Design Patterns

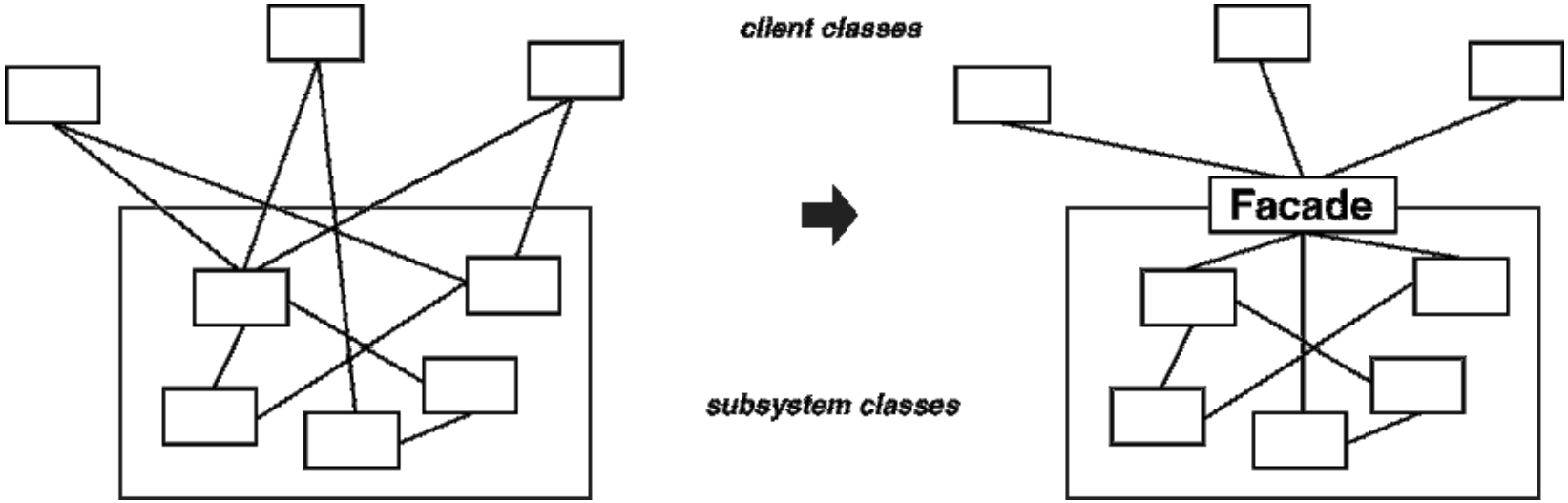
宋 杰
Song Jie
东北大学 软件学院
Software College, Northeastern University

9. Facade Pattern

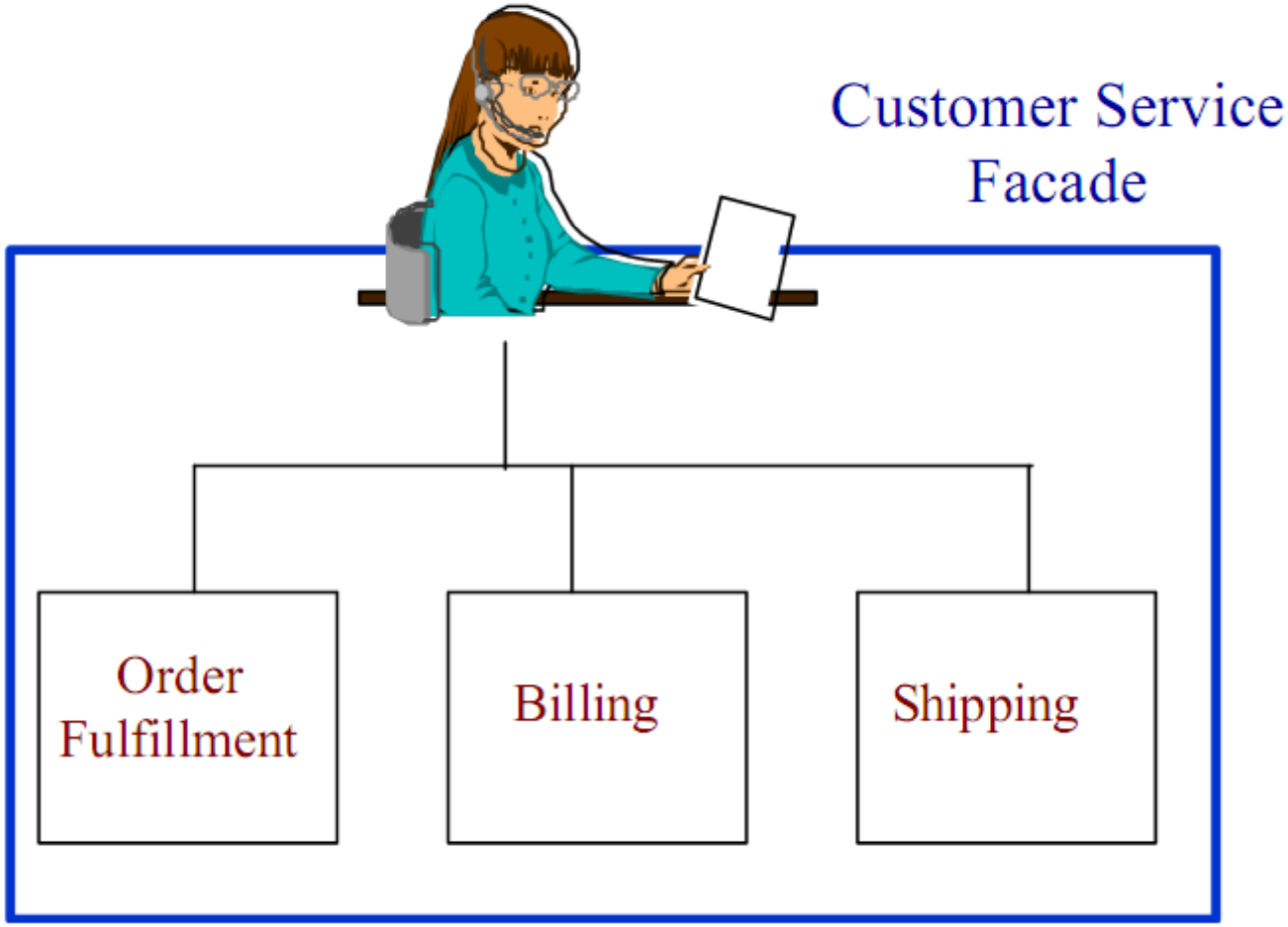
Intent

- Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use.
- 门面模式或外观模式

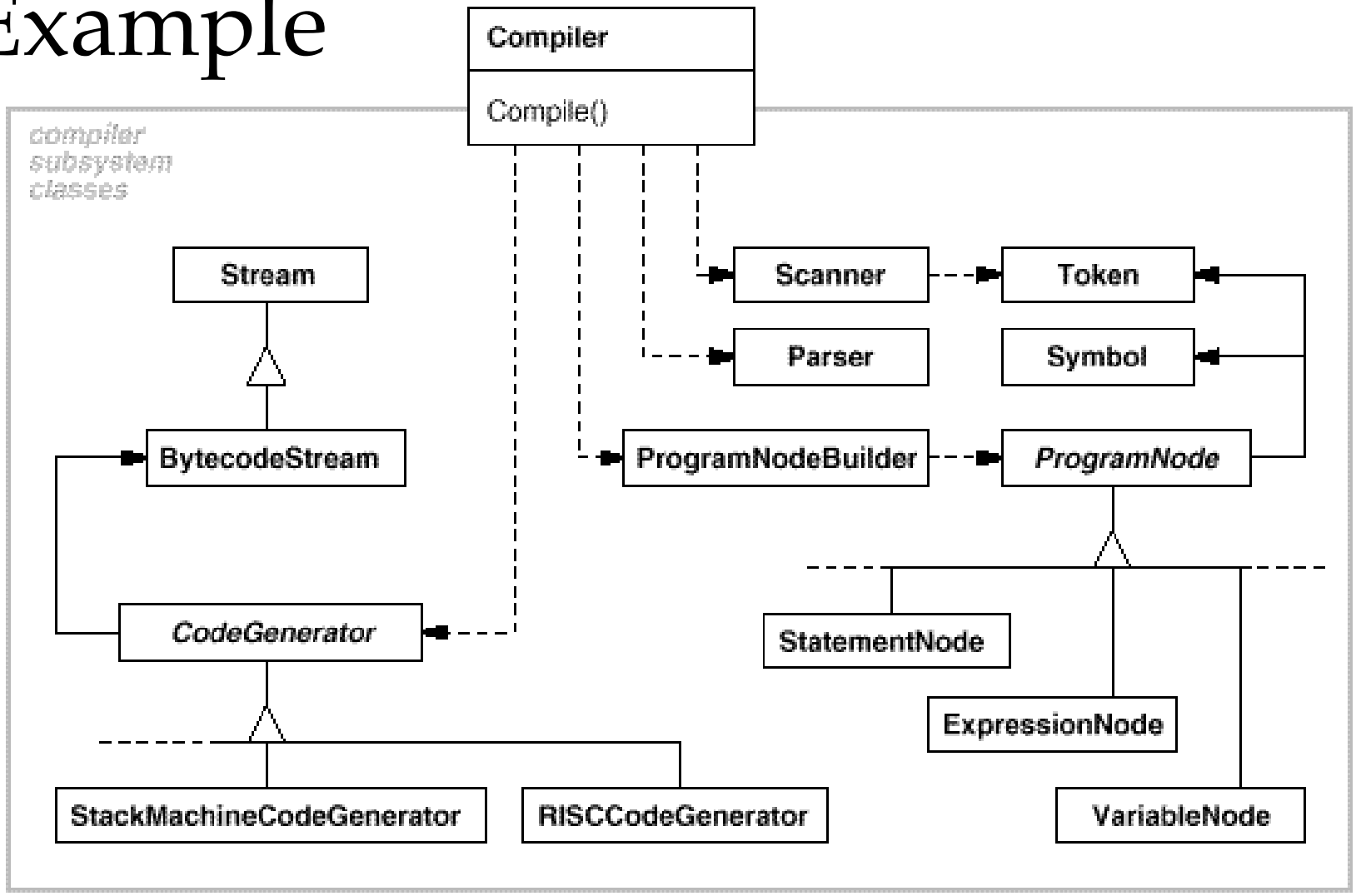
- Structuring a system into subsystems helps reduce complexity.
- A common design goal is to minimize the communication and dependencies between subsystems.
- One way to achieve this goal is to introduce a facade object that provides a single, simplified interface to the more general facilities of a subsystem.



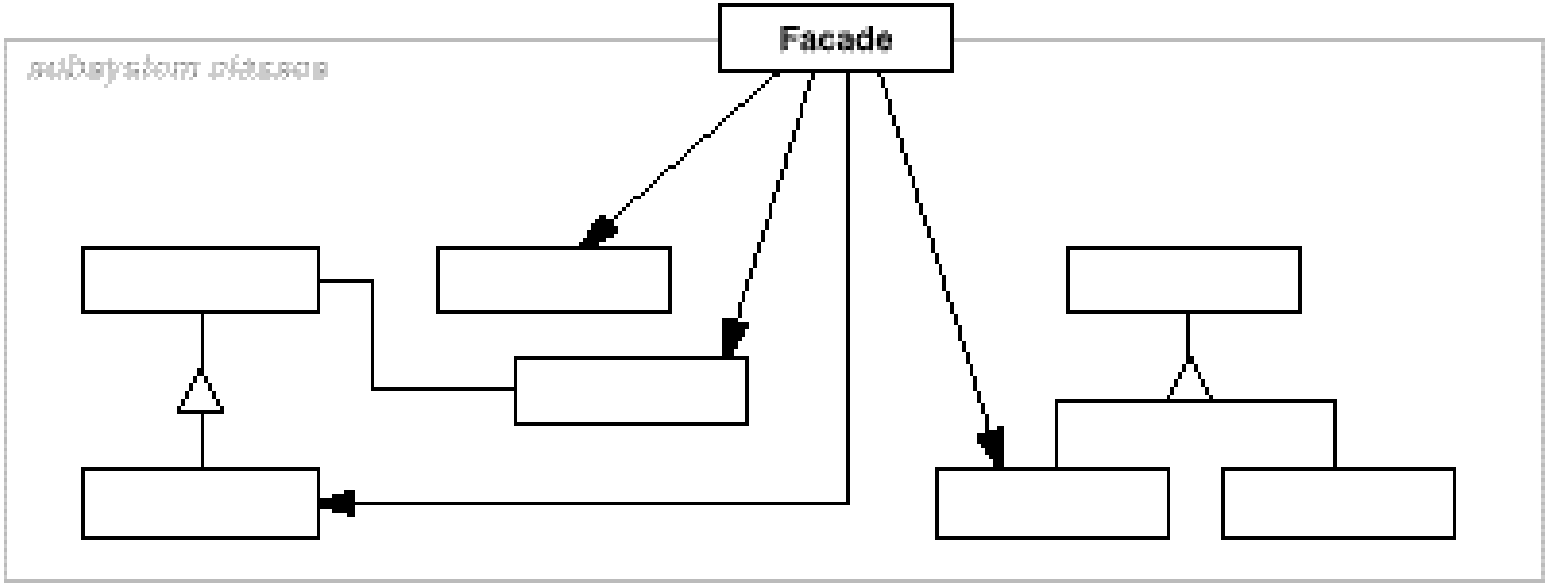
Example


Customer Service
Facade
Order
Fulfillment
Billing
Shipping

Example


Compiler
Compile()
compiler subsystem classes
Stream
Scanner
Token
Parser
Symbol
BytecodeStream
ProgramNodeBuilder
ProgramNode
CodeGenerator
StatementNode
ExpressionNode
StackMachineCodeGenerator
RISCCodeGenerator
VariableNode

Structure



Participants

- Facade
  - Knows which subsystem classes are responsible for a request.
  - Delegates client requests to appropriate subsystem objects.
- Subsystem classes
  - Implement subsystem functionality.
  - Handle work assigned by the Facade object.
  - Have no knowledge of the facade; that is, they keep no references to it.

Consequences

- It shields clients from subsystem components.
- It promotes weak coupling between the subsystem and its clients.
- It doesn't prevent applications from using subsystem classes if they need to.

Applicability

- You want to provide a simple interface to a complex subsystem.
  - Subsystems often get more complex as they evolve(演化).
  - Most patterns, when applied, result in more and smaller classes.
    - This makes the subsystem more reusable and easier to customize,
    - But it also becomes harder to use for clients that don't need to customize it.
  - A facade can provide a simple default view of the subsystem that is good enough for most clients.
  - Only clients needing more customizability will need to look beyond the facade.

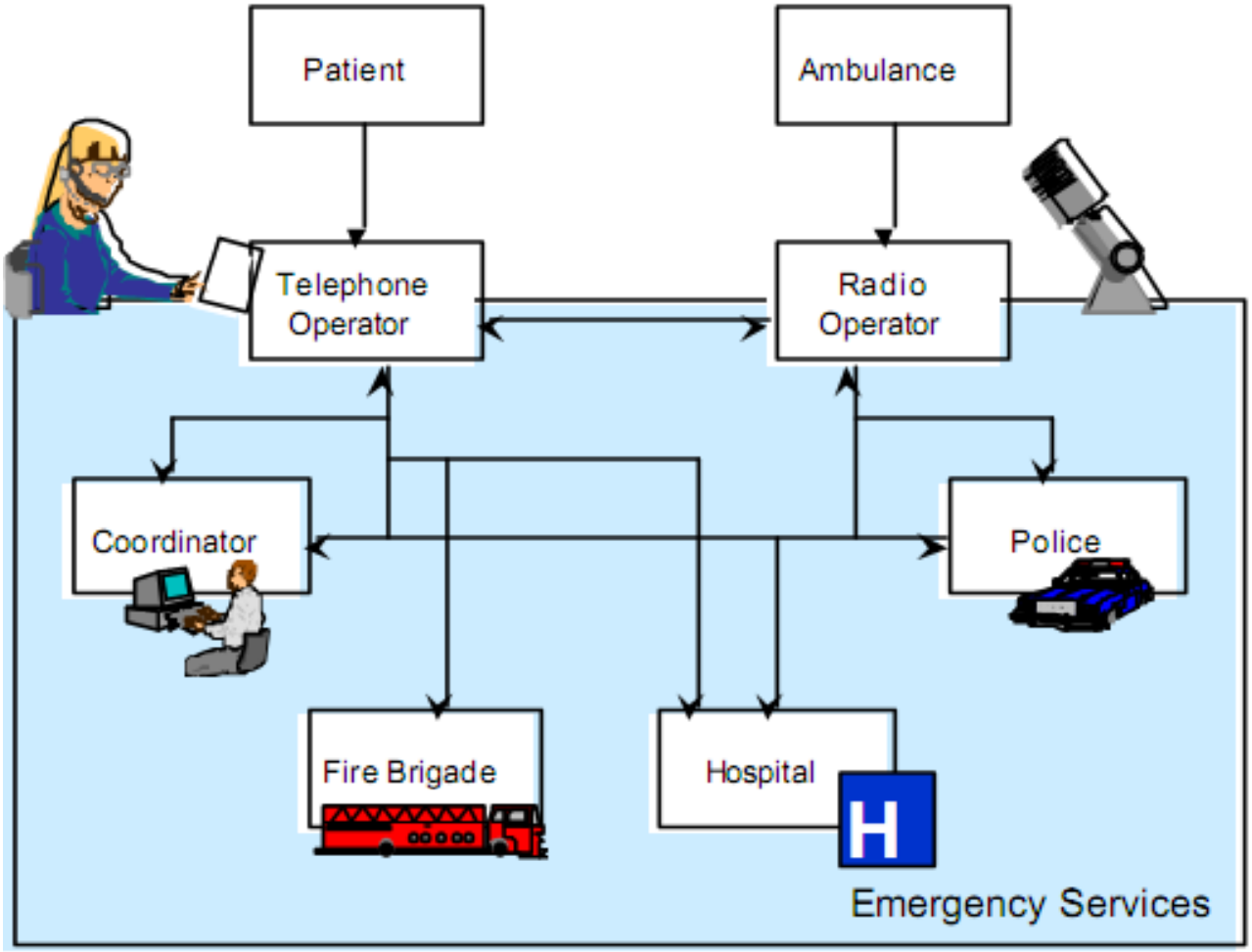
Applicability

- There are many dependencies between clients and implementation.
  - Introduce a facade to decouple the subsystem from clients and other subsystems, thereby promoting subsystem independence and portability.
- You want to layer your subsystems.
  - Use a facade to define an entry point to each subsystem level, simplify the dependencies between them by making them communicate with each other solely through their facades.

Implementation 1 Multiple Facades

- In general, a subsystem only need one facade.
- But in some situation, the multiple facades is meaningful, it should also be considered.

Example


Patient
Ambulance
Telephone Operator
Radio Operator
Coordinator
Police
Fire Brigade
Hospital
H
Emergency Services

Implementation 2 There is not new behaviors defined in Facades

- DO NOT introduce new behaviors to the facade.
- If an facade can not providing the required behaviors:
  - If such behaviors is implemented by subsystem, then extend the facade.
  - If such behaviors is not implemented by subsystem yet, the extend the subsystem and facade both.

Variation: Abstract Facade

- Making Facade an abstract class with concrete subclasses for different implementations of a subsystem. This abstract coupling keeps clients from knowing which implementation of a subsystem is used.
- An alternative to subclassing is to configure a Facade object with different subsystem objects. To customize the facade, simply replace one or more of its subsystem objects.

Extension: public subsystem and private subsystem

- A subsystem is analogous to a class in that both have interfaces (NOT Java interface here), and both encapsulate something
  - A class encapsulates state and operation
  - A subsystem encapsulates classes.
- Think about the public and private interface of a class, AND
- Think about the public and private interface of a subsystem.
  - The public interface to a subsystem consists of classes that all clients can access;
  - The private interface is just for subsystem extenders.
- The Facade class is part of the public interface, of course, but it's not the only part. Other subsystem classes are usually public as well.

Let’s go to next…