Façade

Exposing several components through a single interface.

By Alexander Tilkin

Motivation

- Balancing complexity and presentation/usability
- Typical home
  - Many subsystems (electrical, sanitation, water)
  - Complex internal structure (e.g., floor layers)
  - End users is not exposed to internals
- Same with software
  - Many systems working to provide flexibility, but
  - API consumers want it to 'just work'

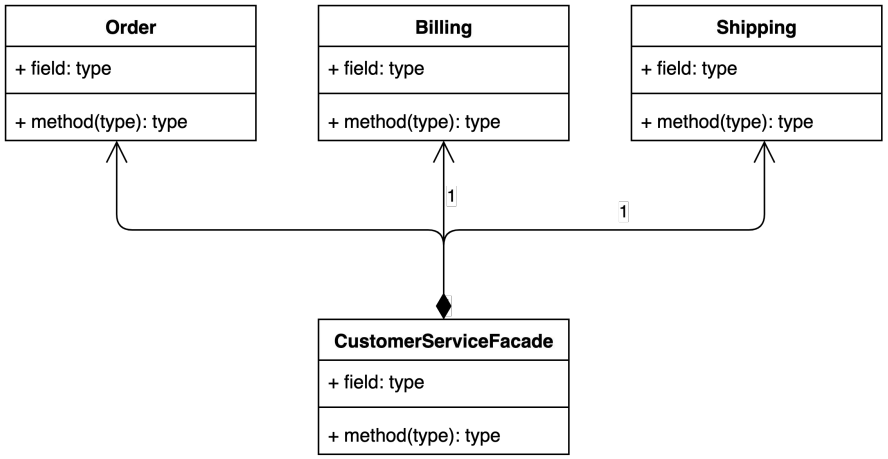
Façade

Provides a simple, easy to understand/user interface over a large and sophisticated body of code.

units.unit_10_facade.FacadeExample.java

Façade Example

- We have three classes: Order, Billing, and Shipping
- Each responsible for a different part of data
- To get the complete information the user needs to query all three applications
- This will make it a bit complicated code
- To ease the use we create a class name CustomerServiceFacade
- This class aggregates the instances of the three classes and exposes interface to the user
- Using this approach the user gets a simple and short code of use
- This design pattern is very useful for use between different layer in the application



Summary

- Build a Façade to provide a simplified API over a set of classes
- May wish to (optionally) expose internals through the façade
- Many allow user to ‘escalate’ to use more complex APIs if they need to