric Freeman, Elisabeth Freeman, Bert Bates, Kathy Sierra
ISBN: 0596007124 Publisher: O'Reilly
Print Publication Date: 2004/10/25