Object Oriented Analysis and Design with Java

UE19CS353

Sindhu R Pai

Department of Computer Science and Engineering

Acknowledgements: Significant portions of the information in the slide sets presented through the course in the class, are extracted from the prescribed text books, information from the Internet. Since these are only intended for presentation for teaching within PESU, there was no explicit permission solicited. We would like to sincerely thank and acknowledge that the credit/rights remain with the original authors/creators only

PES UNIVERSITY ONLINE

UE19CS353: Object Oriented Analysis and Design with Java

Theory and Implementation: Facade pattern

Prof. Sindhu R Pai
Department of Computer Science and Engineering

Agenda

- What is façade?
- Why Façade?
- Pictorial Representation
- Applicability
- Advantages
- Known Uses
- Implementation
- References

What is facade?

Meaning:

- The principal front of a building, that faces on to a street or open space.

  "the house has a half-timbered façade"

- A deceptive outward appearance.

  "her flawless public façade masked private despair"

Facade in Software

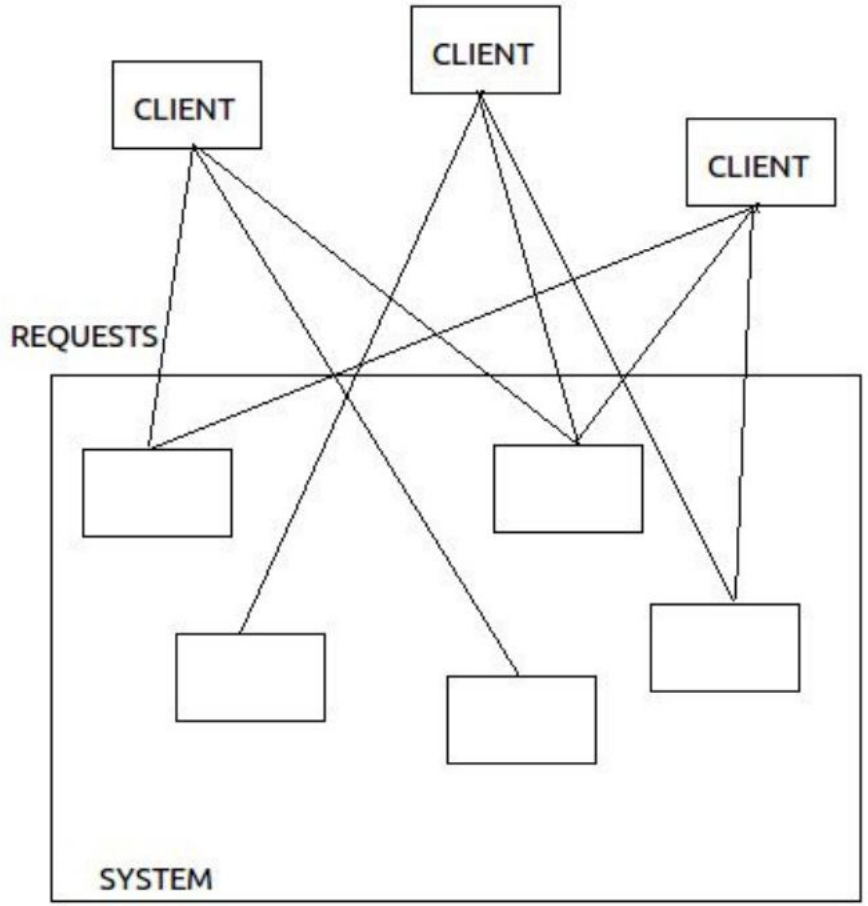
- An object that provides a **simplified interface to a larger body** of code, a library, a framework, or any other complex set of classes.
- The main intent of Façade is to provide a **unified interface to a set of interfaces in a subsystem**. Façade defines a **higher-level interface** that makes the subsystem easier to use.
- Example: fopen is an interface to open and read system commands.
- A structural design pattern, which **wraps a complicated subsystem with a simpler interface.**
- If the Facade is the only access point for the subsystem, it will limit the features and flexibility that **"power users**" may need

Why Facade?

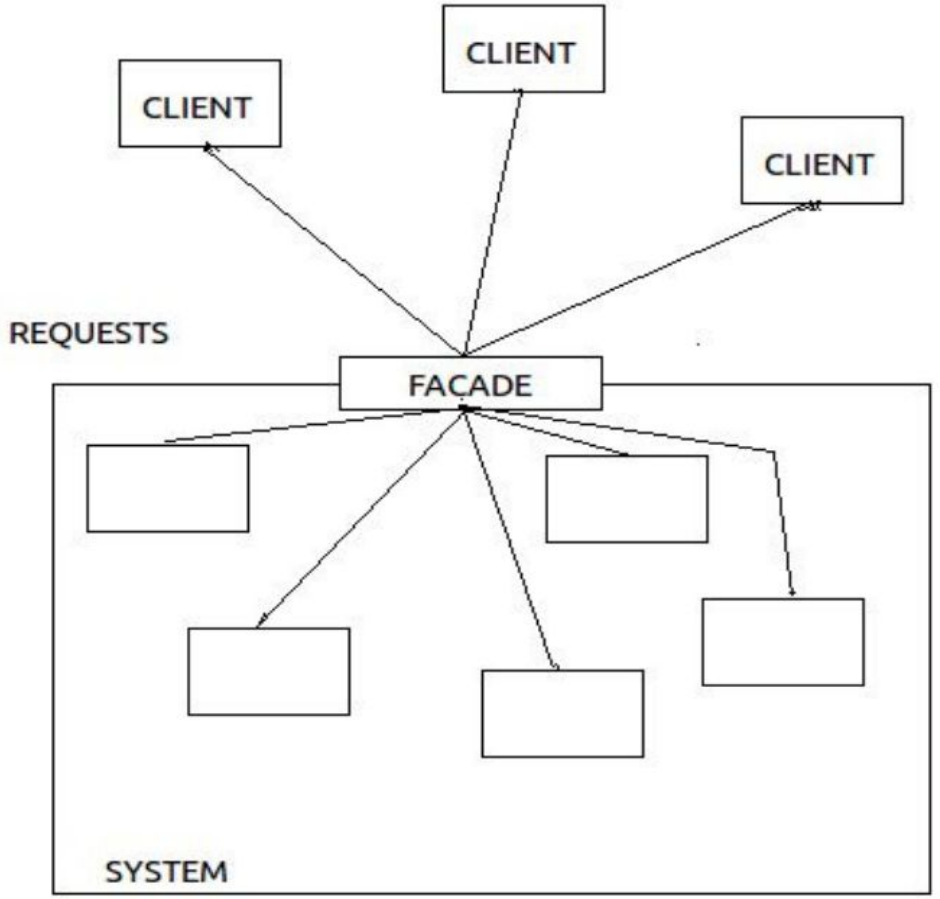
- Structuring a system into subsystems **helps reduce complexity**.

- A common design goal is to **minimize the communication and dependencies between subsystems**. One way to achieve this goal is to introduce a façade object that provides a single, simplified interface to the more general facilities of a subsystem.

Pictorial representation



Without facade



With facade

Applicability

- To provide a **simple interface to a complex subsystem**. A façade can provide **a simple default view of the subsystem** that is good enough for most clients.
- Introduce a façade to **decouple the subsystem from clients and other subsystems**, thereby promoting **subsystem independence and portability**.
- Use a façade to **define an entry point to each subsystem** level.
- To **simplify the dependencies between subsystems** by making them communicate with each other solely through their façades.

Advantages

- It **shields clients from subsystem components**, thereby reducing the number of objects that clients deal with and making the subsystem easier to use.
- It **promotes weak coupling between the subsystem and its clients**. Often the components in a subsystem are strongly coupled.
- This can **eliminate complex or circular dependencies**. This can be an important consequence when the client and the subsystem are implemented independently.
- Reducing compilation dependencies with façades **can limit the recompilation needed for a small change** in an important subsystem.

Known uses

- In the **ET++ application framework**, an application that can have built-in browsing tools for inspecting its objects at run-time. These browsing tools are implemented in a separate subsystem that includes a façade class called "ProgrammingEnvironment." This façade defines operations such as InspectObject and InspectClass for accessing the browsers.

- The **Choices operating system** uses façade to compose many frameworks into one. The key abstractions in Choices are processes, storage, and address spaces. For each of these abstractions there is a corresponding subsystem, implemented as a framework, that supports porting Choices to a variety of different hardware platforms.

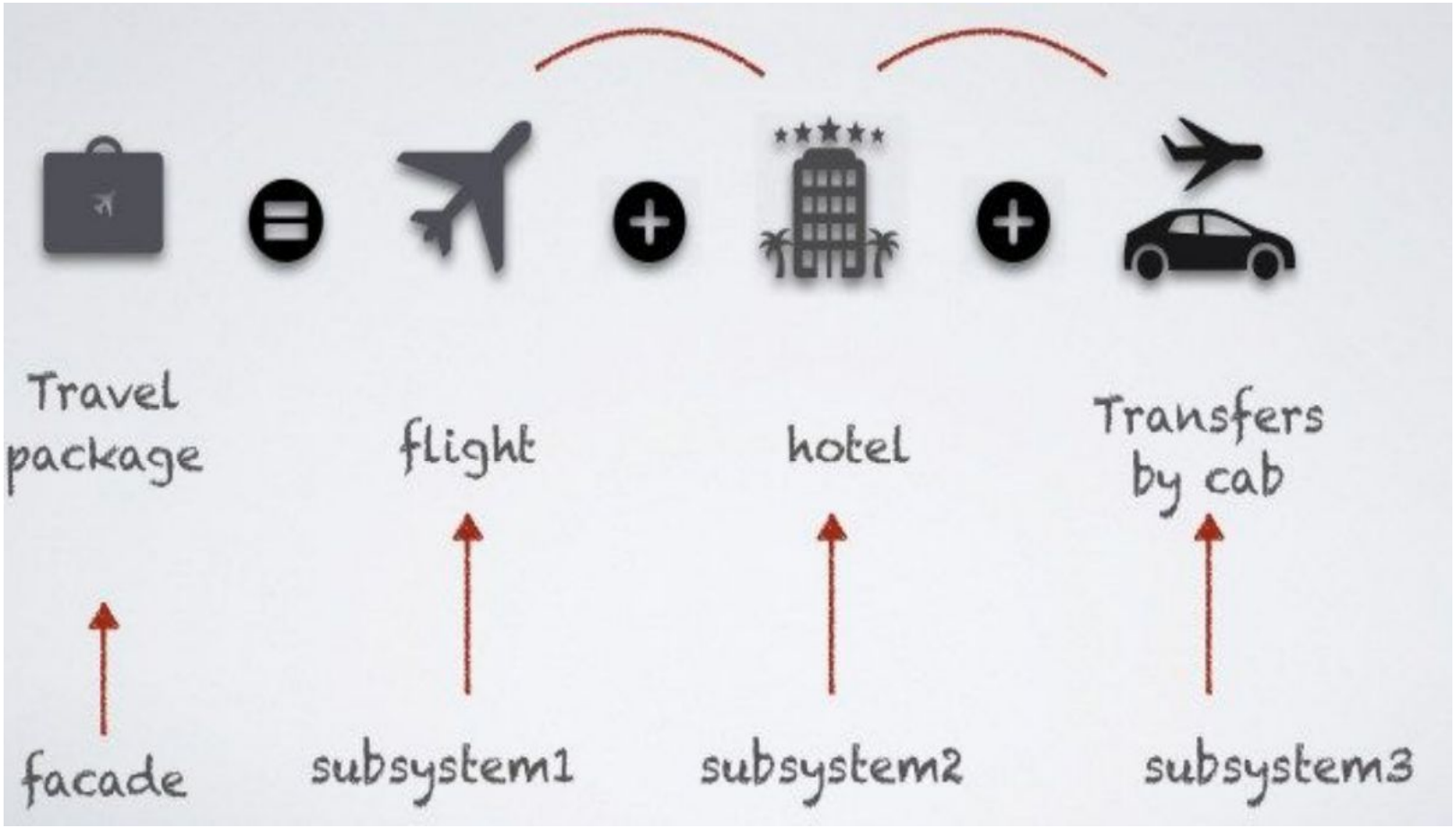
Implementation

- **Reduce client-subsystem coupling**. This can be done by simply **creating a "Façade Abstract Class" and concrete subclasses for the implementation of the subsystem**. Now the client class can communicate with the subsystem through the "**Abstract Façade Class**“.

  An alternative approach is to configure a façade object with different subsystem objects

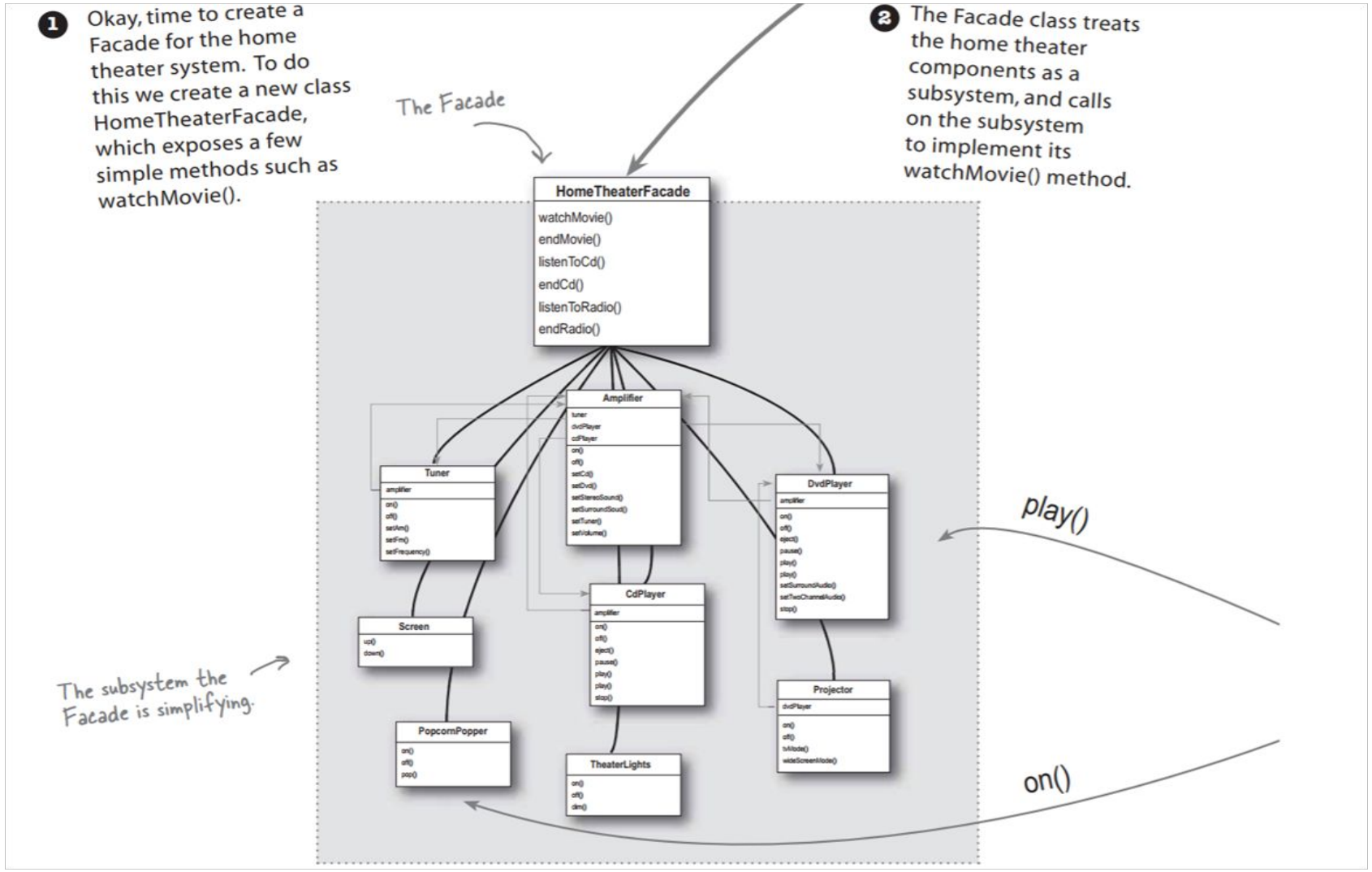
- Public versus Private Interfaces. Classes and Subsystems are similar. Both encapsulate something. Both have private and public interfaces. The **public interface consists of classes accessible by all Clients** whereas the private interface is only for subsystem extenders. **Façade is part of the public interface**

Example -1: Booking a package



Booking a package

Example -2: Designing Hometheater



References

- Facade Pattern from Head First Design Patterns (javaguides.net)
- Facade Design Pattern (sourcemaking.com)
- https://refactoring.guru/design-patterns/java

THANK YOU

Prof. Sindhu R Pai

Department of Computer Science and Engineering

sindhurpai@pes.edu