Ingegneria del Software

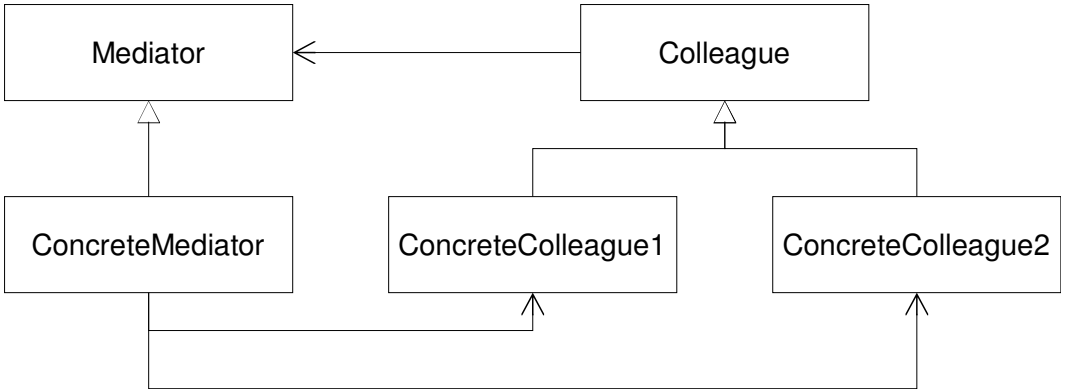
Corso di Laurea in Informatica per il Management

Design Patterns part 4

Davide Rossi
Dipartimento di Informatica
Università di Bologna

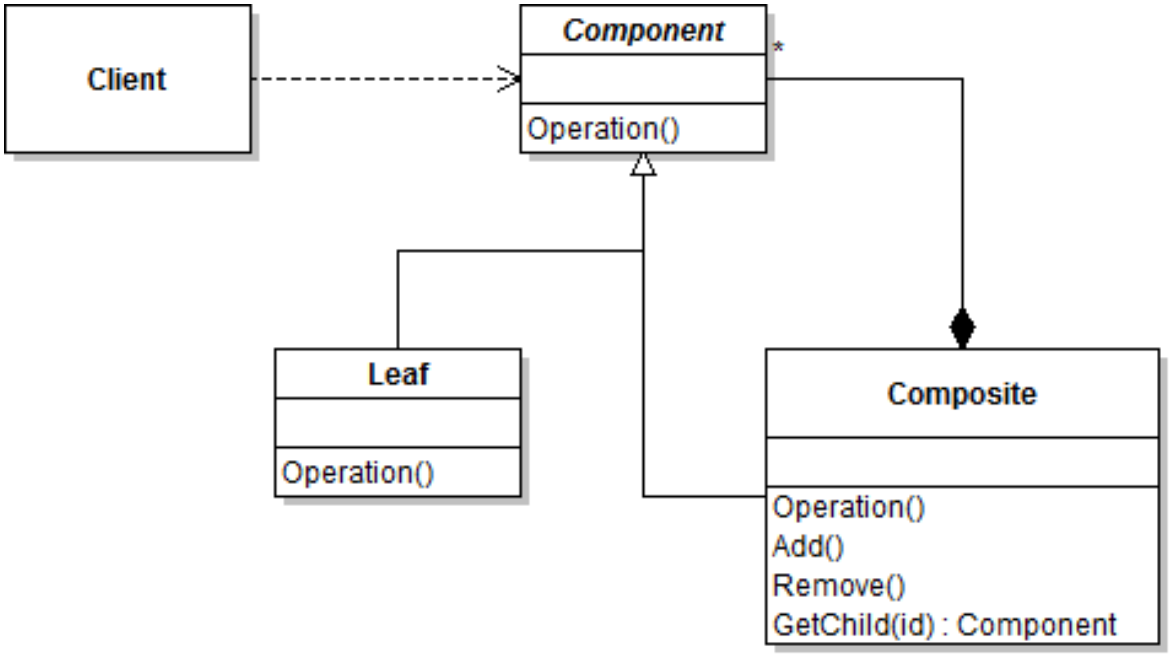
GoF: Mediator

- Define an object that encapsulates how a set of objects interact.
- Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and
- it lets you vary their interaction independently.



GoF: Composite

- Compose objects into tree structures to represent part-whole hierarchies
- Composite lets clients treat individual objects and compositions of objects uniformly

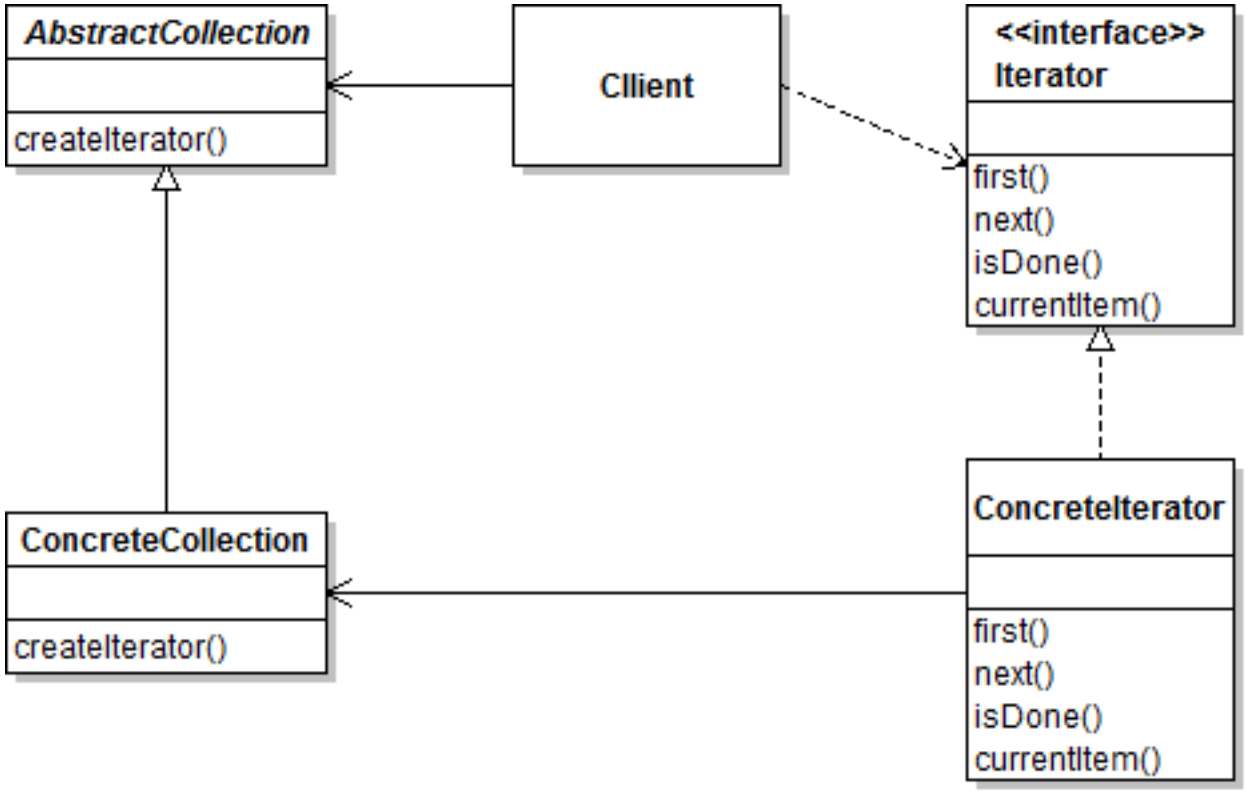


GoF: Memento

- Without violating encapsulation, capture and externalize an object's internal state so that the object can be restored to this state later.
- A *caretaker* asks *originator* for *memento*s that can be stored and used to restore originator’s state.

GoF: Iterator

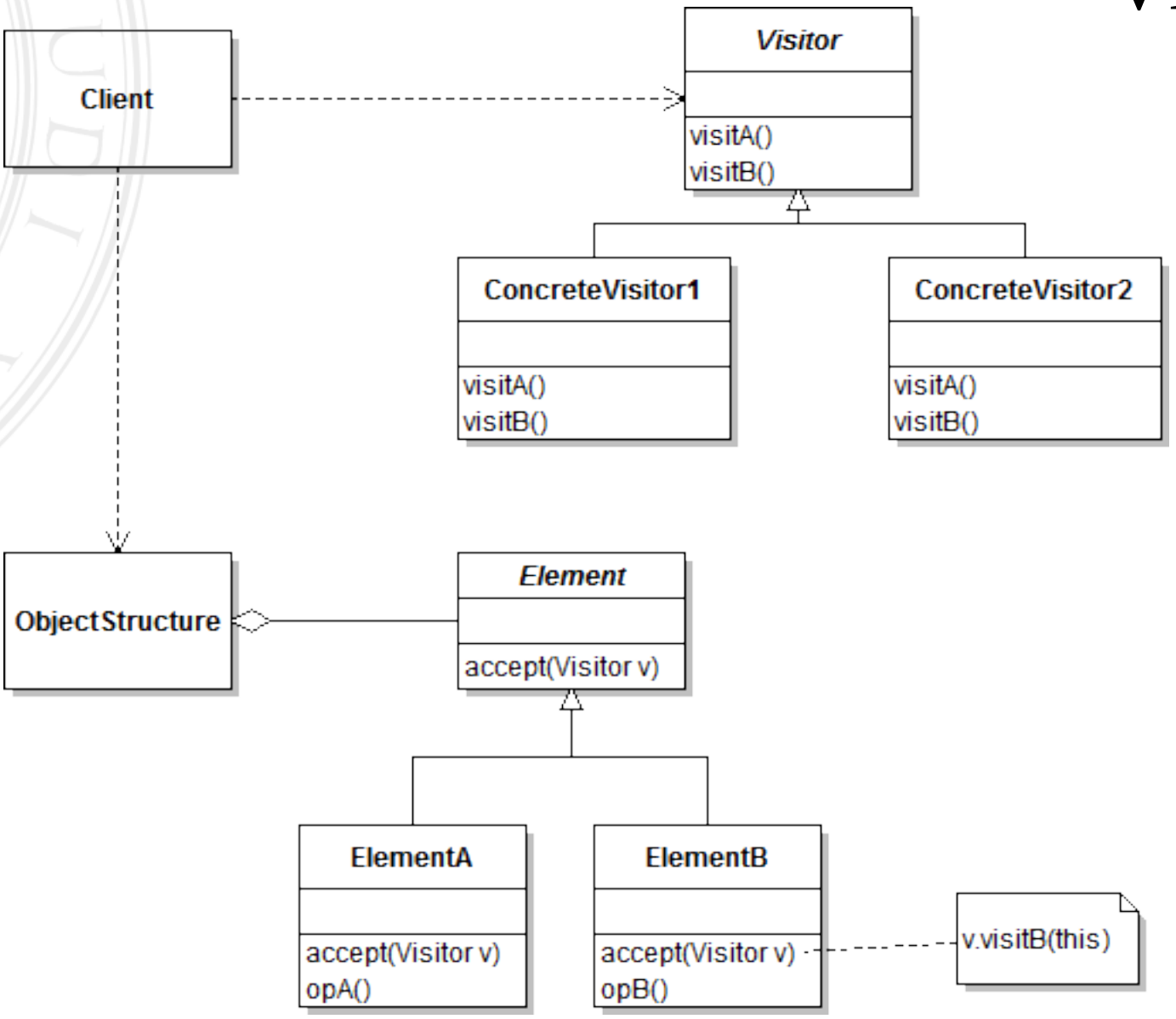
- Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.



GoF: Visitor

- Represent an operation to be performed on the elements of an object structure.
- Visitor lets you define a new operation without changing the classes of the elements on which it operates.
- Based on inversion of control

Visitor


Client
Visitor
visitA()
visitB()
ConcreteVisitor1
visitA()
visitB()
ConcreteVisitor2
visitA()
visitB()
ObjectStructure
Element
accept(Visitor v)
ElementA
accept(Visitor v)
opA()
ElementB
accept(Visitor v)
opB()
v.visitB(this)

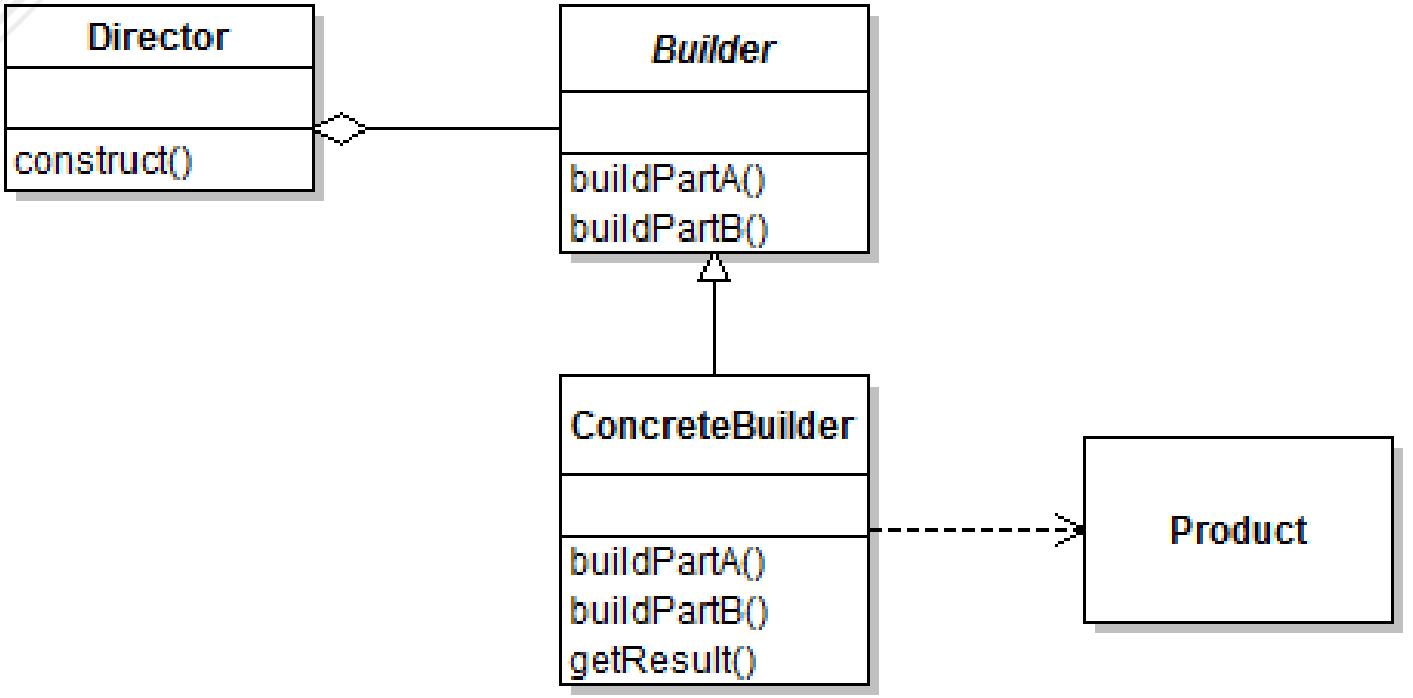
Visitor in the Java API

```
class java.nio.file.Files {
  public static Path walkFileTree
          (Path start, FileVisitor<? super Path> visitor)
  …
}

interface java.nio.file.FileVisitor {
  visitFile(T file, BasicFileAttributes attrs)
  …
}
```

GoF: Builder

- Separate the construction of a complex object from its representation so that the same construction process can create different representations.

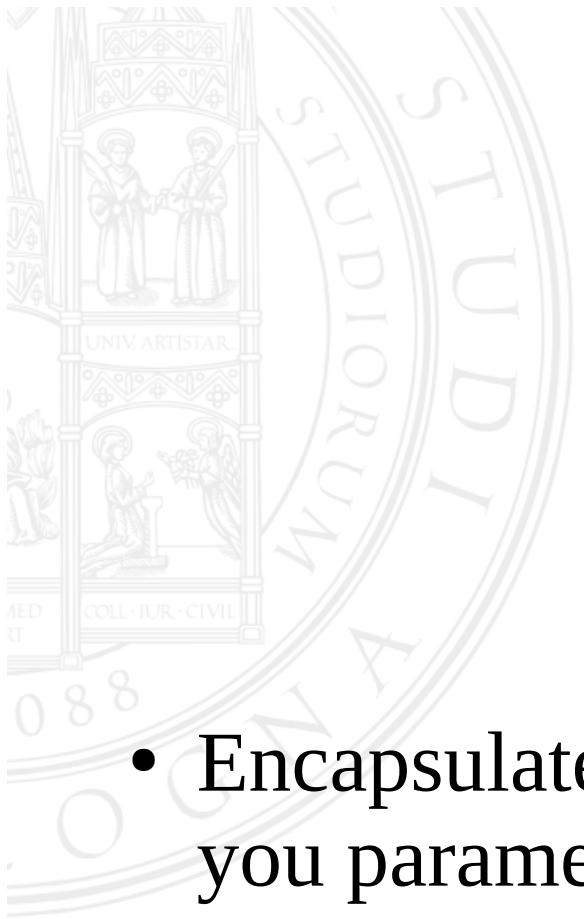


Minor but frequent issue solved by Builder: ugly constructors

- Foo foo = new Foo(a, b, null, null, c, null, d)

Builder in Java

- Foo foo =
  Foo.Builder.createBuilder().
  setWidth(a).setHeight(b).
  setDepth(c).setColor(d).build()



GoF: Command

- Encapsulate a request as an object, thereby letting you parametrize clients with different requests, queue or log requests, and support undoable operations.

GoF: Abstract Factory

- Provide an interface for creating families of related or dependent objects without specifying their concrete classes.

GoF: Prototype

- Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype.
- Create new instances by *cloning* existing ones.

GoF: Flyweight

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

GoF: Chain of Responsibility

- Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request.
- Chain the receiving objects and pass the request along the chain until an object handles it.

GoF: Interpreter

- Given a language, define a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language.

Resources

Books

- Eric Freeman & Elisabeth Robson, **Head First Design Patterns: Building Extensible and Maintainable Object-Oriented Software** (2nd Edition), O'Reilly

Online:

- http://www.vincehuston.org/dp/
- http://www.oodesign.com/
- https://refactoring.guru/design-patterns/
- http://www.informit.com/articles/article.aspx?p=1404056