Tecniche di Progettazione: Design Patterns

GoF: Façade

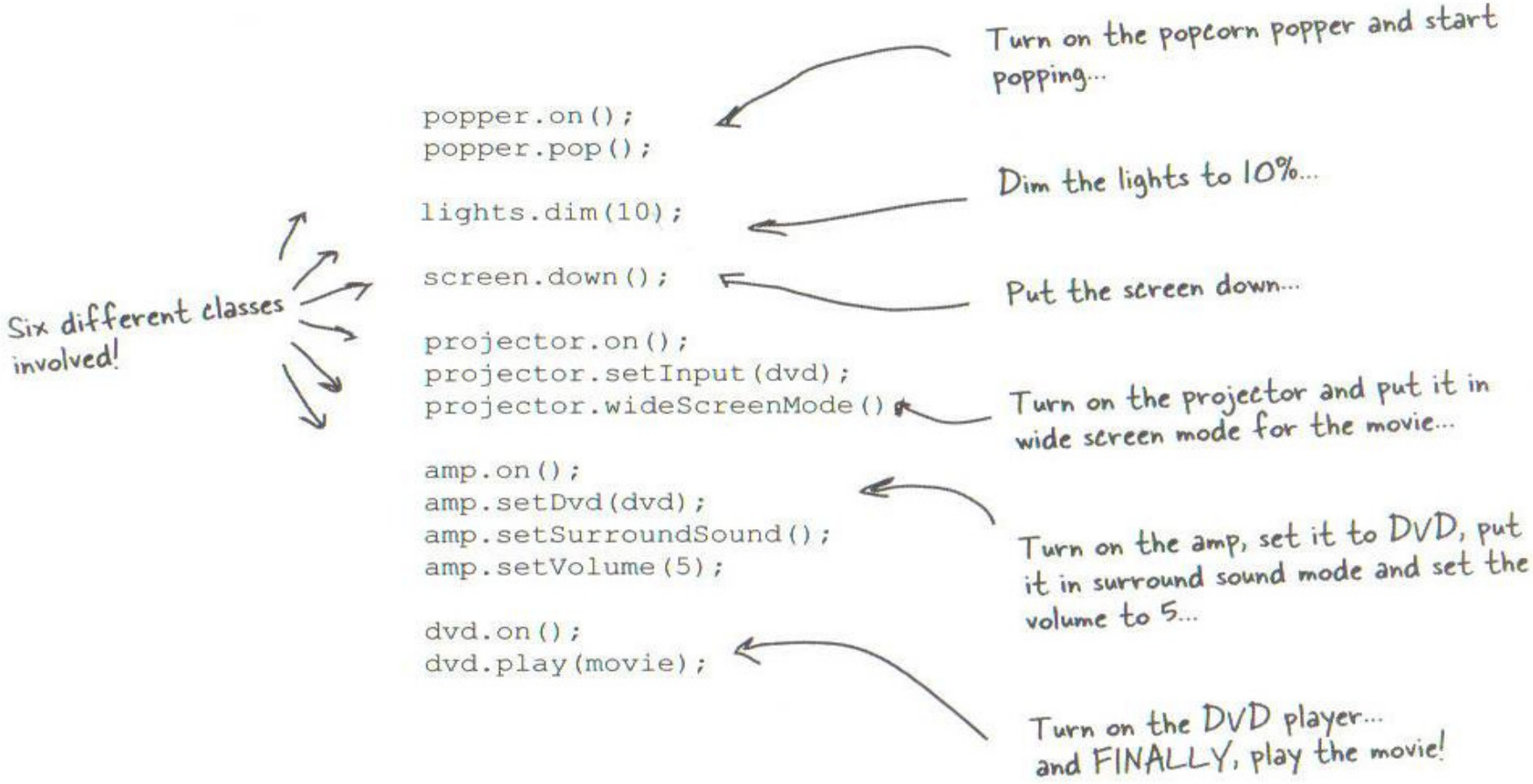
Facade or Façade

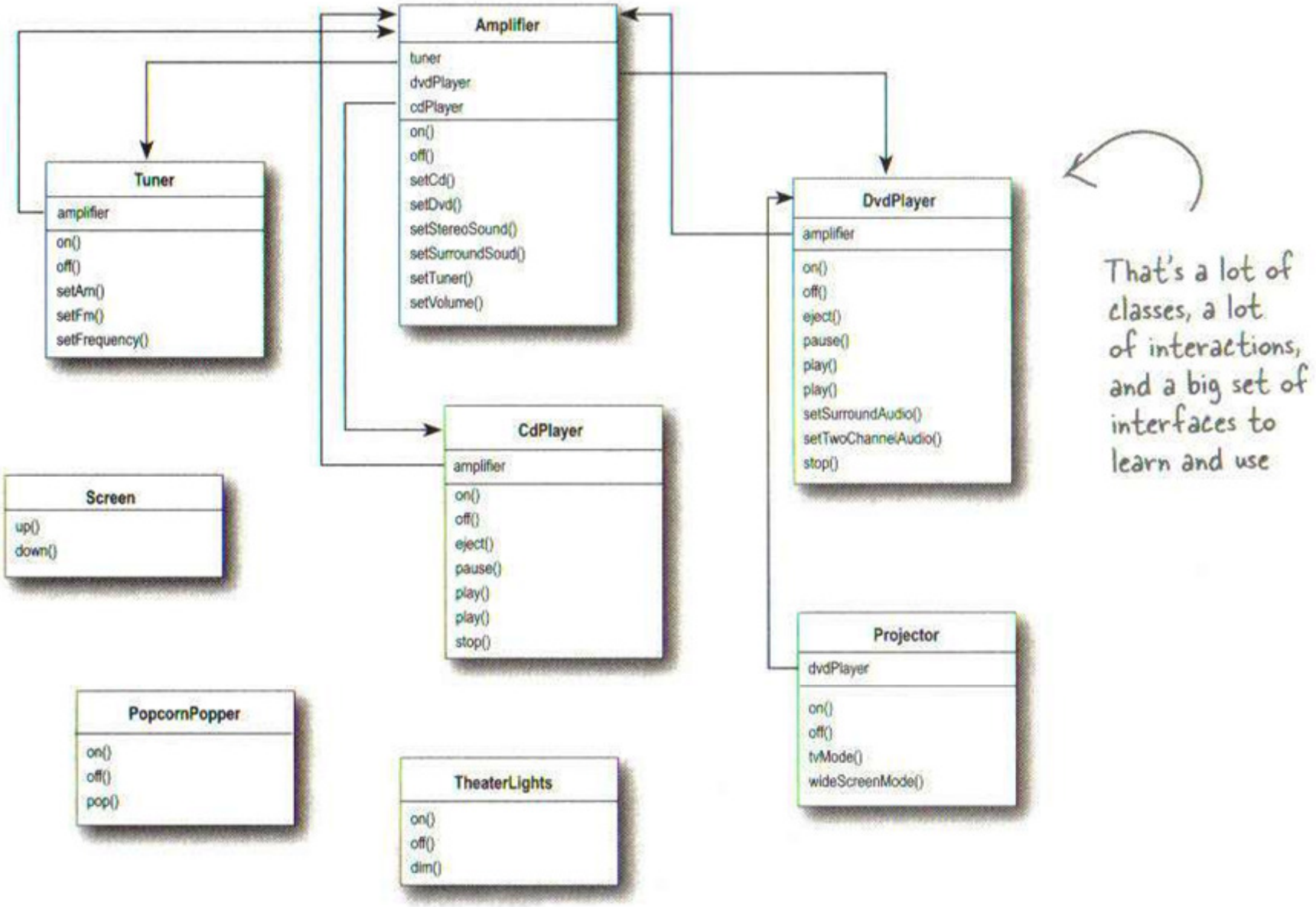
- ***From www.m-w.com***
- Main Entry: **fa.cade**
  Variant(s): *also* **fa.çade** /f&-'säd/
  Function: *noun*
  Etymology: French *façade*, from Italian *facciata*, from *faccia* face, from (assumed) Vulgar Latin *facia*
  Date: circa 1681
  1 : the front of a building; *also* : any face of a building given special architectural treatment <a museum's east *facade*>
  2 : a false, superficial, or artificial appearance or effect

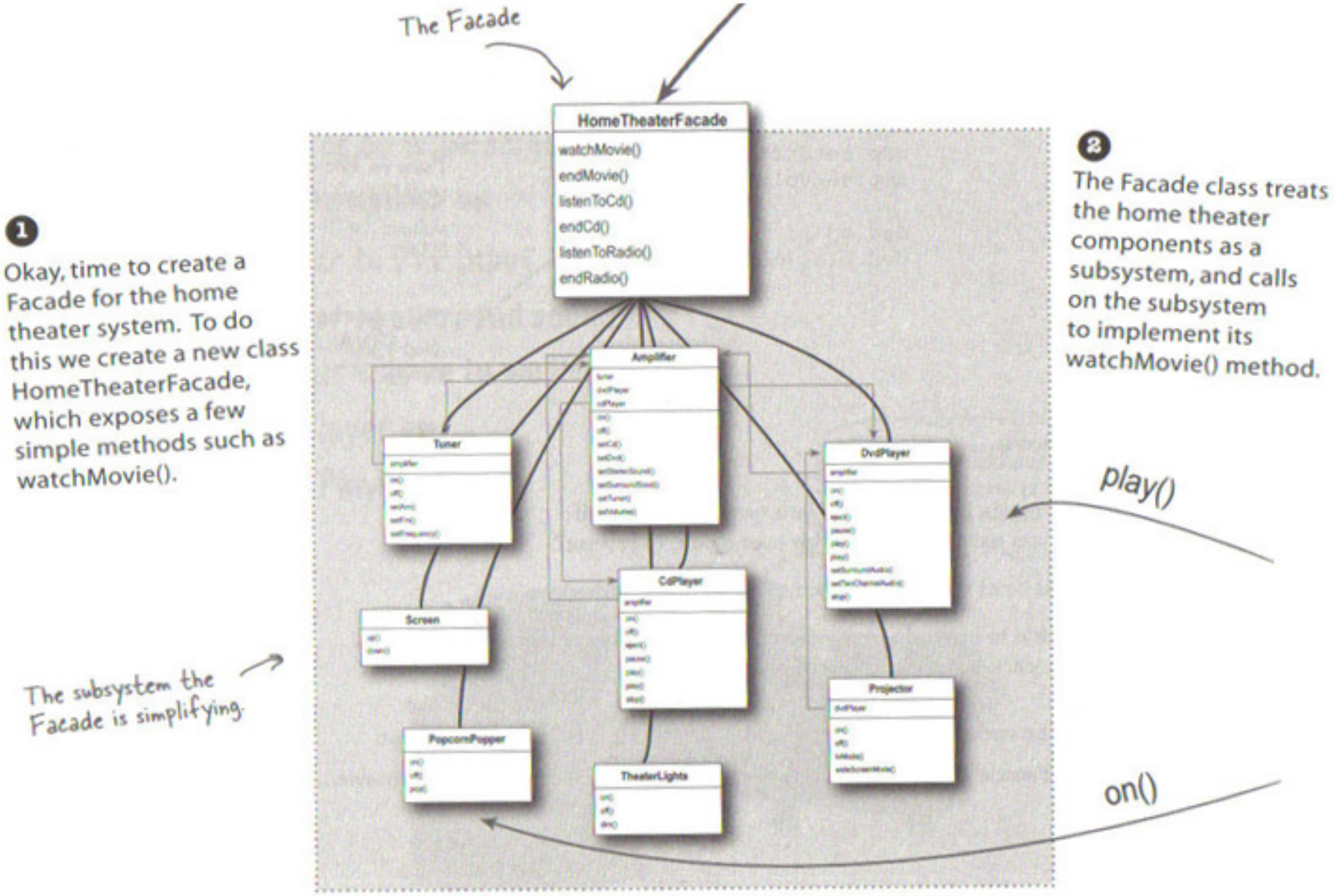
Watching a movie the hard way….

1. Turn on the popcorn popper
2. Start the popper popping
3. Dim the lights
4. Put the screen down
5. Turn the projector on
6. Set the projector input to DVD
7. Put the projector on wide-screen mode
8. Turn the sound amplifier on
9. Set the amplifier to DVD input
10. Set the amplifier to surround sound
11. Set the amplifier volume to medium (5)
12. Turn the DVD Player on
13. Start the DVD Player playing

What needs to be done to watch a movie….



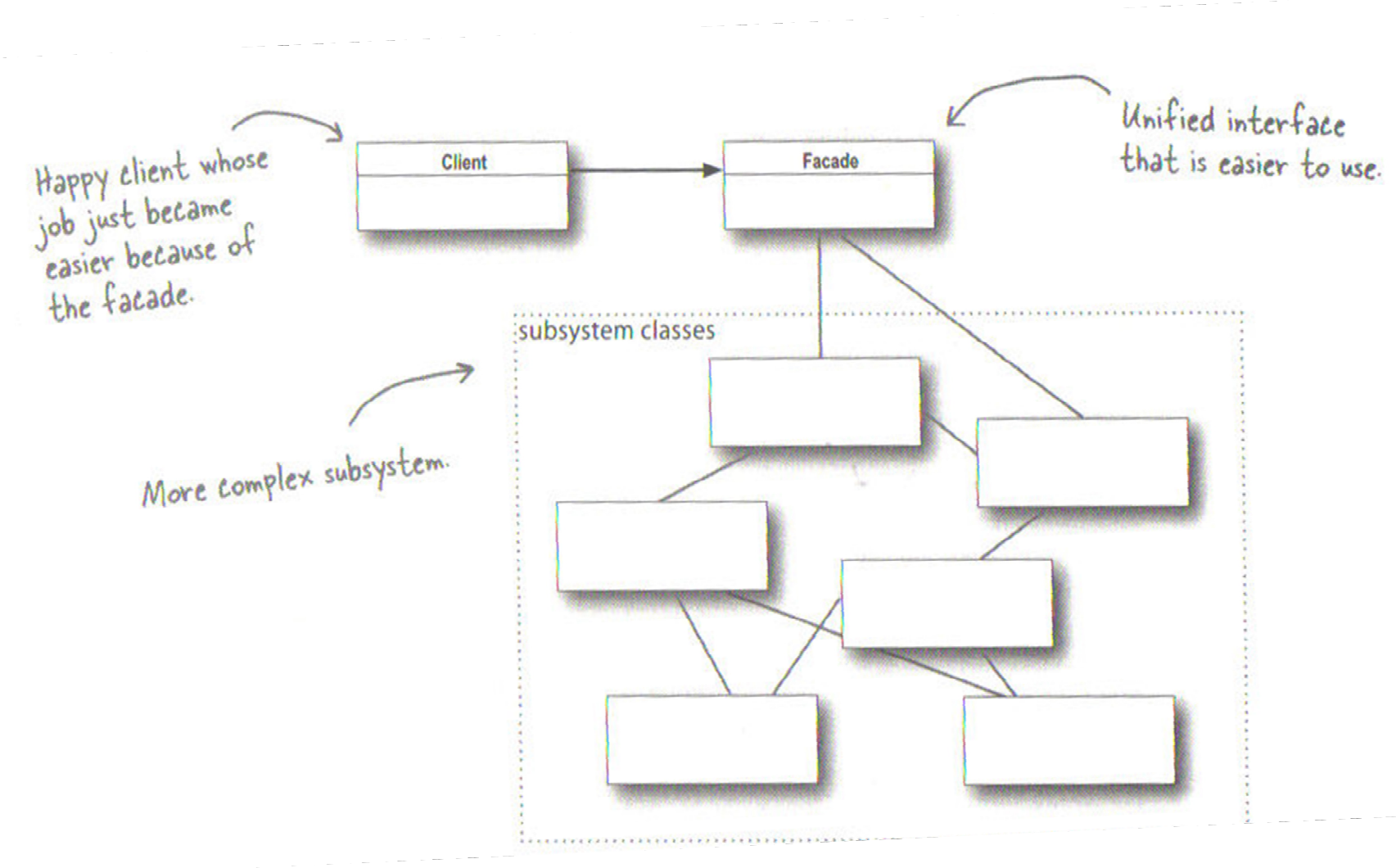

Amplifier
tuner
dvdPlayer
cdPlayer
on()
off()
setCd()
setDvd()
setStereoSound()
setSurroundSoud()
setTuner()
setVolume()
Tuner
amplifier
on()
off()
setAm()
setFm()
setFrequency()
DvdPlayer
amplifier
on()
off()
eject()
pause()
play()
play()
setSurroundAudio()
setTwoChannelAudio()
stop()
That's a lot of classes, a lot of interactions, and a big set of interfaces to learn and use
CdPlayer
amplifier
on()
off()
eject()
pause()
play()
play()
stop()
Screen
up()
down()
Projector
dvdPlayer
on()
off()
tvMode()
wideScreenMode()
PopcornPopper
on()
off()
pop()
TheaterLights
on()
off()
dim()


The Facade
HomeTheaterFacade
watchMovie()
endMovie()
listenToCd()
endCd()
listenToRadio()
endRadio()
1
Okay, time to create a Facade for the home theater system. To do this we create a new class HomeTheaterFacade, which exposes a few simple methods such as watchMovie().
2
The Facade class treats the home theater components as a subsystem, and calls on the subsystem to implement its watchMovie() method.
Amplifier
Tuner
DvdPlayer
CdPlayer
Screen
PopcornPopper
TheaterLights
Projector
play()
on()
The subsystem the Facade is simplifying

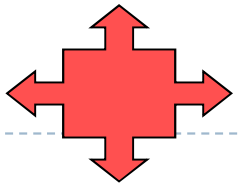
Façade Pattern defined

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

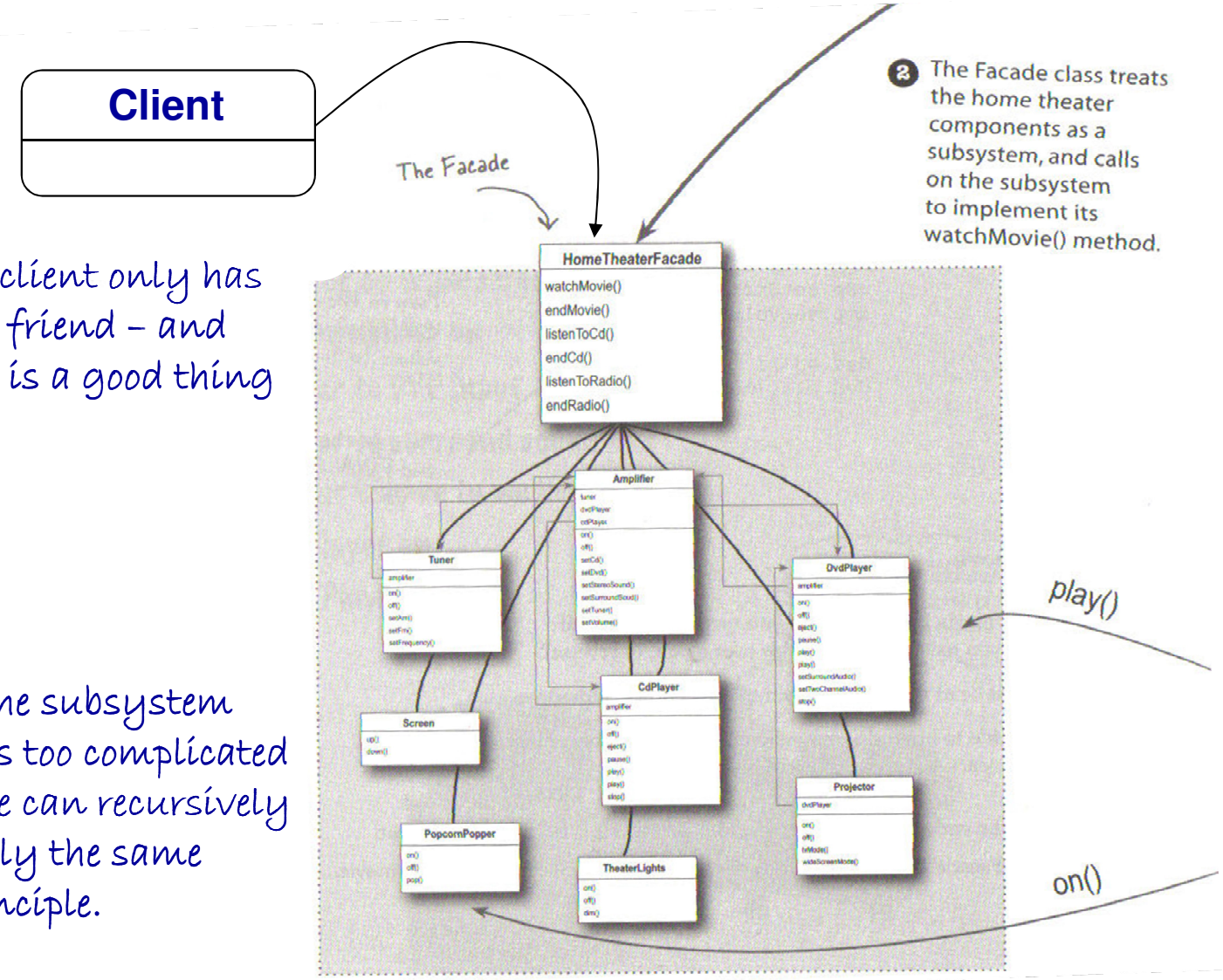
Façade pattern – Class Diagram



Design Principle



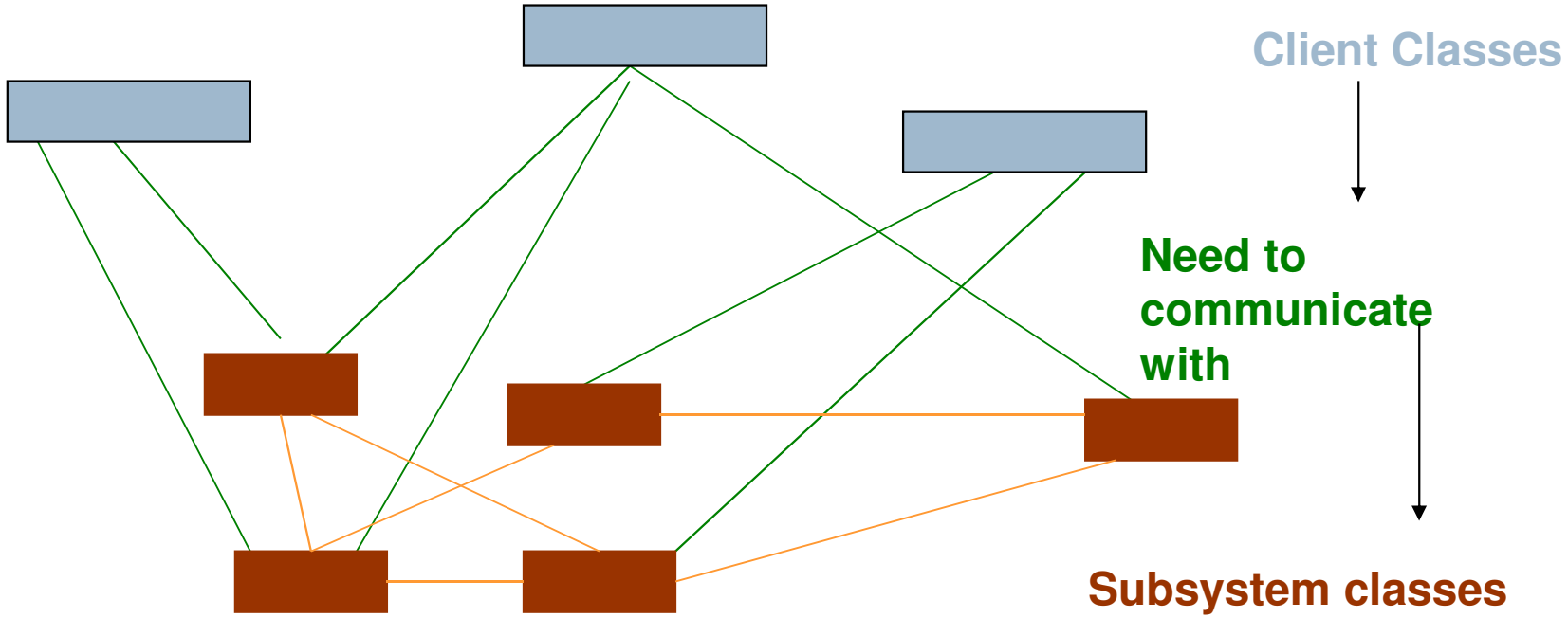
- Law of Demeter
  - aka Principle of Least Knowledge (Head first)
  - talk only to your immediate friends



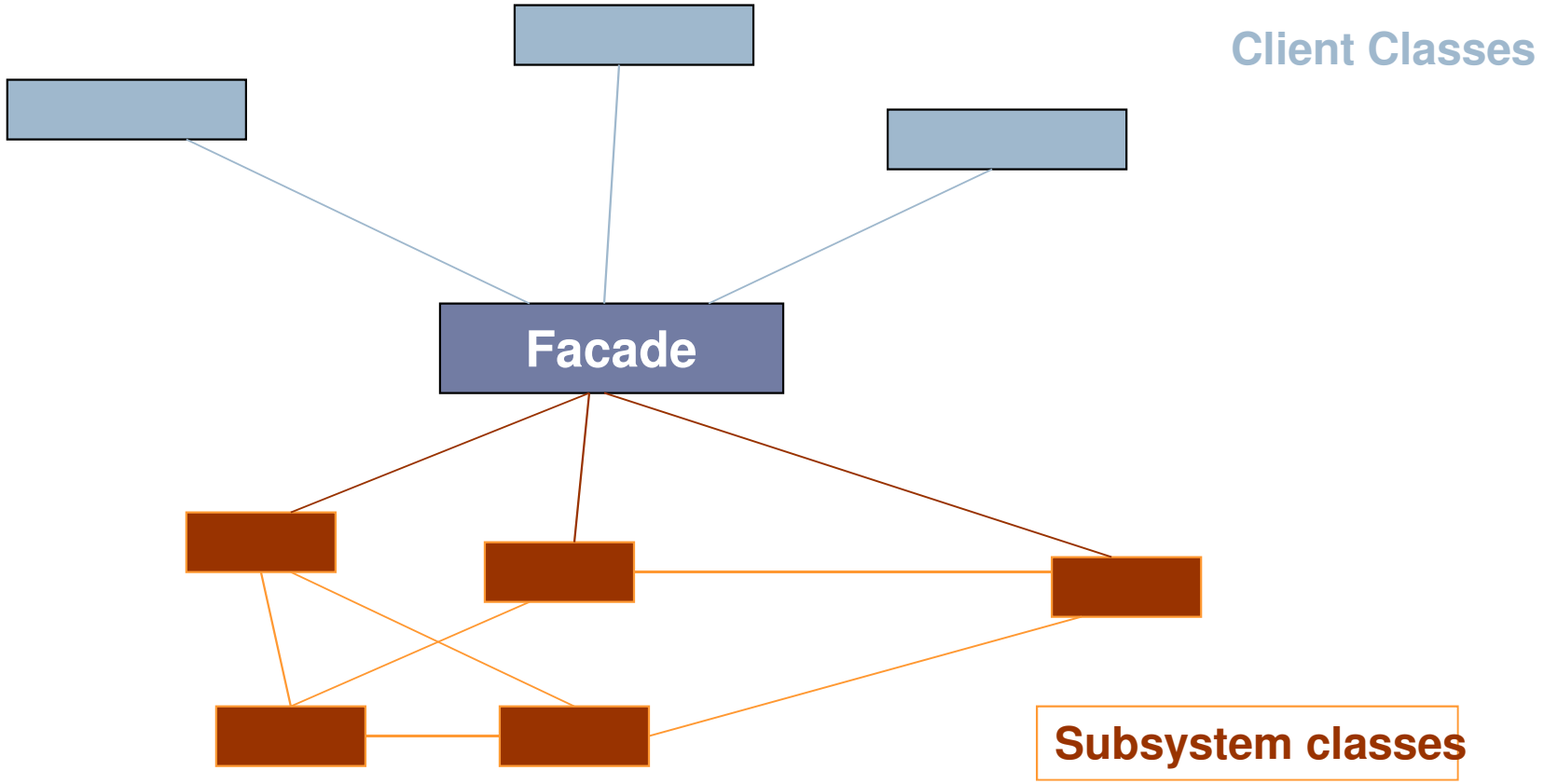
The client only has one friend – and that is a good thing

If the subsystem gets too complicated One can recursively apply the same principle.

Another perspective: The Problem



Another perspective: the solution

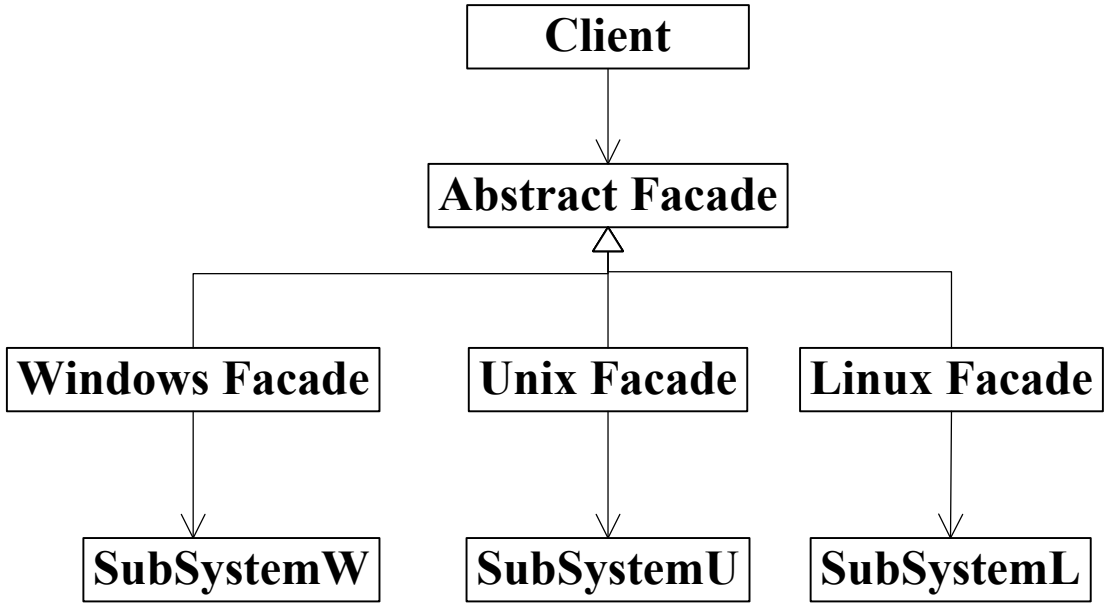


Consequences

- The Facade pattern offers the following benefits:
  - It shields clients from subsystem components.
  - It promotes weak coupling between the subsystem and its clients.
    - help layer a system and the dependencies between objects.
    - reduce compilation dependencies.
    - simplify porting systems to other platforms.
  - It doesn't prevent applications from using subsystem classes

Subclassing: Façade is the common interface of many façades

- Can further decouple clients and subsystem by making Façade an abstract class with concrete subclasses for different implementations of a subsystem.
- Alternative: Configure a Facade object with different subsystem objects.



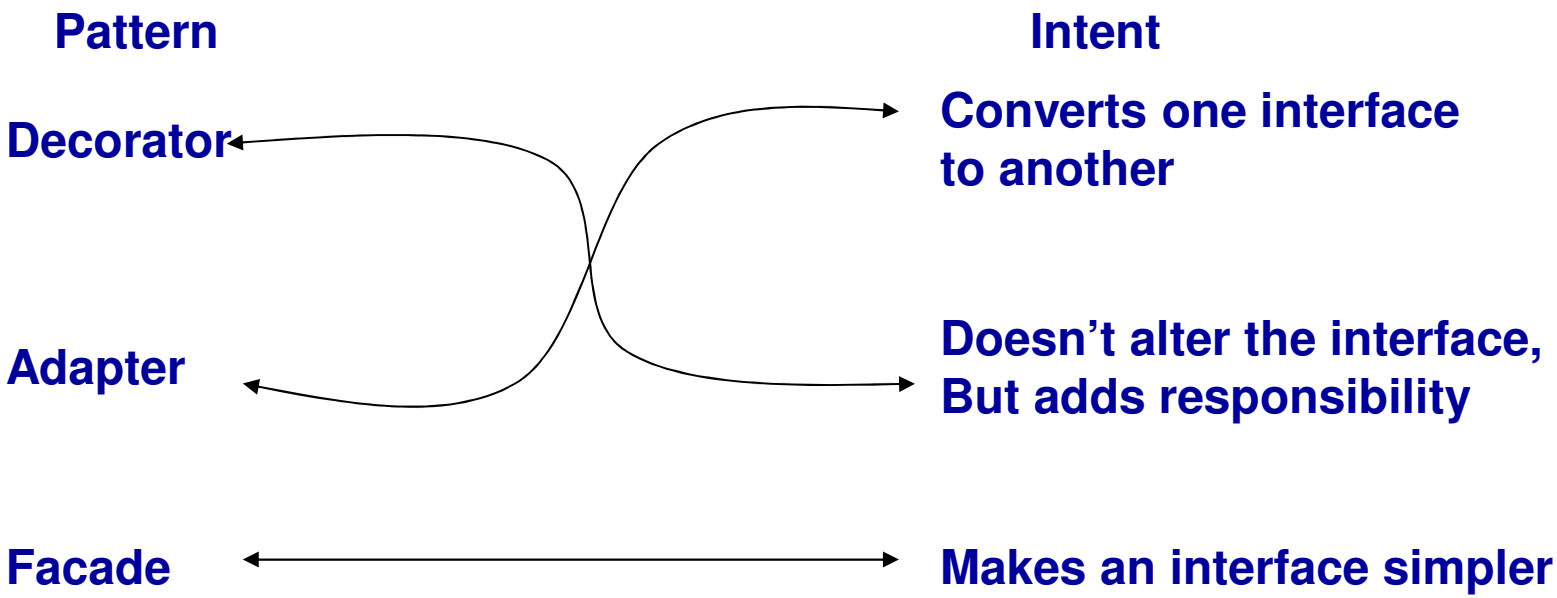
Implementation

- Public versus private subsystem classes.
  - Public interface to a subsystem consists of classes that all clients can access; private interface is just for subsystem extenders.
  - Façade class is part of the public interface. It may use private interfaces provided by subsystems (if they grant friendship)

Façade vs Adapter

- To many peole , these two patterns appear to be similar
  - They both act as wrappers of a preexisting class
  - They both take an interface that we don't need and convert it to an interface that we can use
  - With Facade, the intent is to simplify the existing interface
  - With Adapter, we have a target interface that we are converting to; we often want the adapter to plug into an existing framework and behave polymorphically

A little comparison



Homework: DPHomework4

1. See homework for Adapter
2. Modify the home theater example along the following dimensions:
    1. CD is substituted with a french one
    2. The system prevents you from watching a movie you saw *recently*
3. Create and example where Façade is an abstract class with different concrete subclasses.
    1. Compare with strategy