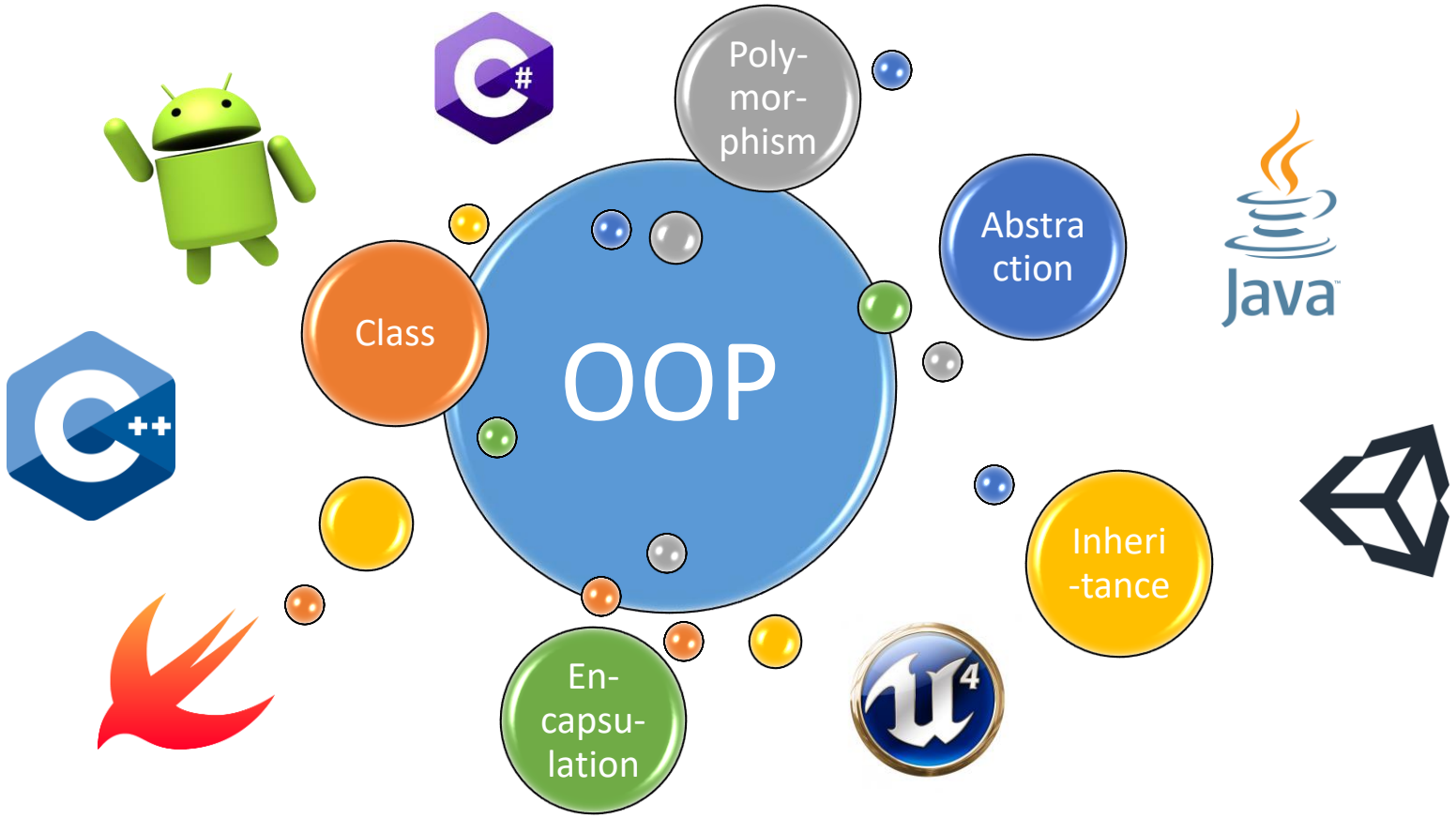


Structural Design Patterns

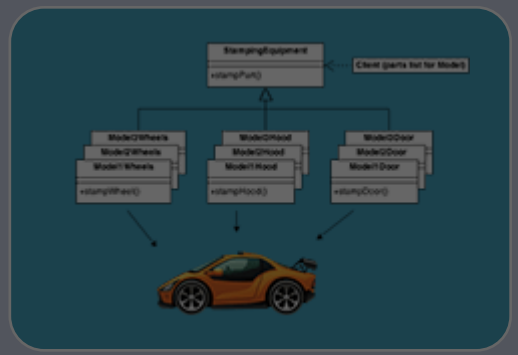
Kuan-Ting Lai

2020/4/12

Structural Design Patterns

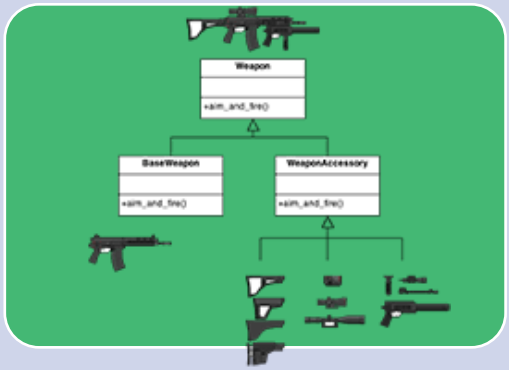
Creational Design Patterns

Initialize objects or create new classes



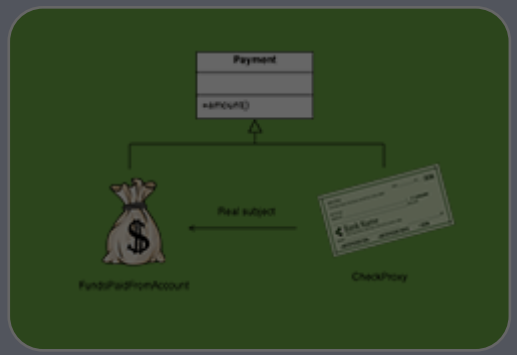
Structural Design Patterns

Compose objects to get new functions



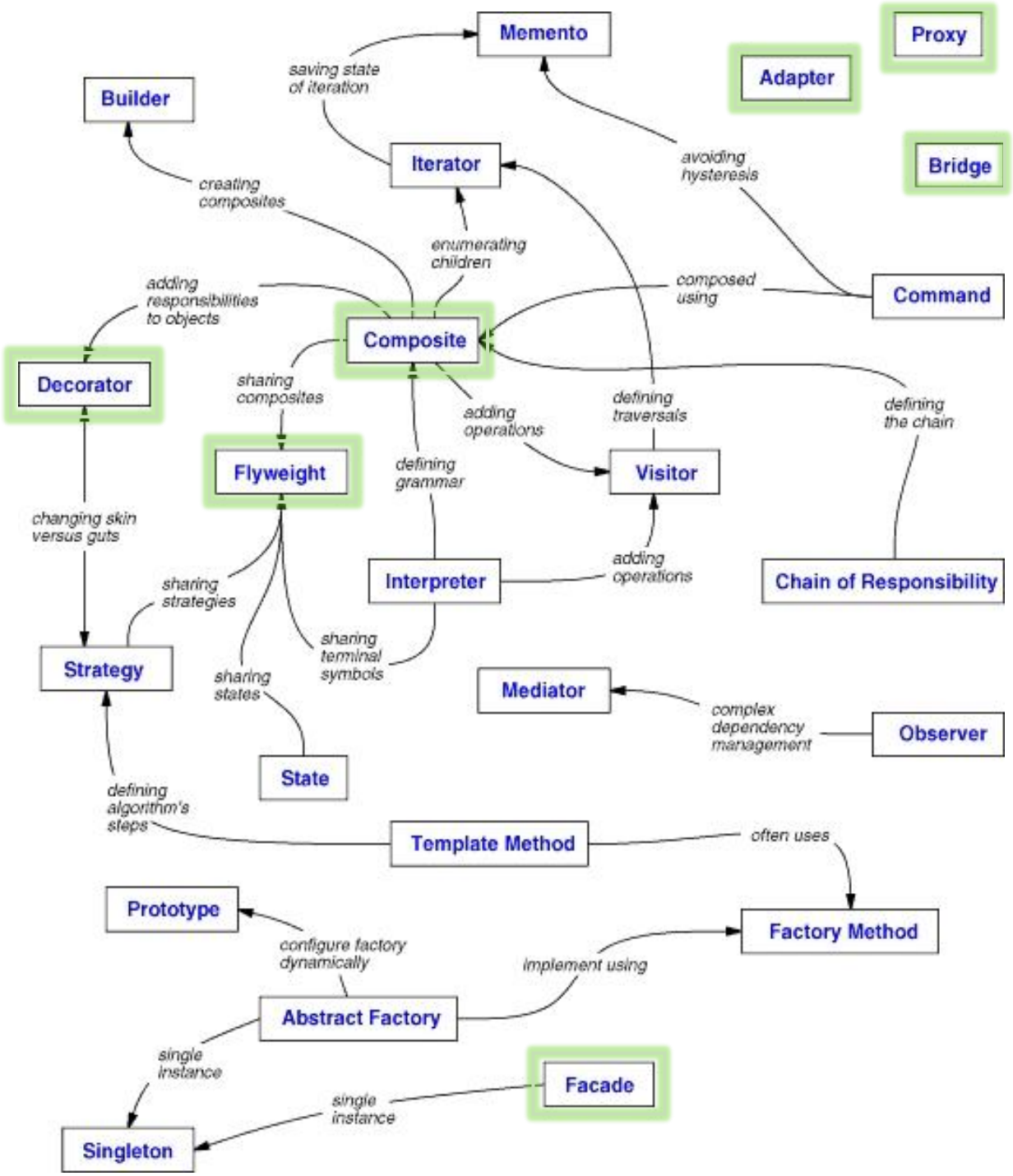
Behavioral Design Patterns

Communication between objects

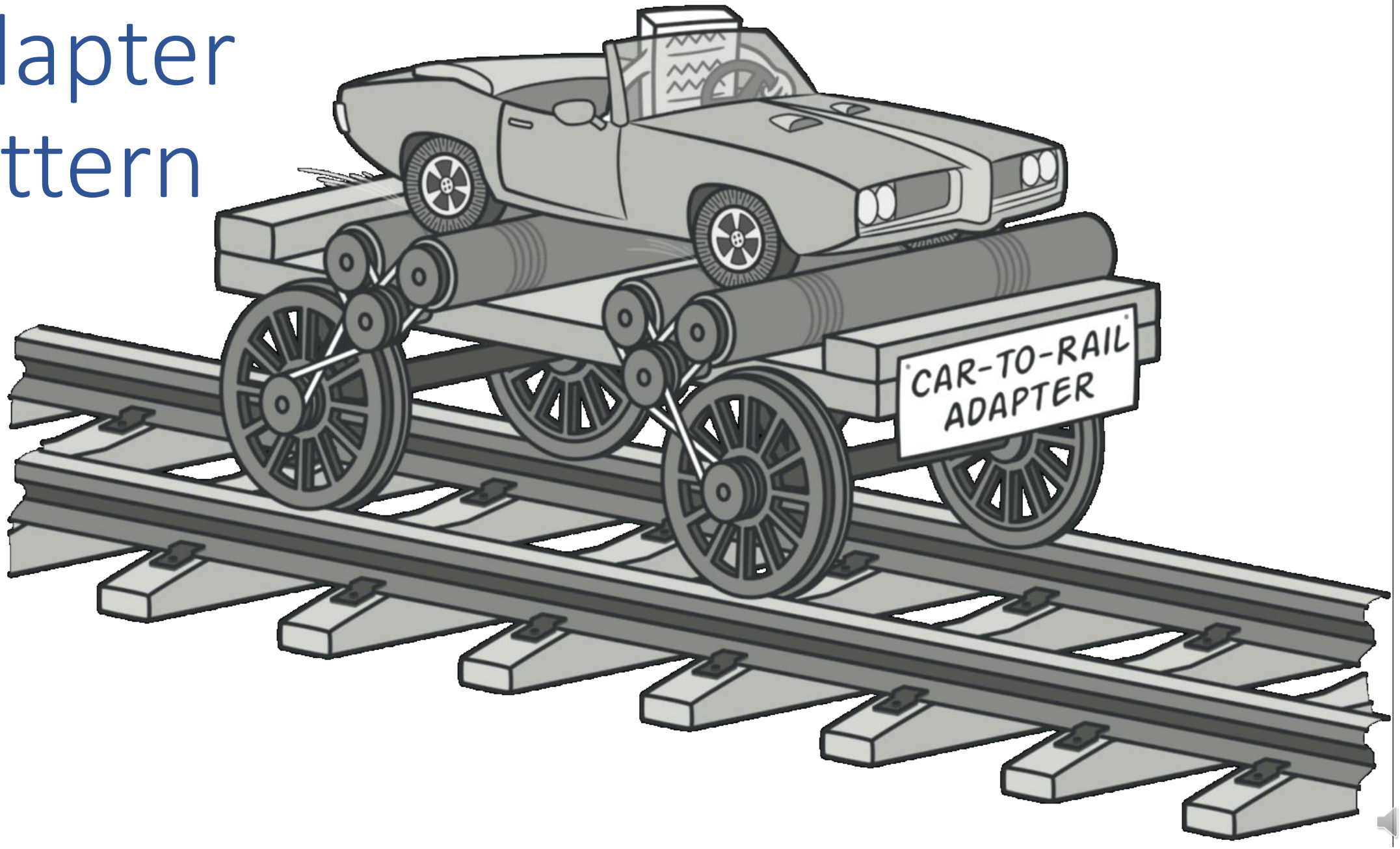


Structural Design Patterns

- Adapter
- Bridge
- Composite
- Decorator
- Façade
- Flyweight
- Proxy

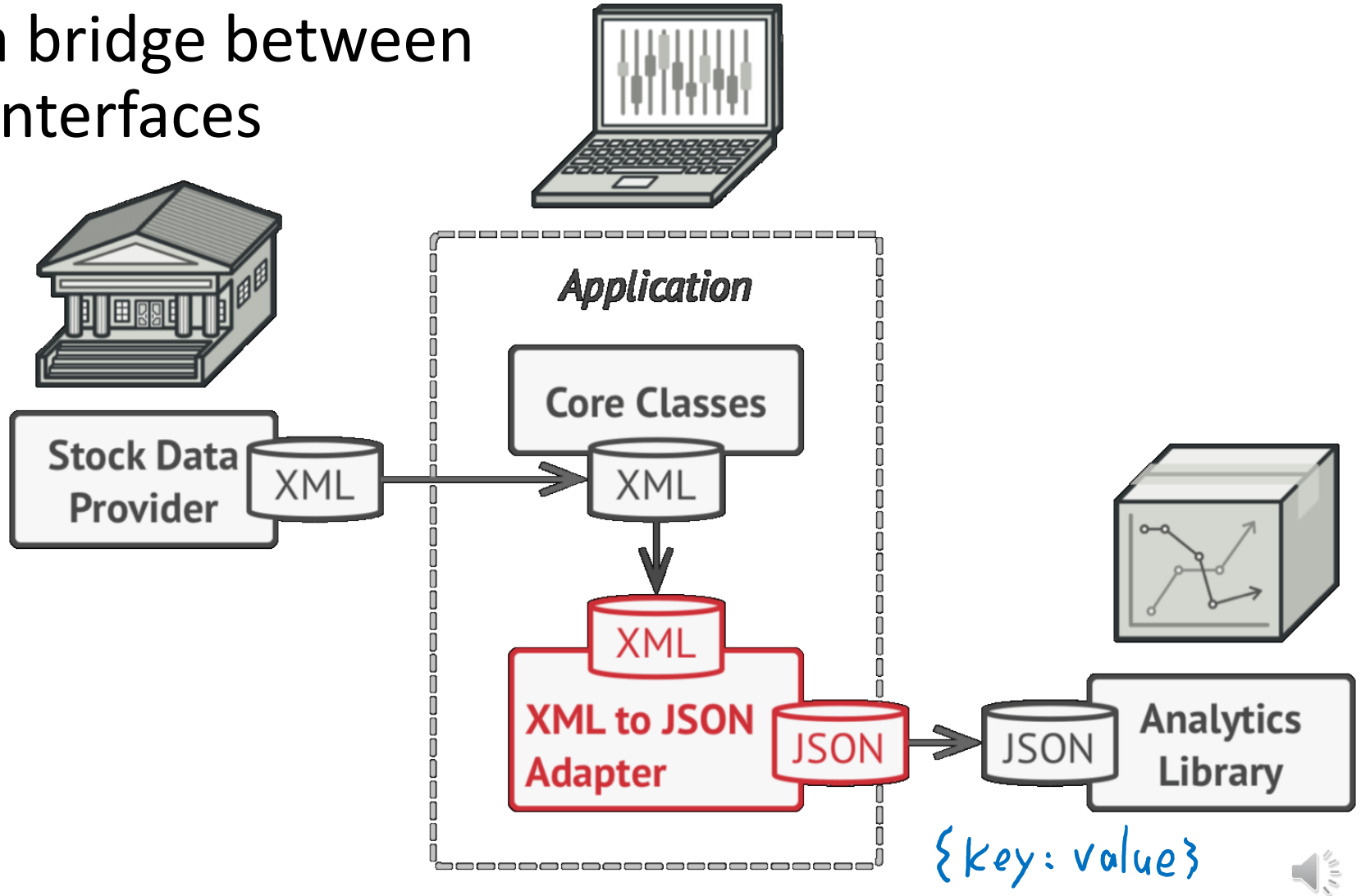

Memento
Proxy
Adapter
saving state of iteration
Builder
Iterator
avoiding hysteresis
Bridge
creating composites
enumerating children
adding responsibilities to objects
composed using
Command
Composite
Decorator
sharing composites
adding operations
defining traversals
defining the chain
Flyweight
defining grammar
Visitor
changing skin versus guts
Interpreter
adding operations
Chain of Responsibility
sharing strategies
sharing terminal symbols
Strategy
sharing states
Mediator
complex dependency management
Observer
State
defining algorithm's steps
Template Method
often uses
Prototype
Factory Method
configure factory dynamically
implement using
Abstract Factory
single instance
Facade
single instance
Singleton

Adapter Pattern

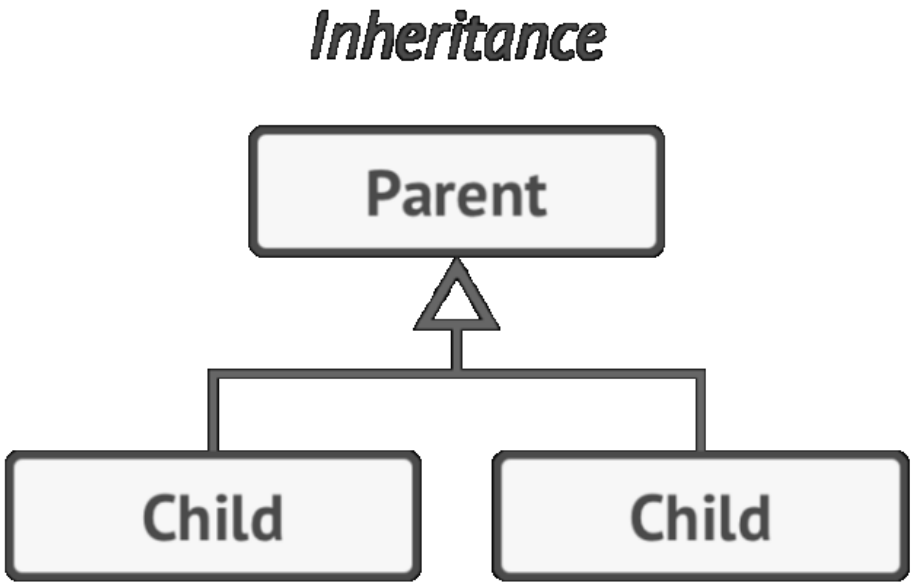


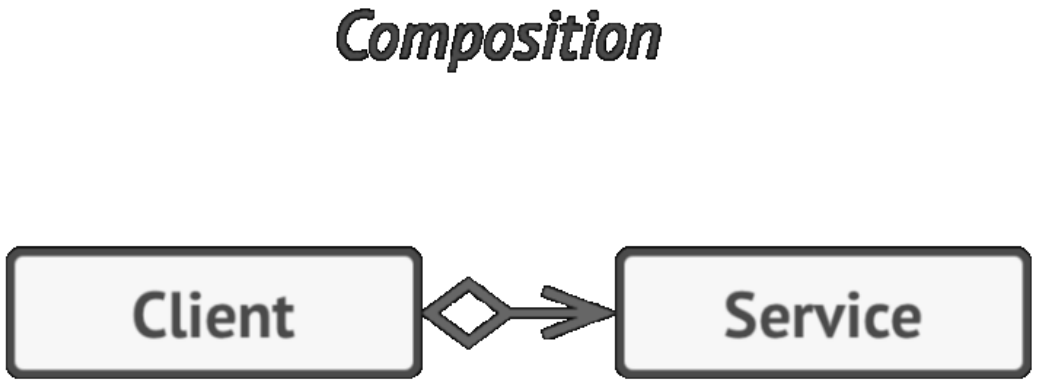
Adapter

- Adapter works as a bridge between two incompatible interfaces
- Object adaptor
- Class adaptor



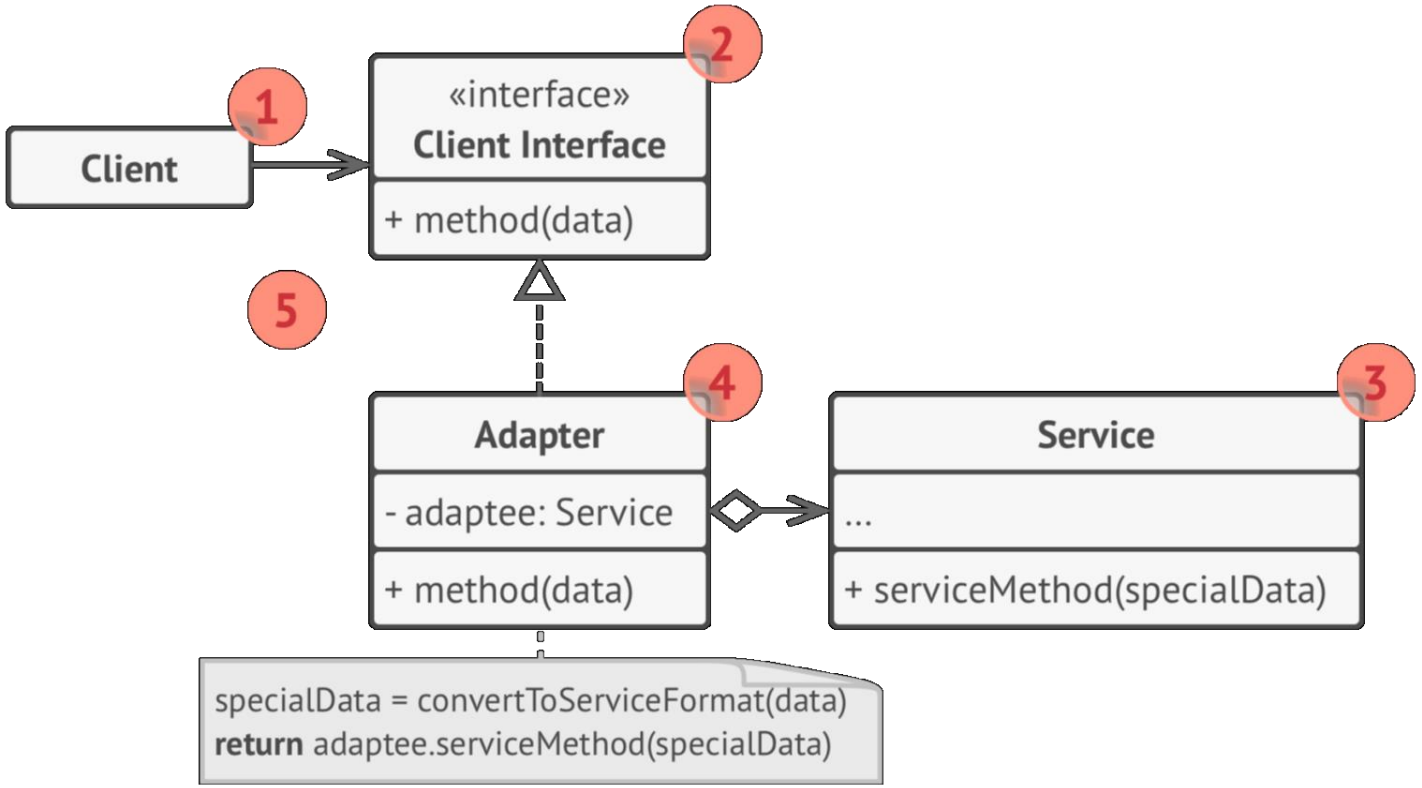
Inheritance vs. Composition





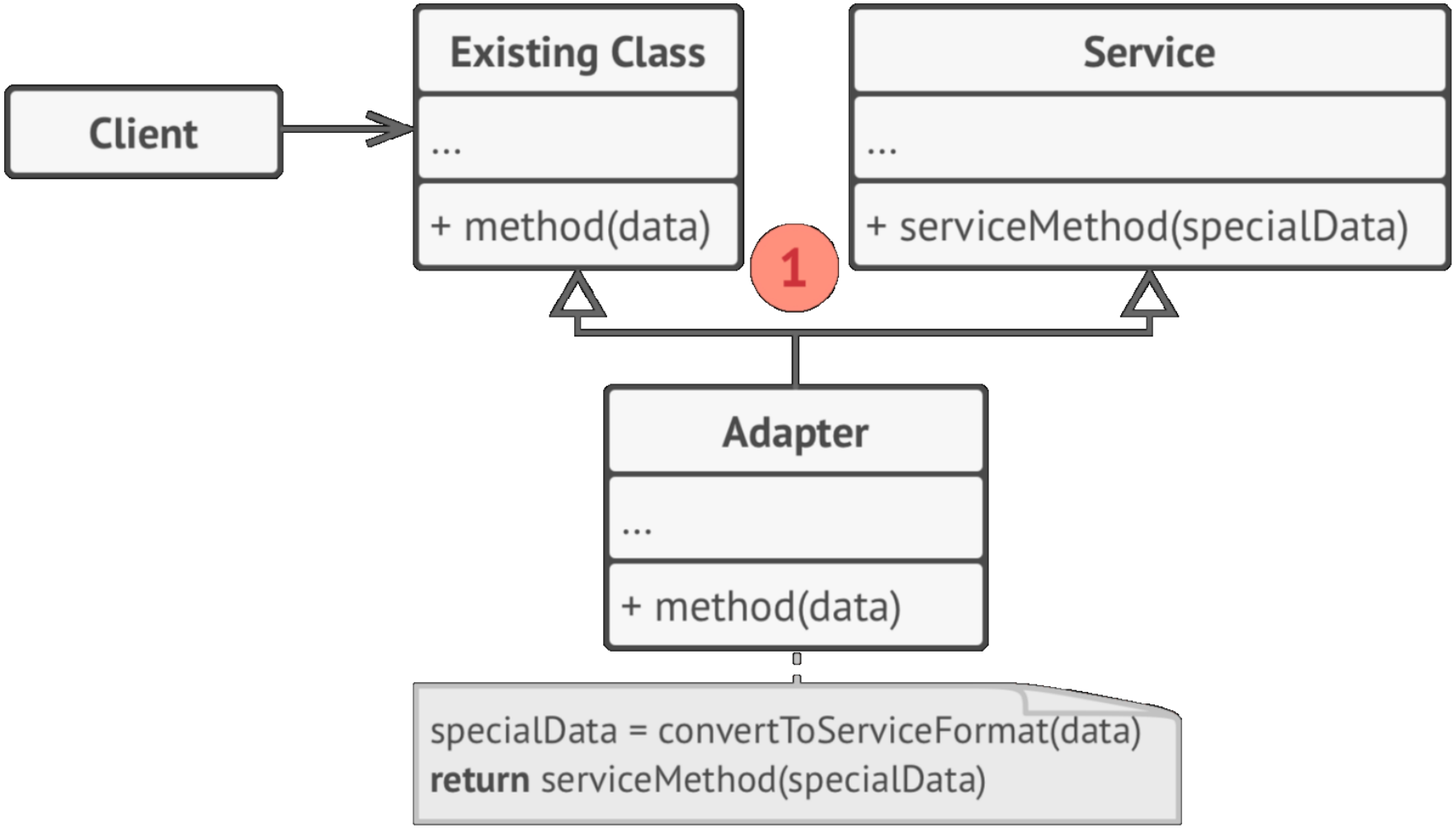
Object Adaptor

- **Client Interface** describes a protocol that other classes must follow
- **Adapter** is a class that implements the client interface and composite a service object

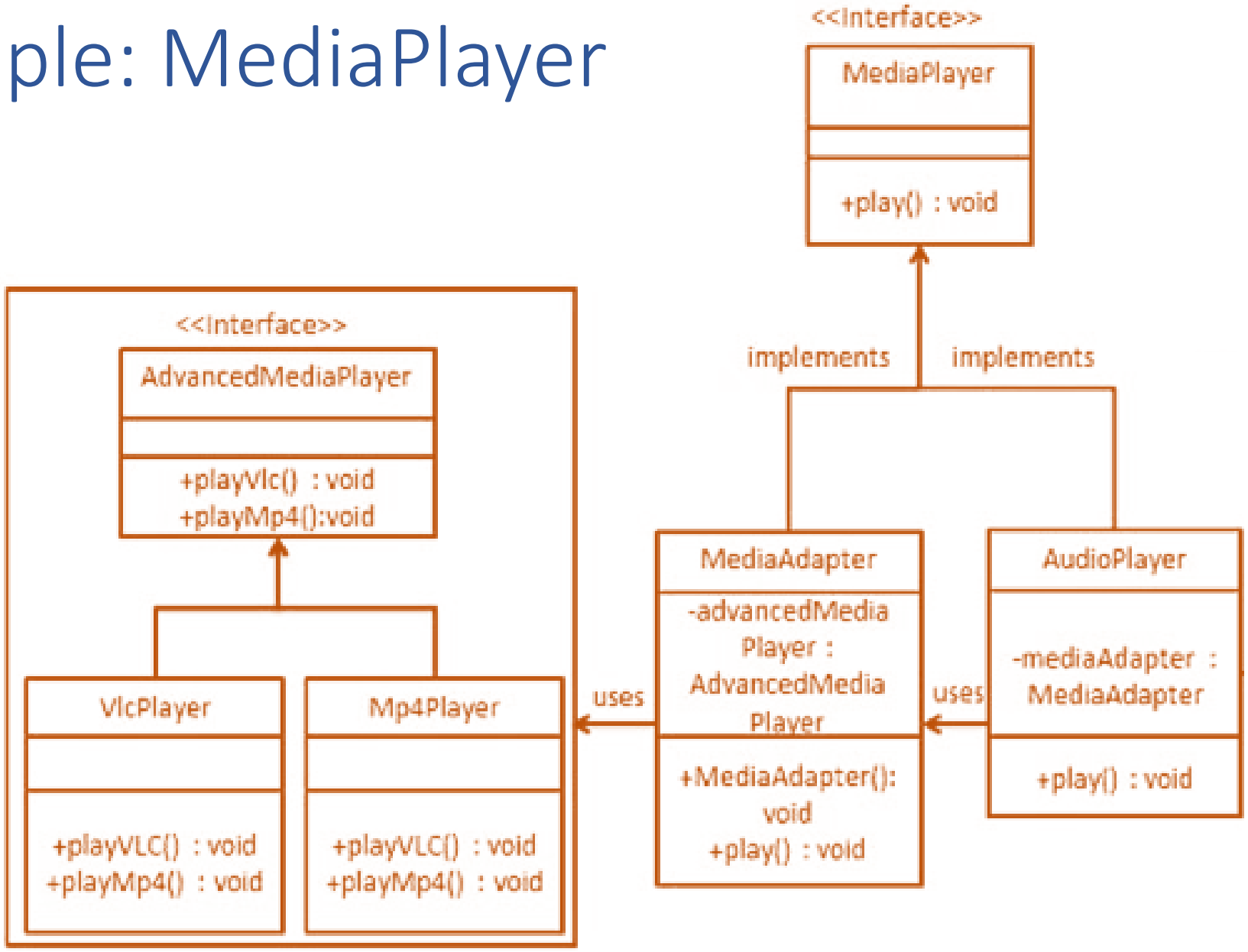


Class Adaptor

- Inherit from both client and service class



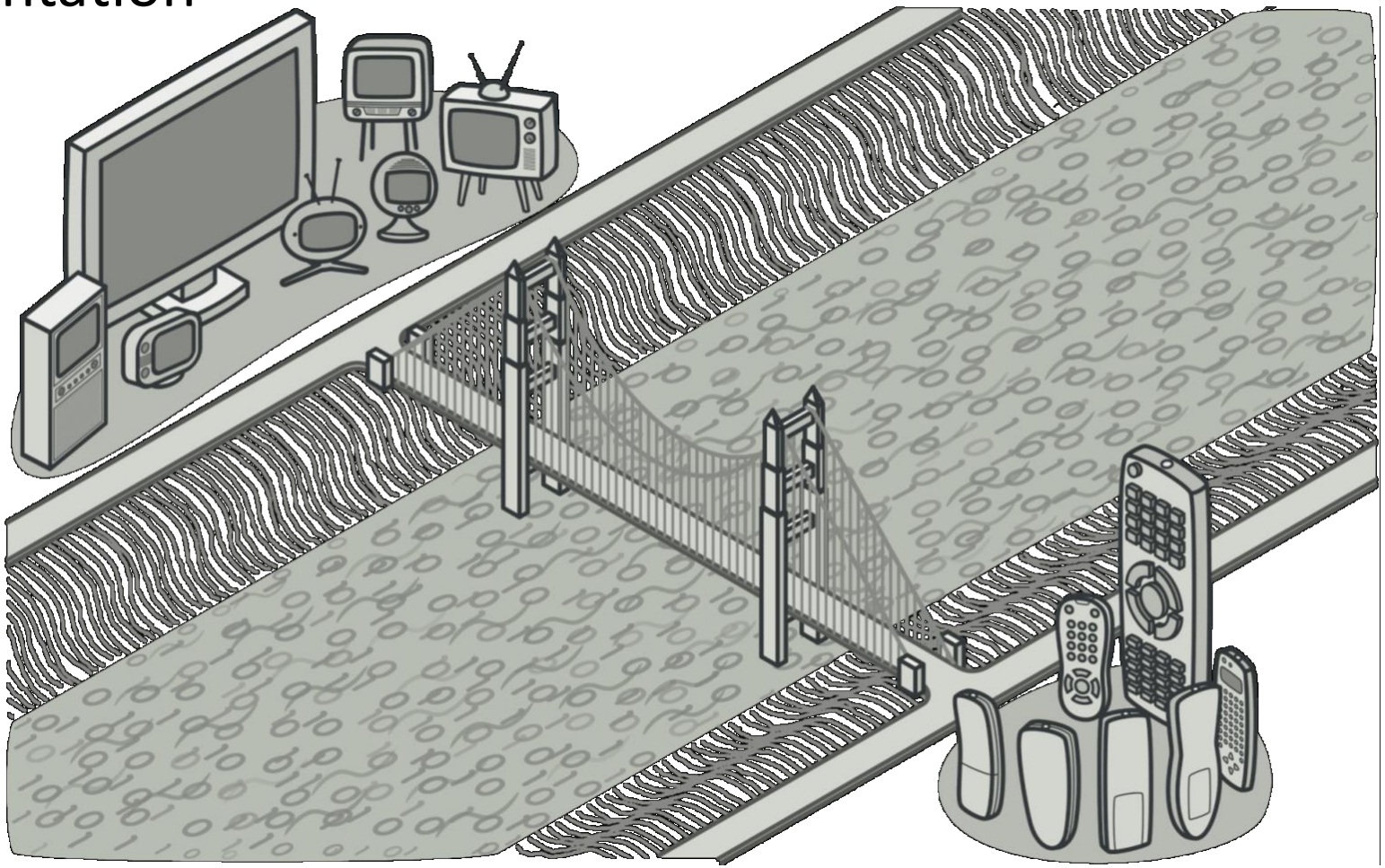
Example: MediaPlayer



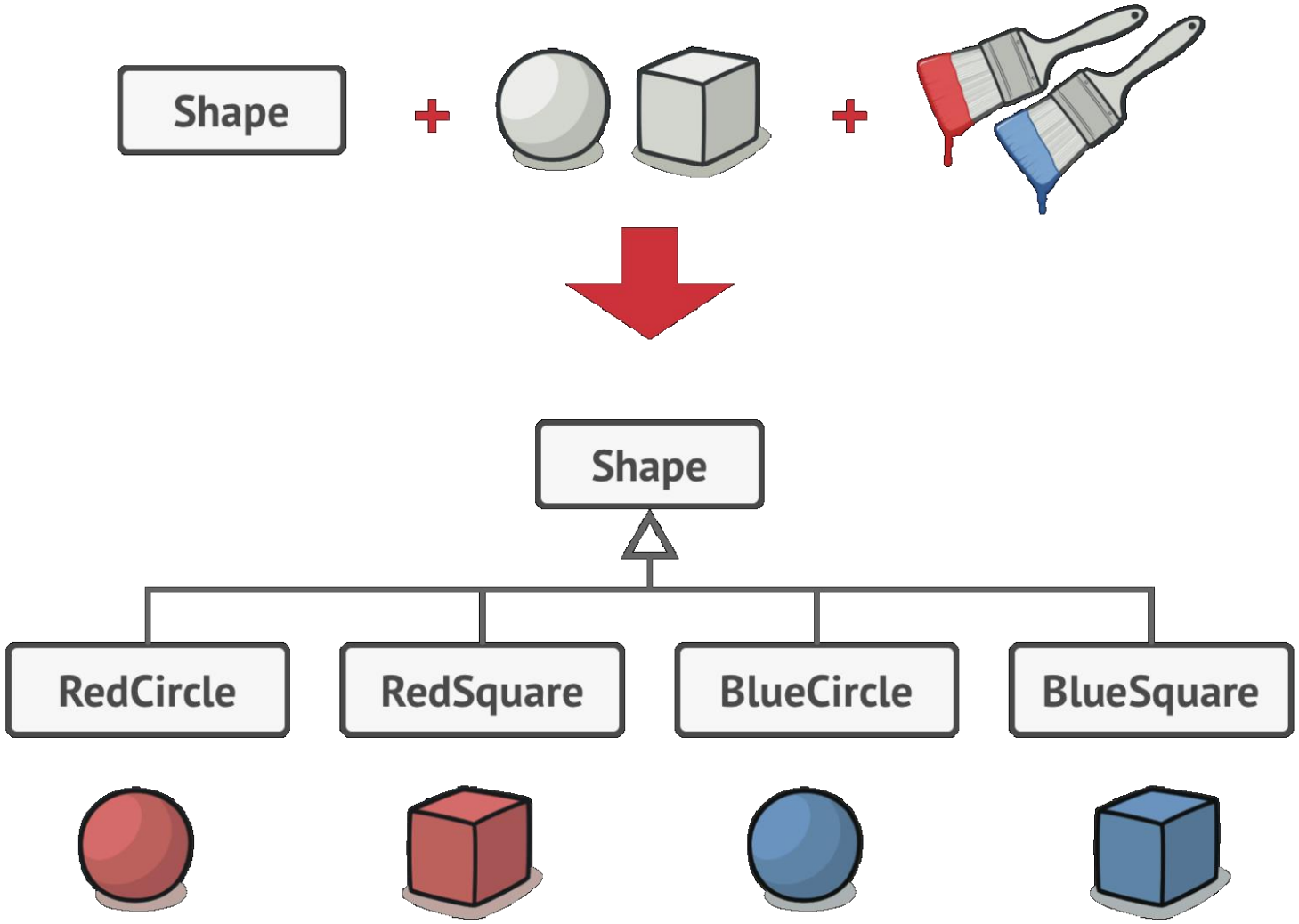
https://www.tutorialspoint.com/design_pattern/adapter_pattern.htm

Bridge Pattern

- Split a large class into two separate hierarchies: abstraction and implementation

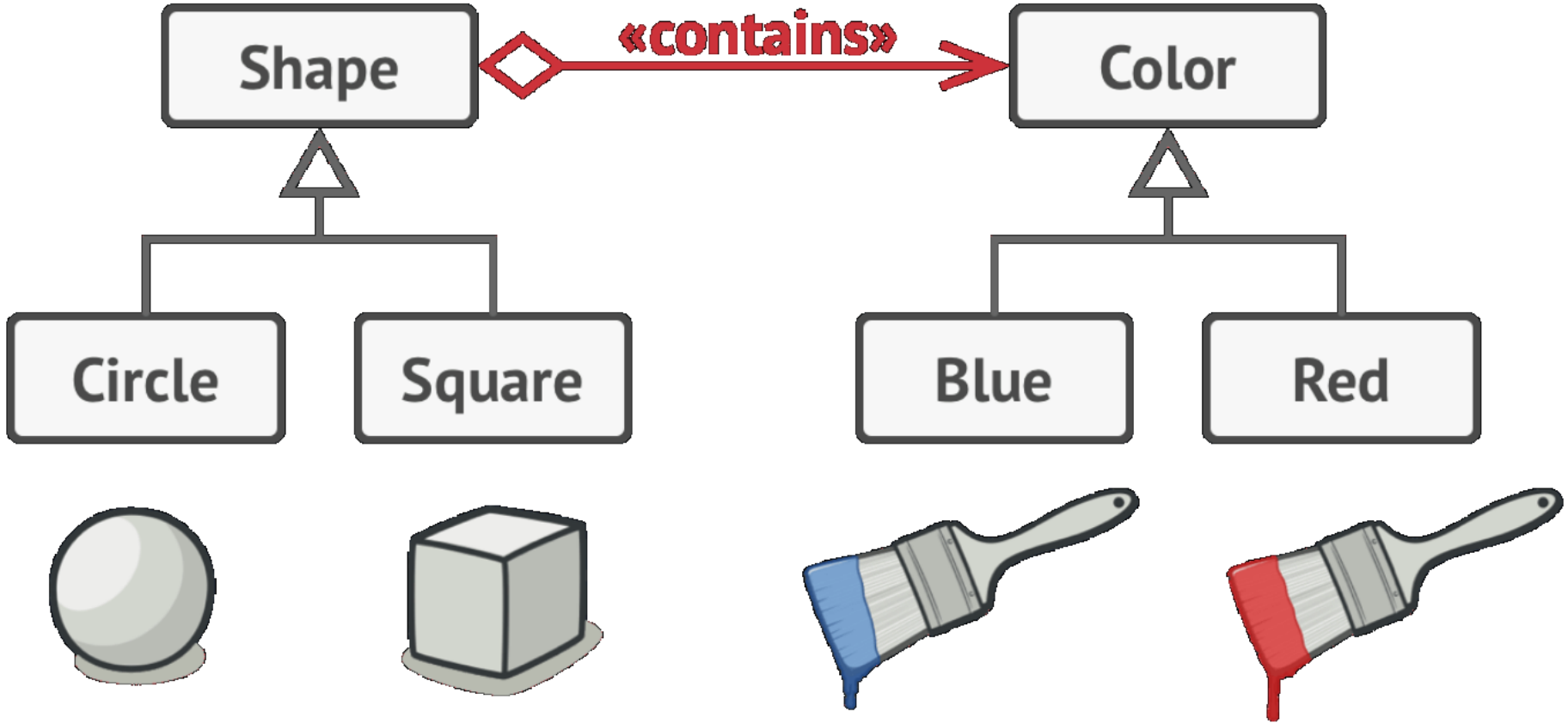


Bridge Example: Colorful Shapes

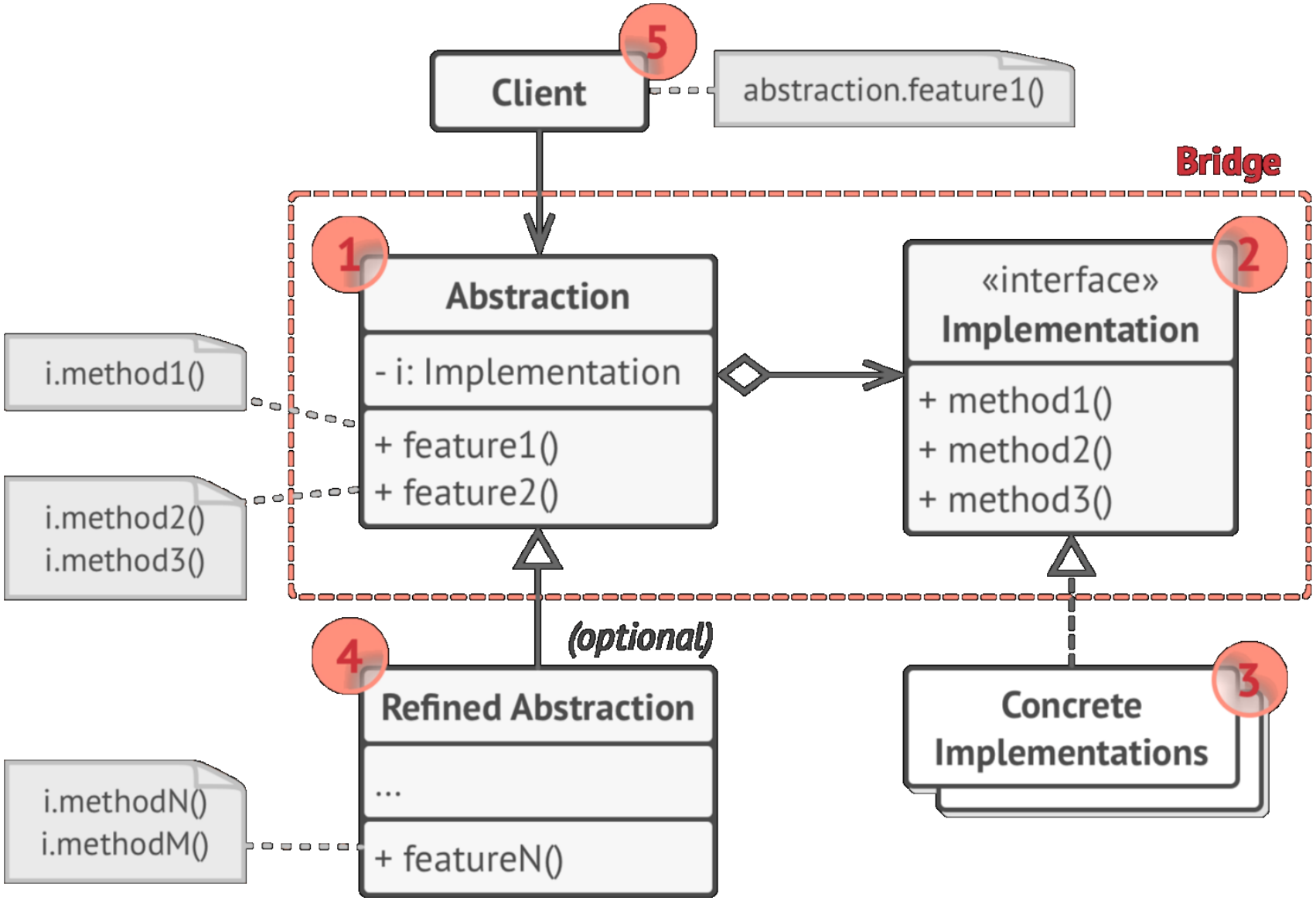


Separating Shape and Color

- The Bridge pattern attempts to solve this problem by switching from inheritance to composition

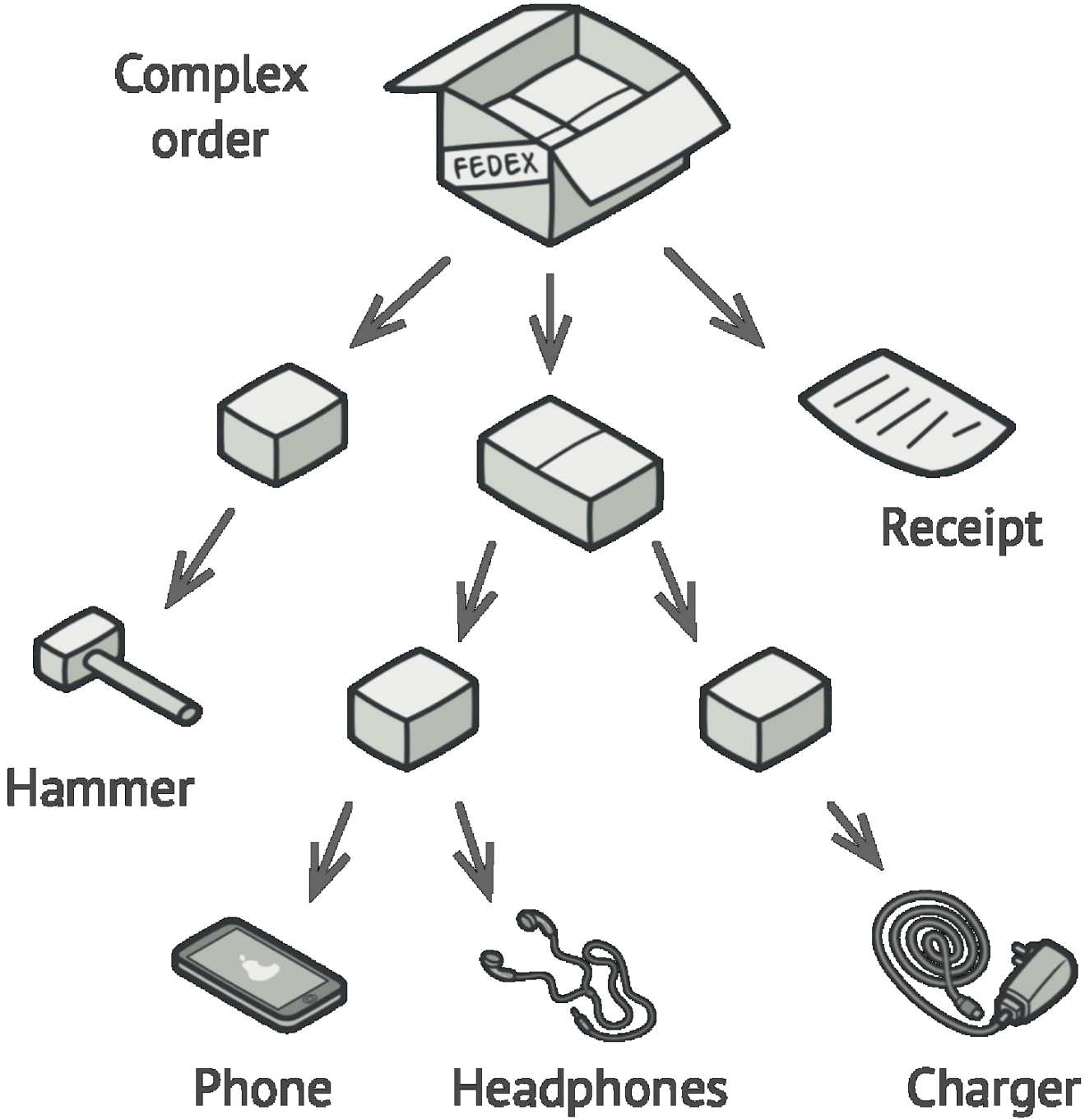


Using Composition to Implement New Function



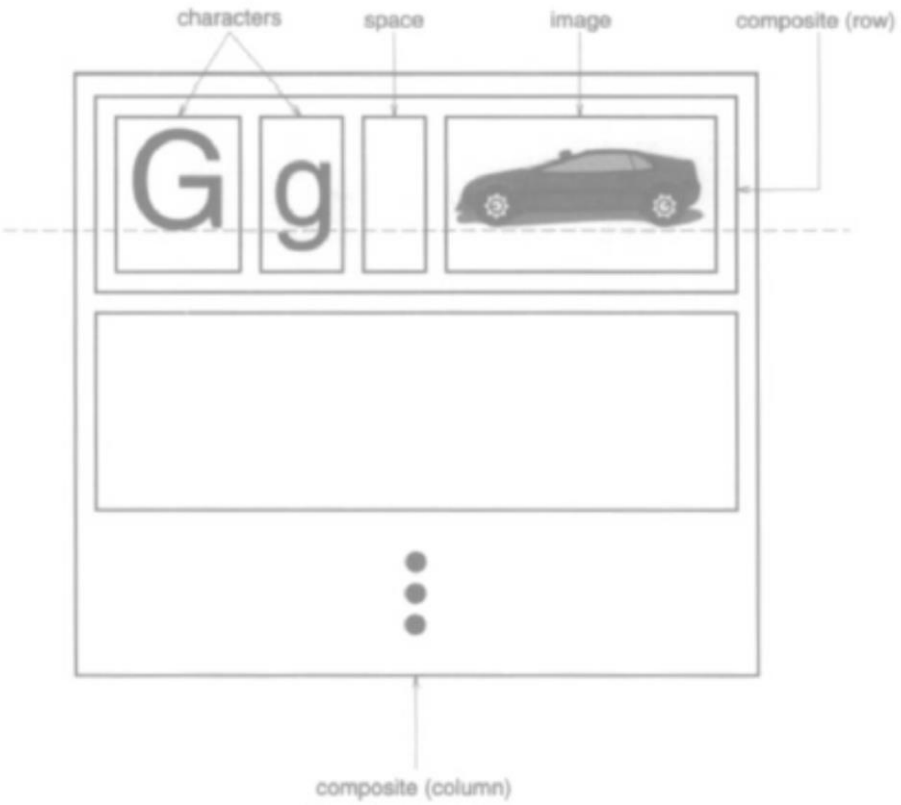
Composite Pattern

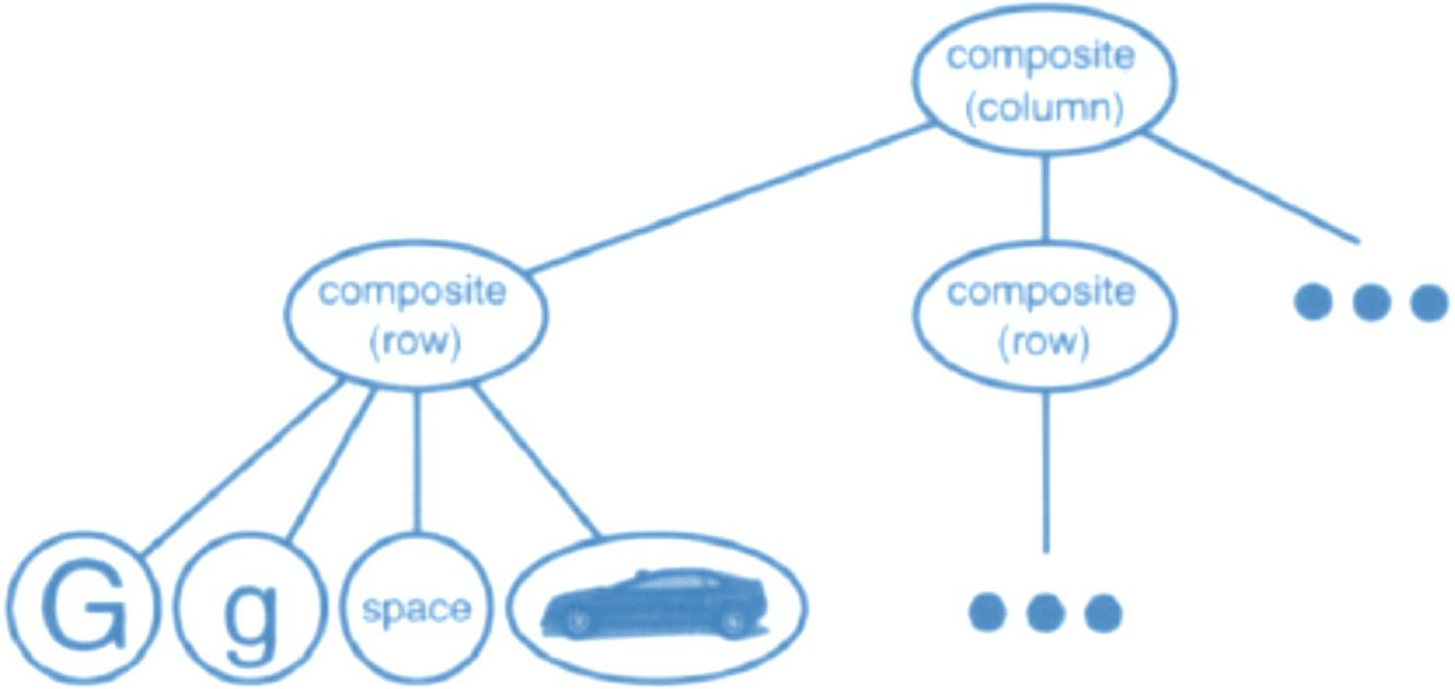
- Compose objects into a tree structure



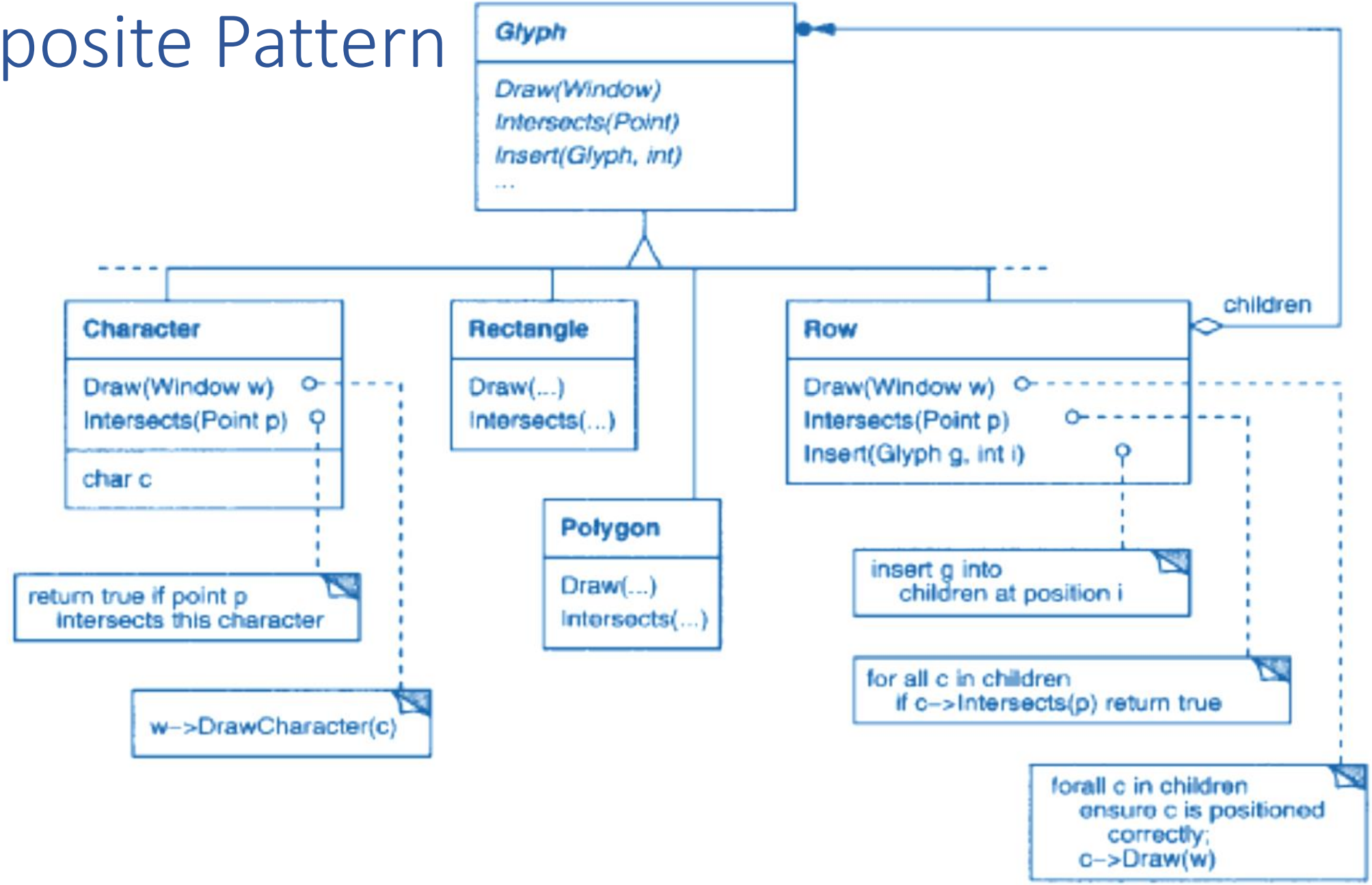
Document Structure

- Recursive composition of text and graphics



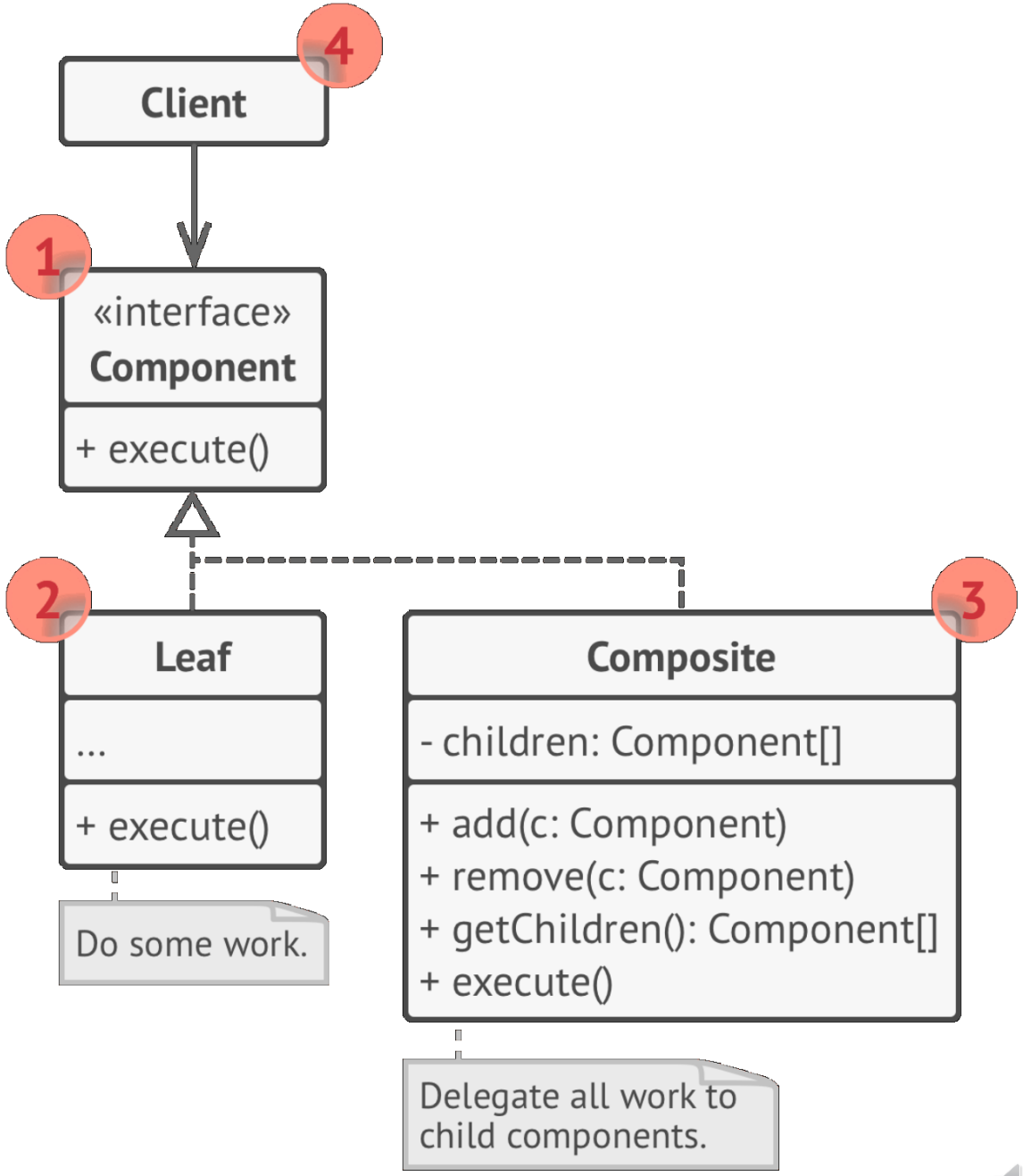


Composite Pattern



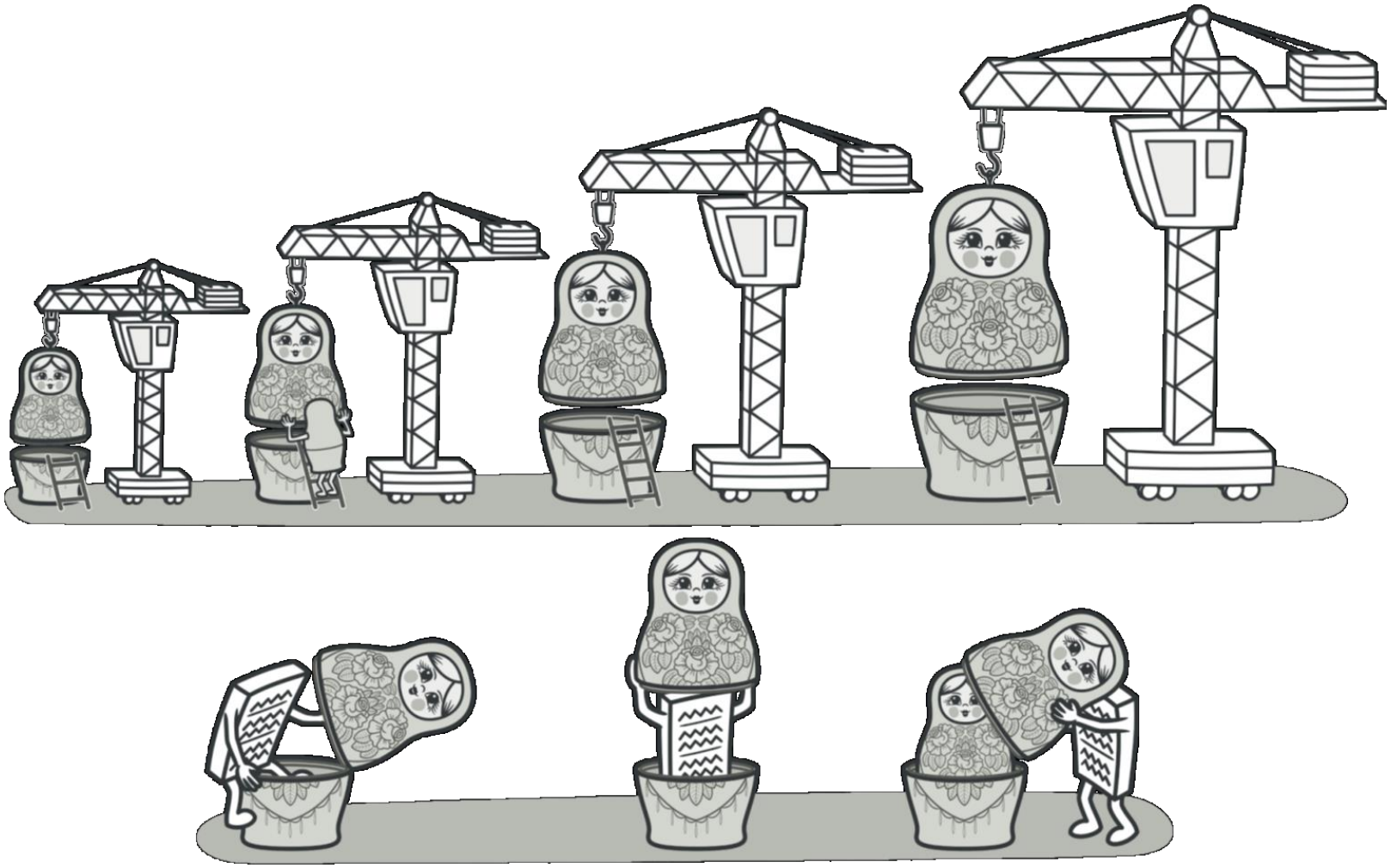
Composite Structure

- **Component** interface describes operations that are common to both simple and complex elements of the tree
- **Leaf** is a basic element without sub-elements
- **Container** (aka *composite*) is an element that has sub-elements



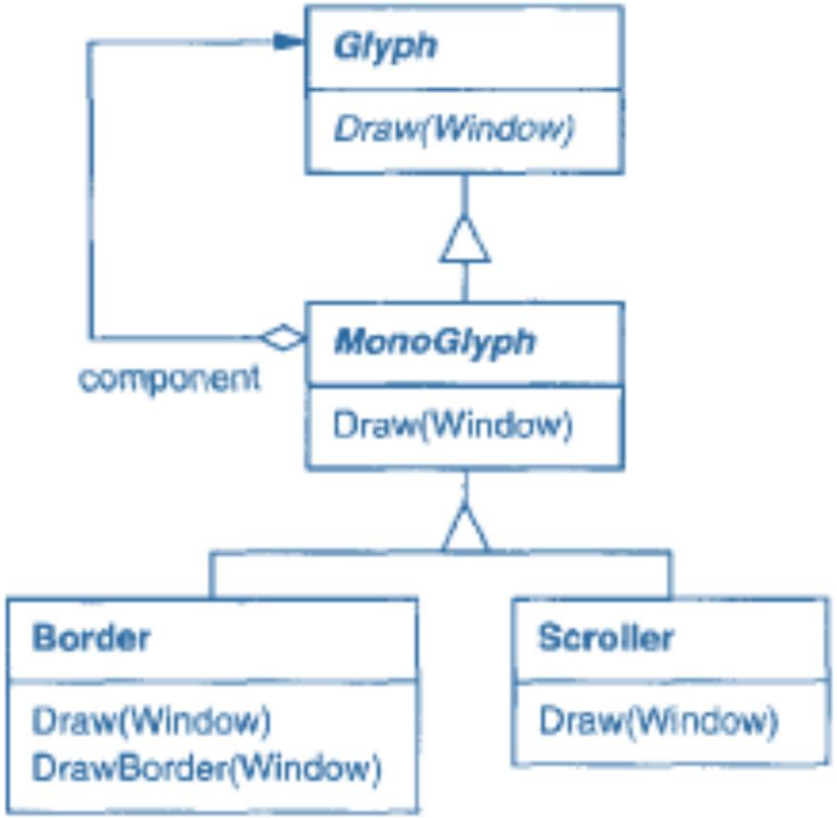
Decorator

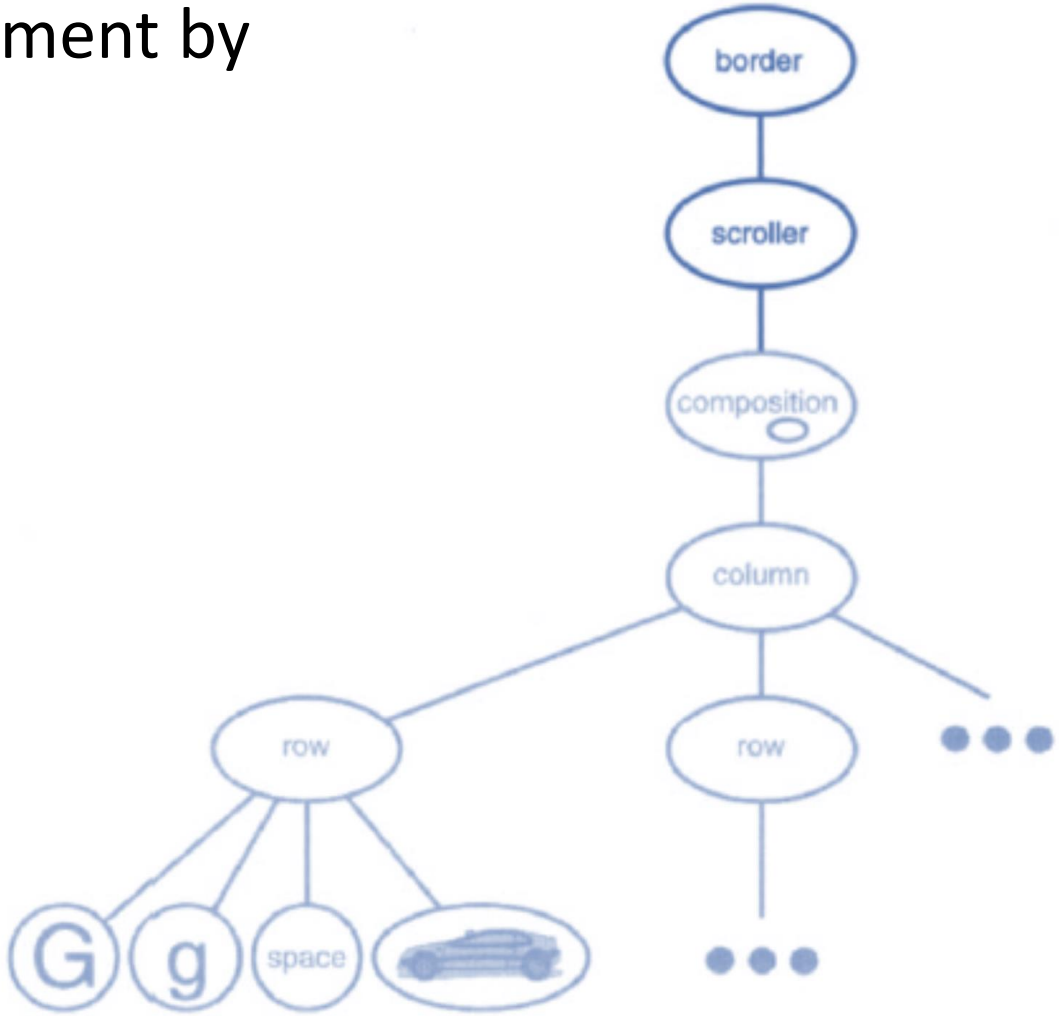
- Attach new behaviors to objects by placing these objects



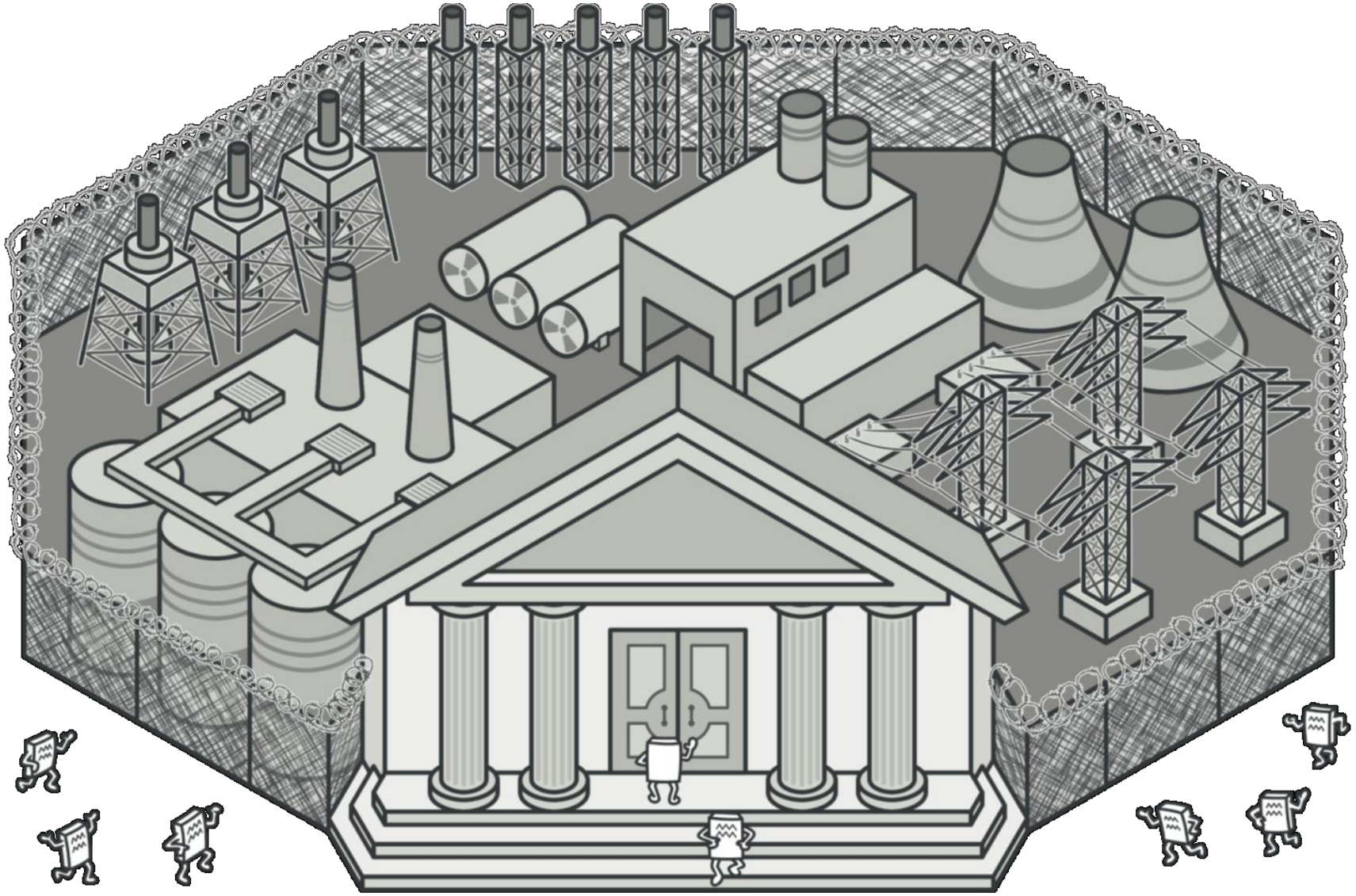
Embellishing the User Interface (Decorator)

- Decorator Pattern: support embellishment by transparent enclosure



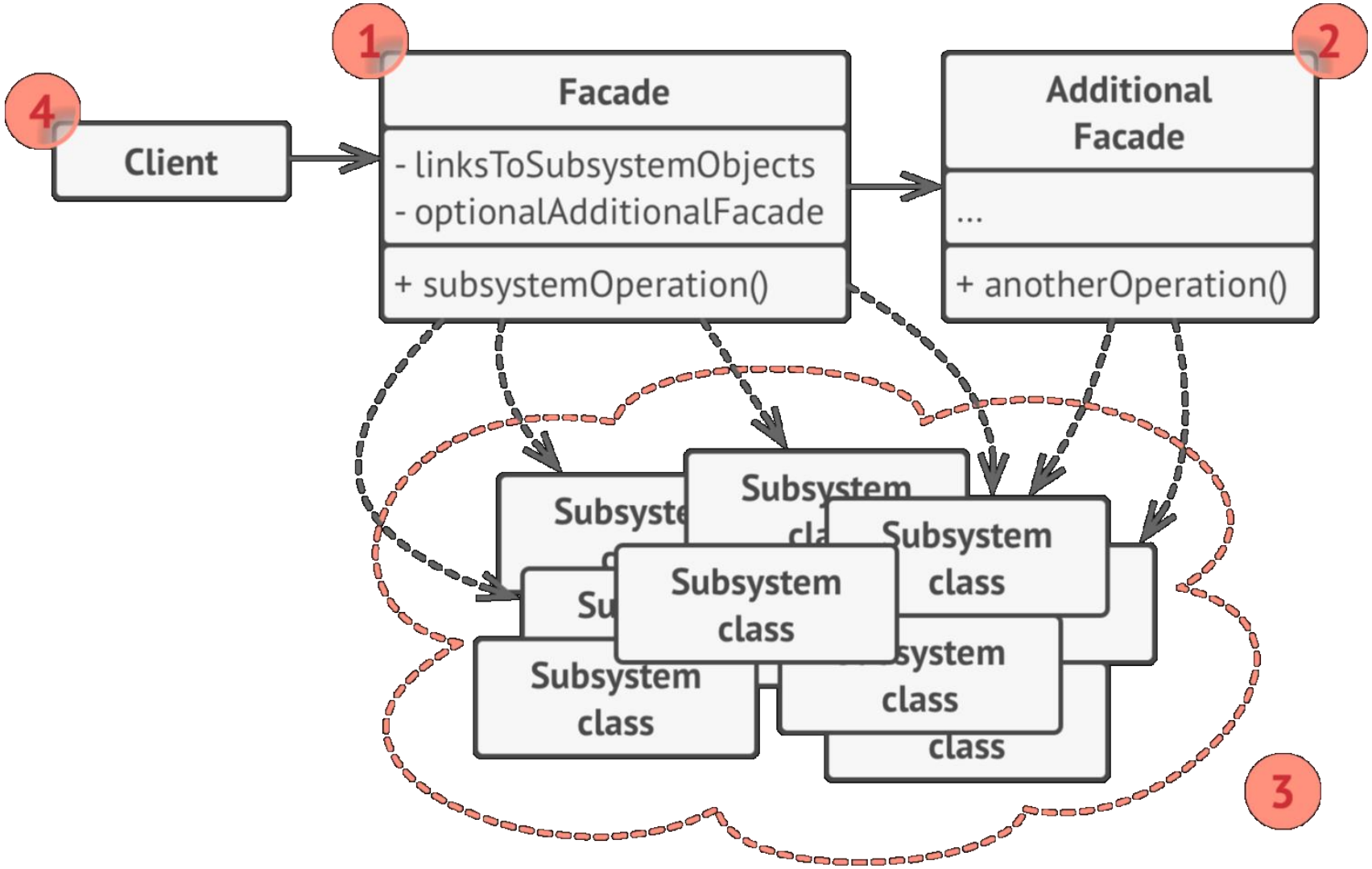


FACADE



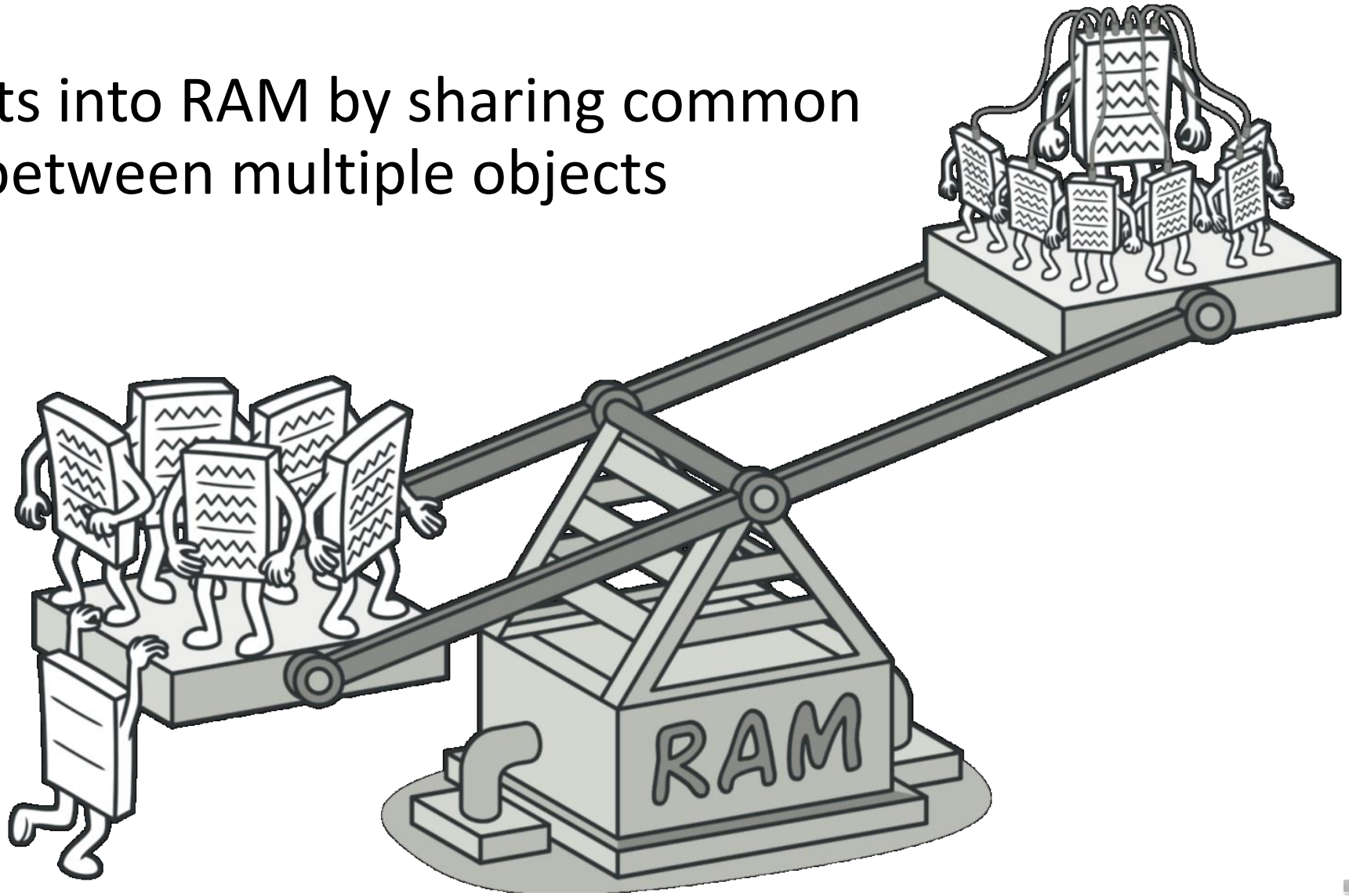
FACADE

- Define a new interface for existing many objects

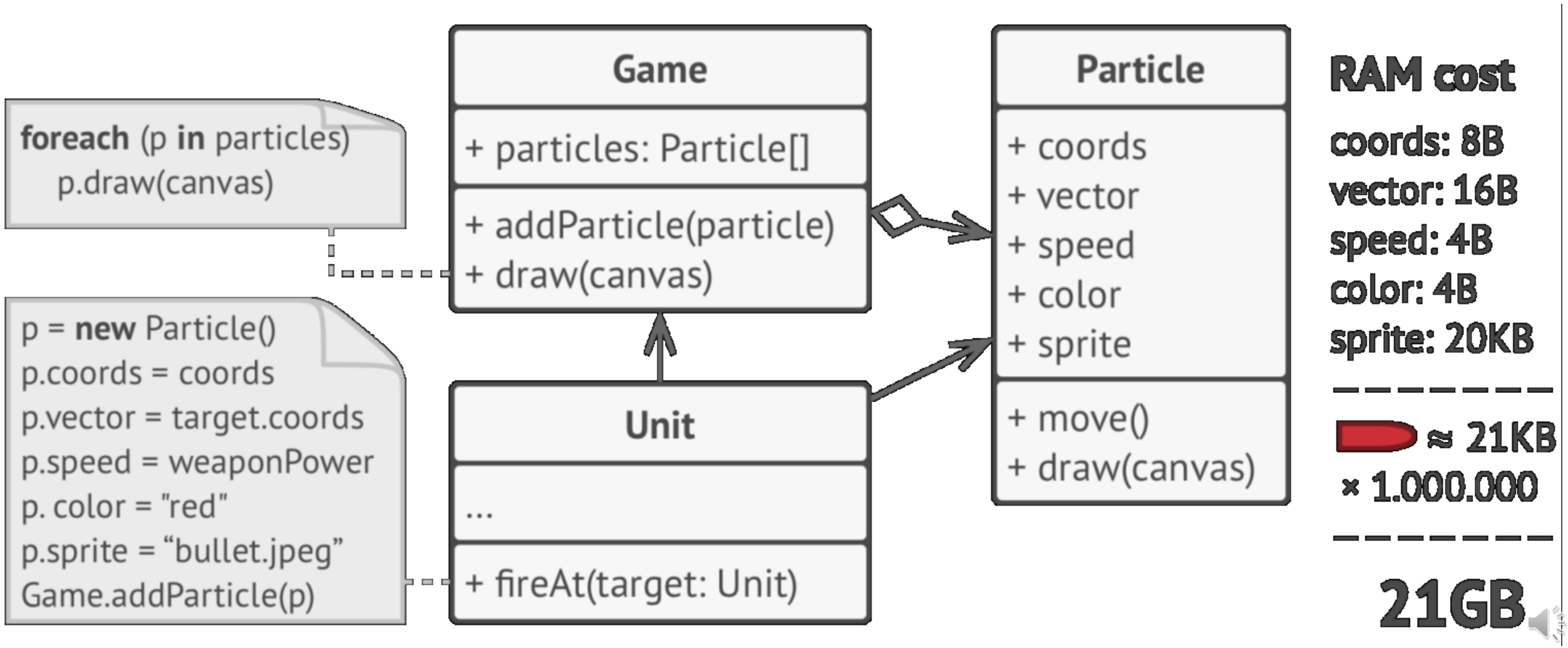


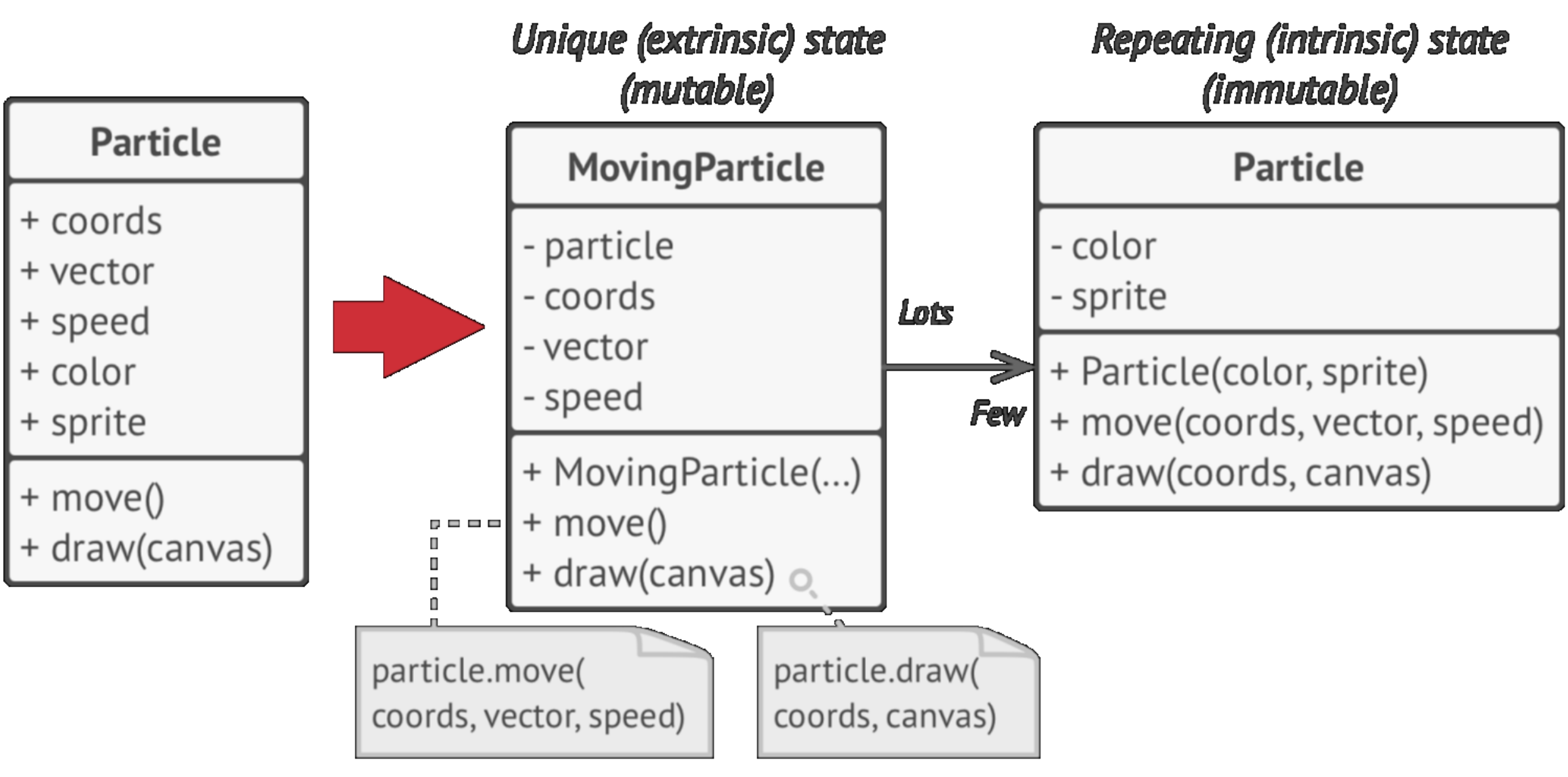
FLYWEIGHT

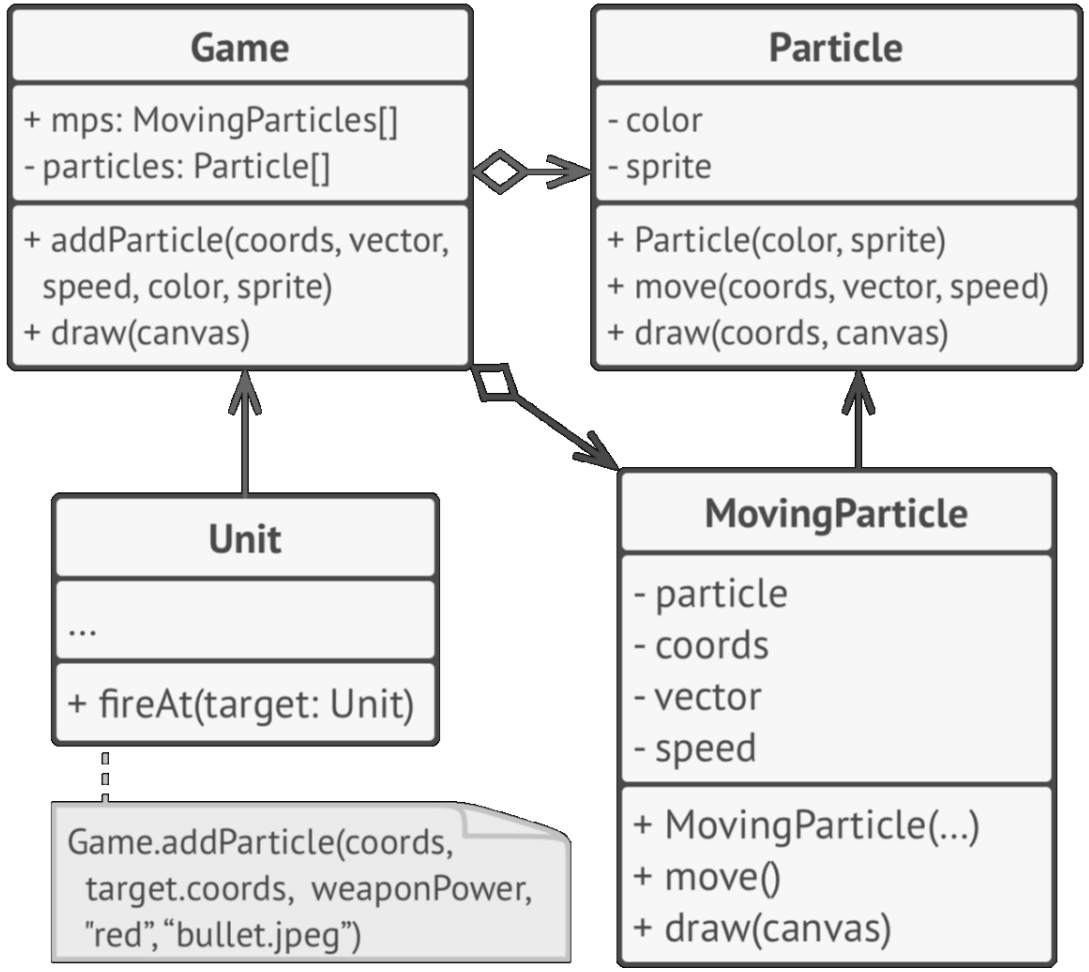
- Fit more objects into RAM by sharing common parts of state between multiple objects

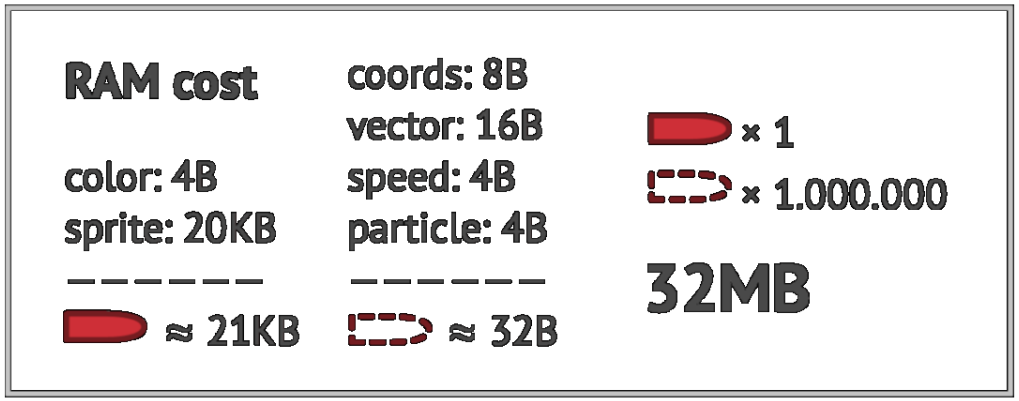


Example: Game Displaying

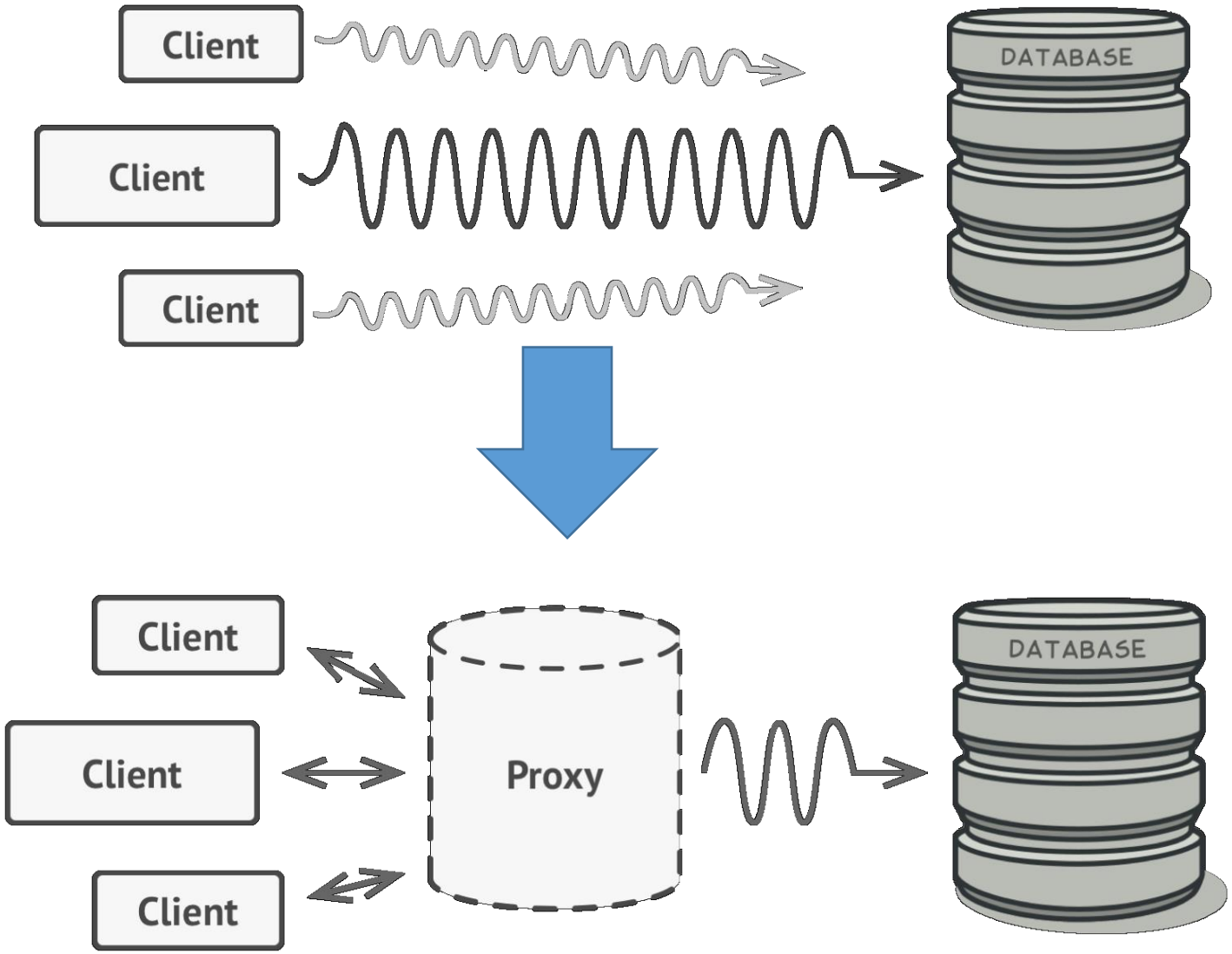



Unique (extrinsic) state (mutable)
Repeating (intrinsic) state (immutable)
Particle
+ coords
+ vector
+ speed
+ color
+ sprite
+ move()
+ draw(canvas)
MovingParticle
- particle
- coords
- vector
- speed
+ MovingParticle(...)
+ move()
+ draw(canvas)
Lots
Few
Particle
- color
- sprite
+ Particle(color, sprite)
+ move(coords, vector, speed)
+ draw(coords, canvas)
particle.move(
coords, vector, speed)
particle.draw(
coords, canvas)


Game
+ mps: MovingParticles[]
- particles: Particle[]
+ addParticle(coords, vector, speed, color, sprite)
+ draw(canvas)
Particle
- color
- sprite
+ Particle(color, sprite)
+ move(coords, vector, speed)
+ draw(coords, canvas)
Unit
...
+ fireAt(target: Unit)
Game.addParticle(coords, target.coords, weaponPower, "red", "bullet.jpeg")
MovingParticle
- particle
- coords
- vector
- speed
+ MovingParticle(...)
+ move()
+ draw(canvas)

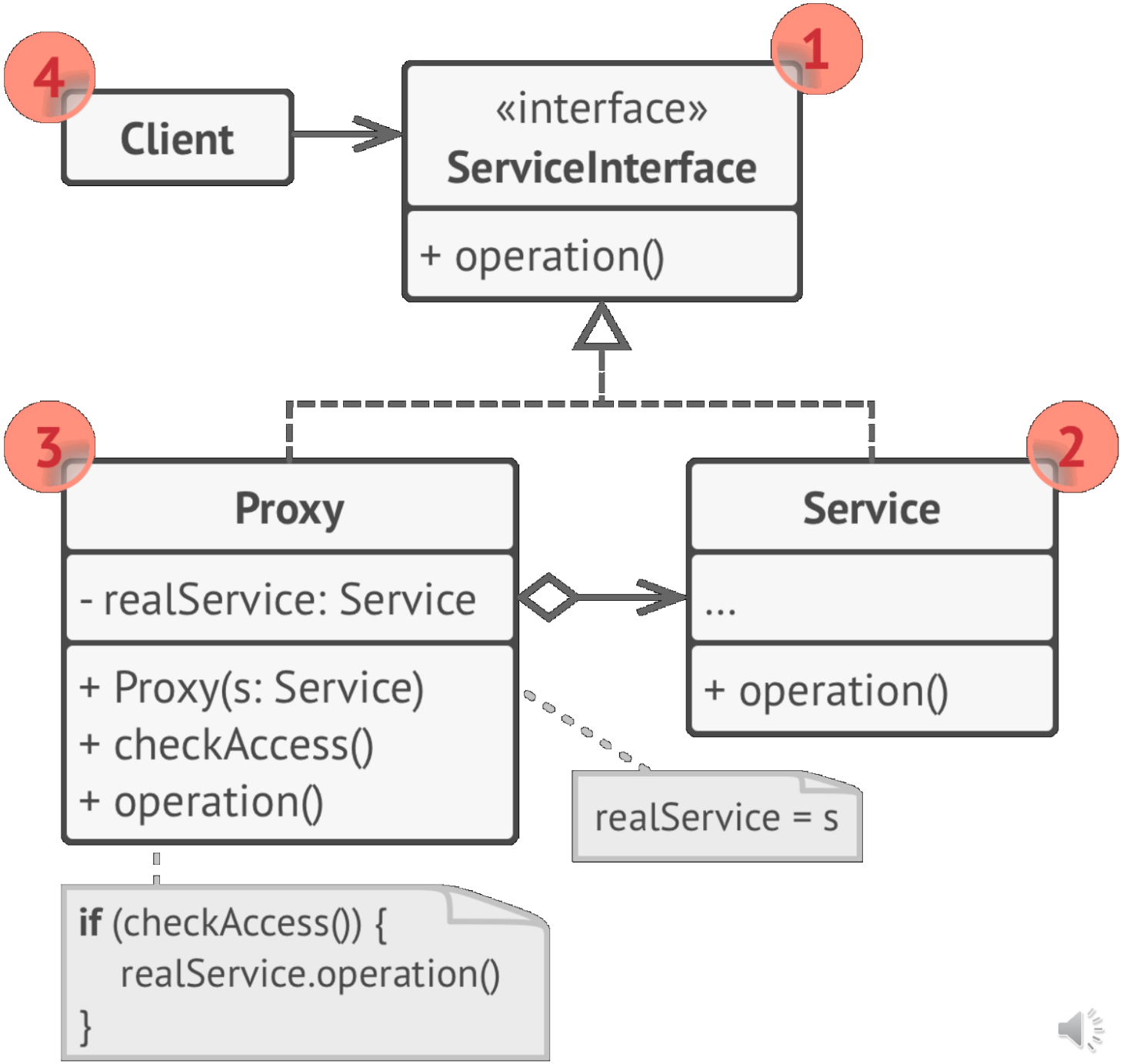

RAM cost
color: 4B
sprite: 20KB
≈ 21KB
coords: 8B
vector: 16B
speed: 4B
particle: 4B
≈ 32B
× 1
× 1.000.000
32MB

Proxy

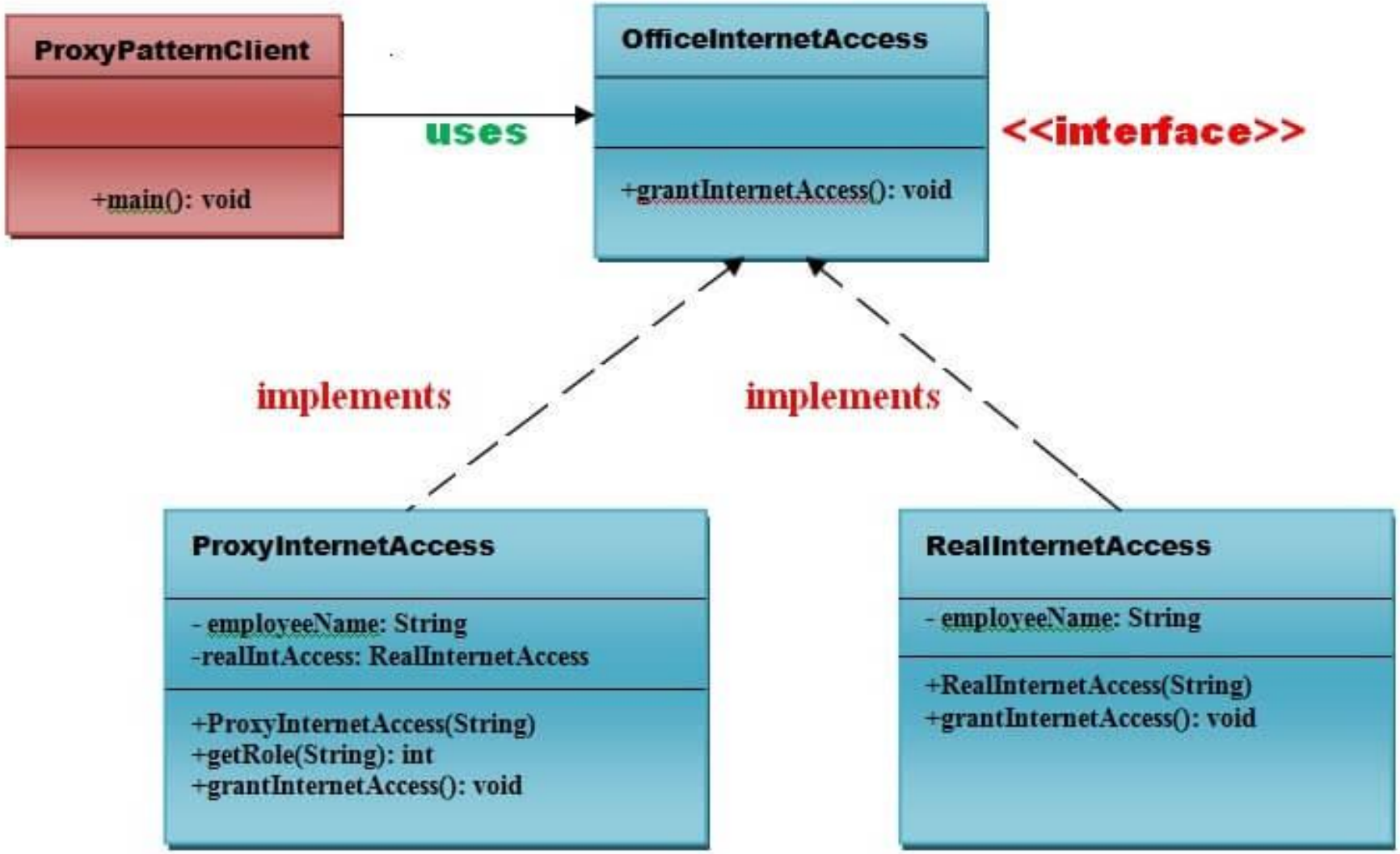

Client
Client
Client
DATABASE
Client
Client
Client
Proxy
DATABASE

Proxy Structure

- Lazy initialization
- Access control
- Local execution of a remote service
- Logging requests
- Caching results
- Garbage collection



Example: Internet Proxy



https://www.javatpoint.com/proxy-pattern

References

- Alexander Shvets, “Dive into Design Patterns,” 2018
- https://www.tutorialspoint.com/design_pattern/index.htm
- https://www.javatpoint.com/design-patterns-in-java
- https://www.startertutorials.com/patterns/select-design-pattern.html