Tecniche di Progettazione: Design Patterns

Laura Semini, Università di Pisa, Dipartimento di Informatica.


ATTENTION
BAD DESIGN
“CALATRAVA BRIDGE”

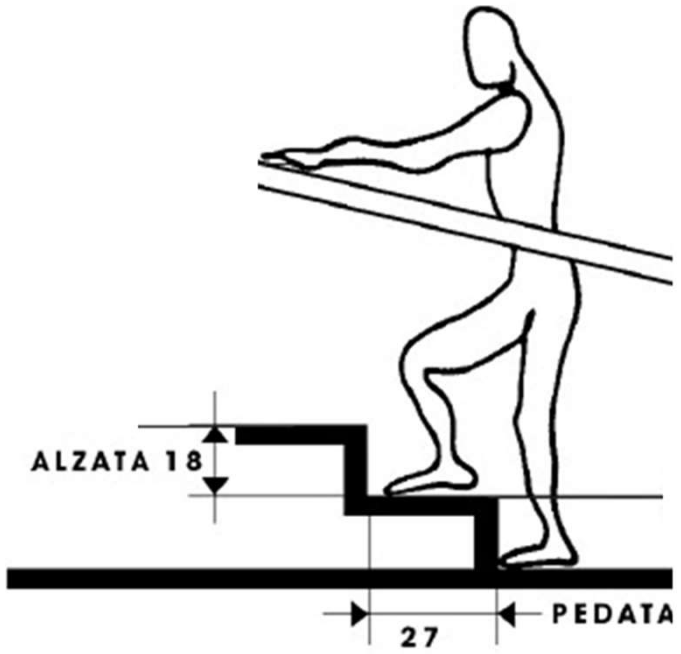
Pattern per un gradino

Formula di Blondel:

2Alzata + Pedata = 62 ÷ 65 cm

Questa formula ha dei limiti sulle piccole altezze di alzata:

> quando i gradini sono bassi conduce a dei **risultati insoddisfacenti.**



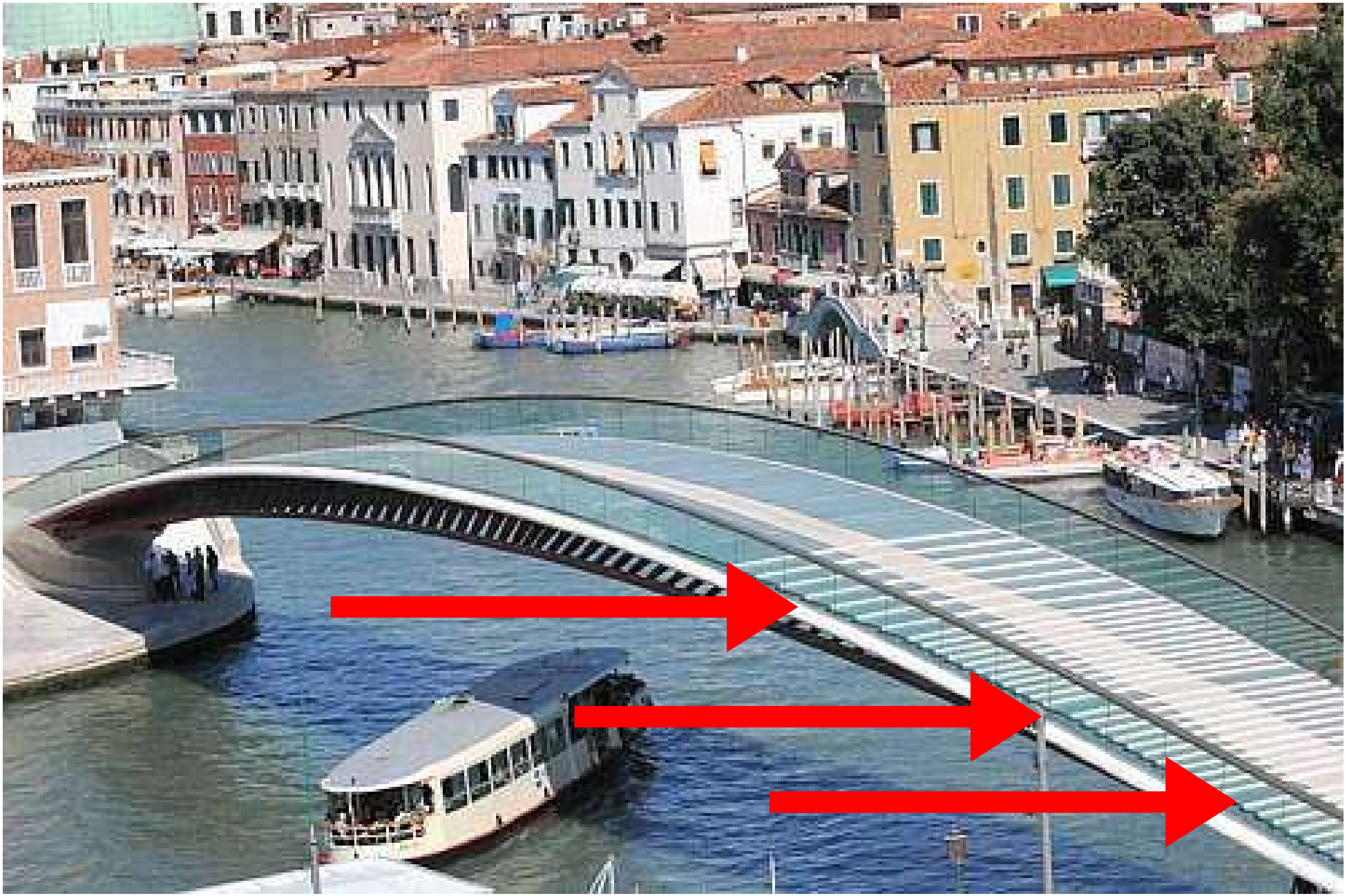
Regola di Hermant

Alzata * Pedata = 600

Ponte di Calatrava

Pedata= 50, Altezza = 8

- Blondel:50+2x8=66 $\approx$ 62 ÷ 65 cm
- Regola di Hermant: 50*8=400 $\neq$ 600



Materiali

Pietra d'Istria alternata alla trachite scura

Soluzione individuata dai veneziani già nel XV secolo.

Dov’è il gradino?

So what?

- Esistono una serie di *regole pratiche* che il progettista può seguire per costruire una scala:
  - Rapporto salita/corsa (Blondel e Hermant)
  - Materiali...
- Queste regole pratiche sono i **design patterns**.
- Sono state definite grazie a secoli di esperienza.

Il design non è soltanto un processo creativo

Carlo Scarpa a un giovane architetto:

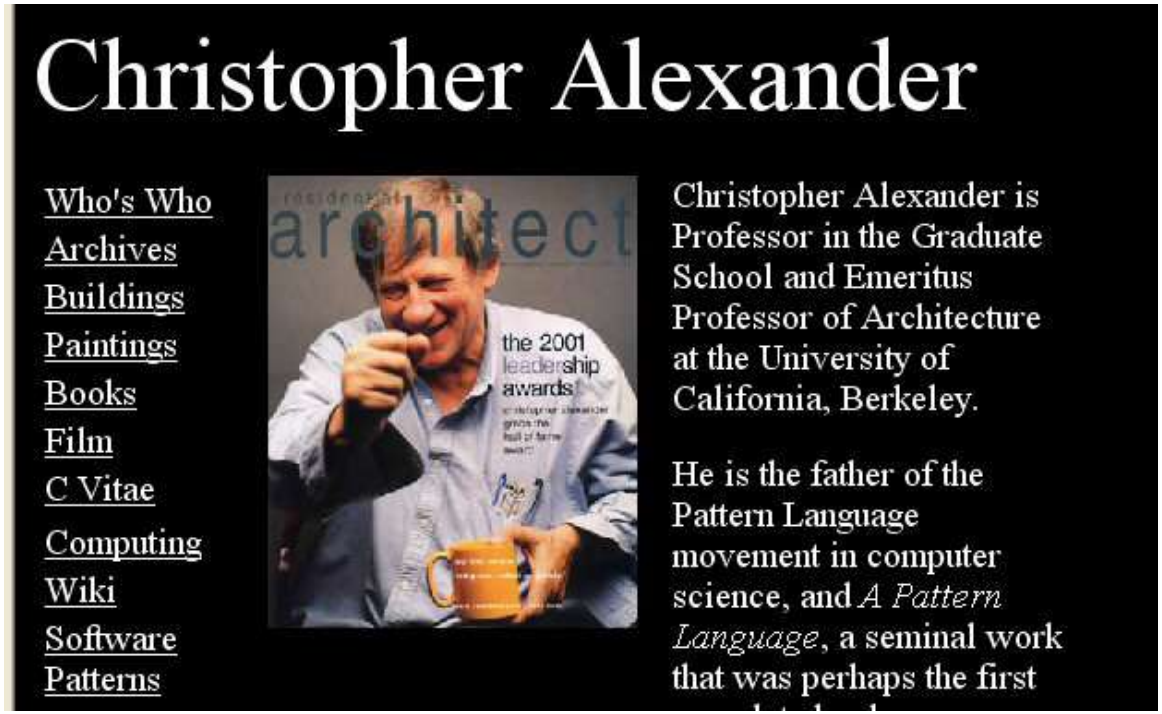
"Leggi cento pagine di architettura al giorno".

Cosa è un (Design) Pattern?

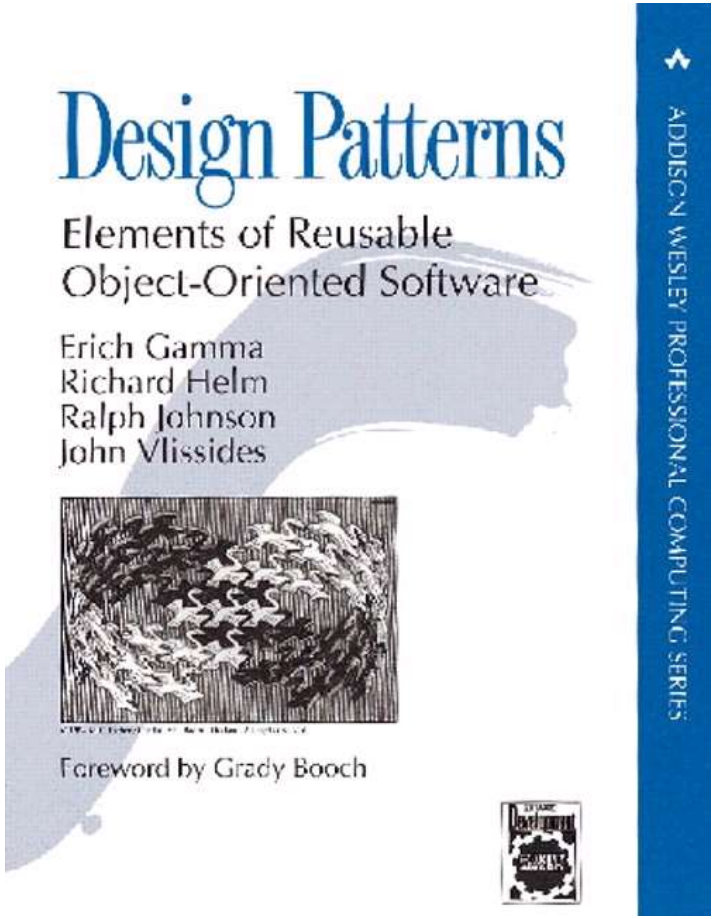
"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice"

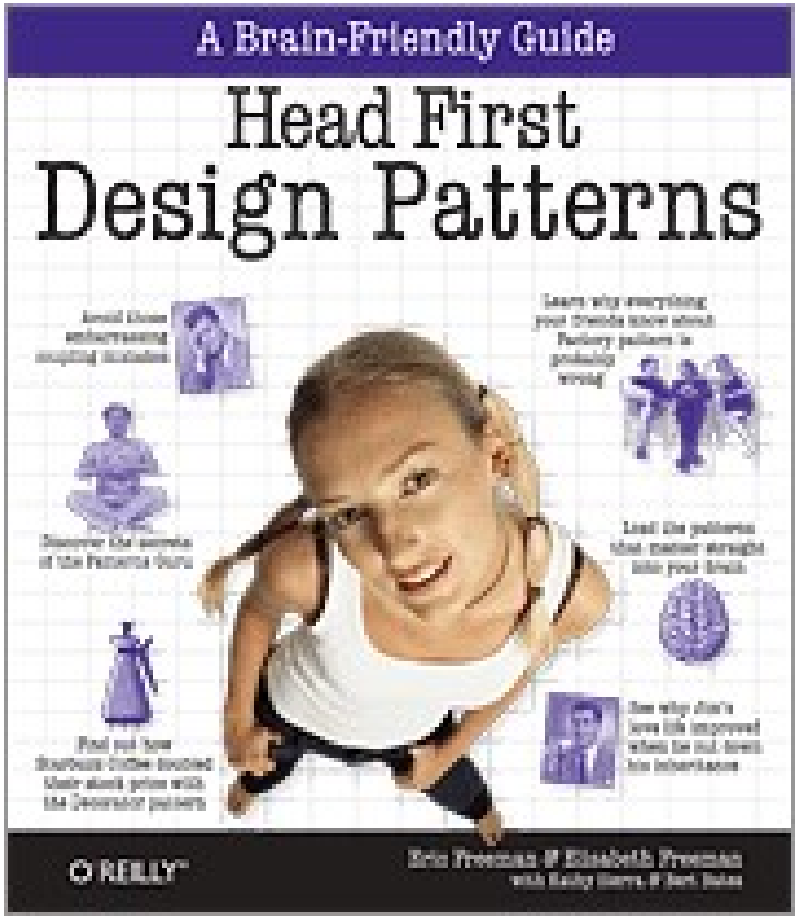
Christopher Alexander

A Pattern Language, 1977

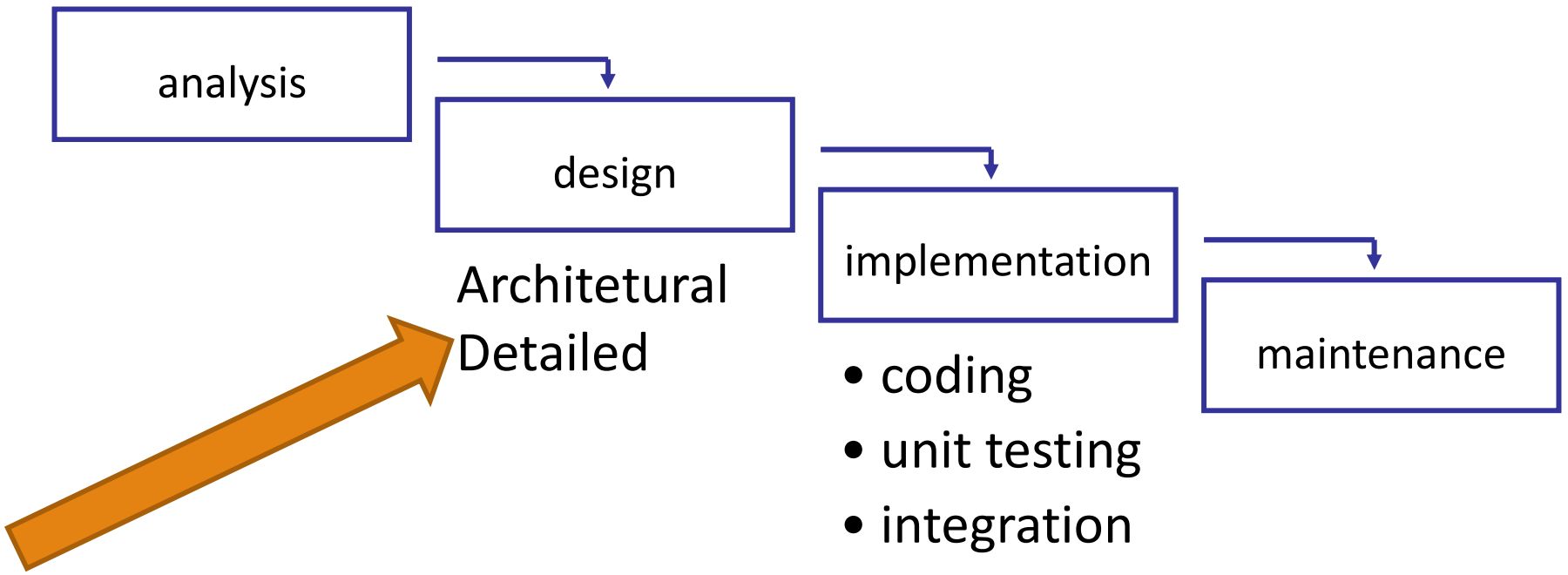


libri suggeriti





In che fase si applicano



GoF Design Patterns

Sono 23 design pattern suddivisi in base al loro scopo

Creazionali:

- propongono soluzioni per creare oggetti

Comportamentali:

- propongono soluzioni per gestire il modo in cui vengono suddivise le responsabilità delle classi e degli oggetti

Strutturali:

- propongono soluzioni per la composizione strutturale di classi e oggetti

Perché i pattern nel software?

- "Progettare software OO è difficile e progettare software OO riutilizzabile è ancora più difficile".
  - - Erich Gamma
- Iprogettisti esperti riutilizzano le soluzioni che hanno funzionato in passato.
- I sistemi OO ben strutturati hanno modelli ricorrenti di classi e oggetti.
- La conoscenza degli schemi che hanno funzionato in passato consente al progettista di essere più produttivo e ai progetti risultanti di essere più flessibili e riutilizzabili.

Software Patterns History

- 1987 - Cunningham and Beck used Alexander's ideas to develop a small pattern language for Smalltalk
- 1990 - The Gang of Four (Gamma, Helm, Johnson & Vlissides) begin compiling a catalog of design patterns
- 1991 - First Patterns Workshop at OOPSLA
- 1993 - Kent Beck and Grady Booch sponsor the first meeting of what is now known as the Hillside Group
- 1994 – 1st Pattern Languages of Programs (PLoP) conf.
- 1995 - The Gang of Four (GoF) *Design Patterns book*

Design Pattern Levels Of Abstraction

Complex design for an entire application or subsystem

Solution to a general design problem in a particular context

Simple reusable design class such as a linked list, hash table, etc.

More Abstract

More Concrete

Architettura-Progettazione Di Dettaglio-Codice

- Patterns o stili architecturali
  - Pipes and Filters, Publish-Subsribe, Model-View-Controller, …
- Design Patterns
  - Progettazione e raffinamento dei componenti.
  - E.g. abstract factory, decorator, …
- Idioms o Coding Design Patterns
  - Pattern di basso livello specifici di un linguaggio di programmazione.
  - Un idioma è più limitato di un modello di progettazione, ma descrive comunque un problema ricorrente
  - Ad esempio, in C, allocazione e deallocazione della memoria, convenzioni sui nomi delle variabili …

GoF Classification Of Design Patterns

Purpose - what a pattern does

- Creational Patterns
  - Concern the process of object creation
  - *Abstract Factory, Builder, Factory Method, Prototype, Singleton.*
- Structural Patterns
  - Deal with the composition of classes and objects
  - *Adapter, Bridge, Composite, Decorator, Façade, Flyweight, Proxy.*
- Behavioral Patterns
  - Deal with the interaction of classes and objects
  - *Chain of responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template, Visitor.*

GoF Pattern Template

Pattern Name and Classification

- A good , concise name for the pattern and the pattern's type

Intent

- Short statement about what the pattern does

Also Known As

- Other names for the pattern

Motivation

- A scenario that illustrates where the pattern would be useful

Applicability

- Situations where the pattern can be used

GoF Pattern Template (Continued)

Structure

- A graphical representation of the pattern

Participants

- The classes and objects participating in the pattern

Collaborations

- How to do the participants interact to carry out their responsibilities?

Consequences

- What are the pros and cons of using the pattern?

Implementation

- Hints and techniques for implementing the pattern

GoF Pattern Template (Continued)

Sample Code

- Code fragments for a sample implementation

Known Uses

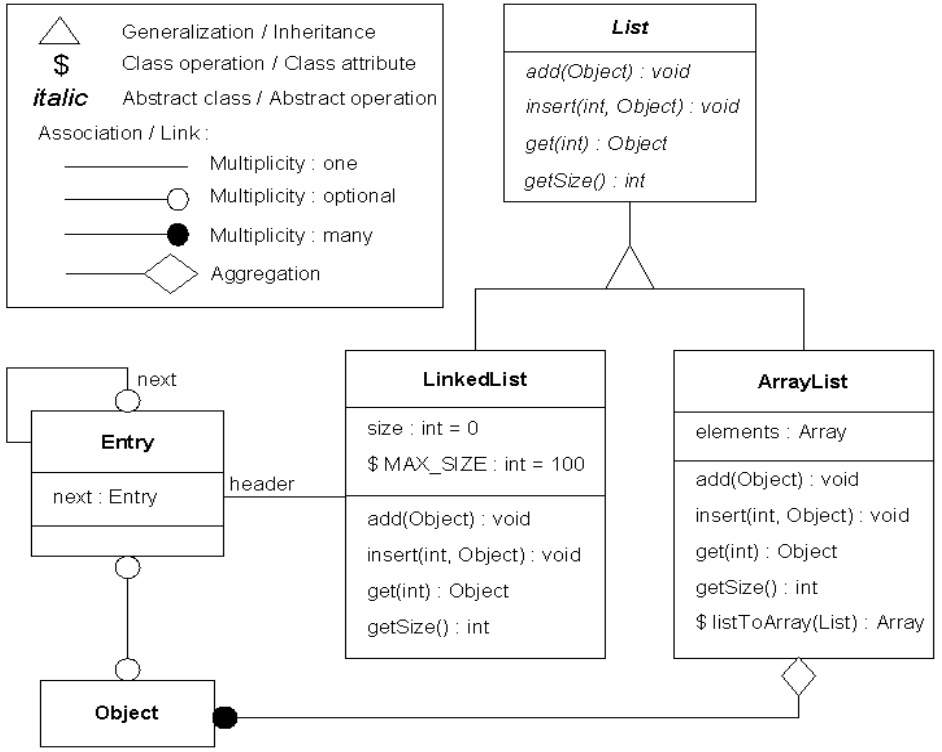
- Examples of the pattern in real systems

Related Patterns

- Other patterns that are closely related to the pattern

GoF Notation

The GoF book uses the Object Modeling Technique (OMT) notation for class and object diagrams



Head first uses UML

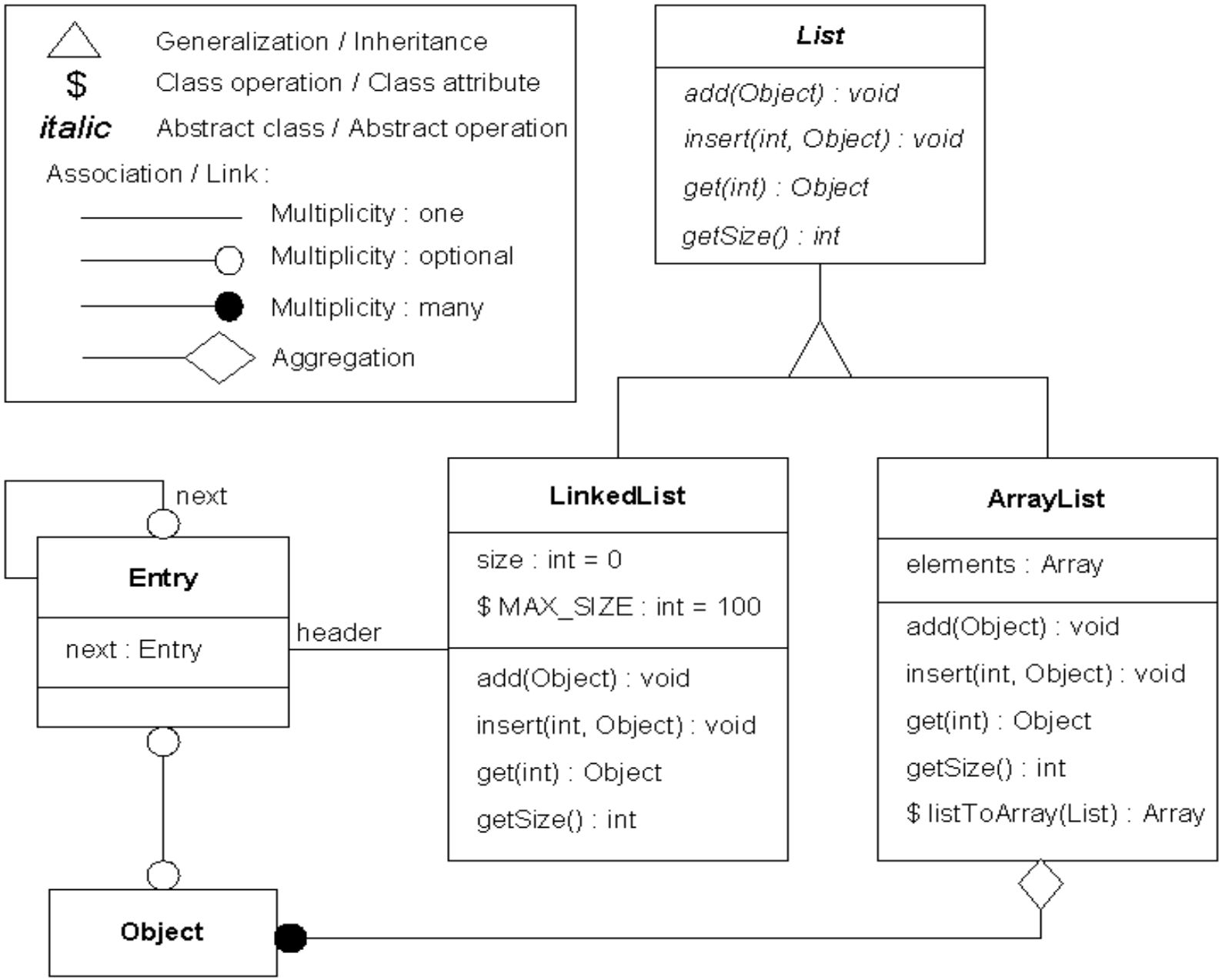
OMT object model

Appendix B of the GoF book.

(a) Abstract and concrete classes

(b) Participant Client class (left) and implicit Client class (right)

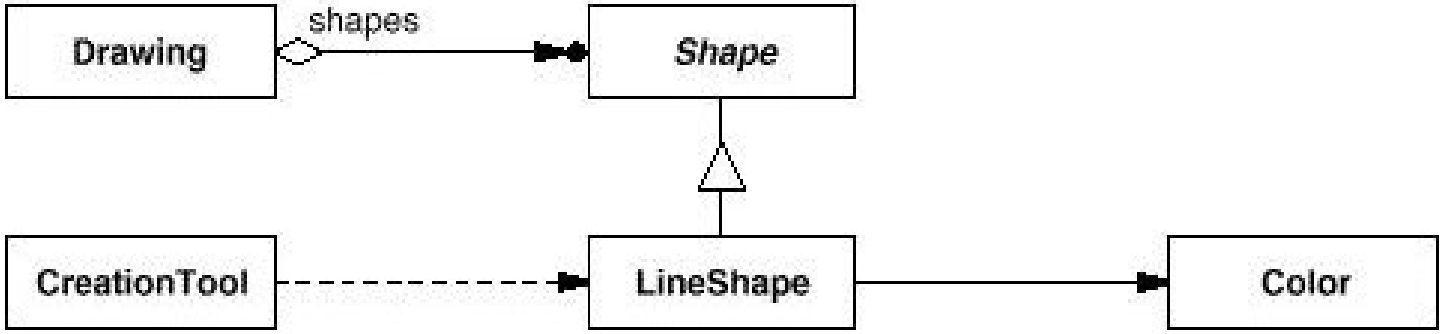
OMT object model (continued)

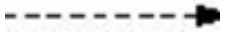


OMT object model (continued)

reference (do not use associations) when describing DP

instantiation relatio

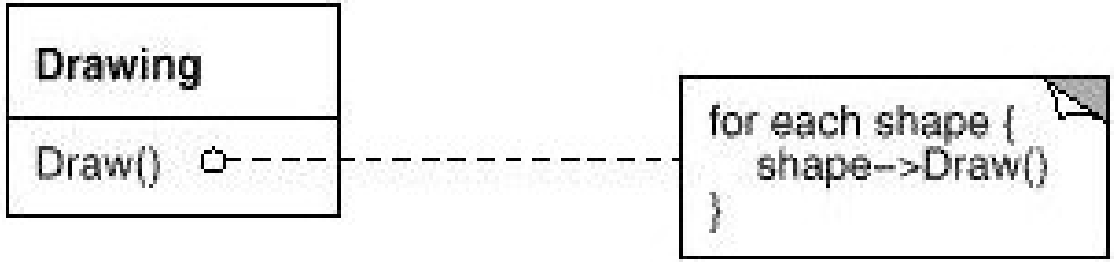




(c) Class relationships

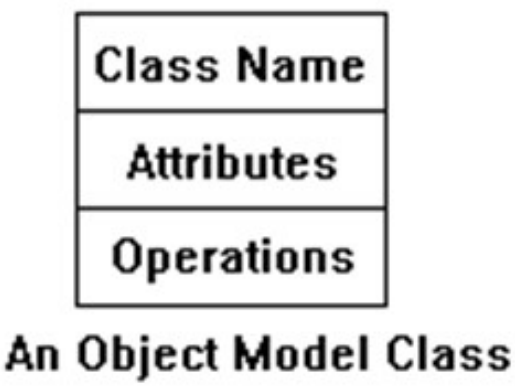
OMT object model (continued)

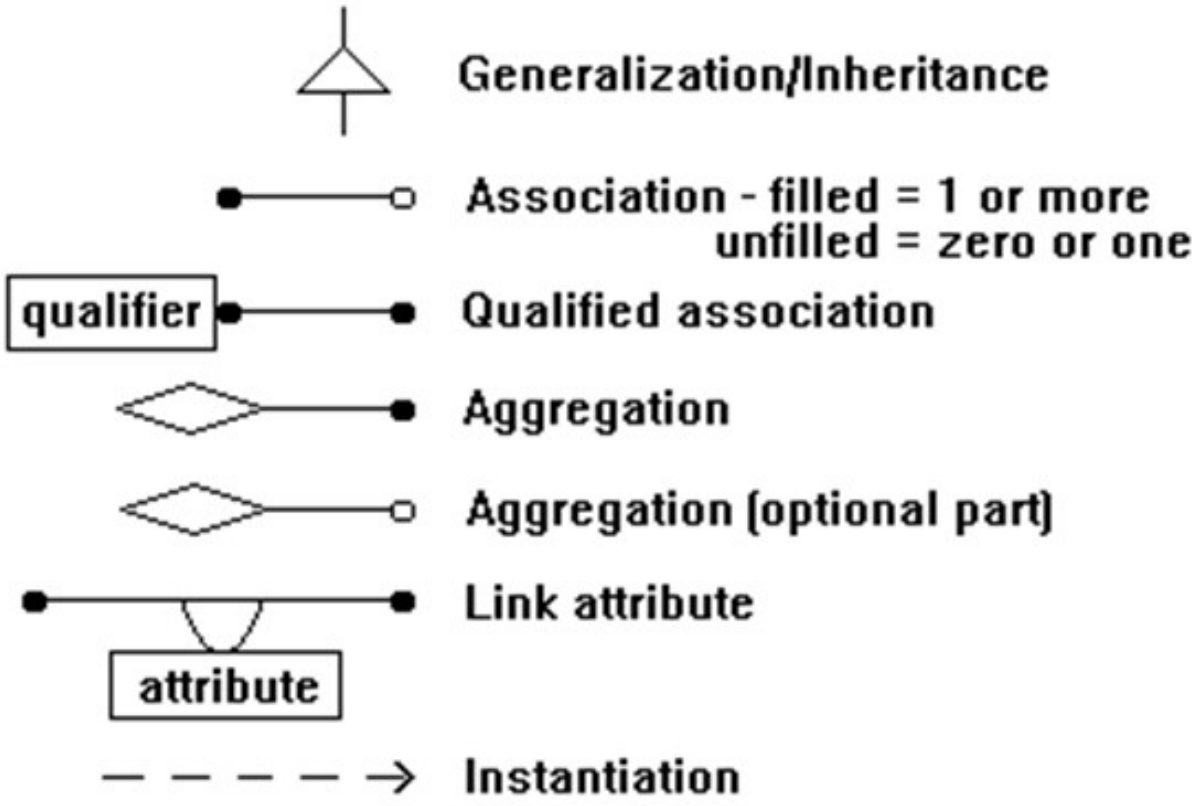
anchor a note



(d) Pseudocode annotation

OMT object model (continued)





Classes & instances (metadata & data)