SHRI RAMDEOBABA COLLEGE OF
ENGINEERING AND MANAGEMENT,
NAGPUR - 440013

DESIGN PATTERNS
V SEMESTER

COURSE COORDINATOR: RINA DAMDOO

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FAÇADE DESIGN PATTERNS

- **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.

OBJECTIVE

- 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.

FAÇADE DESIGN PATTERN

Problem Statement:

Implement a travel app such that a customer can ask an Agent (like MakeMyTrip, Expedia) to provide details of Flights and Hotels available during couple of dates.

Use FAÇADE DESIGN PATTERN

FAÇADE DESIGN PATTERN

Banking System (Loan Application)

- When you apply for a loan, internally the bank may check:
  - Your credit score
  - Account balance
  - Employment history
  - Legal verification

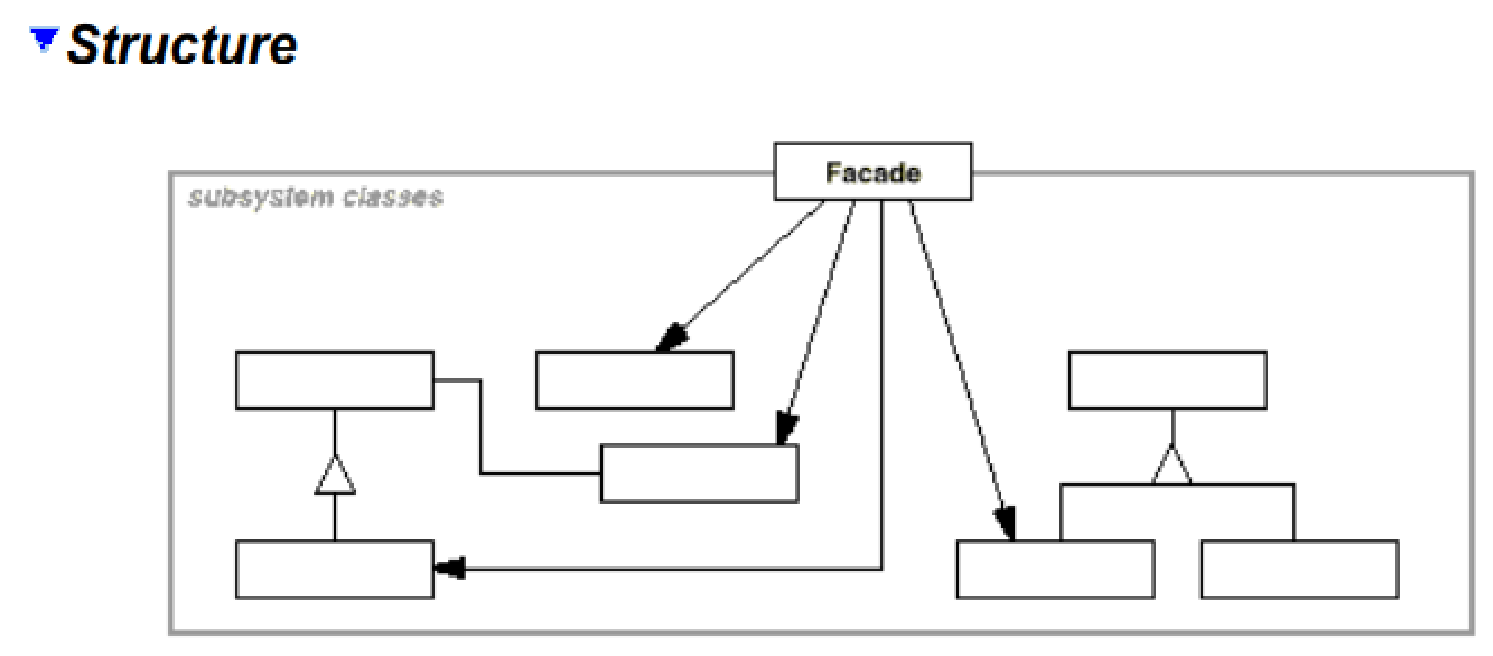
Smart Home Automation

- Turning on "Movie Mode" in a smart home may involve:
  - Closing curtains
  - Dimming lights
  - Turning on AC
  - Starting the projector

E-commerce Checkout

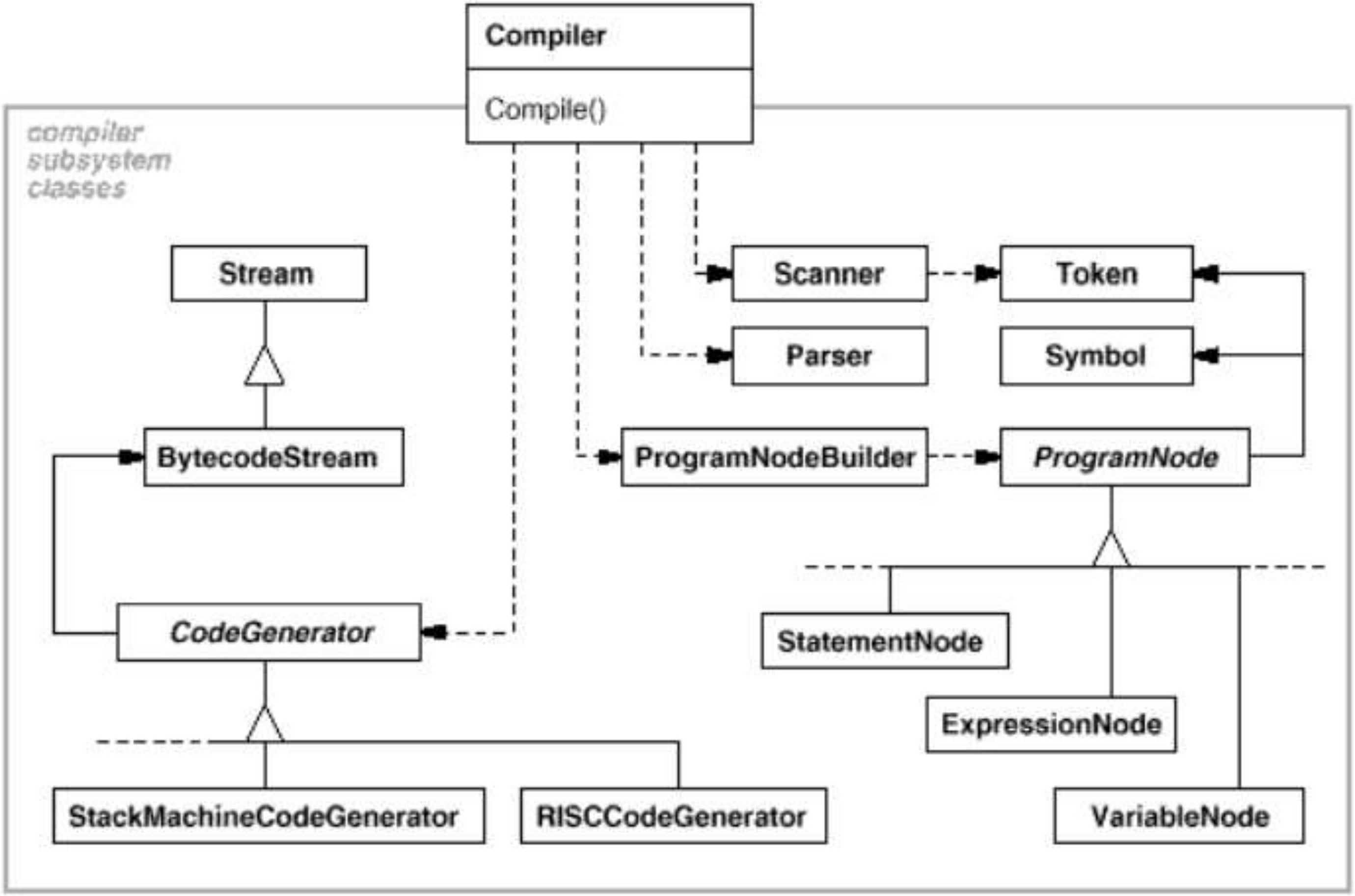
- A checkout process may involve:
  - Payment service
  - Inventory management
  - Invoice generation
  - Shipping service

FAÇADE DESIGN PATTERN

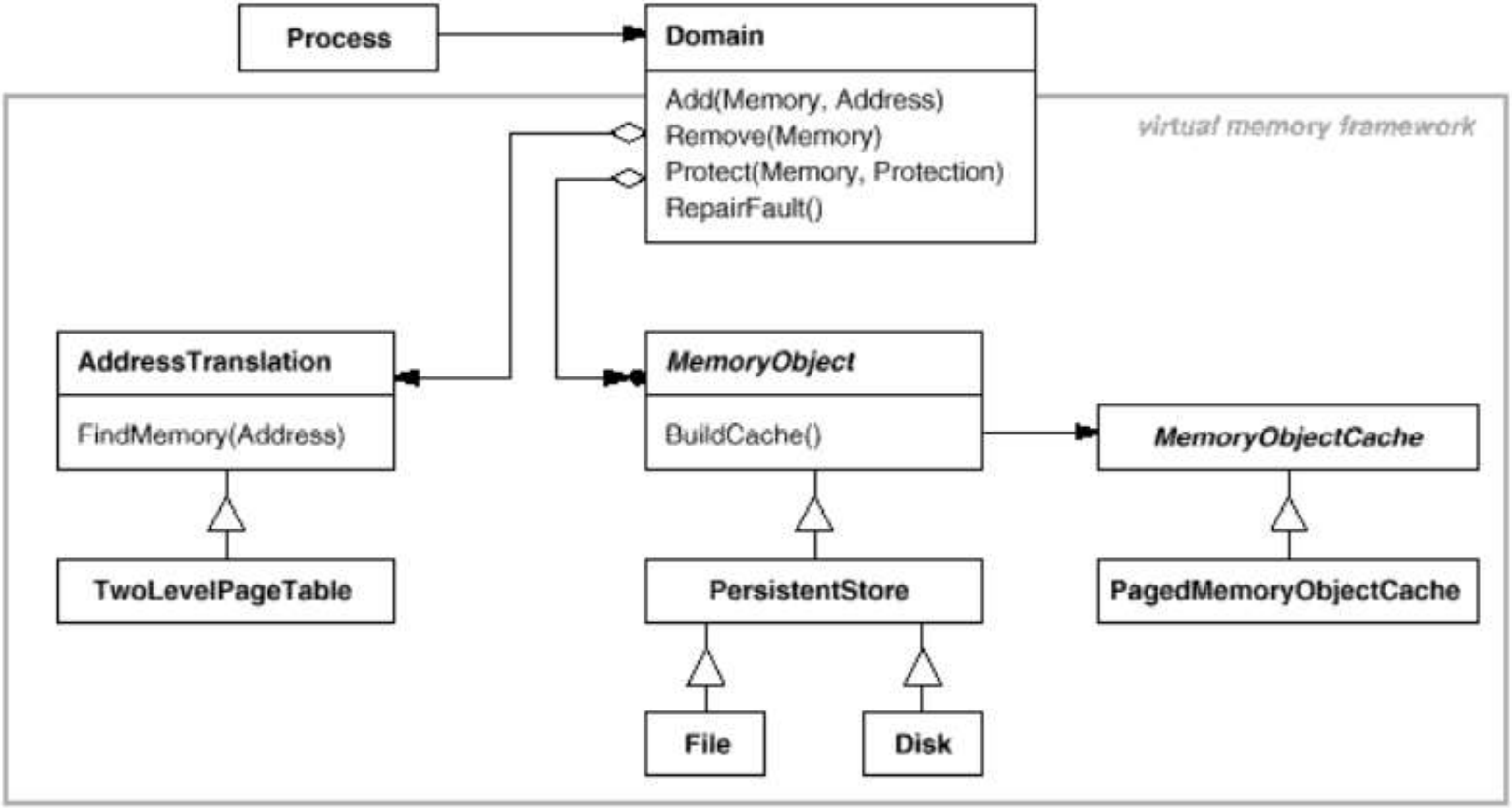


Structure

MOTIVATION


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

Another Example



APPLICABILITY

Use the Façade pattern when:

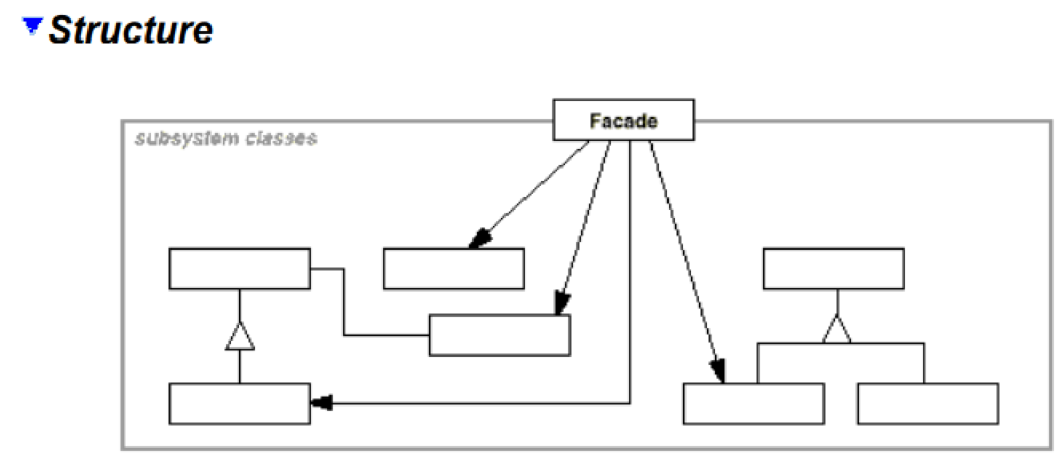
- Provide a simple interface to a complex subsystem.
- There are many Dependencies between clients and the implementation classes
- You want to layer your subsystems. Use a facade to define an entry point to each subsystem level. If subsystems are dependent, then you can simplify the dependencies between them by making them communicate with each other solely through their facades.

PARTICIPANTS

Participant	Responsibility
Facade	• Knows which subsystem classes are responsible for a request. o 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.

COLLABORATIONS

- Clients communicate with the subsystem by sending requests to Facade, which forwards them to the appropriate subsystem object(s). Although the subsystem objects perform the actual work, the facade may have to do work of its own to translate its interface to subsystem interfaces.
- Clients that use the facade don't have to access its subsystem objects directly.



CONSEQUENCES

- 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. Weak coupling lets you vary the components of the subsystem without affecting its clients. Facades help layer a system and the dependencies between objects. They can eliminate complex or circular dependencies. **This can be an important consequence when the client and the subsystem are implemented independently**.
- It doesn't prevent applications from using subsystem classes if they need to. Thus you can choose between ease of use and generality.

IMPLEMENTATION

Implementation Consider the following issues when applying the Facade pattern:

- Reducing client-subsystem coupling.
- Public versus private subsystem classes

KNOWN USES & RELATED PATTERNS

- **MemoryObject represents a data store.**
- **MemoryObjectCache caches the data of MemoryObjects in physical memory.**
- **MemoryObjectCache is actually a Strategy pattern that localizes the caching policy.**
- **AddressTranslation encapsulates the address translation hardware.**

- **Abstract Factory**
- **Mediator**