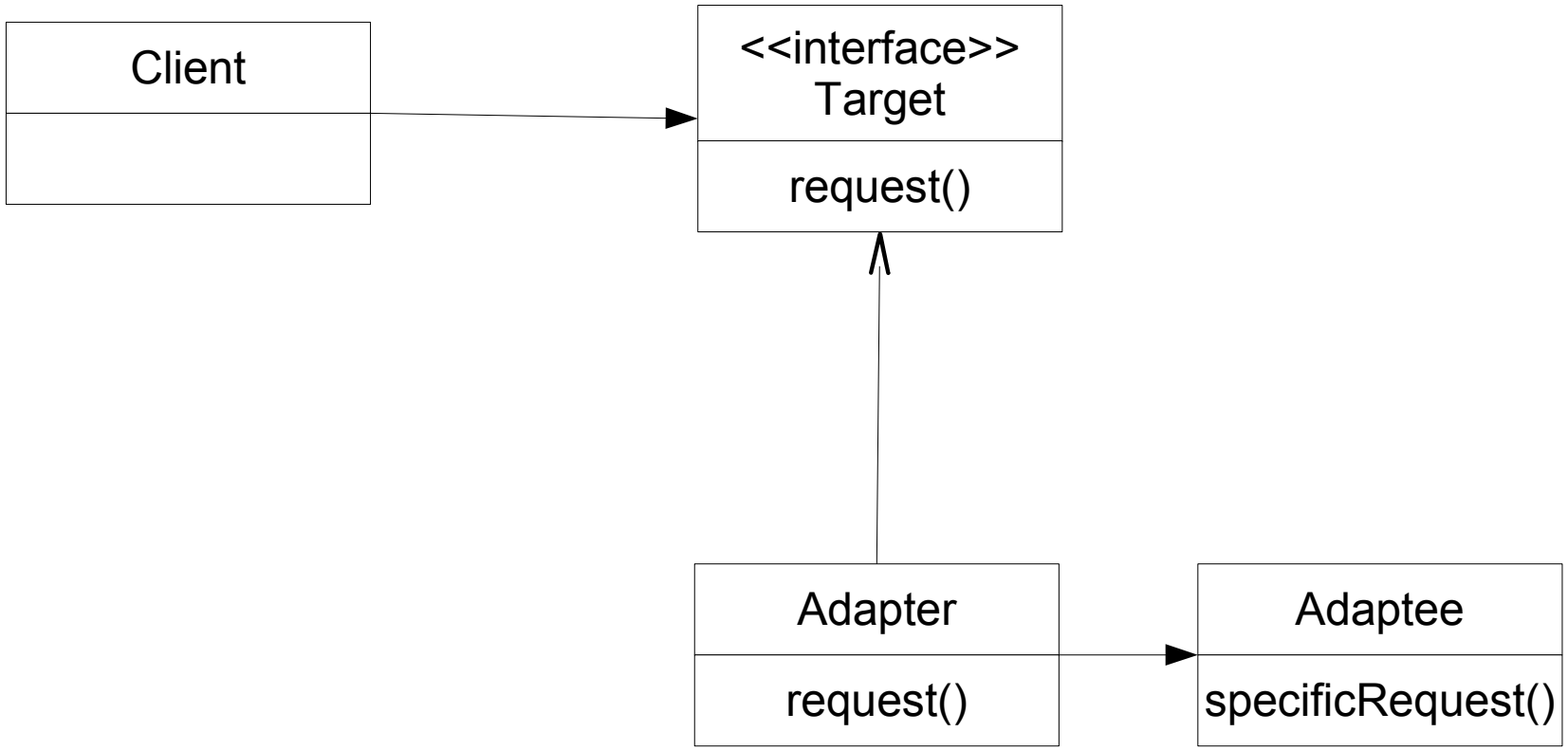
Adapter Pattern

The Adapter Pattern converts the interface of a class into another interface the clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.

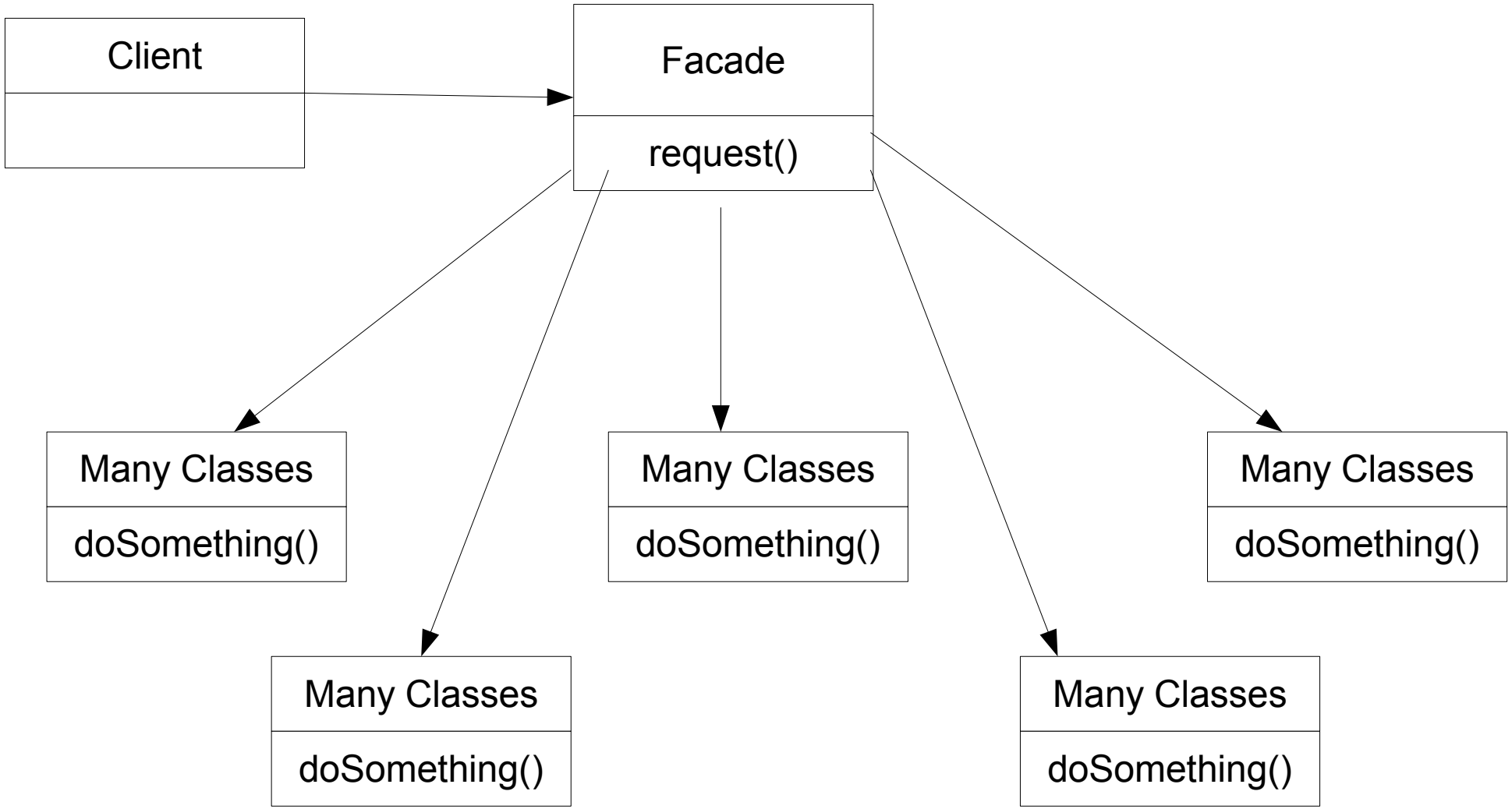

Client
<<interface>>
Target
request()
Adapter
request()
Adaptee
specificRequest()

Questions

- Can an Adpater use more than one class?
  - Yes if necessary.
- What do you do if the Adaptee has no appropriate method for the Adapted Interface?
  - Throw an error – e.g. Iterator vs. Enumerator
- What patterns is it similar to?
  - Facade – but it's intent is to convert not to simplify
  - Decorator – but it always changes an interface and doesn't add responsibility
  - Proxy – but it converts a class rather than duplicates a class's methods

Facade Pattern

- The Facade Pattern provides a unified interface to a set of interfaces in a subsystem. Facade defines a higher level interface that makes the subsystem easier to use.


Client
Facade
request()
Many Classes
doSomething()
Many Classes
doSomething()
Many Classes
doSomething()
Many Classes
doSomething()
Many Classes
doSomething()

Questions

- What if you need access to the Facade's underlying classes?
  - You still have access w/ Facade – it doesn't hide anything
- Does it just simplify?
  - No. It decouples. You could add extra classes to your Facade but the Client would still call the same method.

Design Principle

Principle of Least Knowledge -
talk only to your immediate friends