# CSC207H Lecture 7

Sadia Sharmin

Oct 25, 2016

# Composite Design Pattern

Scenario:

- Need to manipulate hierarchical collection of 'primitive' and 'composite' objects
- Use this pattern whenever you have "composites that contain components, each of which could be a composite".

Source: `https://sourcemaking.com/design_patterns/composite`

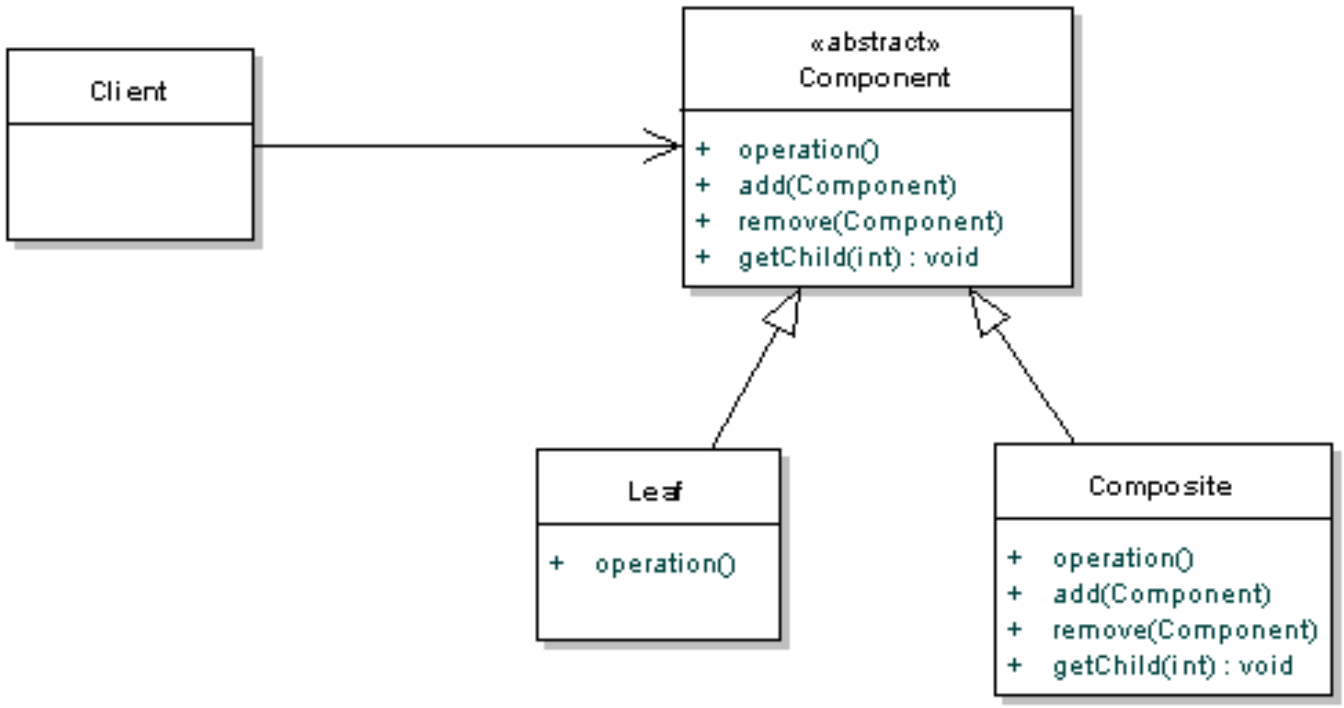
# Composite Design Pattern

- When building a system where a component could either be an individual object or a representation of a collection of objects
- Anything that can be modeled as a tree structure could use the composite pattern
  - e.g. a menu system where a menu bar has a menu with many menu items, which themselves can have submenus
- We can think of a composite as a collection of objects, where any one of these objects could itself be a composite, or a simple object

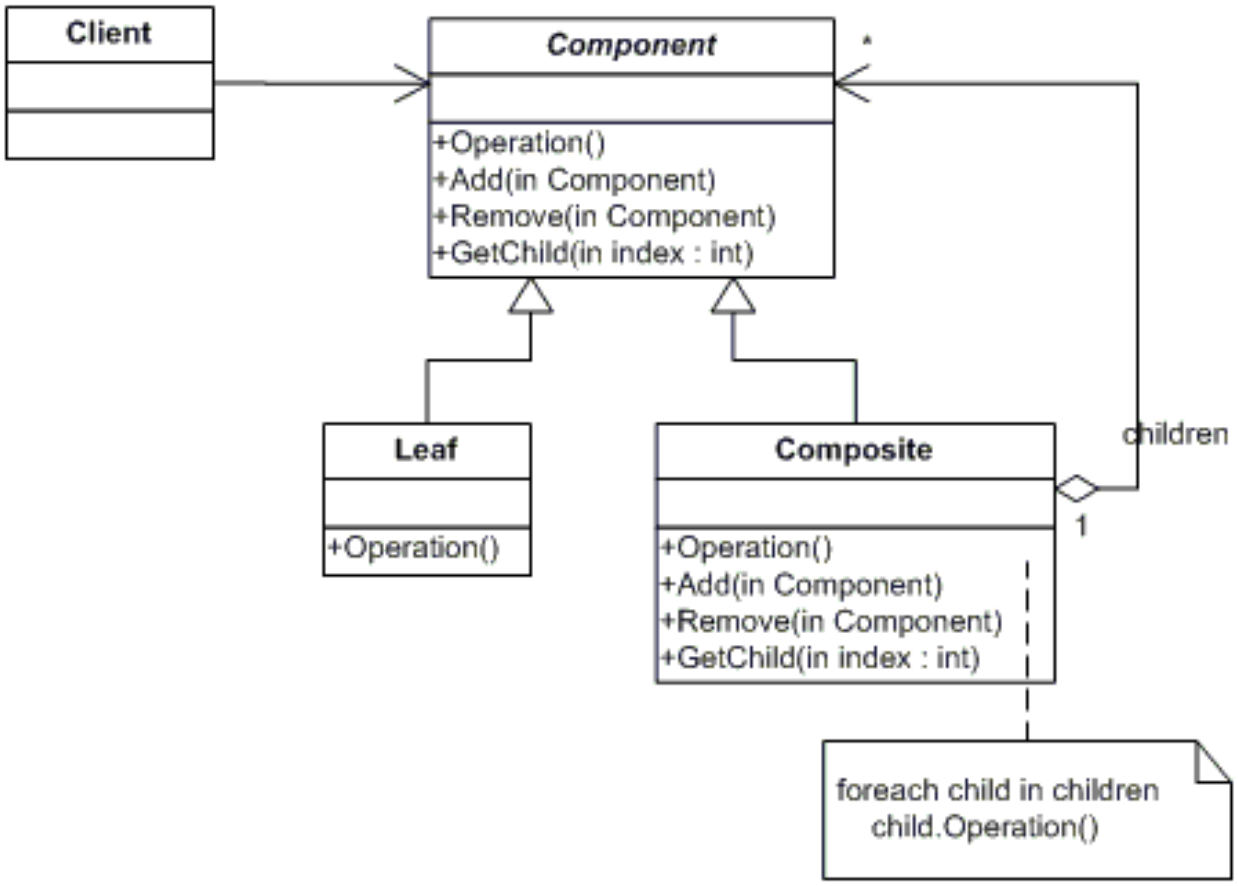
Source:

`https://dzone.com/articles/design-patterns-composite`

# Composite Design Pattern



# Composite Design Pattern



# Composite: What You Need

- A Component interface: includes the operations that both leafs (simple objects) or compositions need to use
- Usually implemented as abstract class with default behaviour for add, remove and getChild

# Command Design Pattern

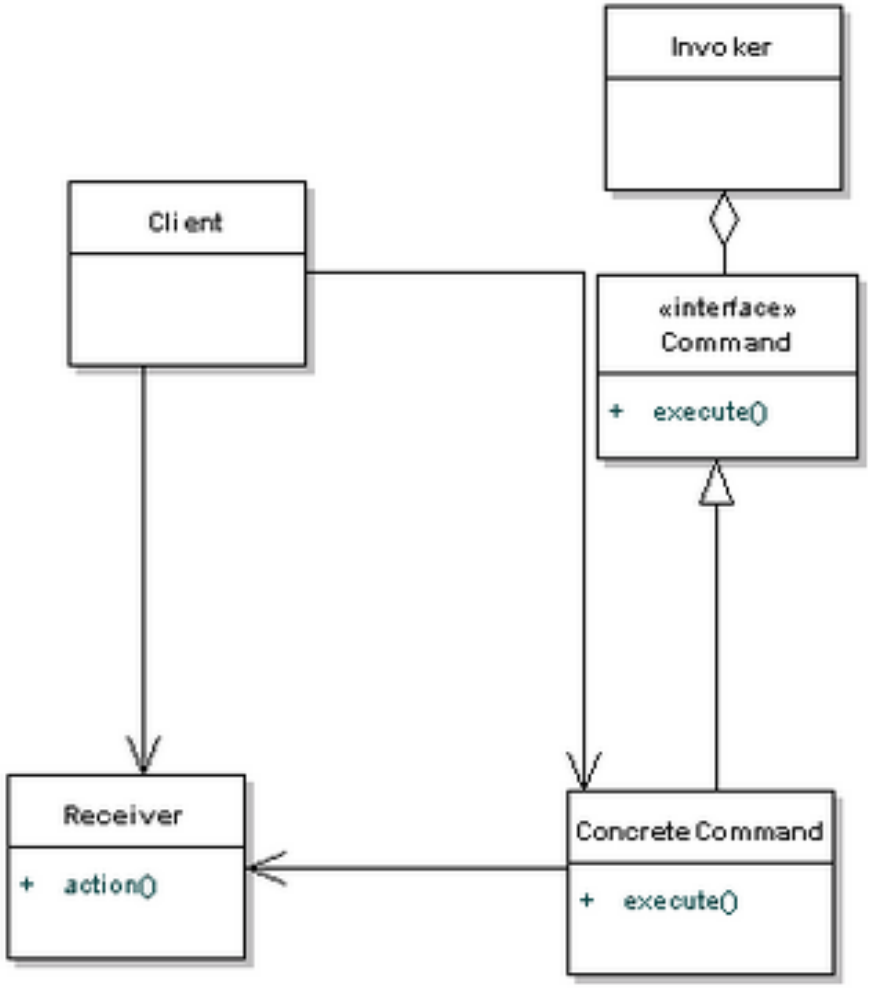
Scenario:

- Need to issue requests to objects
- Keep the object that invokes the operation from the one that knows how to perform it

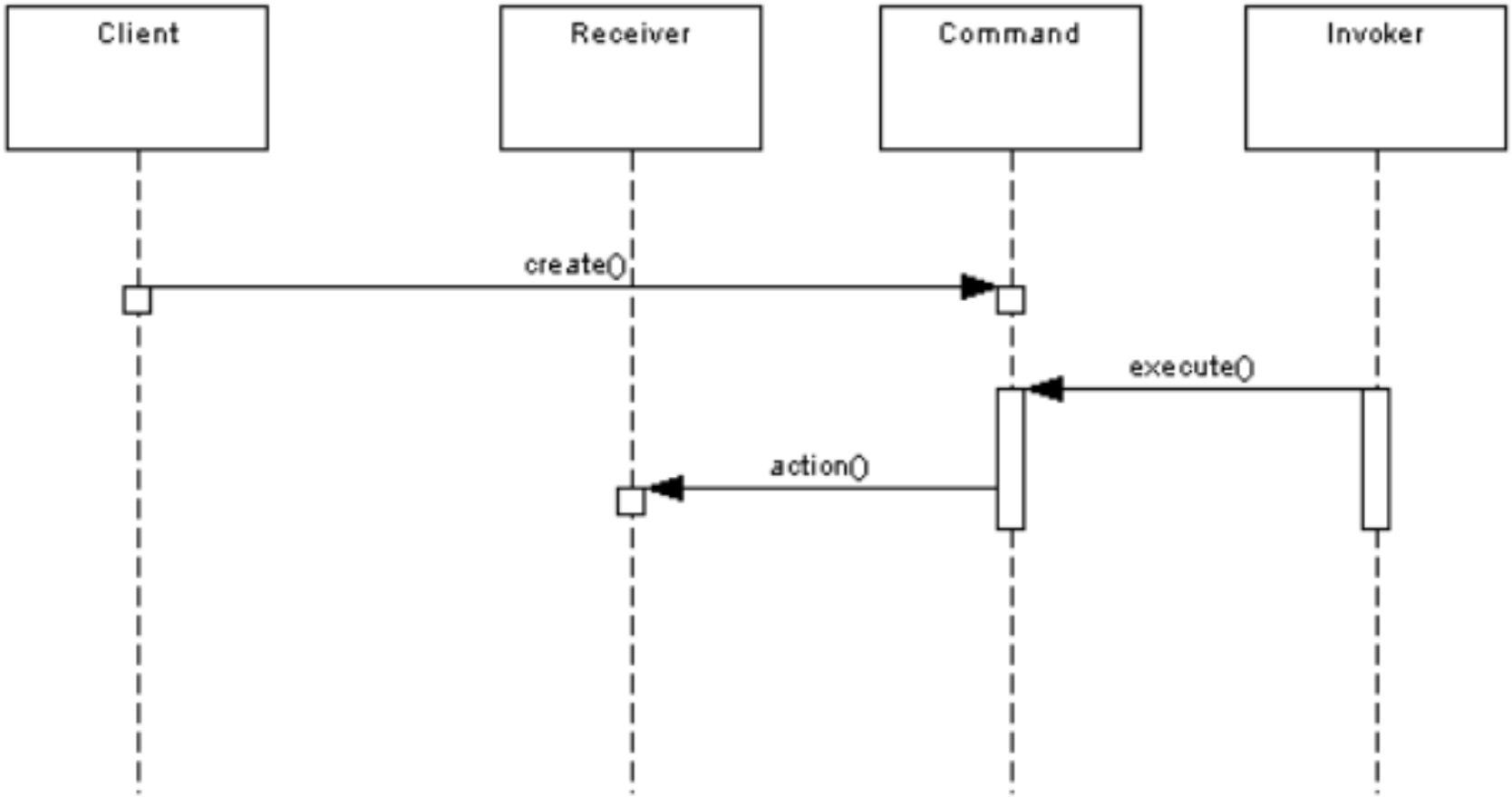
# Command: What You Need

- Command interface for executing operations: has an execute() method
- Actual concrete commands: implements Command
- Invoker: asks the command to carry out the request
- Receiver (the main thing that's being dealt with): knows how to carry out requests when a command is executed
- Client: creates a command and sets its receiver

# Command Design Pattern



# Command Design Pattern

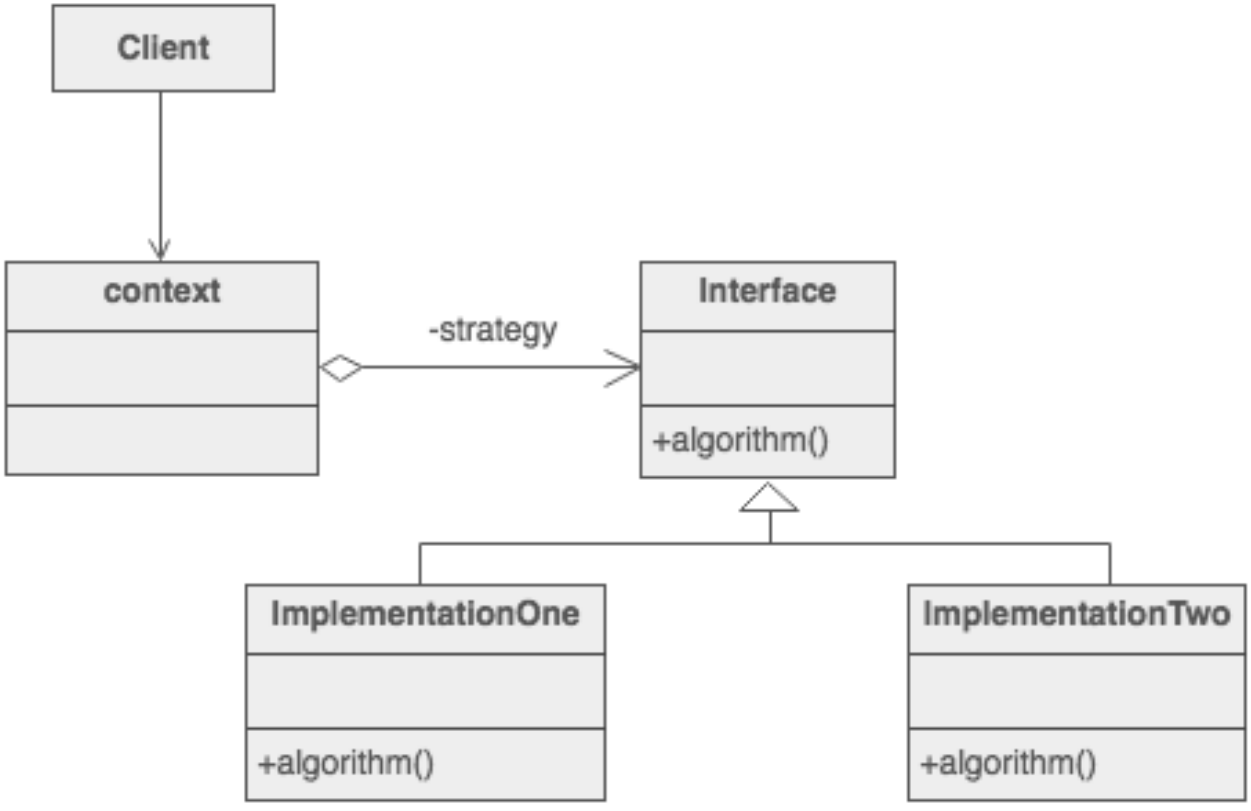


# Strategy Design Pattern

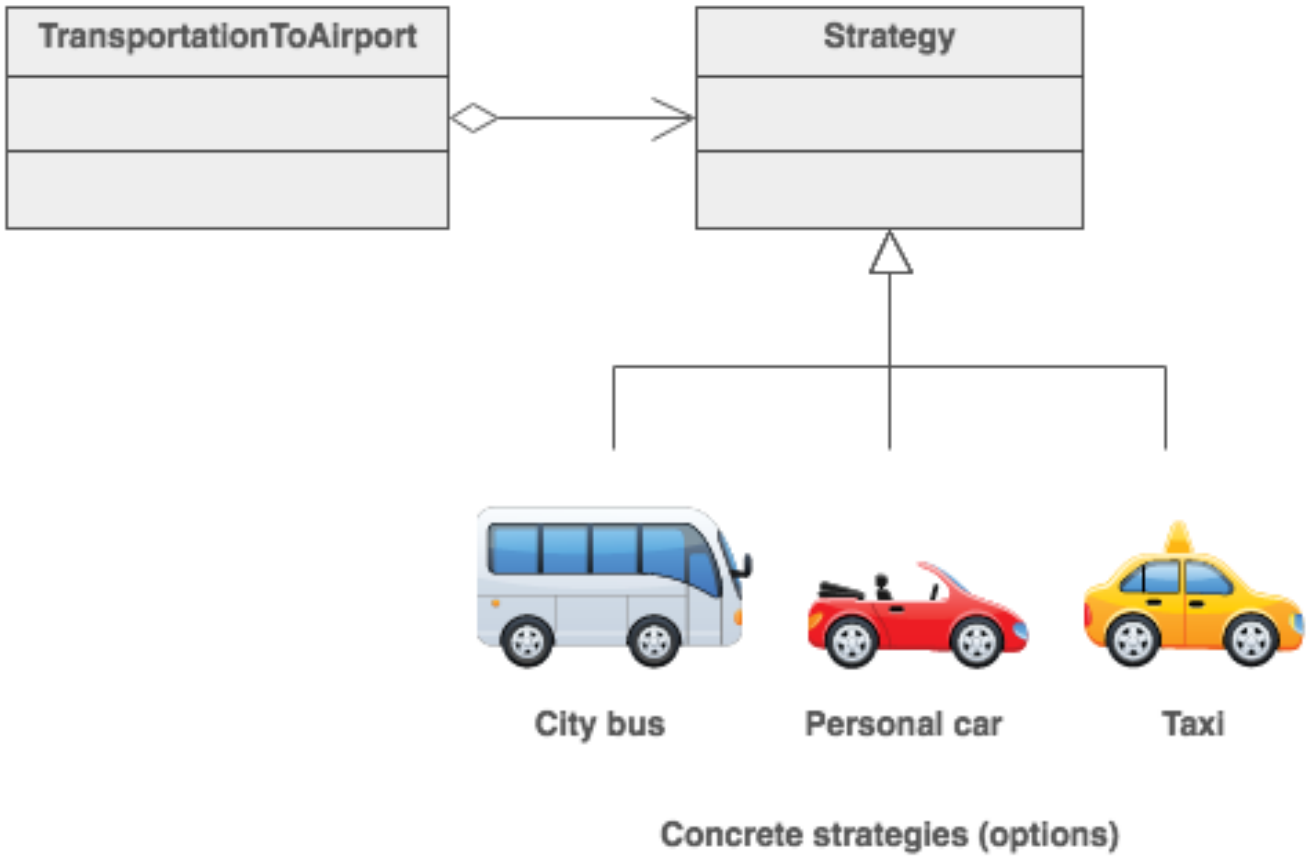
Scenario:

- Involves a family of algorithms (different approaches to same problem) that can be interchangeable
- Need an application's behaviour to be set at runtime (allows the client to choose which algorithm to use)

# Strategy Design Pattern



# Strategy Design Pattern



# Strategy: What You Need

- Strategy: a common interface for all the algorithms in this family
- Concrete strategies that use the above interface
- Context: sets the strategy and uses it