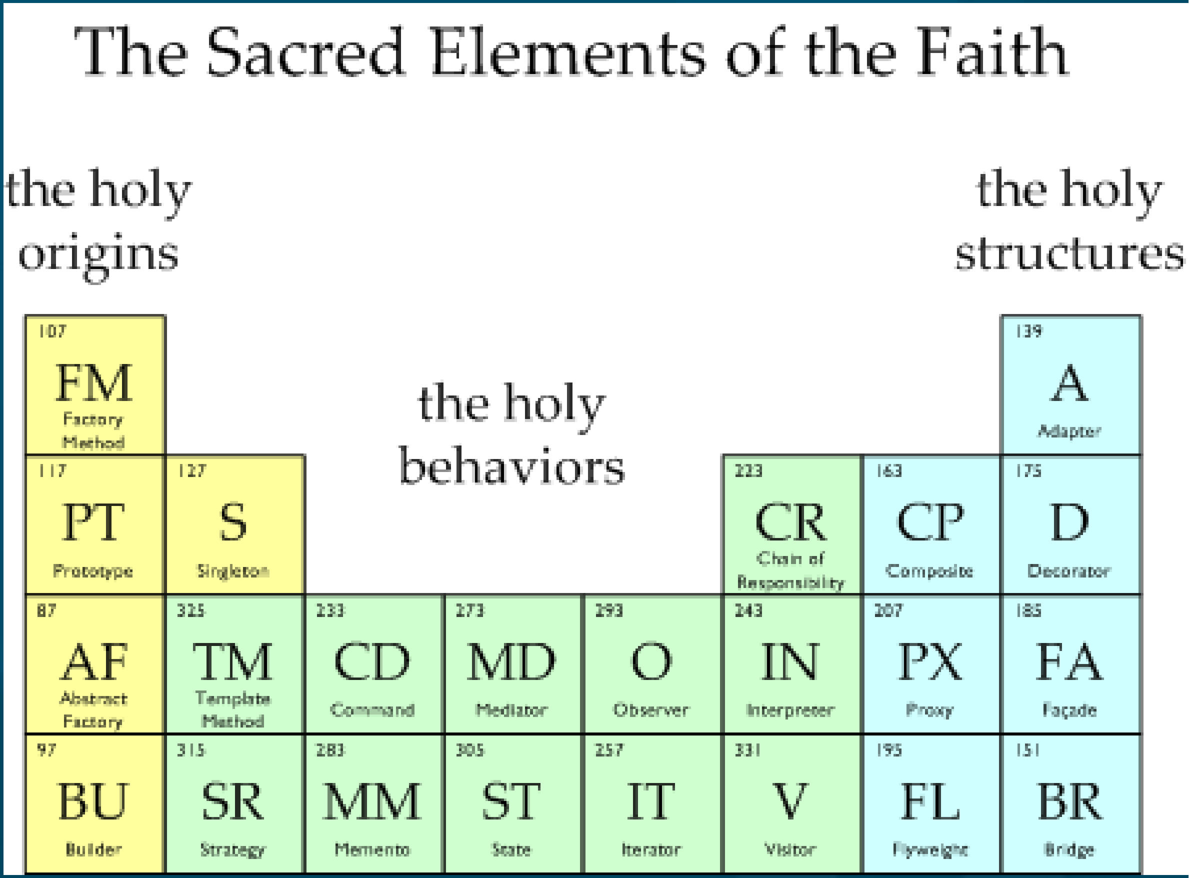
Design Patterns: Structural and Behavioural

9 April 2010
CMPT166
Dr. Sean Ho
Trinity Western University

See also:
Vince Huston,
JavaCamp

Design patterns (GoF)

- Reusable templates for designing programs
  May be very high-level, indep. of prog. language
- Creational patterns
  - Factory method
  - Abstract factory
  - Builder
  - Prototype
  - Singleton



The Sacred Elements of the Faith

the holy origins

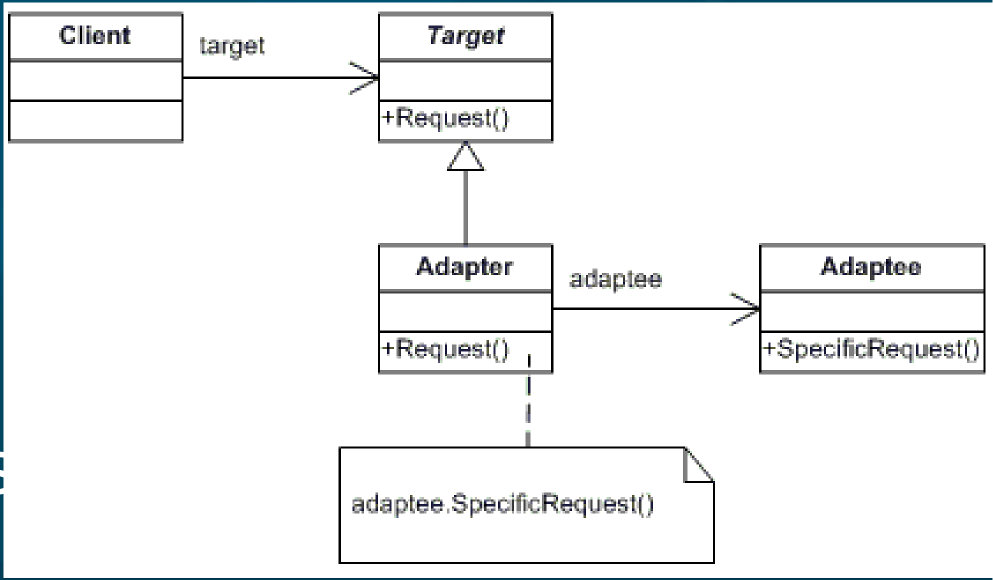
the holy behaviors

the holy structures

107 FM Factory Method							139 A Adapter
117 PT Prototype	127 S Singleton				223 CR Chain of Responsibility	163 CP Composite	175 D Decorator
87 AF Abstract Factory	325 TM Template Method	233 CD Command	273 MD Mediator	293 O Observer	243 IN Interpreter	207 PX Proxy	185 FA Façade
97 BU Builder	315 SR Strategy	283 MM Memento	305 ST State	257 IT Iterator	331 V Visitor	195 FL Flyweight	151 BR Bridge

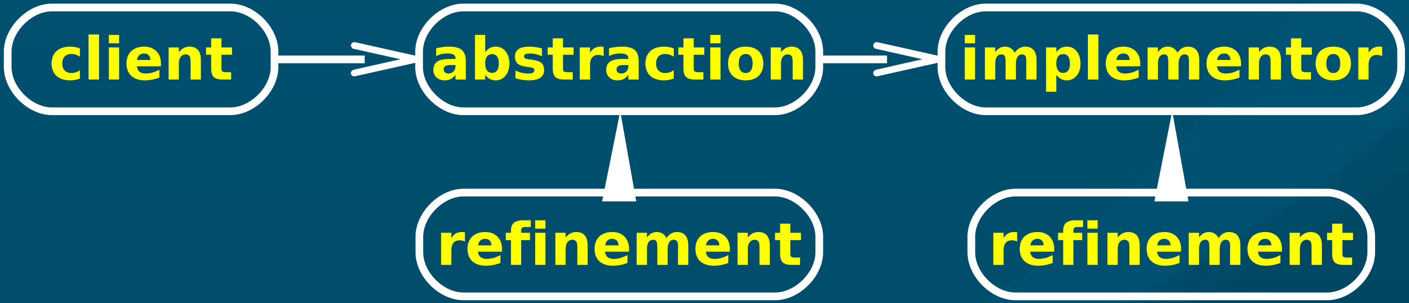
Structural pattern: Adapter

- Convert interface of a class so that two incompatible classes can work together
- Like converting 3-prong plug to 2-prong socket, or impedance matching electrical signals
- e.g., buy prepackaged software system, get it working with your existing system
- e.g., WindowAdapter provides empty implementations of all WindowListener methods



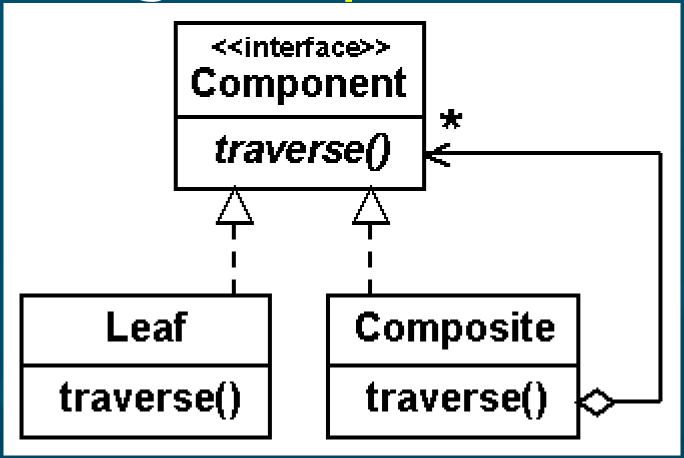
Structural pattern: Bridge

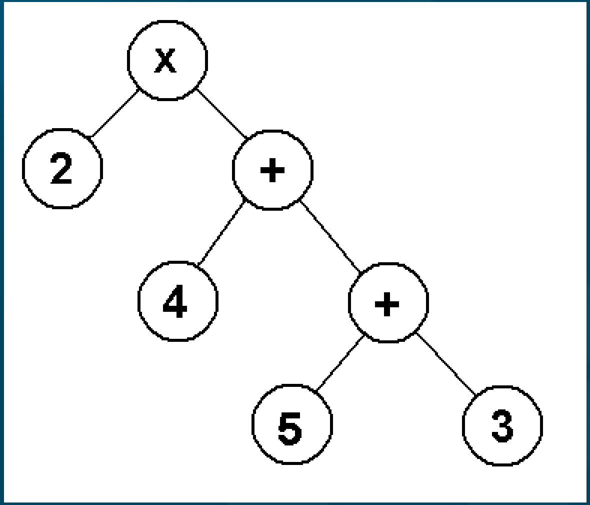
- Decouple an abstraction from its implementation so that the two can vary independently
- e.g., light switch abstract concept vs. implementation of kinds of switches



Structural pattern: Composite

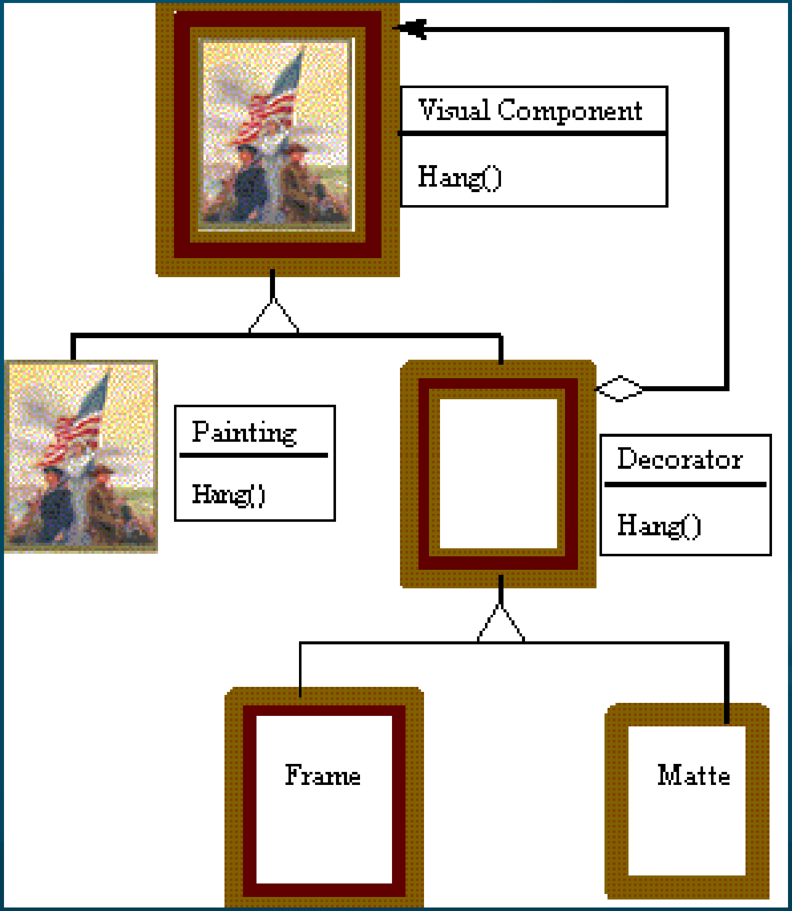
- Tree structure for objects: treat individual objects and composites in the same way
- e.g., file directories have entries, each of which may themselves be directories
- e.g., widgets and containers (Android Views)
- e.g., expression trees

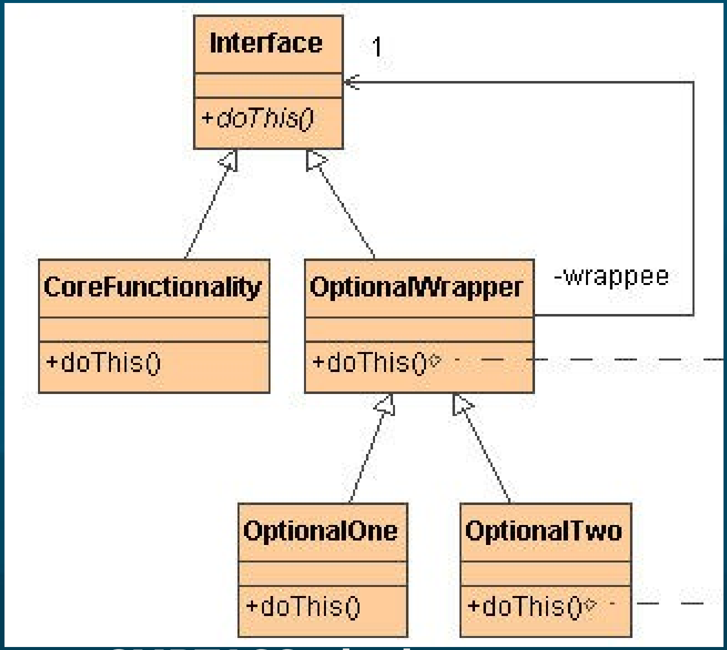




Structural pattern: Decorator

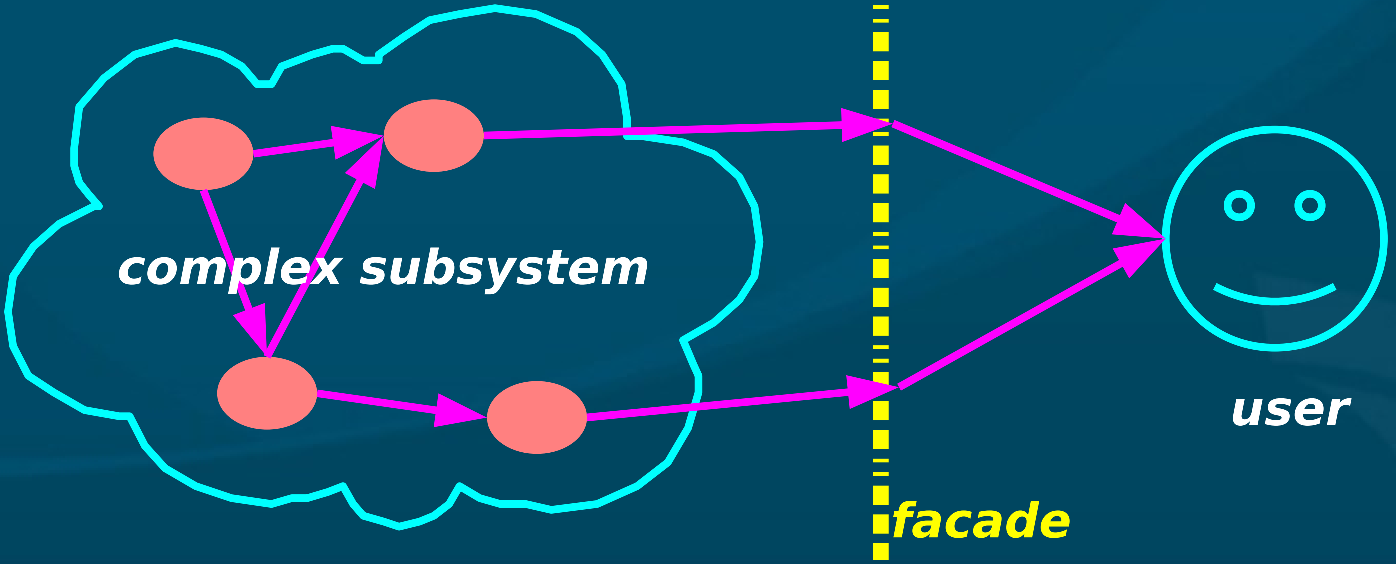
- Dynamically add functionality via a wrapper
  - More flexible than static subclassing
- e.g., JScrollPane for widgets
- e.g., ObjectOutputStream on a FileOutputStream





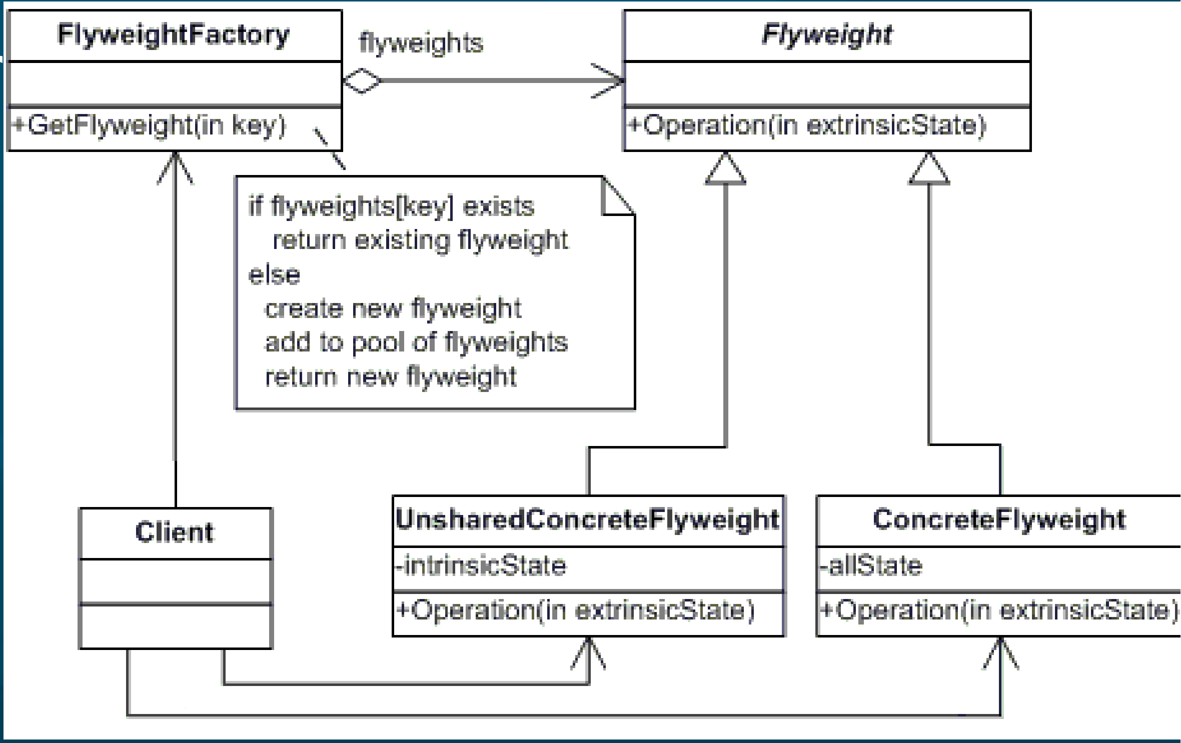
Structural pattern: Facade

- Provide a unified interface to a set of interfaces in a subsystem
  - High-level interface: system is easier to use
  - e.g., web front-end to complex database:
    - want minimal number of widgets, input boxes



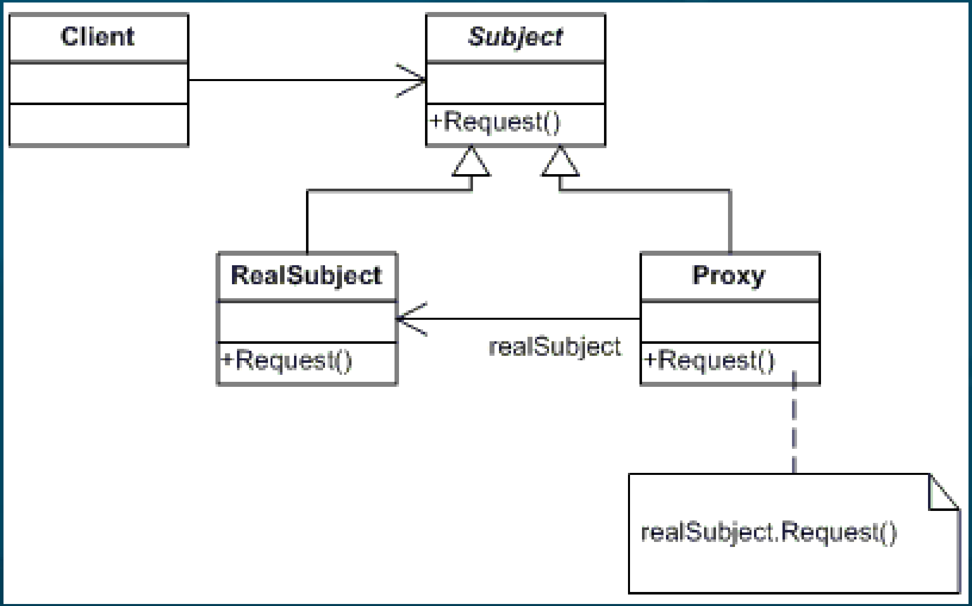
Structural pattern: Flyweight

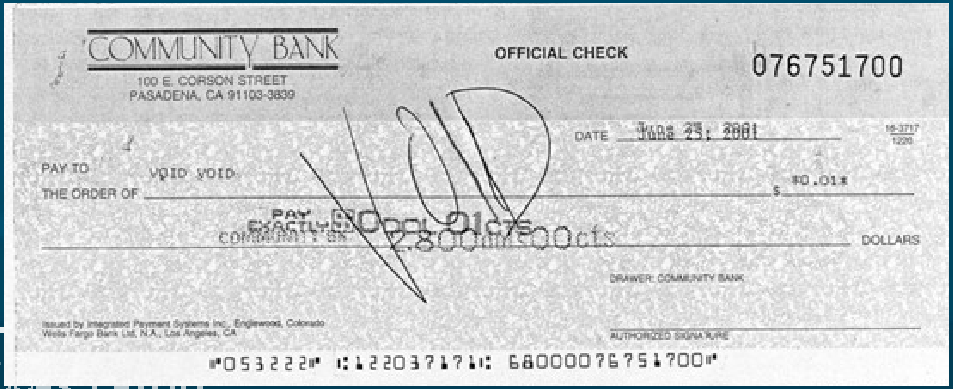
- Use sharing to support lots of "small" objects
- When more objects needed, draw from shared pool on demand
- Often use factory to create initial pool
- e.g., thread pool for multithread server
- Row of bank tellers


FlyweightFactory
+GetFlyweight(in key)
flyweights
Flyweight
+Operation(in extrinsicState)
if flyweights[key] exists
return existing flyweight
else
create new flyweight
add to pool of flyweights
return new flyweight
Client
UnsharedConcreteFlyweight
-intrinsicState
+Operation(in extrinsicState)
ConcreteFlyweight
-allState
+Operation(in extrinsicState)

Structural pattern: Proxy

- Surrogate for the real object
- Control access to the real object, but still let clients think they are talking directly to it
- Use superclass over both real object and proxy
- Contrast with Adapter, Bridge?
- e.g., proxy HTTP server
- e.g., bank cheque




COMMUNITY BANK
100 E. CORSON STREET
PASADENA, CA 91103-3839

OFFICIAL CHECK

076751700

DATE June 25, 2001

16-3717
1220

PAY TO THE ORDER OF VOID VOID

$ *0.01*

PAY EXACTLY [illegible] DOLLARS

DRAWER: COMMUNITY BANK

Issued by Integrated Payment Systems Inc., Englewood, Colorado
Wells Fargo Bank Ltd, N.A., Los Angeles, CA

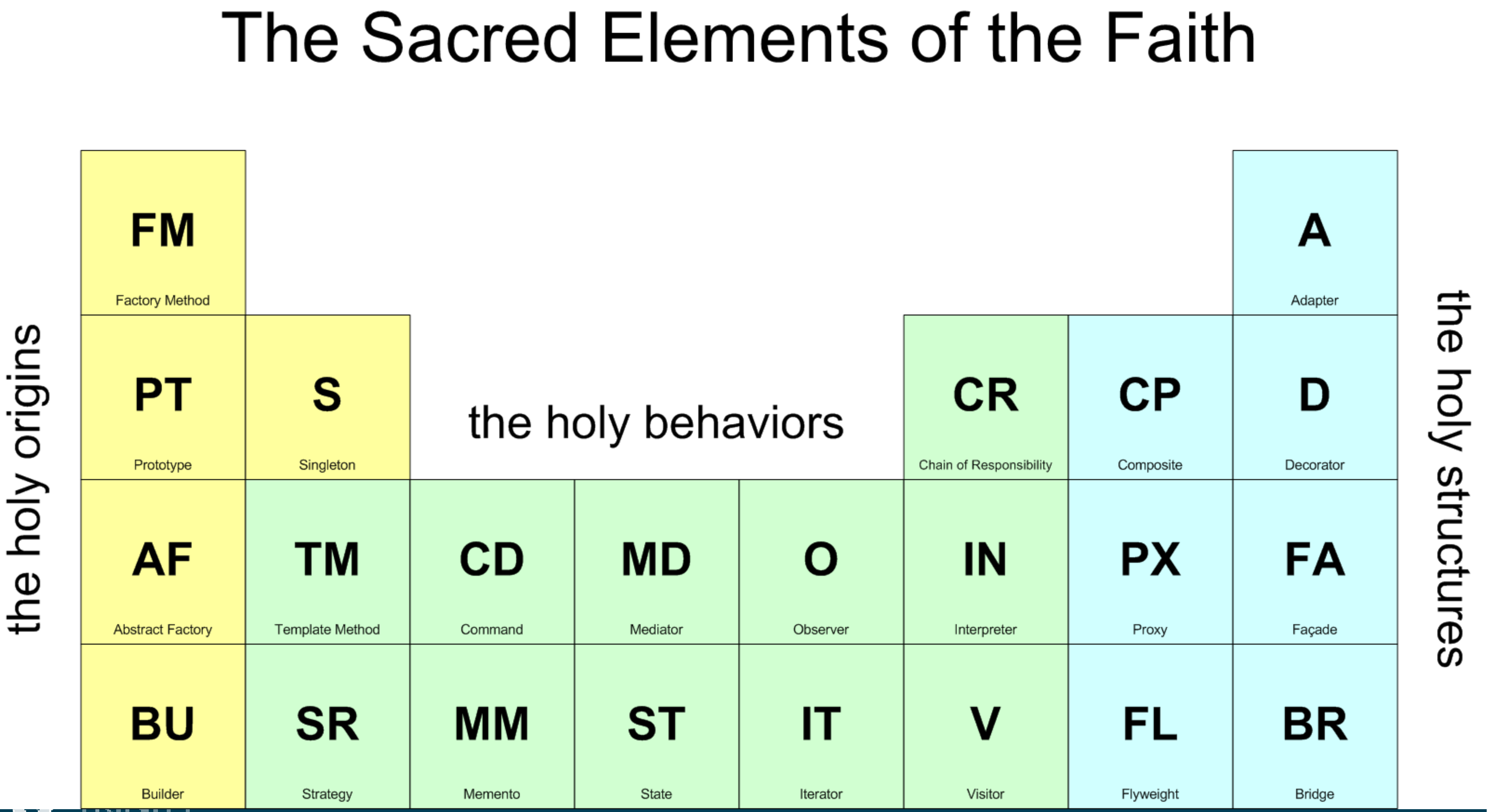
AUTHORIZED SIGNATURE

⑈053222⑈ ⑆122037171⑆ 680000767517OO⑈

Structural patterns

- Adapter/ wrapper: Convert the interface of a class into another interface clients expect
- Bridge: split abstraction from implementation
- Composite: organize objects into trees
- Decorator: dynamically add responsibilities / functionality to an object
- Facade: hide complexities behind simple interface
- Flyweight: use sharing to support large numbers of fine-grained objects efficiently
- Proxy: surrogate/placeholder for another object

Design patterns (GoF)



The Sacred Elements of the Faith

the holy origins

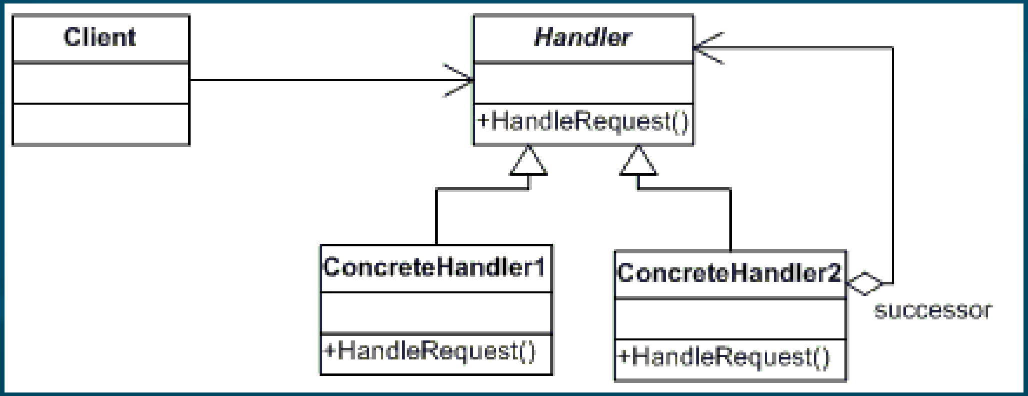
FM Factory Method							**A** Adapter
PT Prototype	**S** Singleton	the holy behaviors			**CR** Chain of Responsibility	**CP** Composite	**D** Decorator
AF Abstract Factory	**TM** Template Method	**CD** Command	**MD** Mediator	**O** Observer	**IN** Interpreter	**PX** Proxy	**FA** Façade
BU Builder	**SR** Strategy	**MM** Memento	**ST** State	**IT** Iterator	**V** Visitor	**FL** Flyweight	**BR** Bridge

the holy structures

Behavior: Chain of responsibility

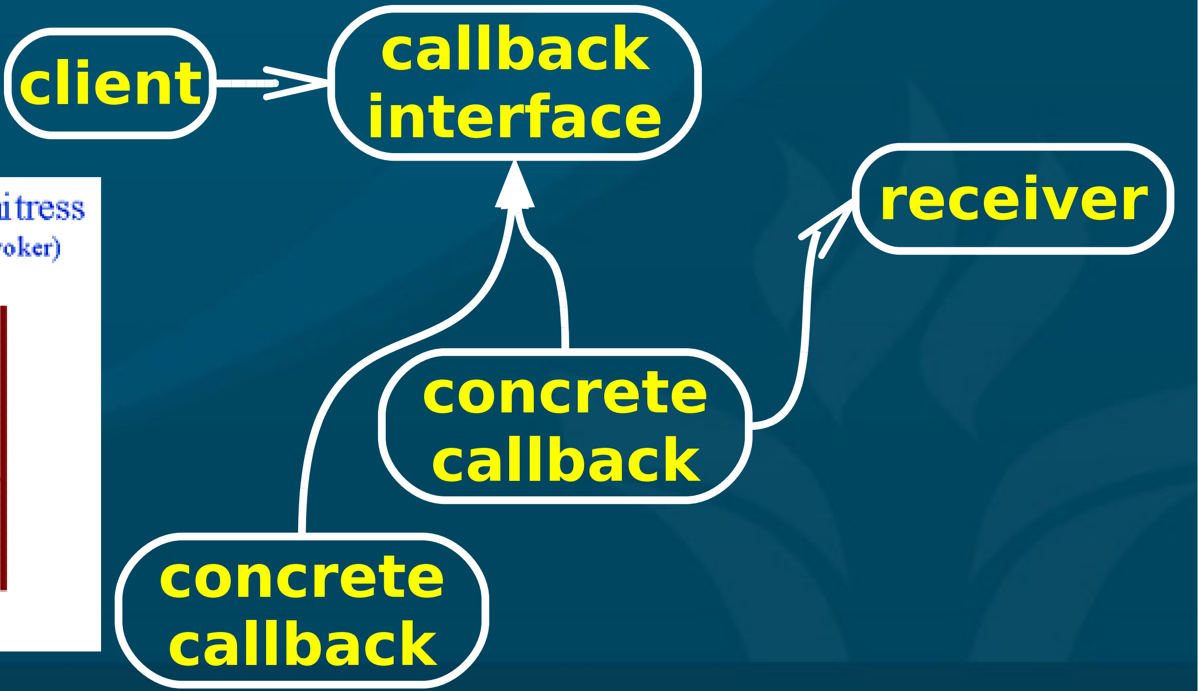
- Decouple sender from receiver by passing request along a chain of intermediate handlers
- Chain may be reconfigured dynamically
- Single pipeline, but many possible handlers
- e.g., coin passing through vending machine

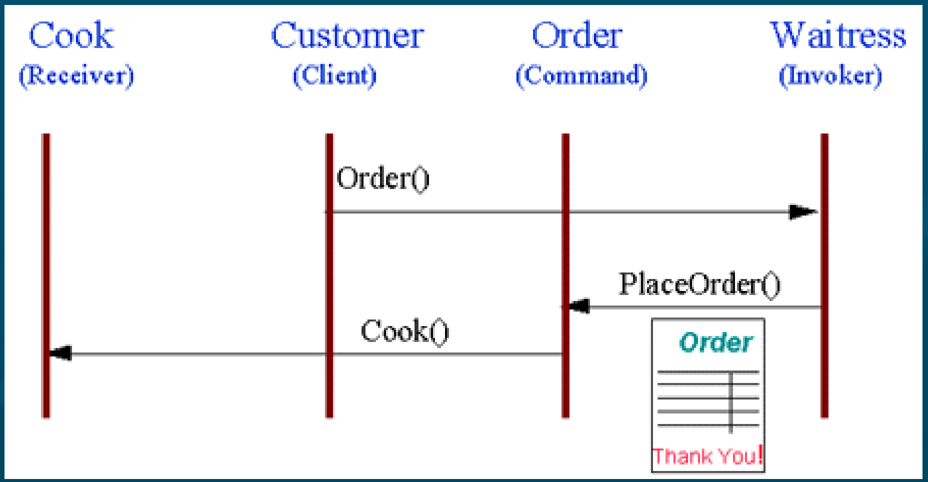
$$\begin{tikzcd}
\text{me} \arrow[r] & \text{my boss} \arrow[r] & \text{2nd level mgr} \arrow[r] & \text{regional mgr} \arrow[d] \\
 & & \text{CEO} & \text{vice president} \arrow[l]
\end{tikzcd}$$



Behavioural pattern: Command

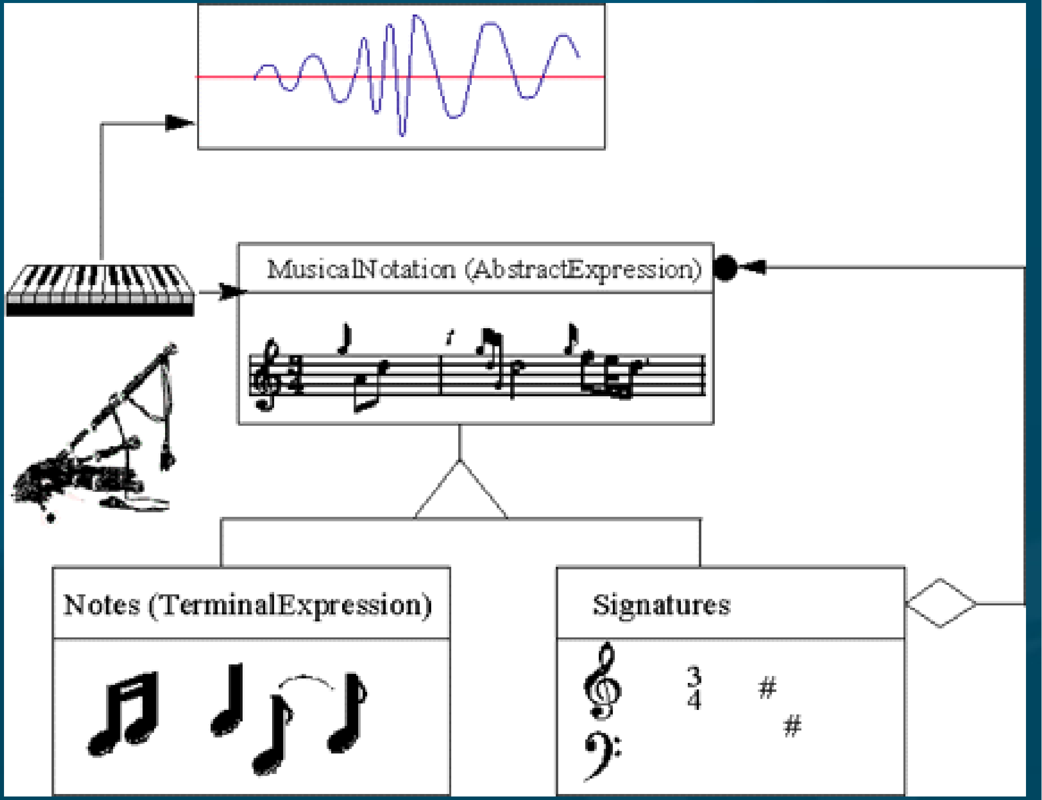
- Encapsulate a request as an object
  - e.g., function objects, callbacks
- Specify: object, method, arguments
- e.g., meal order at restaurant
- Support undo/redo





Behavioural: Interpreter

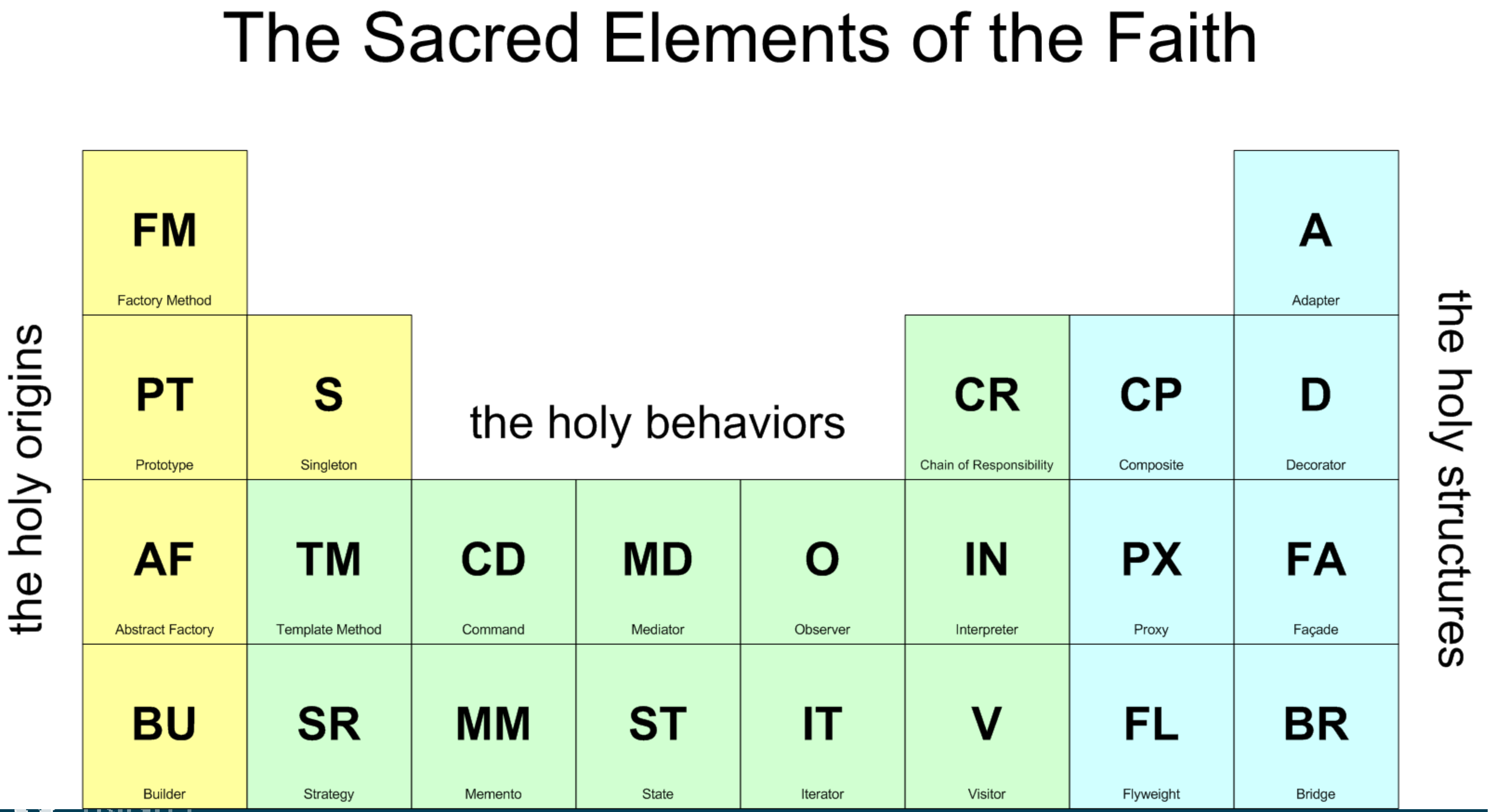
- Given a domain-specific language, define a grammar for the language and an engine to translate into objects
- Vocabulary + syntax
- e.g., parse config file
- e.g., read music → produce sound
- Useful for repeated, similar problems within a well-defined domain


MusicalNotation (AbstractExpression)
Notes (TerminalExpression)
Signatures

Behavioural patterns

- Chain of responsibility: avoid coupling sender directly to receiver by passing through chain
- Command: make requests into objects
- Interpreter: define macro language + parser
- Iterator: access all elements of a collection
- Mediator: object encapsulating the interactions of a set of objects: promotes loose coupling
- Memento: save/restore state of object
- Observer: decouple viewers from the subject

Design patterns (GoF)



The Sacred Elements of the Faith

the holy origins

the holy behaviors

the holy structures

FM Factory Method							**A** Adapter
PT Prototype	**S** Singleton		the holy behaviors		**CR** Chain of Responsibility	**CP** Composite	**D** Decorator
AF Abstract Factory	**TM** Template Method	**CD** Command	**MD** Mediator	**O** Observer	**IN** Interpreter	**PX** Proxy	**FA** Façade
BU Builder	**SR** Strategy	**MM** Memento	**ST** State	**IT** Iterator	**V** Visitor	**FL** Flyweight	**BR** Bridge