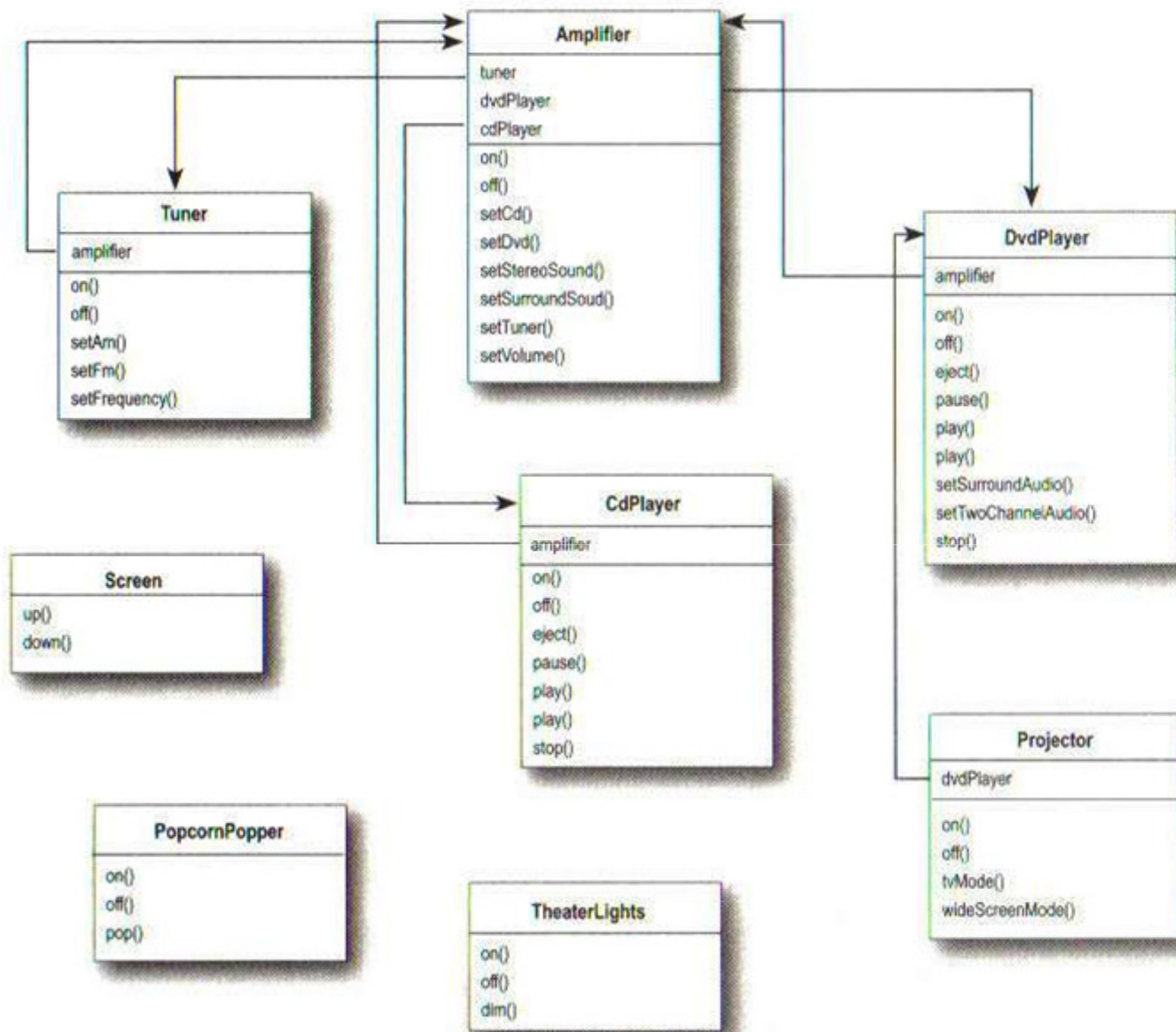


# Tecniche di Progettazione: Design Patterns

GoF: Façade



That's a lot of classes, a lot of interactions, and a big set of interfaces to learn and use



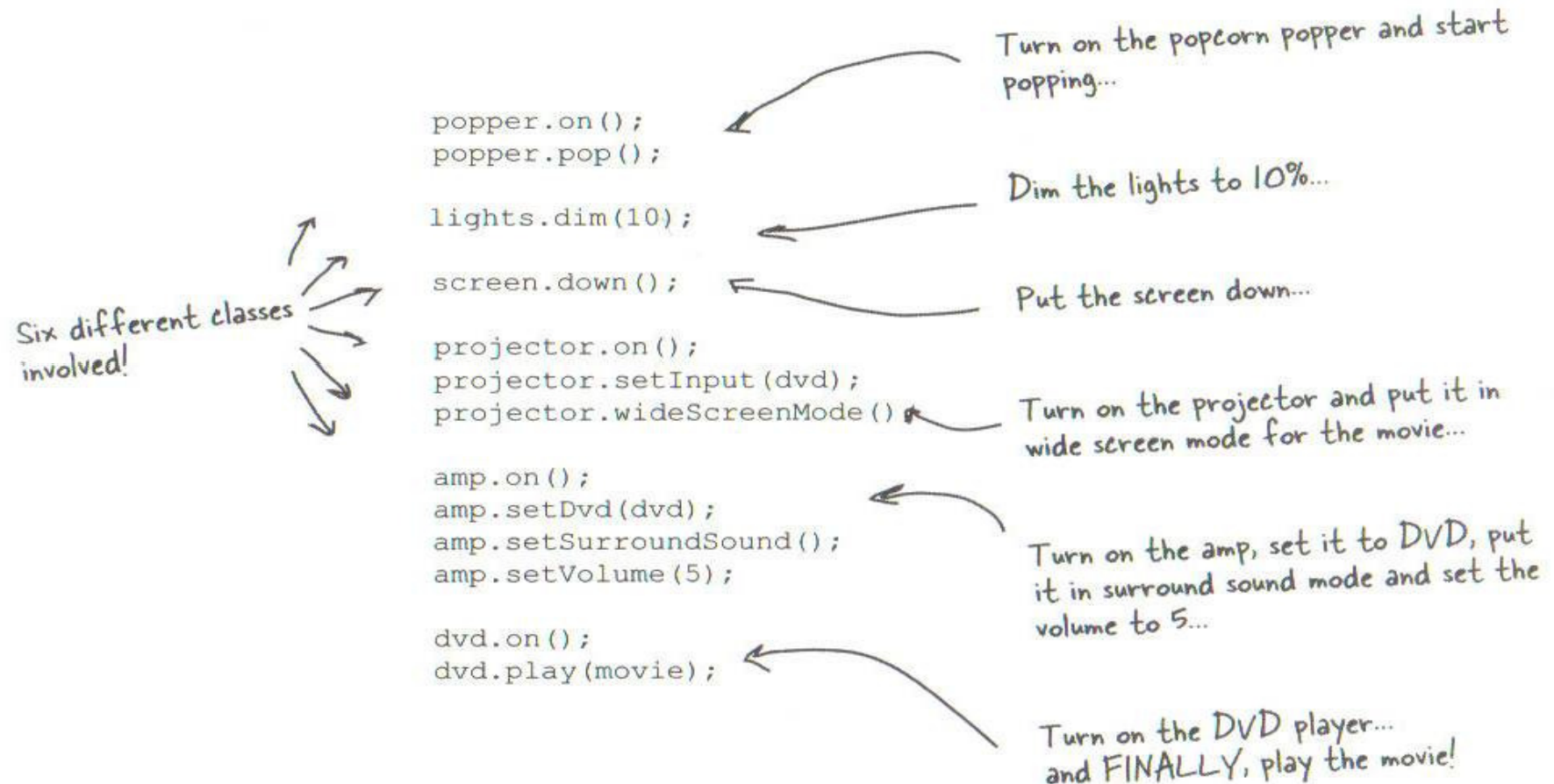
# Watching the movie the hard way....

---

- 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

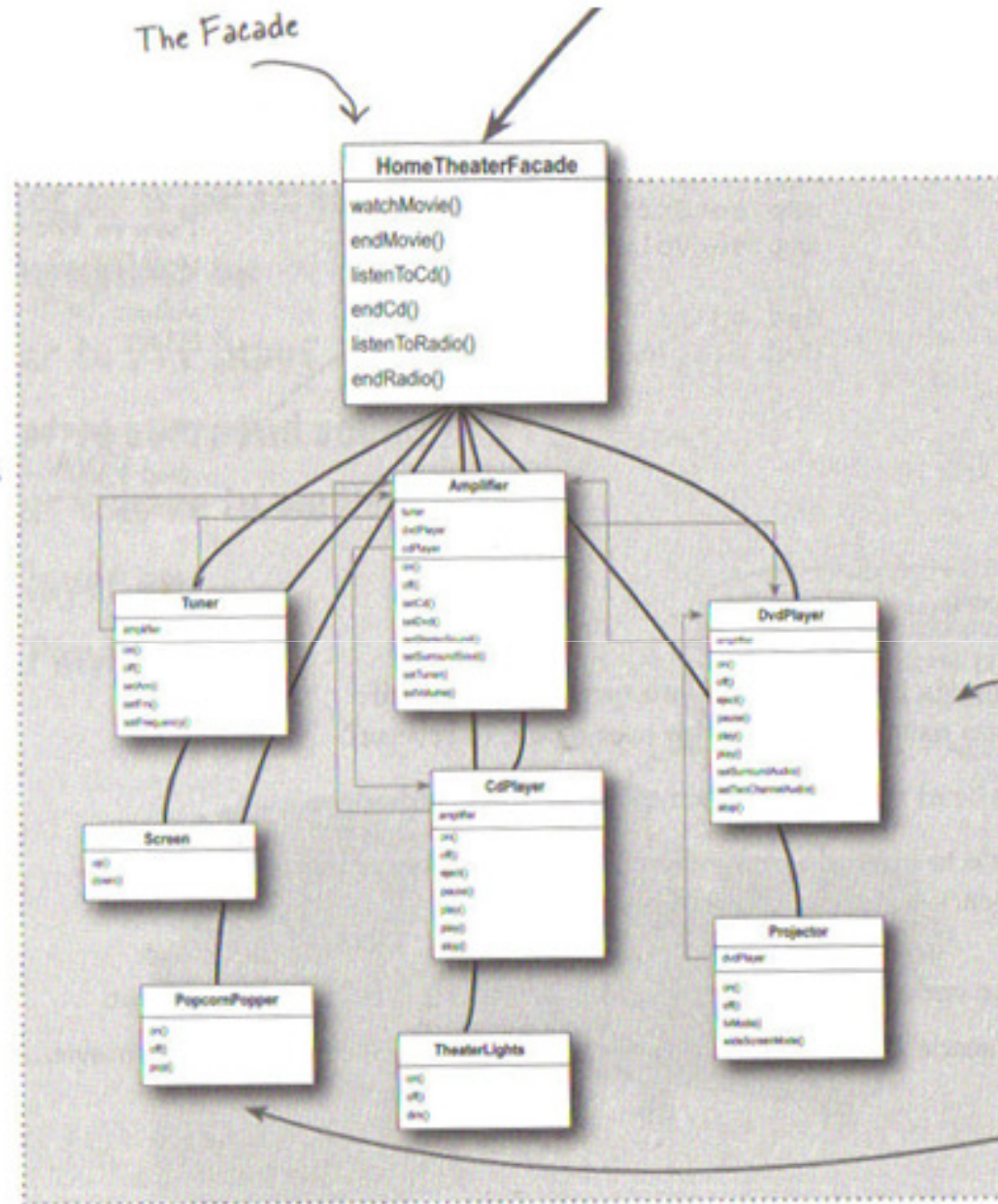
# What needs to be done to watch a movie....

---



**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().

The Facade



**2** The Facade class treats the home theater components as a subsystem, and calls on the subsystem to implement its watchMovie() method.

play()

on()

The subsystem the Facade is simplifying.

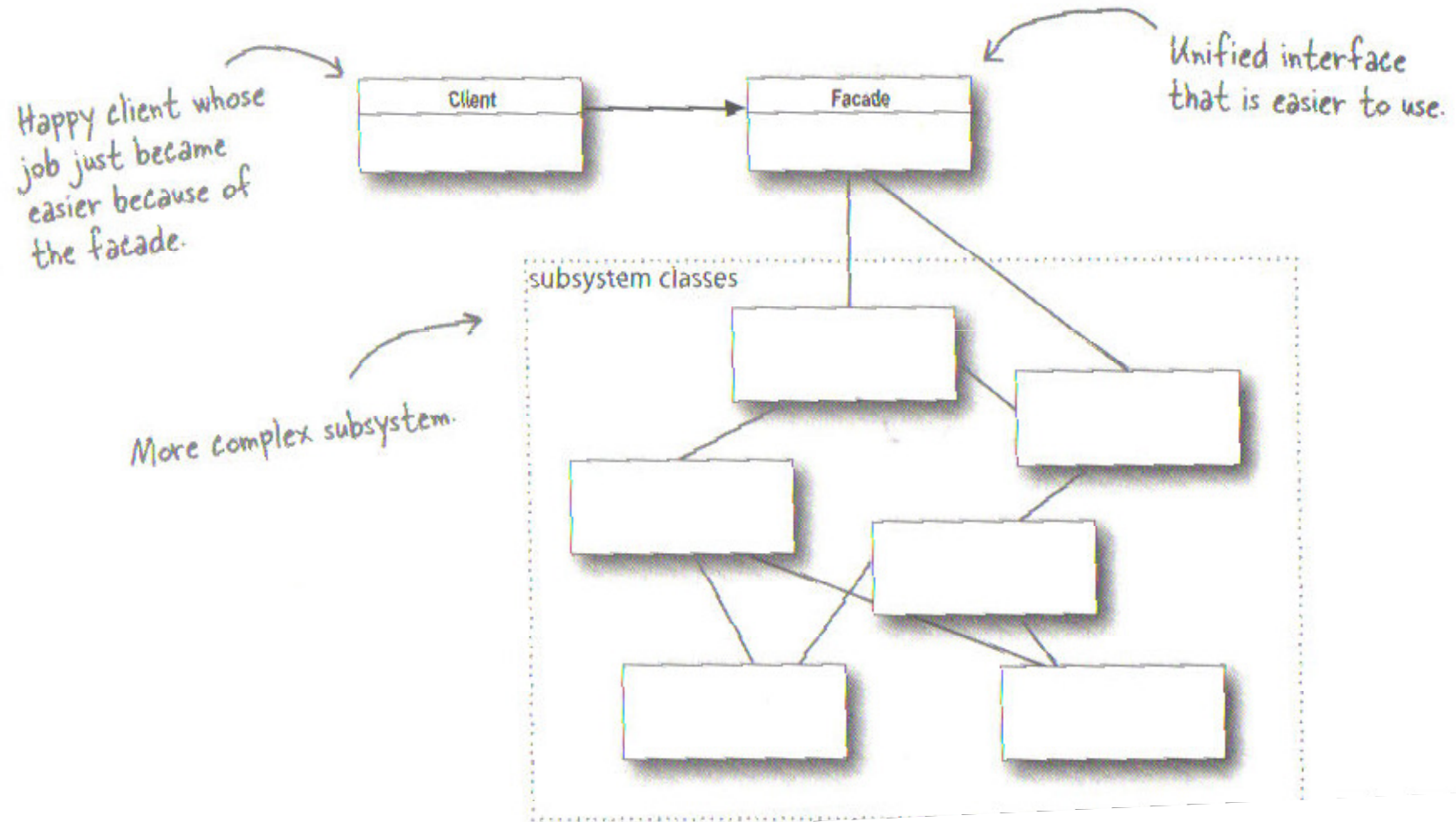
# Façade Pattern defined

---

**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.**

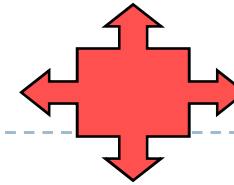


# Façade pattern – Class Diagram



# Design Principle

---



## Principle of Least Knowledge

talk only to your immediate friends

Basically this says minimize your dependencies

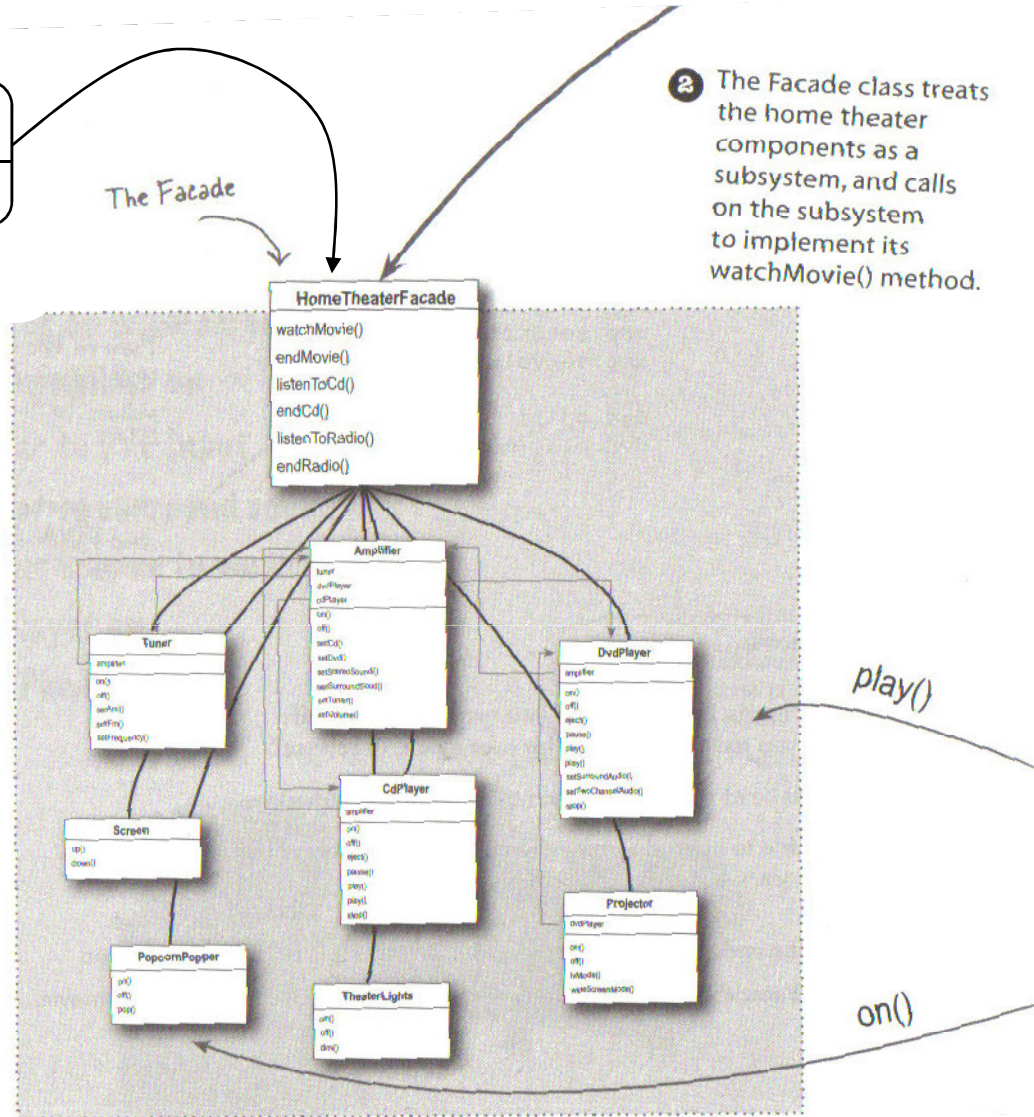




**Client**

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.



# A little comparison

---

