Esercizi design patterns

Angelo Di Iorio, diiorio@cs.unibo.it

Esercizio 1

- *Una parete, che contiene porte e finestre, deve essere dipinta con una vernice. Ogni barattolo contiene una data quantità di vernice, che permette di dipingere una data superficie.*
- Rispondere alle domande:
  - A chi assegnare la responsabilità di calcolare la quantità di vernice necessaria per una data superficie?
  - A chi assegnare la responsabilità di calcolare la quantità di vernice necessaria per dipingere una parete?

Esercizio 2

- Data la seguente rappresentazione UML di una tabella (es. HTML), a chi assegnare la responsabilità di creare una riga? E una cella?
- Le responsabilità cambierebbero se la Tabella fosse composta solo da celle, non organizzate in righe (la classe Riga non esiste)?

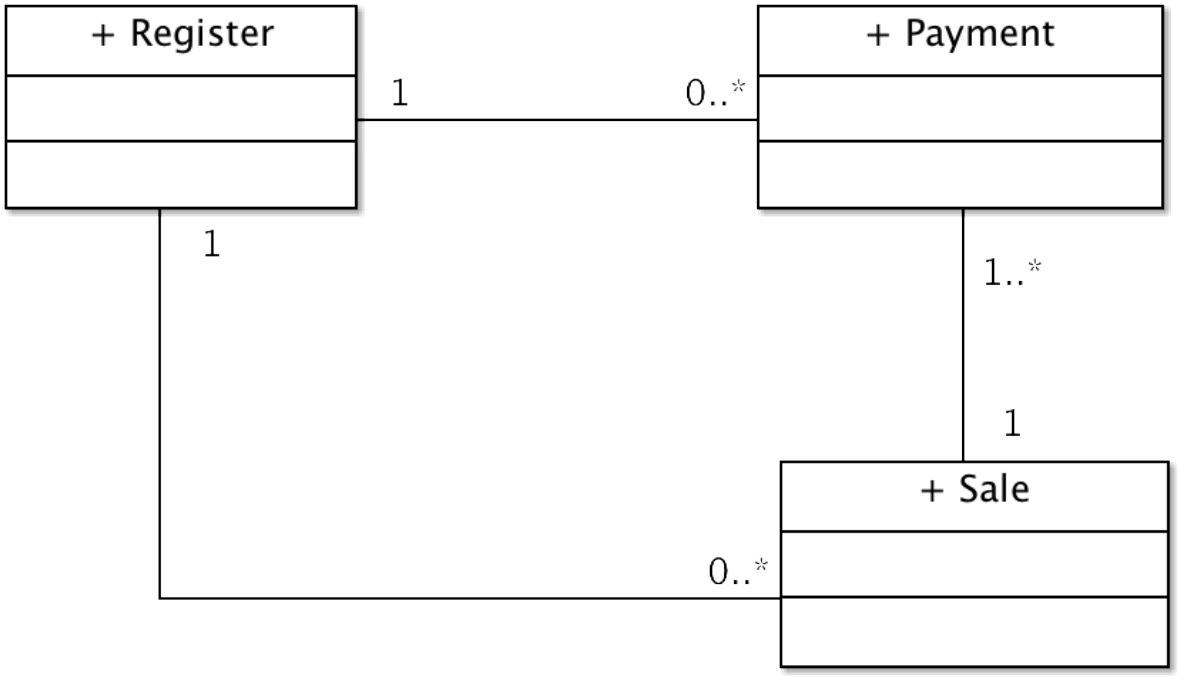
GRASP Creator

- *Problem*: Who should be responsible for creating a new instance of some class?
- *Solution*: Assign class B the responsibility to create an instance of class A if one or more of the following is true:
  - B *aggregates A objects*
  - B *contains A objects*
  - B *records instances of A objects*
  - B *has the inizializing data* passed to *A* when it is created

Esercizio 3

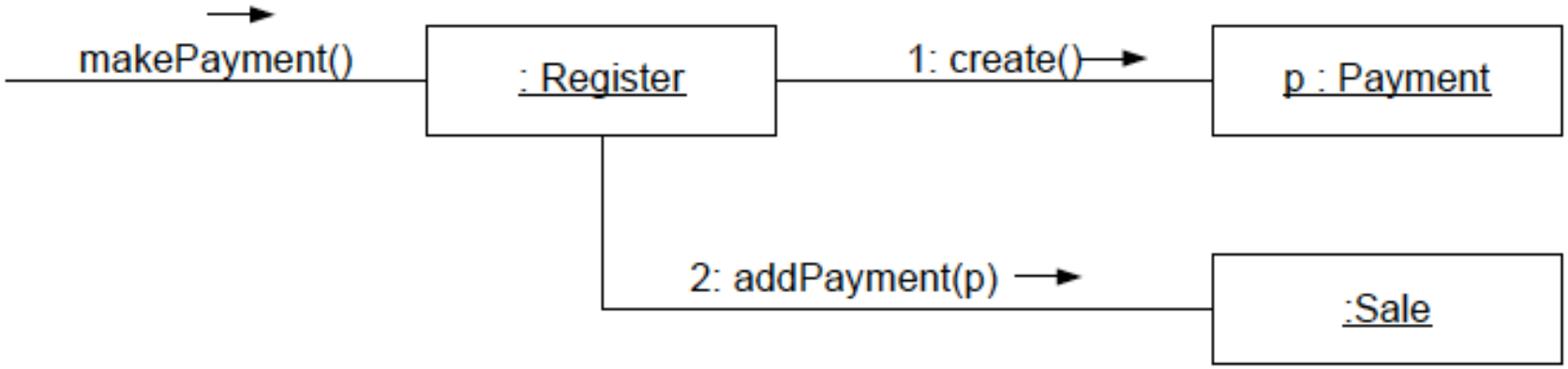
- Si consideri il seguente dominio:
  - Un *Registro (Register)* tiene traccia dei *Pagamenti (Payment)*
  - Ogni *Vendita (Sale)* è associata a un insieme di *Pagamenti*
- A chi assegnare la responsabilità di creare un'istanza di Pagamento?
- Disegnare un diagramma di comunicazione che descrive la soluzione data
  - Utile partire dal diagramma delle classi

Classi di analisi



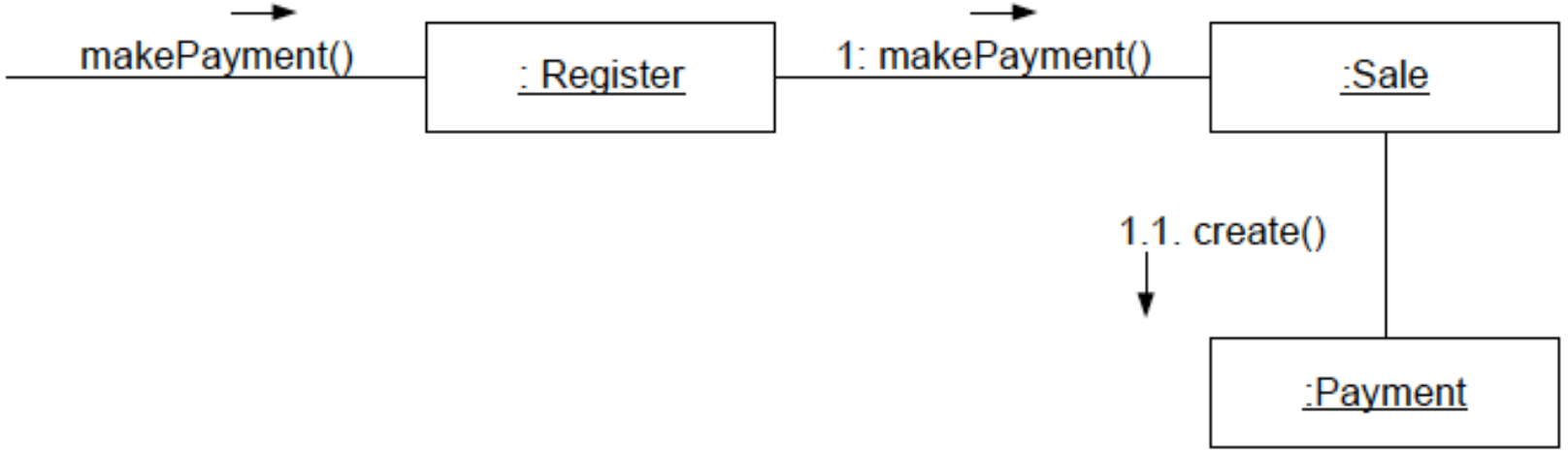
Soluzione 3.1

- Problema?



Soluzione 3.2

- Contraddice il pattern Creator?

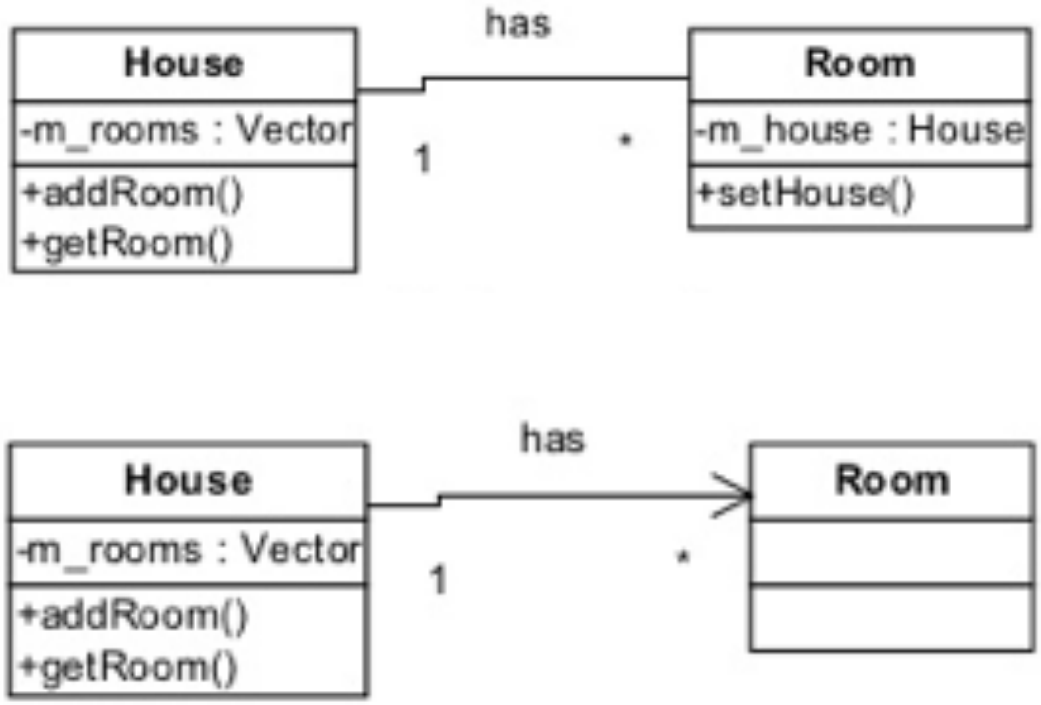


GRASP Low Coupling

- *Problem*: How to support low dependency, low change impact, and increased reuse?
- *Solution*: Assign a responsibility so that coupling remains low.

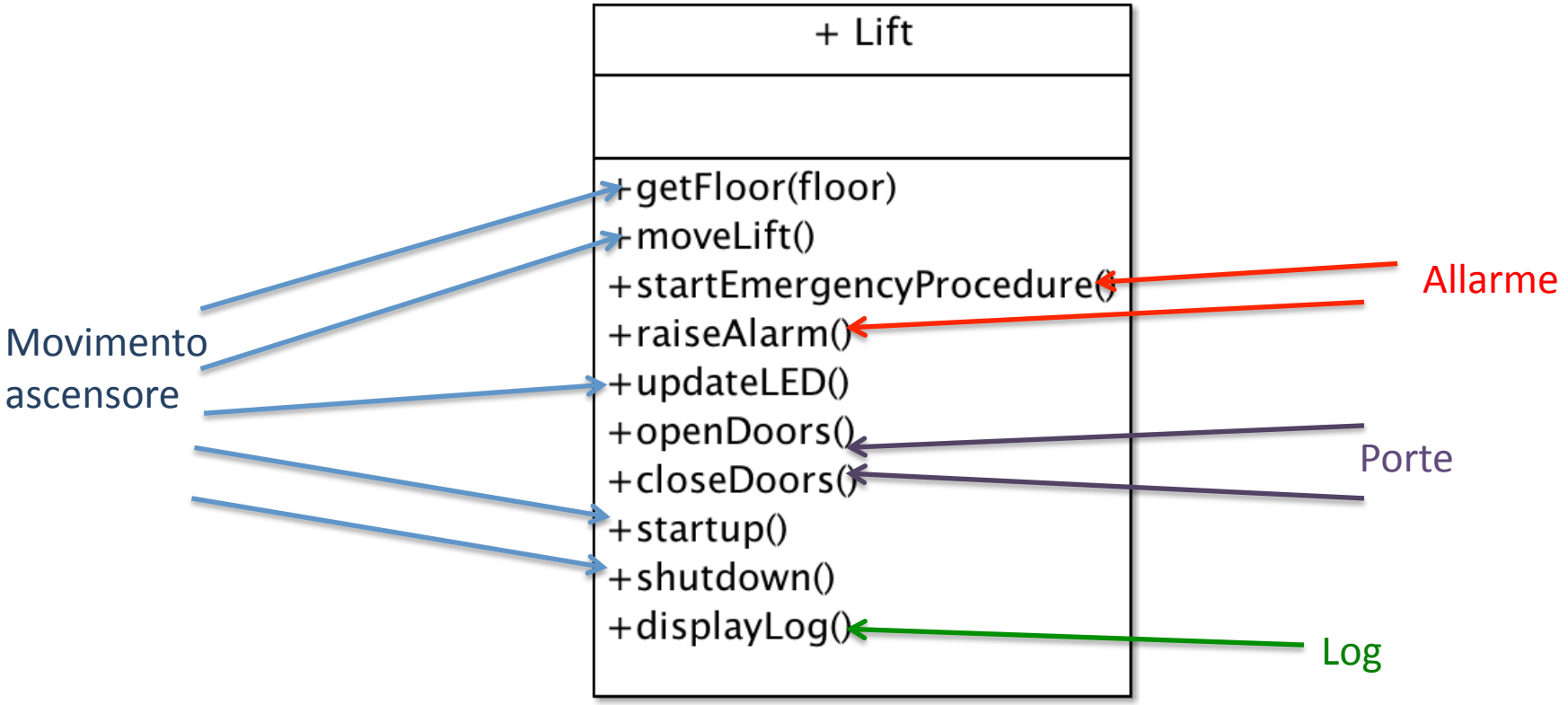
(Coupling is a measure of how strongly one element is connected to, has knowledge of, or relies on other elements)

Quale preferire?



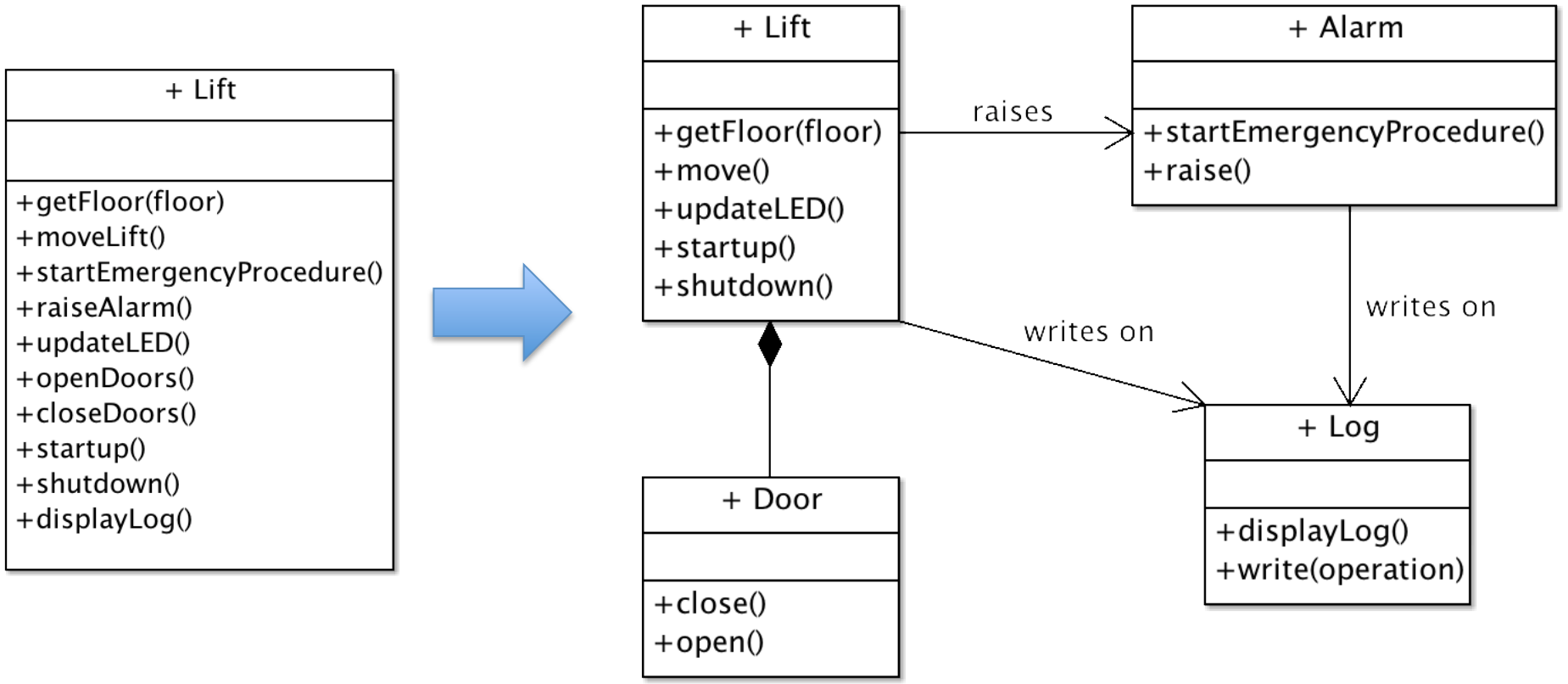
Esercizio 4

- Si può progettare meglio questa classe, che modella un ascensore?



Possibile soluzione

- Coesione?
- Ulteriore refactoring?



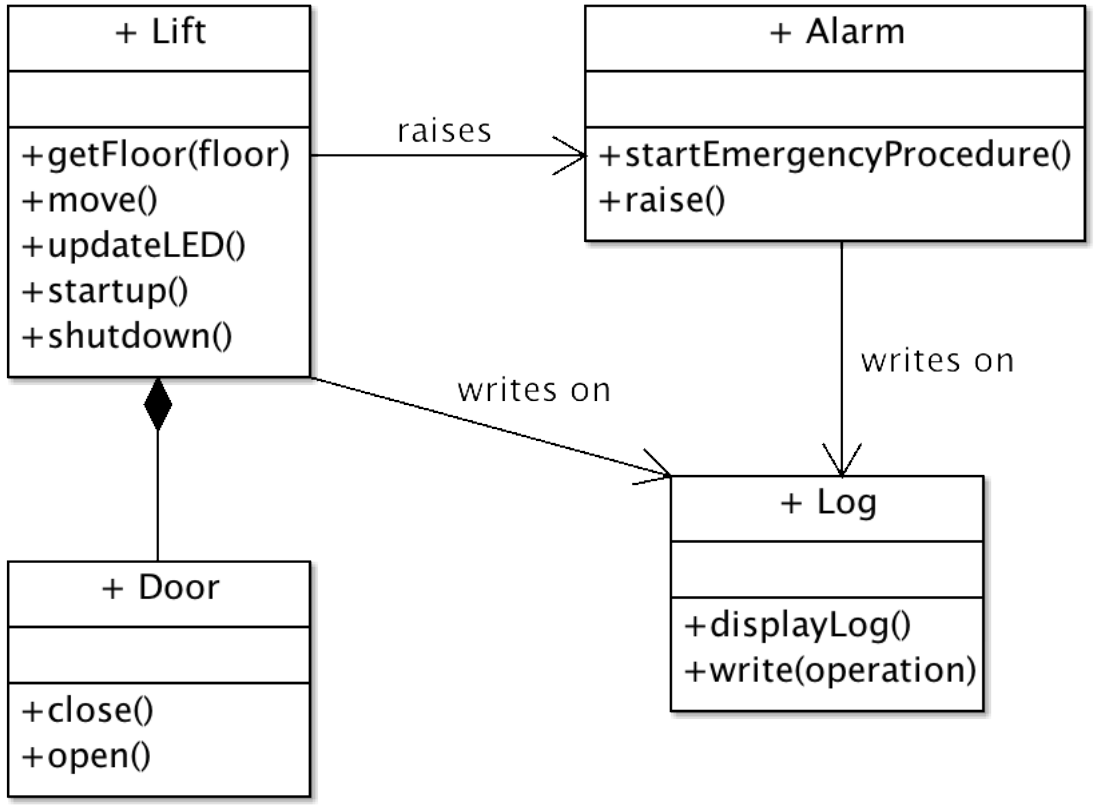
GRASP High Coesion

- *Problem*: How to keep complexity manageable?
- *Solution*: Assign a responsibility so that cohesion remains high.

(cohesion, or more specifically *functional cohesion*, is a measure of how strongly related and focused the responsibilities of an element are.)

Interazione con il sistema

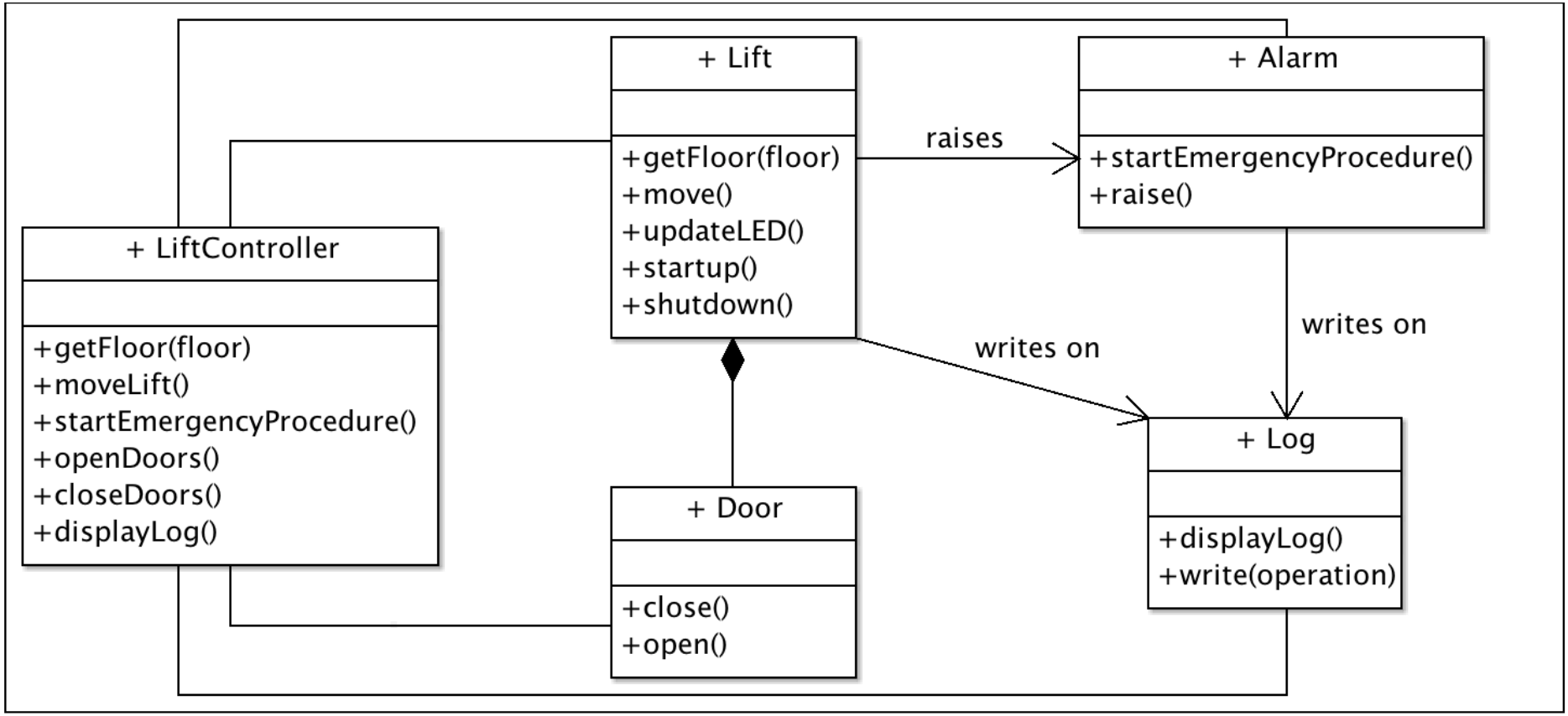
- Quali metodi rispondono ad eventi sollevati dall'utente?



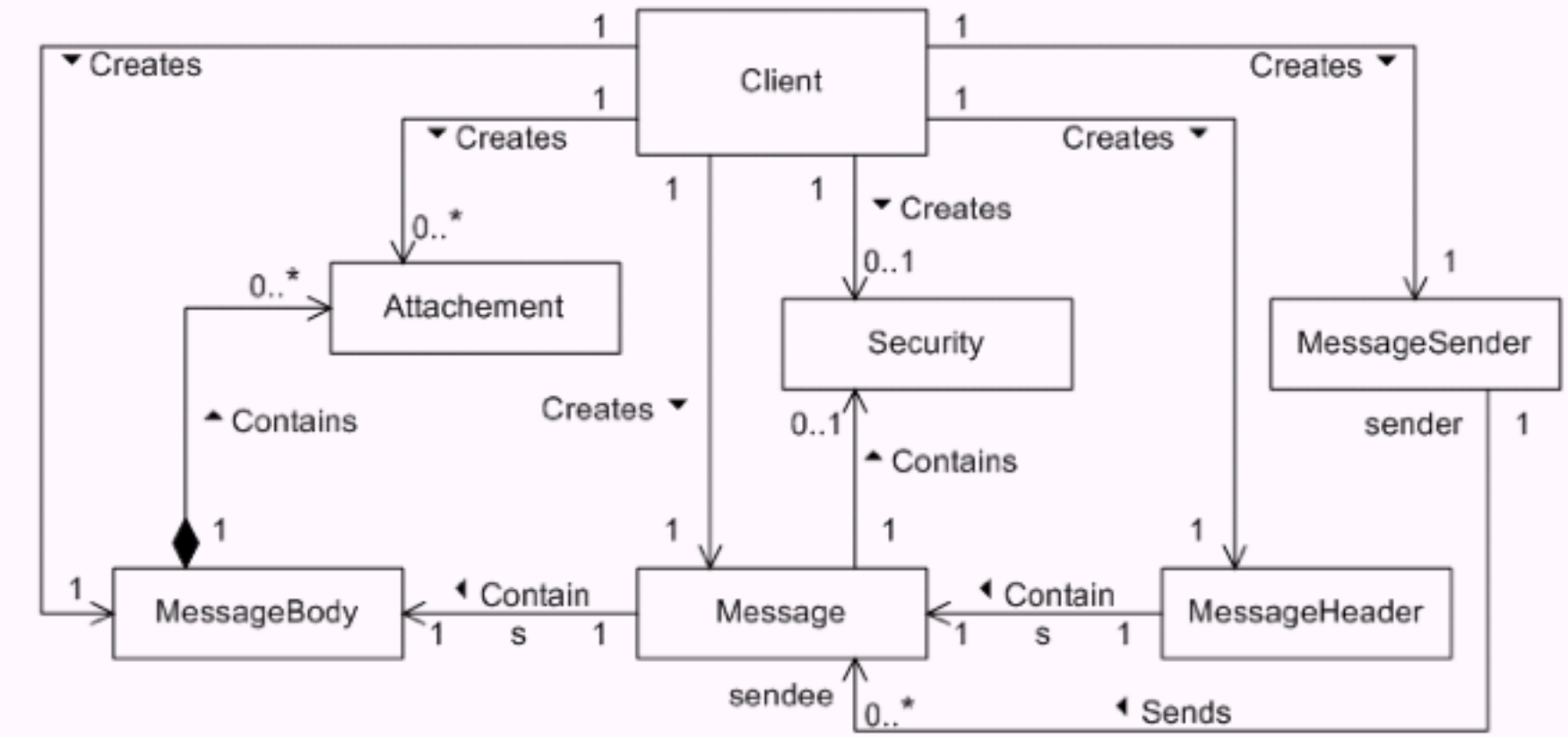
GRASP Controller

- *Problem*: Who should be responsible for handling an input system event?
- *Solution*: Assign the responsibility for receiving or handling a system event message to a class representing one of the following choices:
  - Represents the overall system, device, or subsystem
  - Represents a use case scenario within which the system event occurs

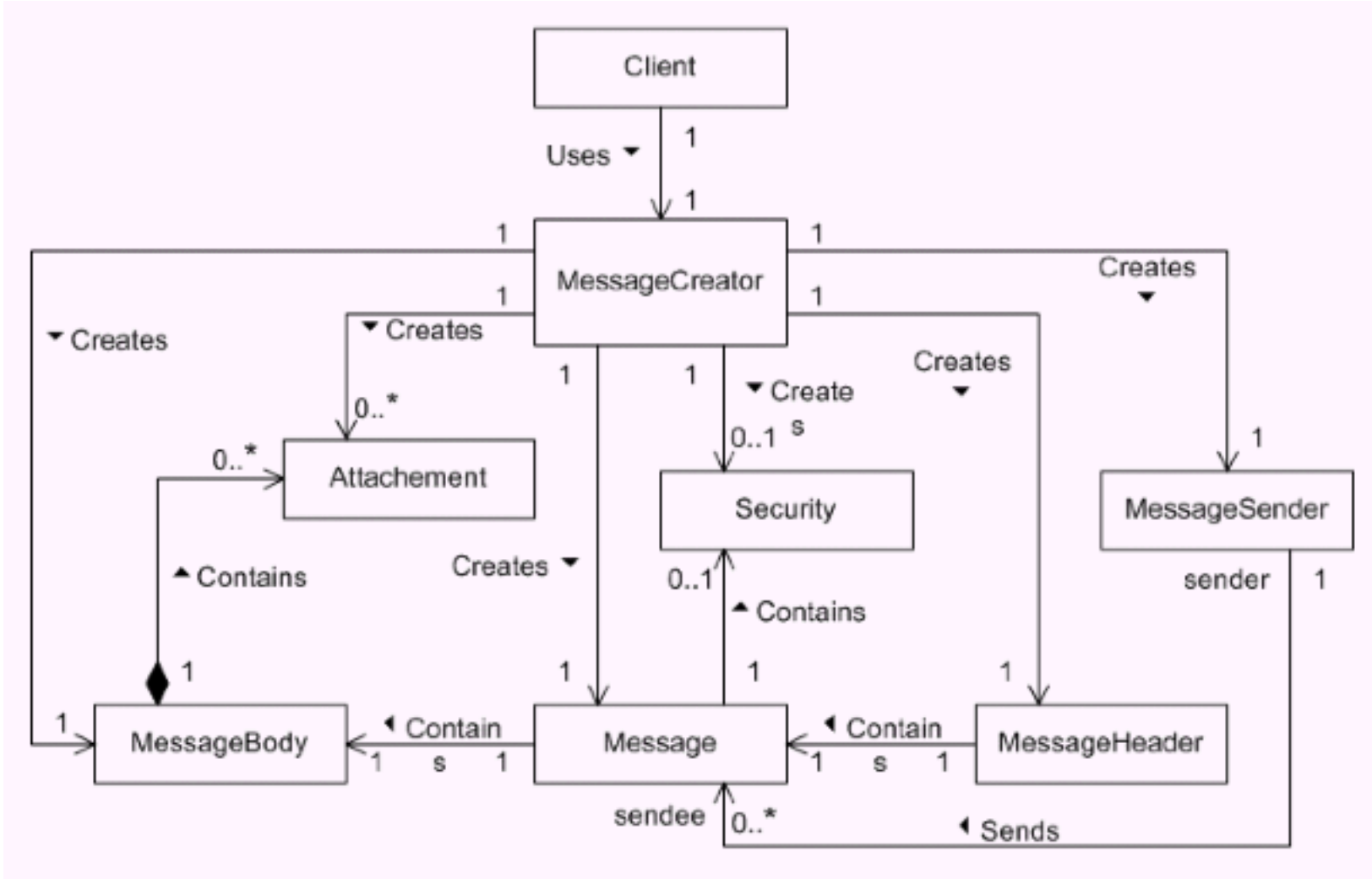
Controller Ascensore



Controller?


Client
Attachement
Security
MessageSender
MessageBody
Message
MessageHeader
Creates
Contains
Contain
s
sendee
Sends
sender
1
0..*
0..1

GRASP Controller (GoF Façade)

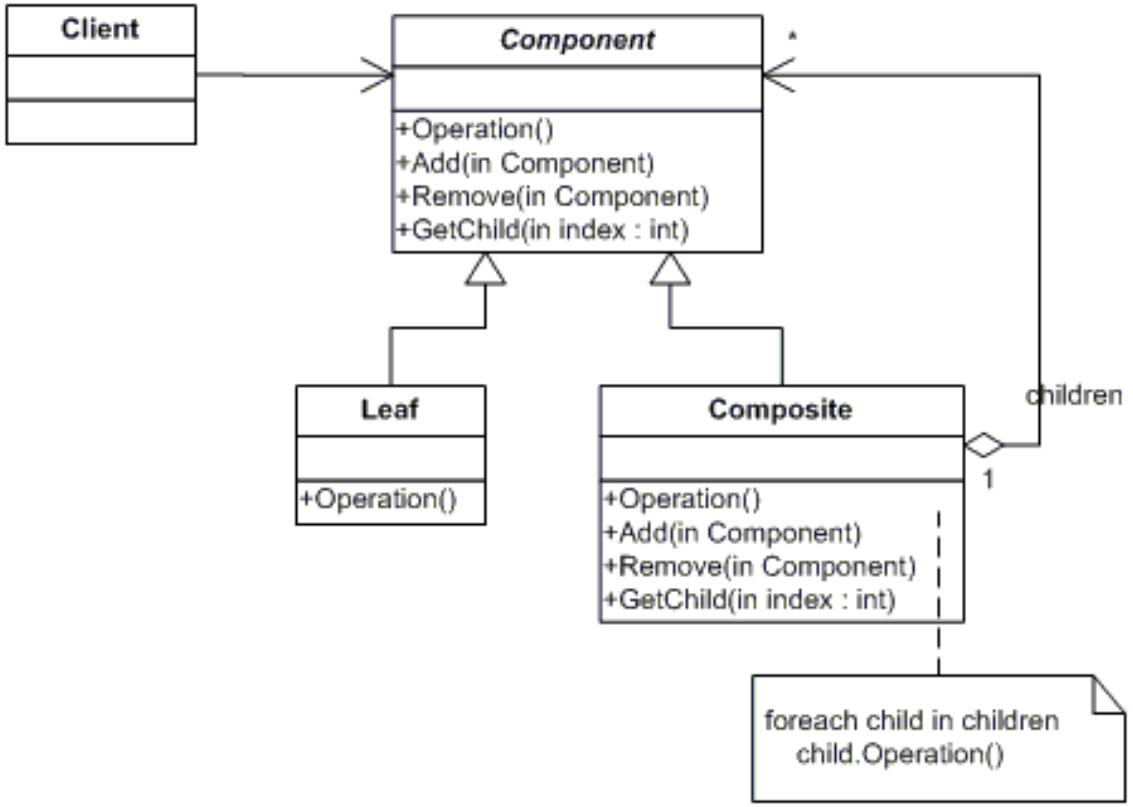


Esercizio 5: file system

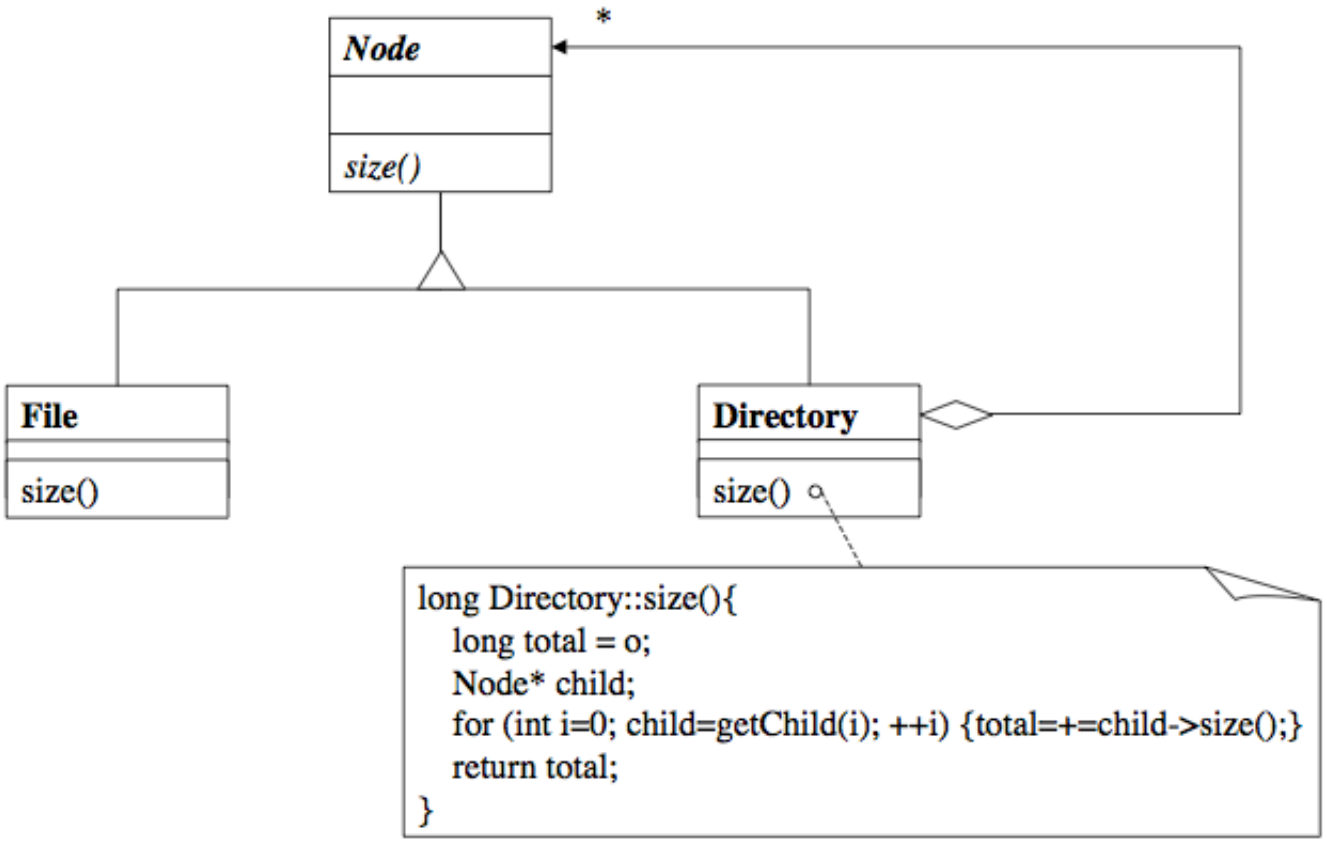
- Come organizzare al meglio un diagramma delle classi (e relativo codice) per modellare un file-system, in cui è possibile conoscere le dimensioni di ogni file e/o directory?

Composite Pattern

- *Problema*: creare una gerarchia di oggetti (elementari o contenitori) in cui il client "usa" allo stesso modo sia gli oggetti elementari che i contenitori



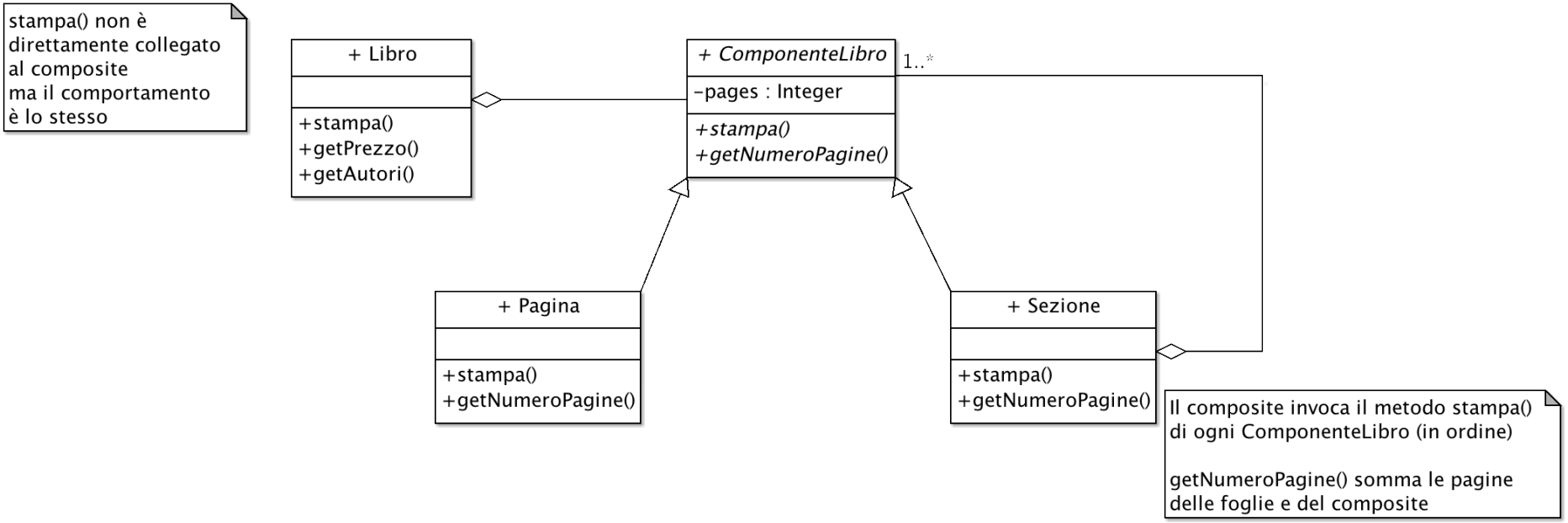
File system


*
Node
size()
File
size()
Directory
size()
long Directory::size(){
long total = o;
Node* child;
for (int i=0; child=getChild(i); ++i) {total=+=child->size();}
return total;
}

Esercizio 6: Libro

- Disegnare una diagramma delle classi per modellare questo dominio:
  - Un libro è composto da pagine, eventualmente organizzate in sezioni. Ogni sezione può contenere sezioni (una o più) e pagine semplici.
  - E’ possibile stampare una pagina singola, una sezione o l’intero libro.

Soluzione


stampa() non è
direttamente collegato
al composite
ma il comportamento
è lo stesso
+ Libro
+stampa()
+getPrezzo()
+getAutori()
+ ComponenteLibro
-pages : Integer
+stampa()
+getNumeroPagine()
1..*
+ Pagina
+stampa()
+getNumeroPagine()
+ Sezione
+stampa()
+getNumeroPagine()
Il composite invoca il metodo stampa()
di ogni ComponenteLibro (in ordine)
getNumeroPagine() somma le pagine
delle foglie e del composite

Riferimenti

- E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.*
- Larman, *Applying UML and patterns, Pearson 2005.*
- Head First Design Patterns By Eric Freeman, Elisabeth Freeman, Kathy Sierra, Bert Bates, first edition 2004.