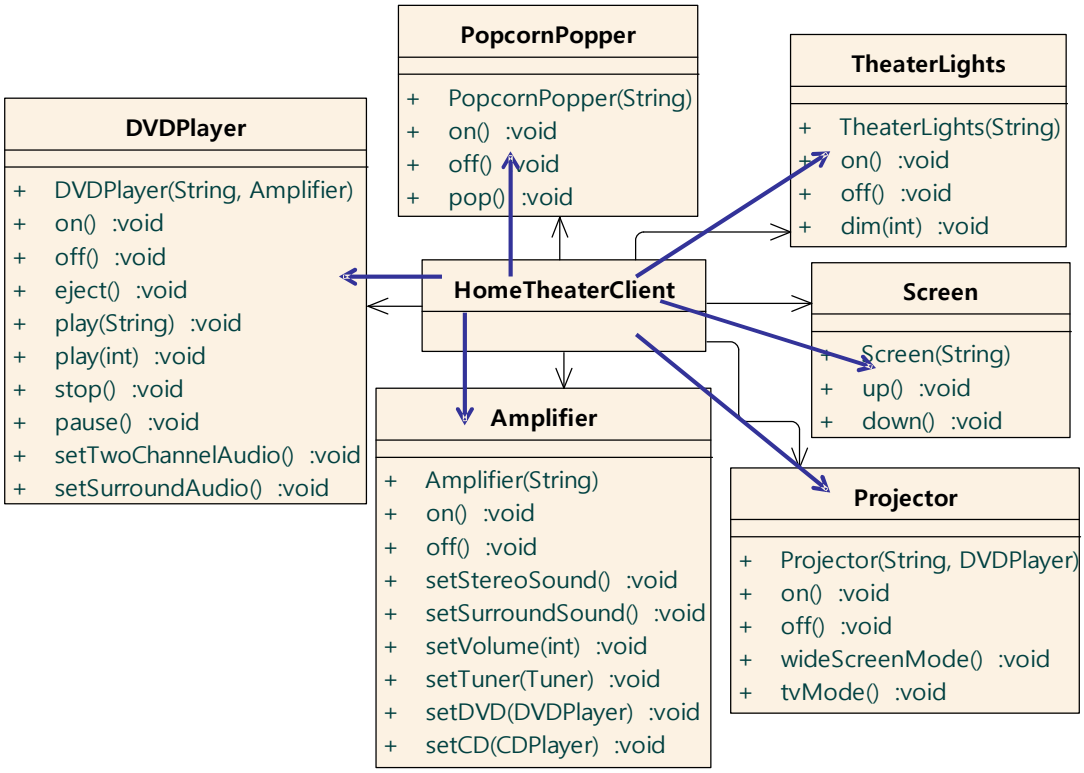
# Façade pattern

## Motivating Example – Home Theater

- client needs to control many components



# Home Theater Client Source Code

```
class HomeTheaterClient {
  Amplifier amp = new Amplifier("Top-O-Line Amplifier");
  DVDPlayer dvd = new DVDPlayer("Top-O-Line DVD Player", amp);
  Projector projector = new Projector("Top-O-Line Projector", dvd);
  TheaterLights lights = new TheaterLights("Theater Ceiling Lights");
  Screen screen = new Screen("Theater Screen");
  PopcornPopper popper = new PopcornPopper("Popcorn Popper");
  public void watchMovie(String movie) {
    popper.on(); popper.pop();
    lights.dim(10);
    screen.down();
    projector.on();
    ...
    amp.setVolume(5);
    dvd.on(); dvd.play(movie);
  }
  public void endMovie() {
    popper.off();
    lights.on();
    screen.up();
    projector.off();
    ...
    dvd.eject(); dvd.off();
  }
}
```

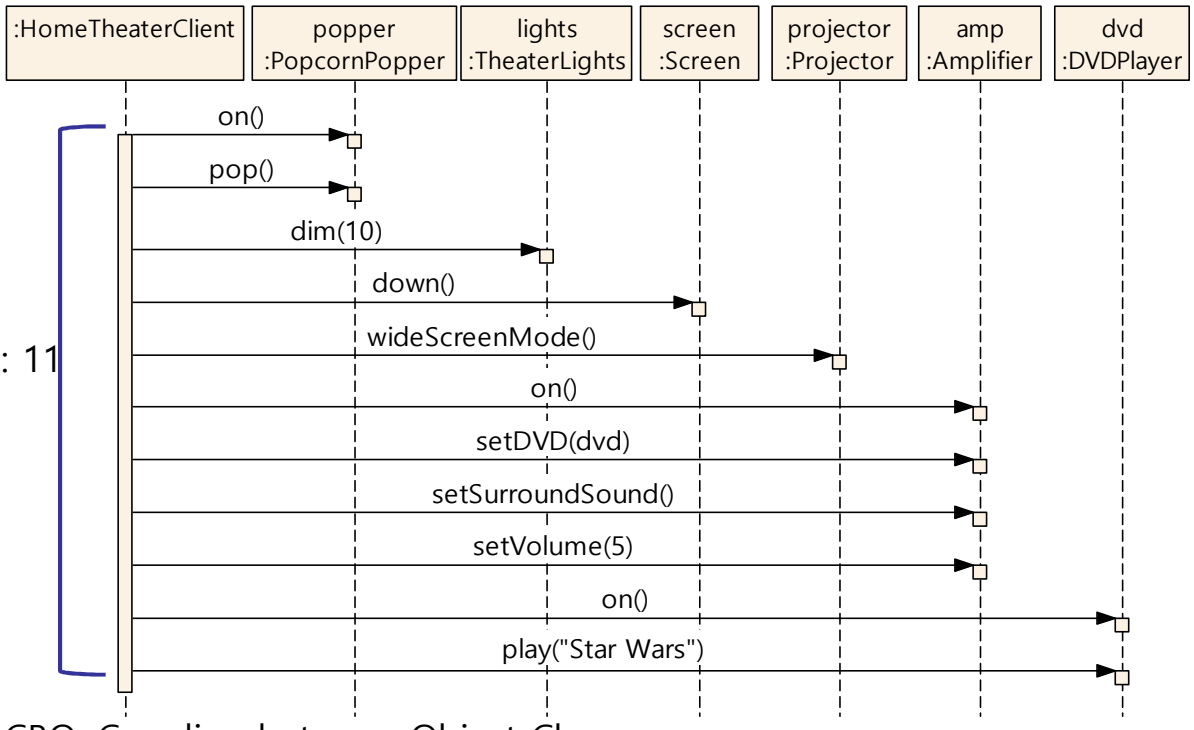
Need to interact with many components

Need to interact with many components



# Home Theater Client Design

- To watch movie

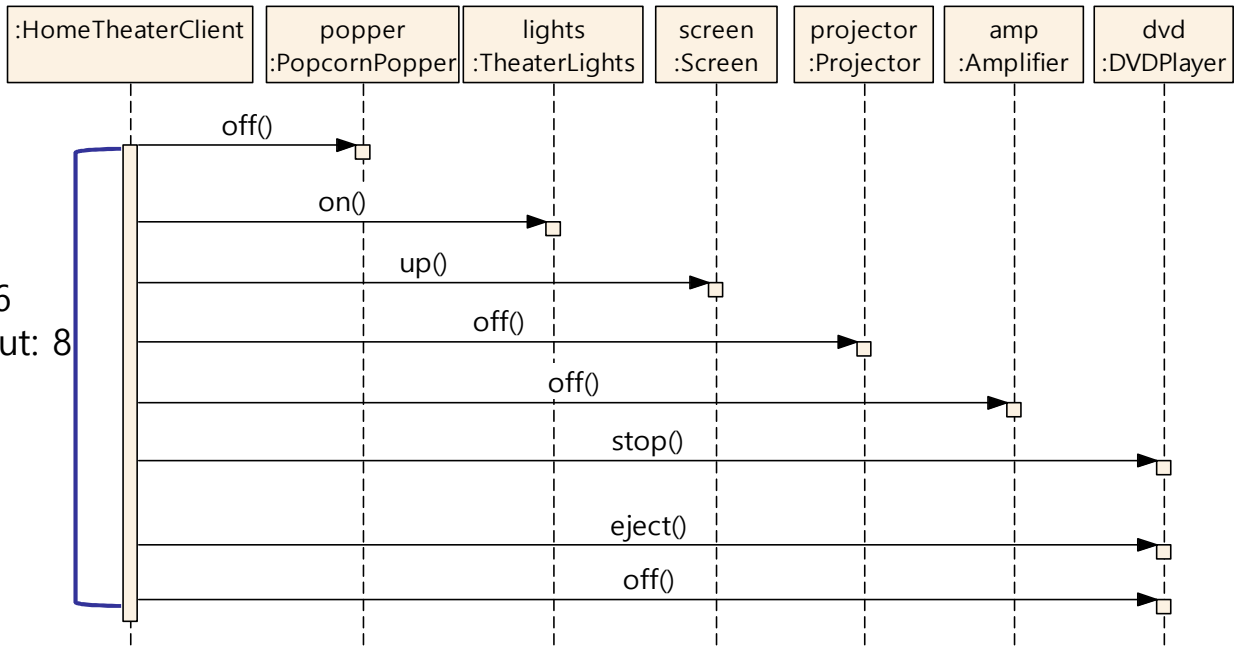



- CBO: 6
- Fan-Out: 11

CBO: Coupling between Object Classes

# Home Theater Client Design

- To end movie

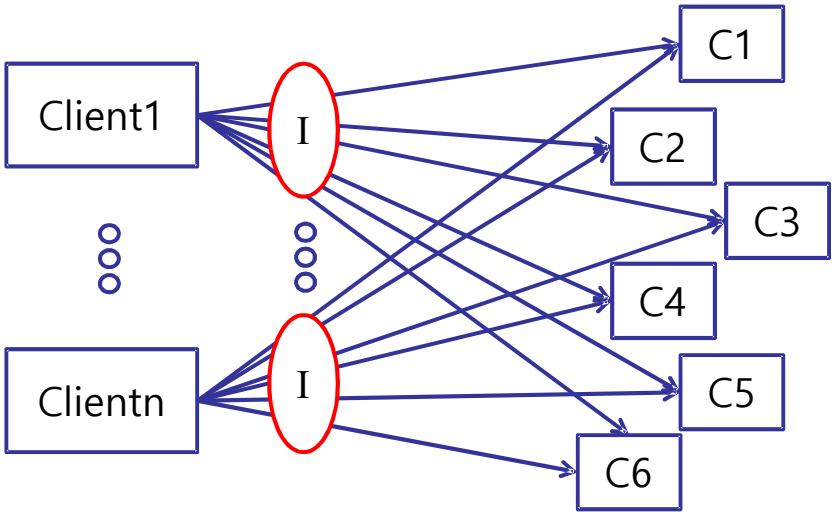



- CBO: 6
- Fan-Out: 8



# Problems with Home Theater

- Client has complex interaction with many components
- The code for interaction can be duplicated and will not be reused
- Thus, the interaction is not easy to extend



# Problems with Home Theater

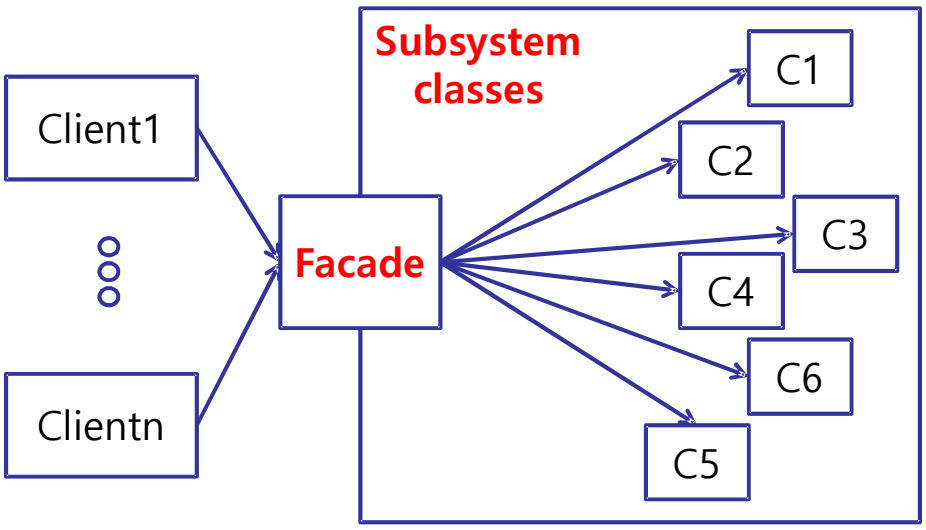
```
class RemoteController {
  void watchMovie(String movie) {
    popper.on();
    popper.pop();
    lights.dim(10);
    screen.down();
    projector.on();
    ...
    amp.setVolume(5);
    dvd.on();
    dvd.play(movie);
  }
  void endMovie() {
    popper.off();
    lights.on();
    screen.up();
    projector.off();
    ...
    dvd.eject();
    dvd.off();
  }
}
```

```
class HomeTheaterApp {
  void watchMovie(String movie) {
    popper.on();
    popper.pop();
    lights.dim(10);
    screen.down();
    projector.on();
    ...
    amp.setVolume(5);
    dvd.on();
    dvd.play(movie);
  }
void endMovie() {
    popper.off();
    lights.on();
    screen.up();
    projector.off();
    ...
    dvd.eject();
    dvd.off();
  }
}
```

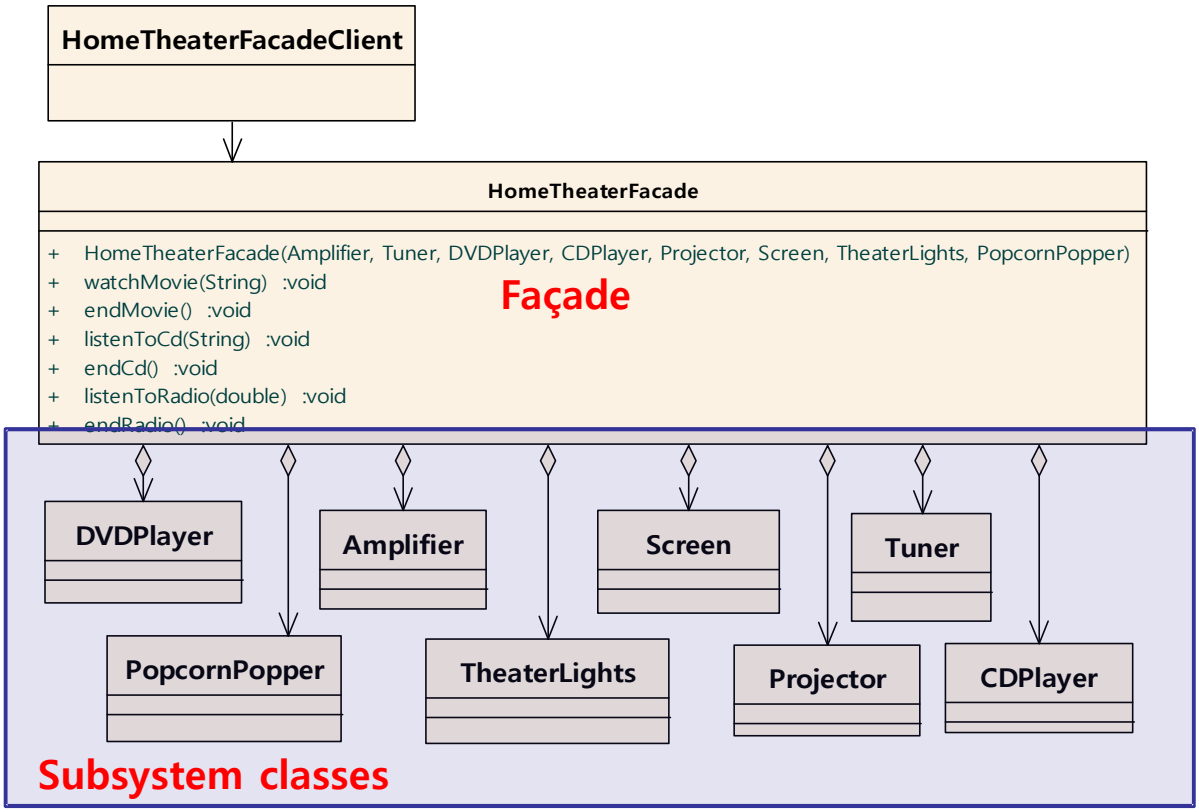


# Solution – Façade Pattern

- Facade defines a higher-level interface that makes the subsystem easier to use.
- Wrap a complicated subsystem with a simpler interface.



# Home Theater with Façade






# Home Theater Façade Source Code

```
public class HomeTheaterFacade {
  private Amplifier amp;
  private Tuner tuner;
  private DVDPlayer dvd;
  private CDPlayer cd;
  private Projector projector;
  private TheaterLights lights;
  private Screen screen;
  private PopcornPopper popper;

  public HomeTheaterFacade(Amplifier amp, Tuner tuner, DVDPlayer dvd,
         CDPlayer cd, Projector projector, Screen screen, TheaterLights lights,
          PopcornPopper popper) {
    this.amp = amp;
    this.tuner = tuner;
    this.dvd = dvd;
    this.cd = cd;
    this.projector = projector;
    this.screen = screen;
    this.lights = lights;
    this.popper = popper;
  }
```

# Home Theater Façade Source Code

```
public void watchMovie(String movie) {
  System.out.println("Get ready to watch a movie...");
  popper.on(); popper.pop();
  lights.dim(10);
  screen.down();
  projector.on(); projector.wideScreenMode();
  amp.on(); amp.setDvd(dvd); amp.setSurroundSound(); amp.setVolume(5);
  dvd.on(); dvd.play(movie);
}
public void endMovie() {
  System.out.println("Shutting movie theater down...");
  popper.off();
  lights.on();
  screen.up();
  projector.off();
  amp.off();
  dvd.stop(); dvd.eject();  dvd.off();
}
```

Façade implements high-level interface by using subsystem classes



# Home Theater Façade Client Code

```
class HomeTheaterClient {
  Amplifier amp = ... ;
  DVDPlayer dvd = ... ;
  Projector projector = ... ;
  TheaterLights lights = ... ;
  Screen screen = ... ;
  PopcornPopper popper = ... ;
  public void watchMovie(String movie) {
    popper.on(); popper.pop();
    lights.dim(10);
    screen.down();
    projector.on();
    ...
    amp.setVolume(5);
    dvd.on(); dvd.play(movie);
  }
  public void endMovie() {
    popper.off();
    lights.on();
    screen.up();
    projector.off();
    ...
    dvd.eject(); dvd.off();
  }
}
```

```
class HomeTheaterClient {
  HomeTheaterFacade façade =
    new HomeTheaterFacade
      (amp, dvd, ...) ;

  public void watchMovie(String movie)
    façade.watchMove(movie) ;
  }

  public void endMovie() {
    façade.endMovie() ;
  }
}
```