Design patterns
Part III

Bartosz Walter
<Bartek.Walter@man.poznan.pl>

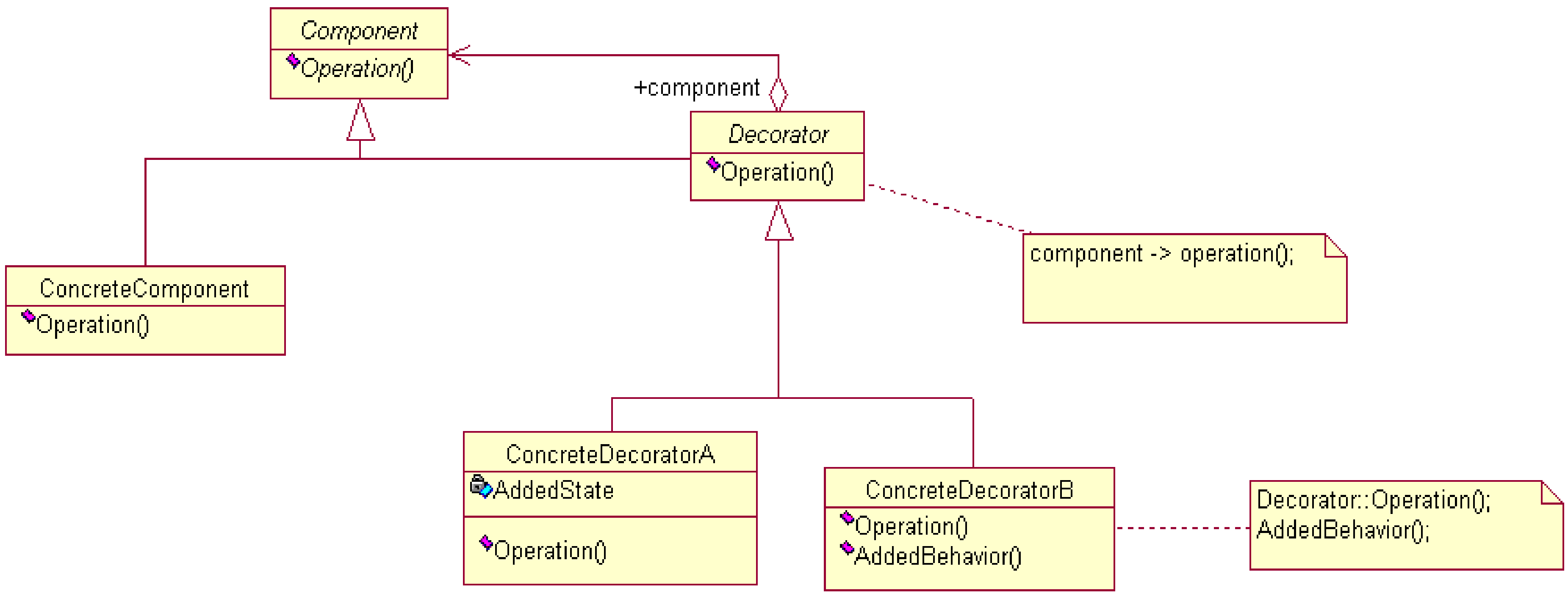
Catalog of Design Patterns

Decorator: Intent

- **Attach additional responsibilities to an object dynamically.**
- Provide a flexible alternative to subclassing for extending functionality.

by the Gang of Four

Decorator: Structure



by the Gang of Four

Decorator: Participants

- **Component**

  declares an interface for objects rhat can have additional responsibilities

- **Concrete Component**

  implements the Component interface

- **Decorator**

  declares an interface that conforms to Component's interface

  knows about Component

- **Concrete Decorator**

  adds responsibilities to the component

by the Gang of Four

Decorator: Consequences

- **more flexible than static inheritance**
  - responsibilities can be added at run-time
  - less complex class hierarchy
- **pay-as-you-go**
  - **features are added whenever needed**
- **object identities differ**
  - **object identity is different from Decorator's identity**
- **lot of small decorating classes**

by the Gang of Four

Decorator: Implementation

- **Conformance of interfaces**

  Decorator and Component must implement same interface

- **Default Decorator**

- **Keeping Component lightweight**

  the data storage should be deferred to subclasses

- **Strategy vs. Decorator**

  Strategy deals with changing the guts

  Decorator deals with changing the skin

by the Gang of Four

Decorator: Example of use

The hierarchy of java.io.* classes.

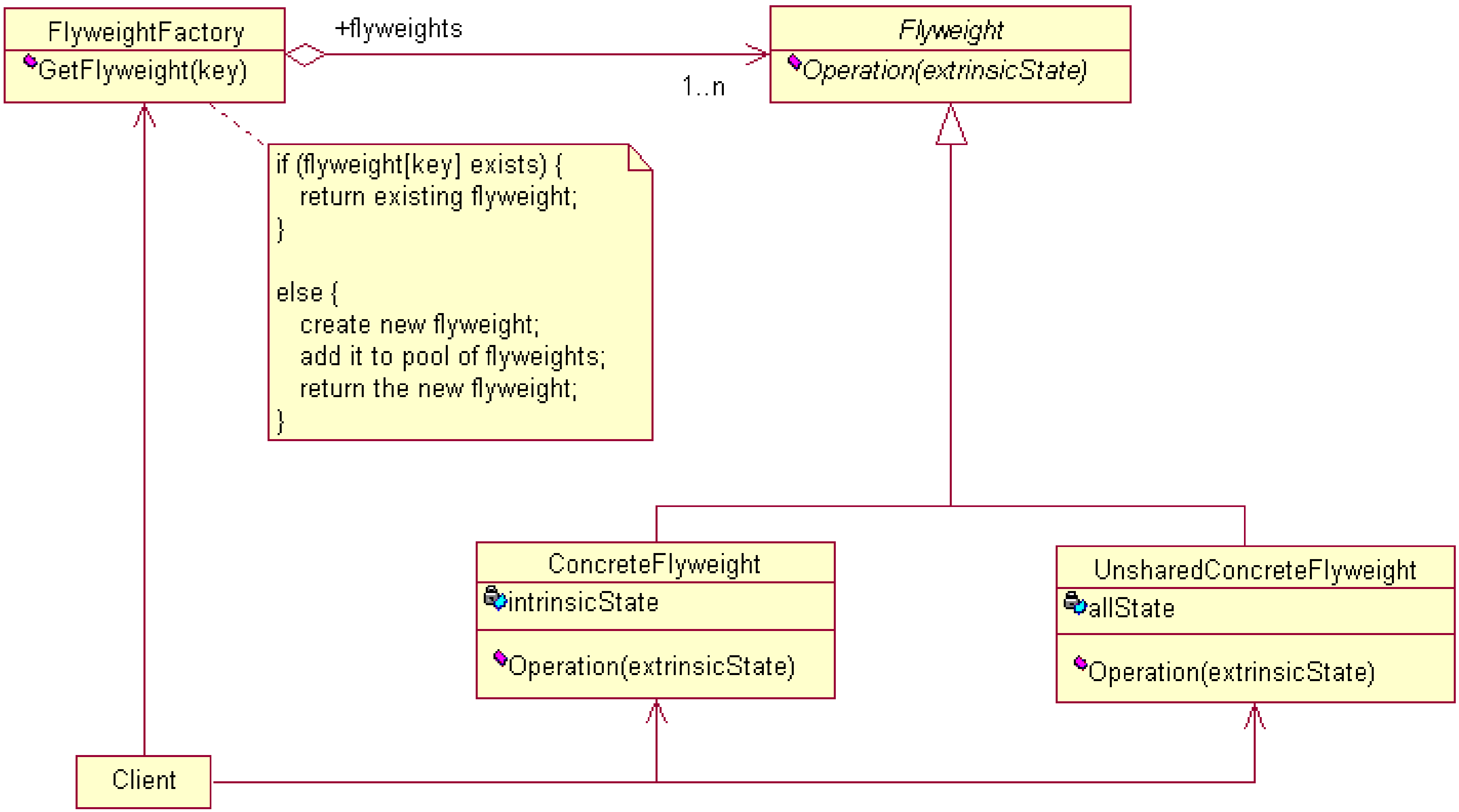
FilterInputStream is a decorator for InputStream. Its subclasses provide decorated implementations for the InputStream methods, and then forward the requests to the component object (InputStream).

Flyweight: Intent

Use sharing to **support large numbers of fine-grained objects efficiently**

by the Gang of Four

Flyweight: Structure



by the Gang of Four

Flyweight: Participants

- **Flyweight**

  declares an interface through which flyweights can receive and act on extrinsic state

- **Concrete Flyweight**

  adds storage for intrinsic state (if any)

  must be independent of its context

- **Unshared Concrete Flyweight**

  non-sharable flyweight

- **Flyweight Factory**

  creates and manages flyweight objects

  ensures that flyweights are shared properly

- **Client**

  implements Implementor interface

by the Gang of Four

Flyweight : Consequences

- **growing space savings**
  - reduction of total number of instances
  - reduction of intrinsic state
  - extrinsic state may be computed or stored
- **run-time costs**
  - managing the extrinsic state

by the Gang of Four

Flyweight: Example of use

Dealing with icons displayed on the screen.

The icons share most of their data, except for the name and pointer to the underlying object. Therefore they can implement the flyweight: the core of the object is shared among them, and parametrized with their intrinsic state.

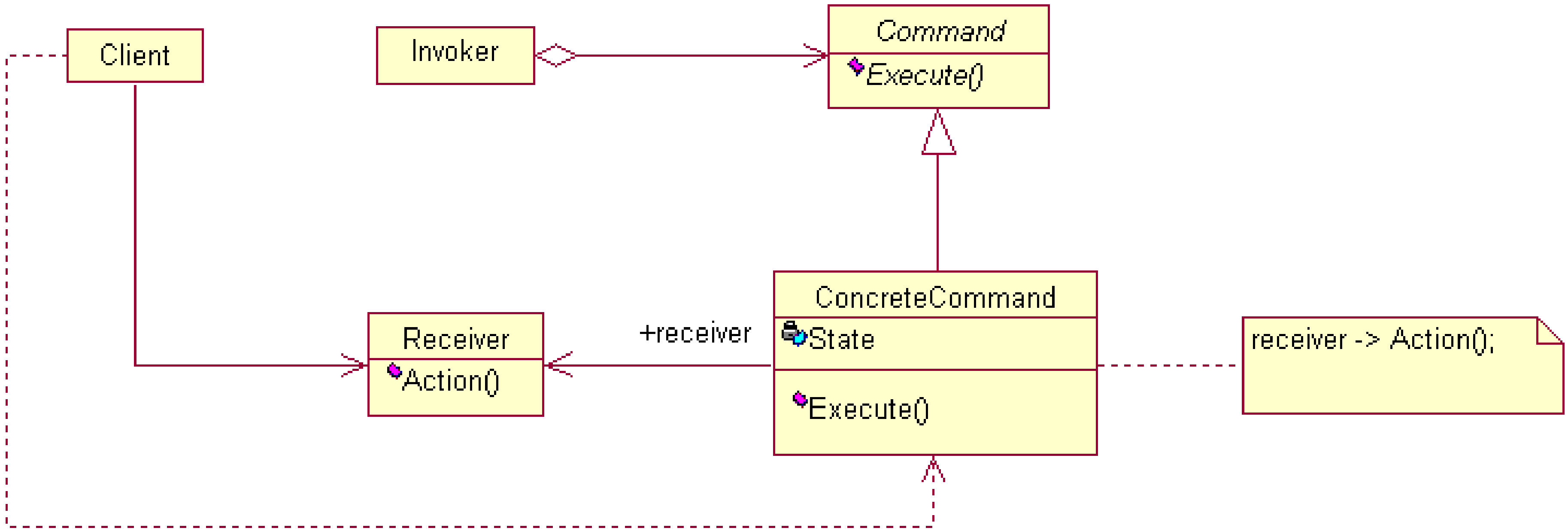
by the Gang of Four

Command: Intent

- **Encapsulate a request as an object.**
- Allow **parametrizing clients with different requests**.
- Support **undoable operations**

by the Gang of Four

Command: Structure



by the Gang of Four

Command: Participants

- **Command**

  declares an interface for executing an operation

- **ConcreteCommand**

  defines a binding between a Receiver and an action

  implements *execute()* method

- **Client**

  creates a ConcreteCommand object and sets its receiver

- **Invoker**

  asks the command to carry out the request

Receiver

knows how to perform the concrete operations

by the Gang of Four

Command: Consequences

- decoupling the sender from receiver
- Commands can be manipulated and extended like any other object
- Commands can be assembled into a composite command
- adding new Commands is easy

by the Gang of Four

Command: Example of use

In the menu bar, after an item is clicked, an appropriate command is issued and executed.

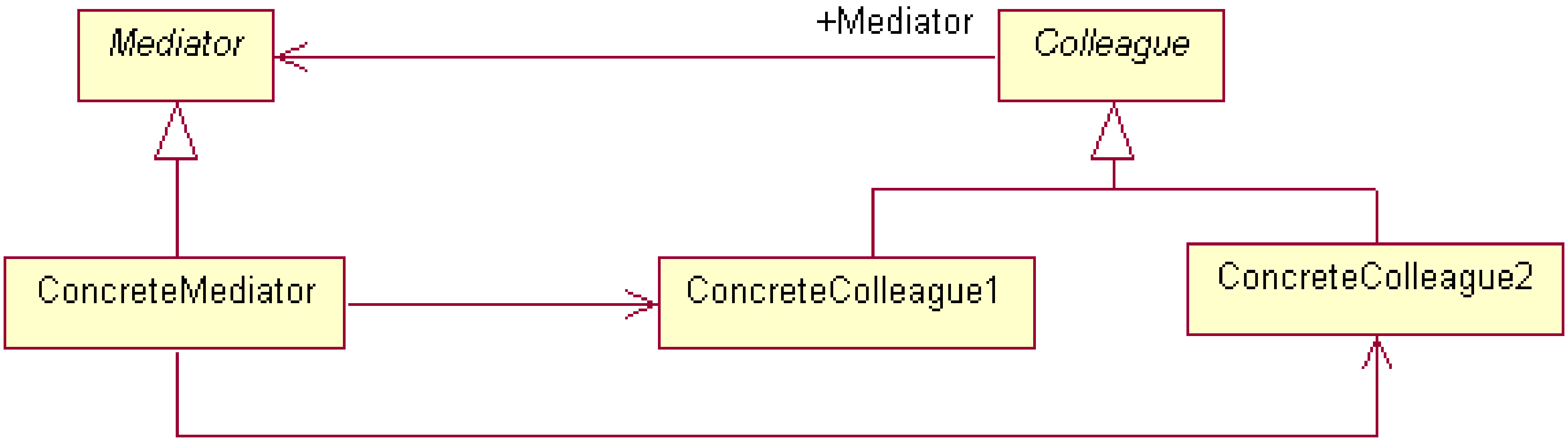
The MenuItem stores commands and invokes appropriate one for the given receiver (a document etc.). Concrete Commands handles the request.

Mediator: Intent

- Define an object that **encapsulates how a set of objects interact**.
- Promote loose coupling by **keeping objects from referring to each other explicitly**.
- Allow **varying their interaction independently**.

by the Gang of Four

Mediator: Structure



by the Gang of Four

Mediator: Participants

- **Mediator**

  defines an interface for communicating woth Colleague objects

- **Concrete Mediator**

  implements cooperative behaviour by coordinating Colleagues

  knows and maintains its Colleagues

- **Colleague classes**

  each of them knows its Mediator

  colleage communicates with Mediator instead of another Colleague

by the Gang of Four

Mediator: Consequences

- **limited subclassing**
  - Mediator localizes behavior that otherwise would be distributed
  - changing the behavior requires subclassing the Mediator only
- **decoupling the colleagues from each other**
- **simplified object protocols**
  - Mediator replaces a many-to-many associations with one-to-many
- **centralized control**
  - trade-off between complexity of interaction with compexity of mediator
  - mediator becomes a hard to maintain monolith

by the Gang of Four

Mediator: Example of use

Communication of related widgets in a graphical application.

Routing the requests

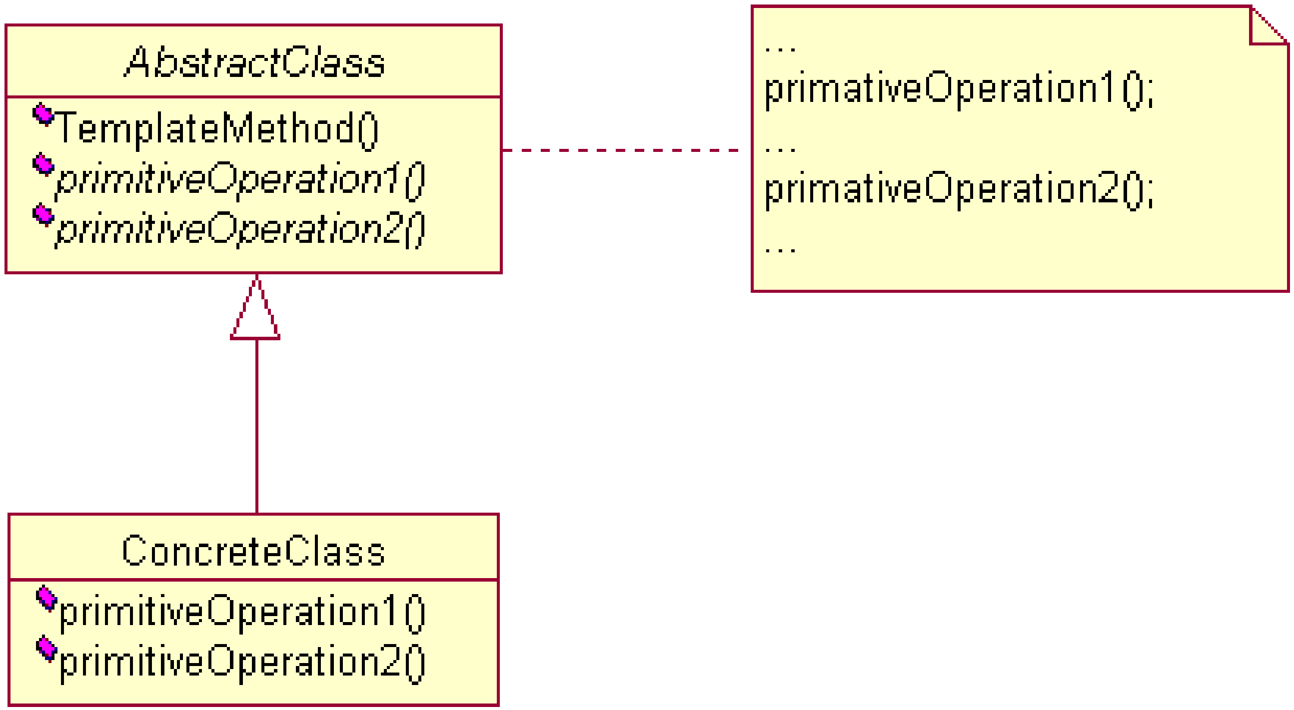
by the Gang of Four

Template Method: Intent

- Define the **skeleton of an algorithm in an operation**, deferring some steps to subclasses.
- **Allow subclasses redefining certain steps** of an algorithm without changing the algorithm's structure.

by the Gang of Four

Template Method: Structure



by the Gang of Four

Template Method: Participants

- **AbstractClass**

  defines abstract primitive operations

  implements a template method defining the skeleton of an algorithm

- **ConcreteClass**

  implements primitive operations

by the Gang of Four

Template Method: Consequences

- **inverted control structure**

  don't call us, we will call you

- **different kinds of operations called by TM**

  concrete operations (on Concrete Class or client classes)

  concrete Abstract Class operations (methods useful for subclasses)

  primitive operations (abstract methods)

  factory methods

  hook operations (default behavior that can be extended)

by the Gang of Four

Template Method: Example of use

Generic algorithms with hooks and abstract methods.

Sorting algorithms can extend a common class which specifies methods to implement and reuse.

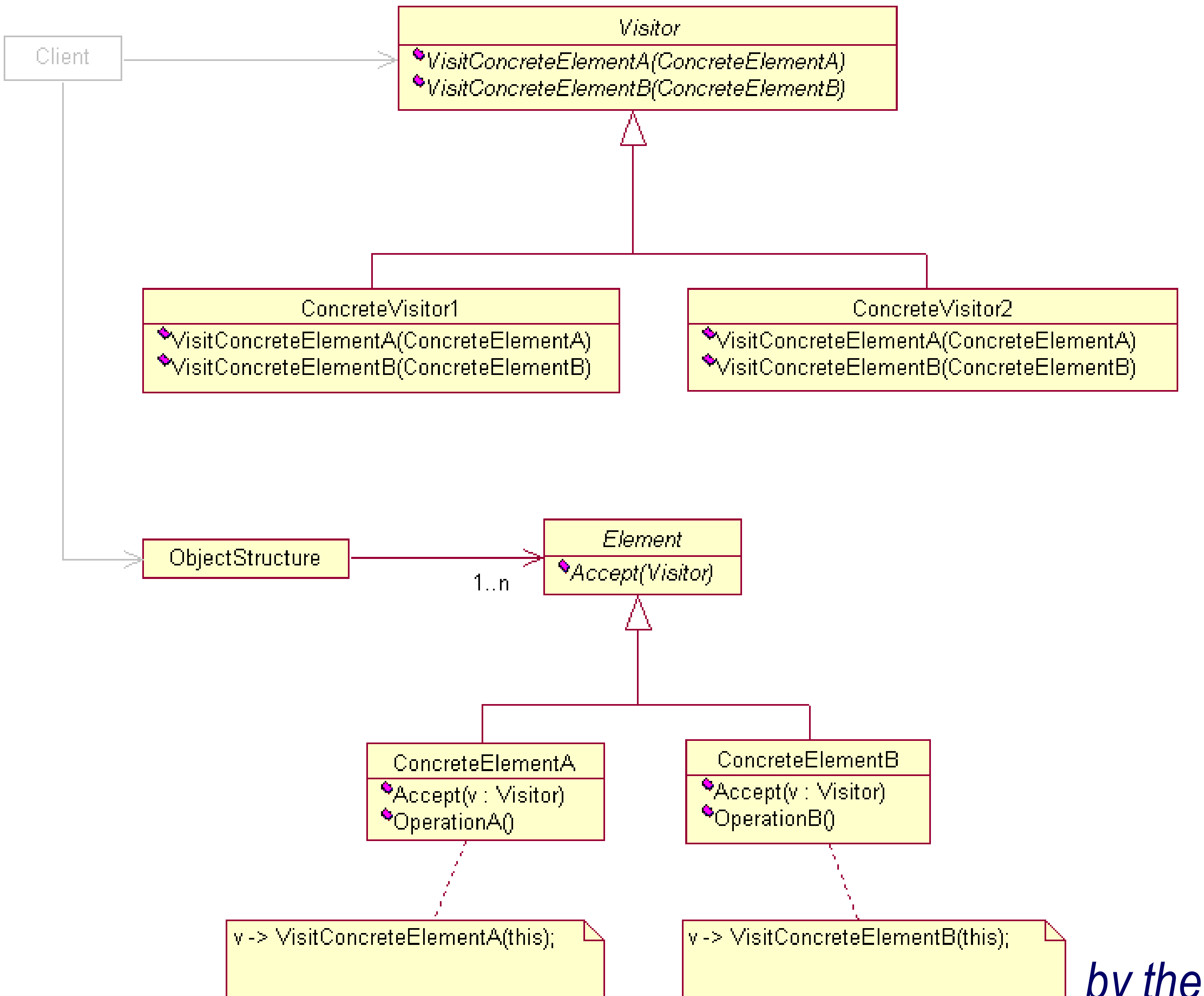
by the Gang of Four

Visitor: Intent

- **Represent an operation to be performed on the elements of an object structure**.
- Allow **defining a new operation without changing the classes of the elements** on which it operates.

by the Gang of Four

Visitor: Structure



by the Gang of Four

Visitor: Participants

- **Visitor**

  declares operations for every ConcreteElement to be visited

- **Concrete Visitor**

  implements the operations

- **Element**

  defines *accept()* operation parametrized with Visitor

- **Concrete Element**

  implements *accept()* operation

- **Object Structure**

  can enumerate its elements

  may provide a high-level interface to allow the visitor to visit its elements

by the Gang of Four

Visitor: Consequences

- **easy adding new operations**

  new Visitors can traverse the object structure

- **gathering related operations and separation of unrelated ones**

  related behavior is localized in a Visitor

  unrelated sets of behavior are partitioned in their own Visitor subclasses

- **difficult adding new Concrete Elements**

  each Concrete Element gives rise to a new operation on Visitor and corresponding Concrete Visitors

- **visiting across class hierarchies**

  unlike Iterator, the Visitor can visit objects of different classes

by the Gang of Four

Visitor: Consequences (cont.)

- **accumulating state**

  Visitors can accumulate state during the object traversal

- **breaking encapsulation**

  pattern often enforces existance often public operations that access an element's internal state

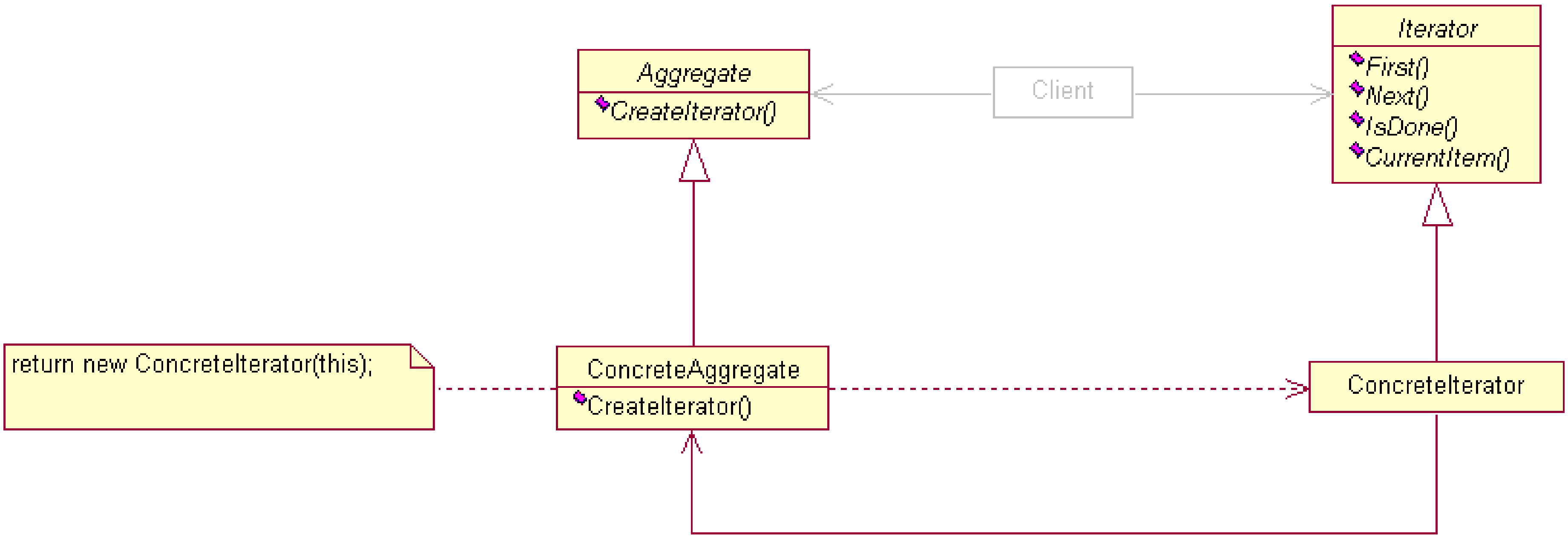
by the Gang of Four

Iterator: Intent

Provide a way **to access elements of an aggregate sequentially** without exposing its internal structure.

by the Gang of Four

Iterator: Structure



by the Gang of Four

Iterator: Participants

- **Iterator**

  declares an interface for accessing and iterating through aggregates

- **Concrete Iterator**

  performs necessary computations

- **Aggregate**

  declares an interface for creating Iterator

- **Concrete Aggregate**

  implements Iterator interface

by the Gang of Four

Iterator: Consequences

- **supports multiple variations of traversing the aggregate**

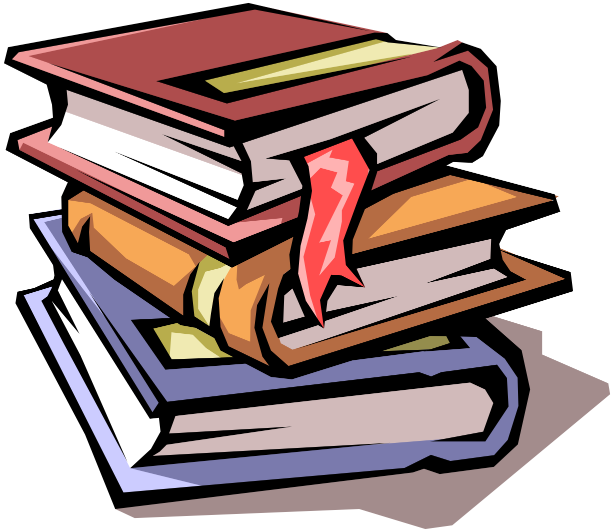
  adapts only to a concrete class, not subclasses

- **multiple traversal allowed**

  each Iterator keeps track of the running traversal

by the Gang of Four

Readings



1. Gamma E. et al., *Design Patterns. Elements of Reuseable Object-Oriented Software.* Addison-Wesley, 1995
2. Eckel B., *Thinking in patterns*. http://www.bruceeckel.com
3. Cooper J., *Java. Wzorce Projektowe.* Helion, 2001
4. Shalloway A., Trott J., *Projektowanie zorientowane obiektowo. Wzorce projektowe.* Helion, 2001

Q & A