Sistemi Informativi:

Progetto del sistema

Argomenti

- schemi di progetto (pattern)
  - utilita’
  - categorie
- tipi
  - composto: modella aggregati dinamici
  - facciata: interfaccia a sottosistemi
  - adattatore: interfaccia a sistemi esistenti
  - ponte: interfaccia a sistemi esistenti e futuri

Richiamo: trovare gli oggetti

- analisi requisiti
  - casi d’uso
  - analisi testuale flusso eventi (sostantivi ..)
  - interviste per il dominio applicativo
- progetto sistema
  - decomposizione
  - strati e partizioni
- progetto degli oggetti
  - usare conoscenza del dominio implementativo per trovare altri oggetti

I **design patterns** (schemi di progetto)

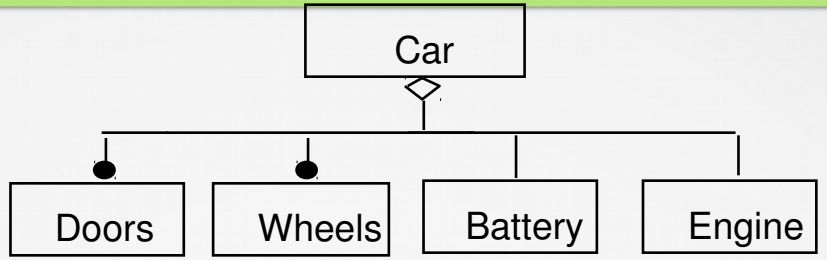
- modellare accuratamente il mondo attuale non porta ad un sistema per il futuro
- c'e' bisogno di progetti flessibili e riutilizzabili
- schema di progetto:
  - descrive un problema che si verifica molto spesso e la cui soluzione si puo' riapplicare con opportune variazioni a problemi diversi

notazione per schemi di progetto

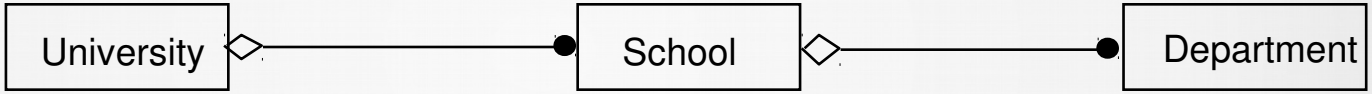
- Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley, 1995
- piccole differenze con UML

modellare un'aggregazione

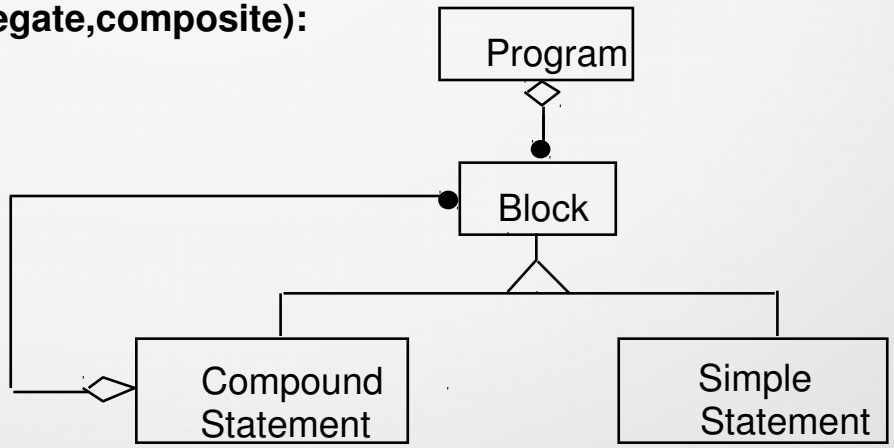
Fixed Structure:



Organization Chart (variable aggregate):

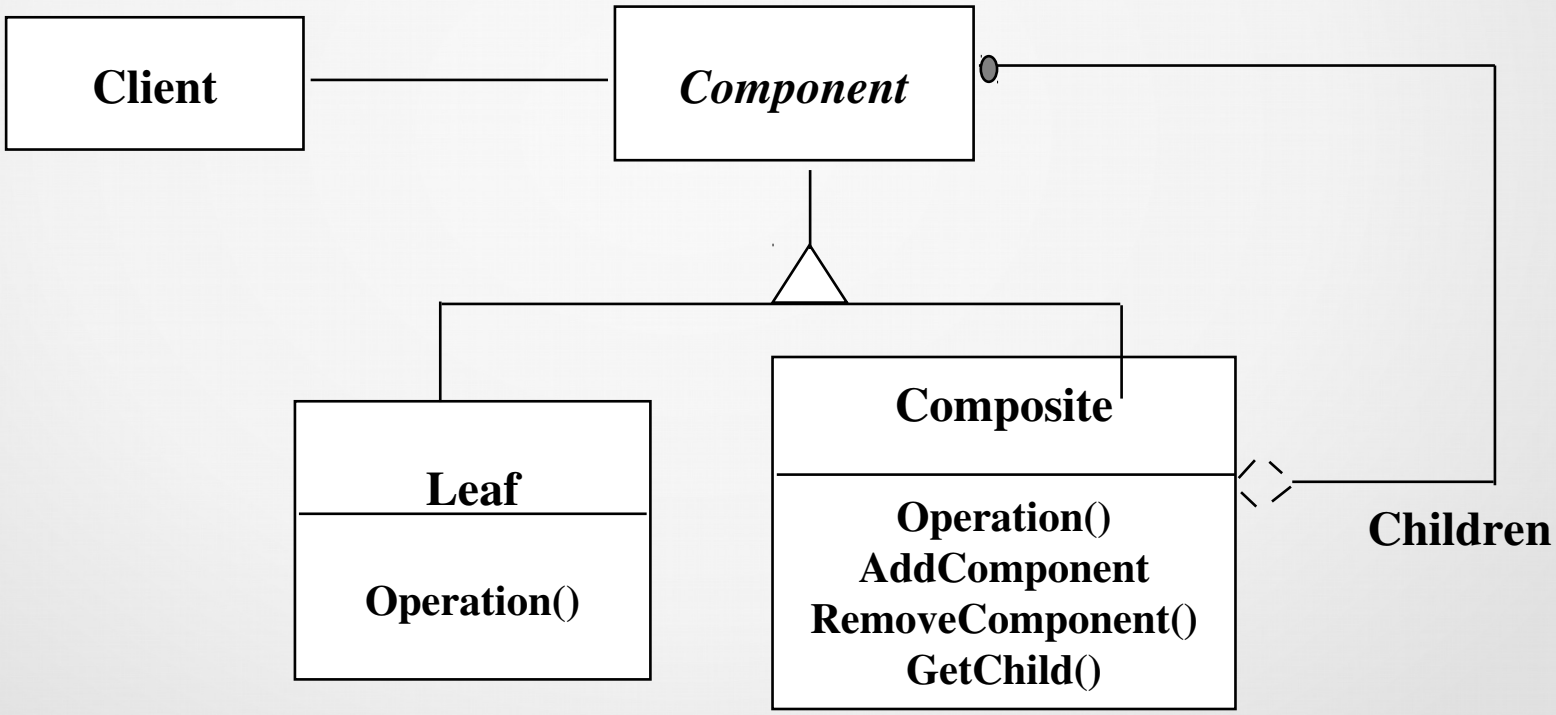


Dynamic tree (recursive aggregate,composite):



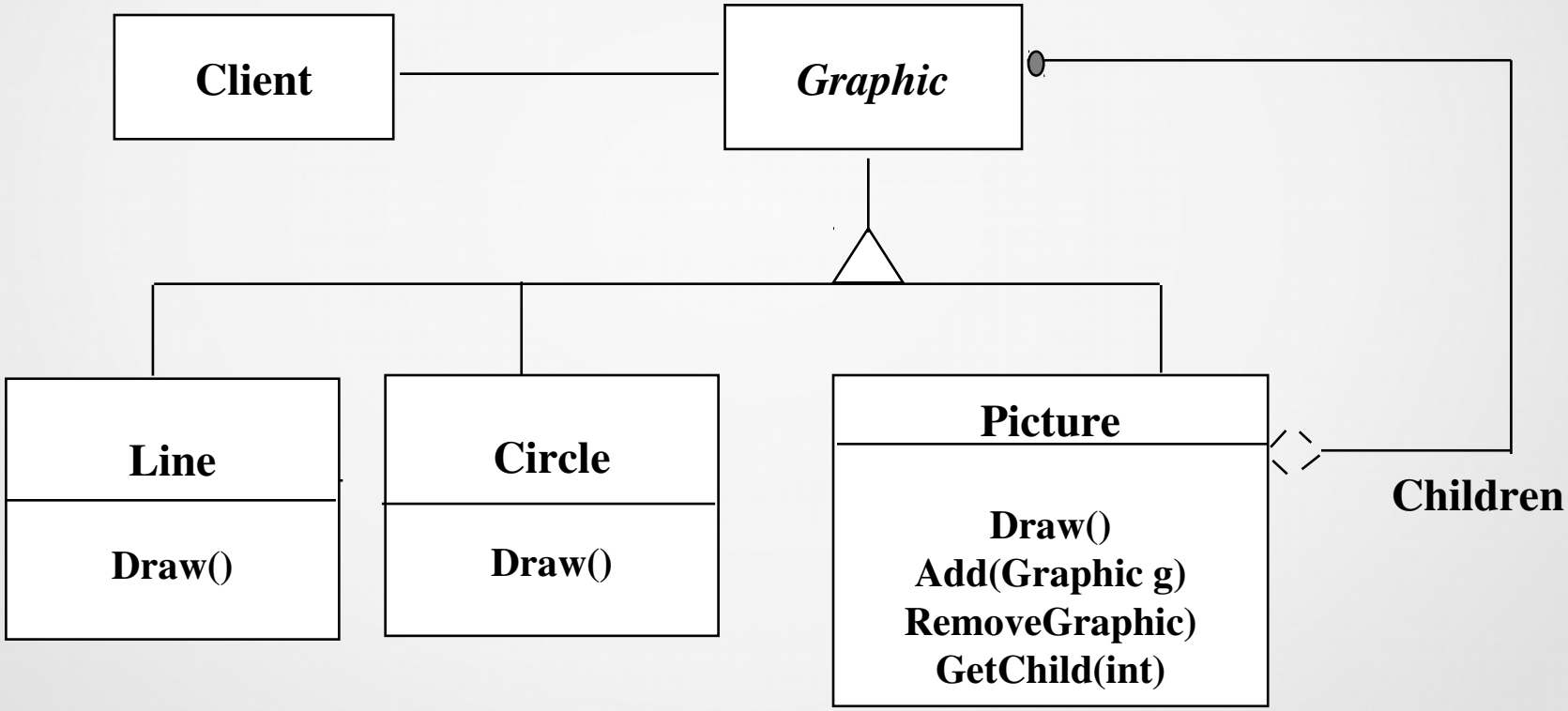
schema Composto

- compone oggetti in strutture ad albero di dimensione arbitraria (ricorsione)
- permette di trattare i singoli oggetti in modo uniforme



le applicazioni grafiche usano lo schema Composto

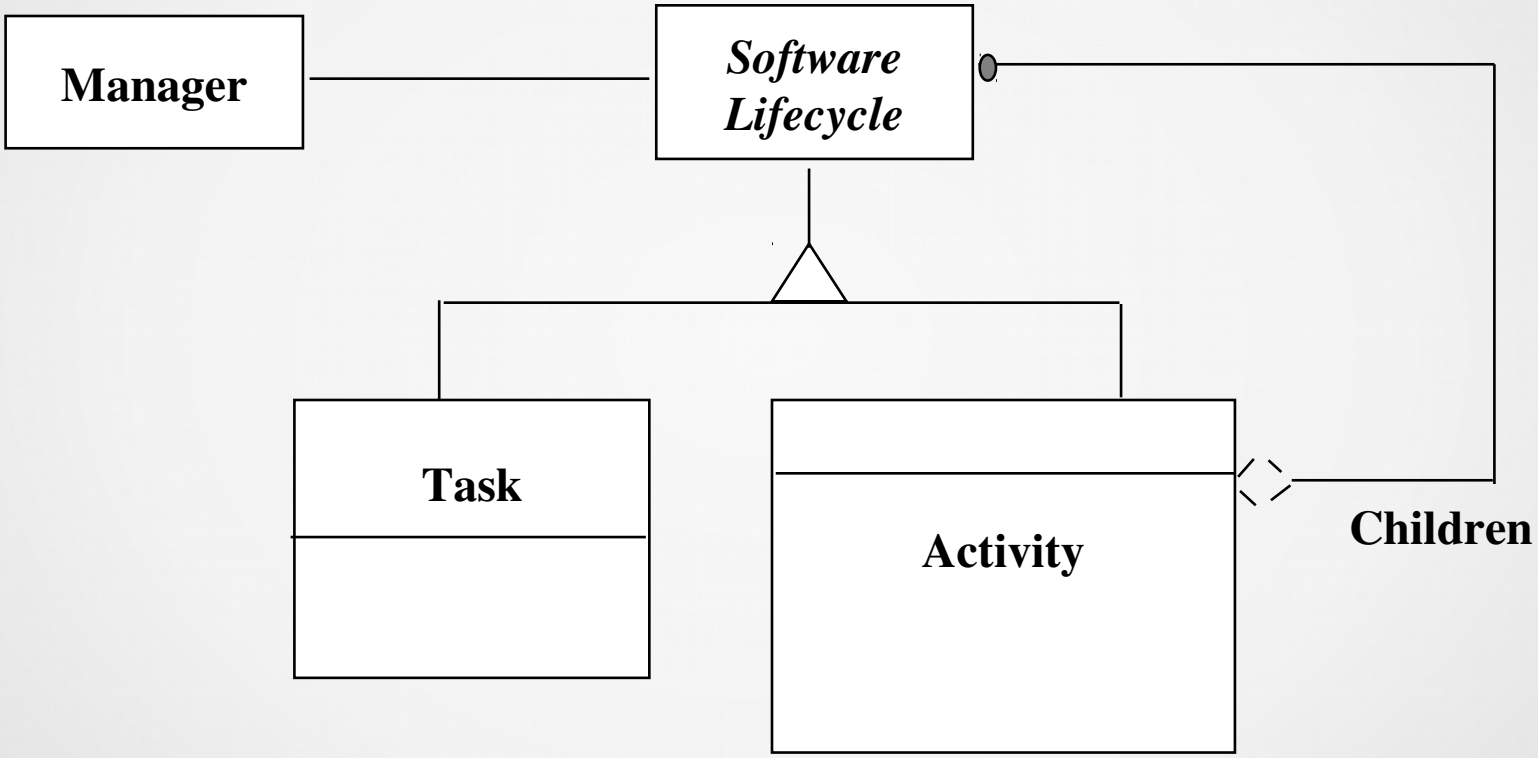
- la classe Graphic rappresenta sia oggetti elementari (linea , punto) che complessi (disegno)



esempio d’uso dello schema composto

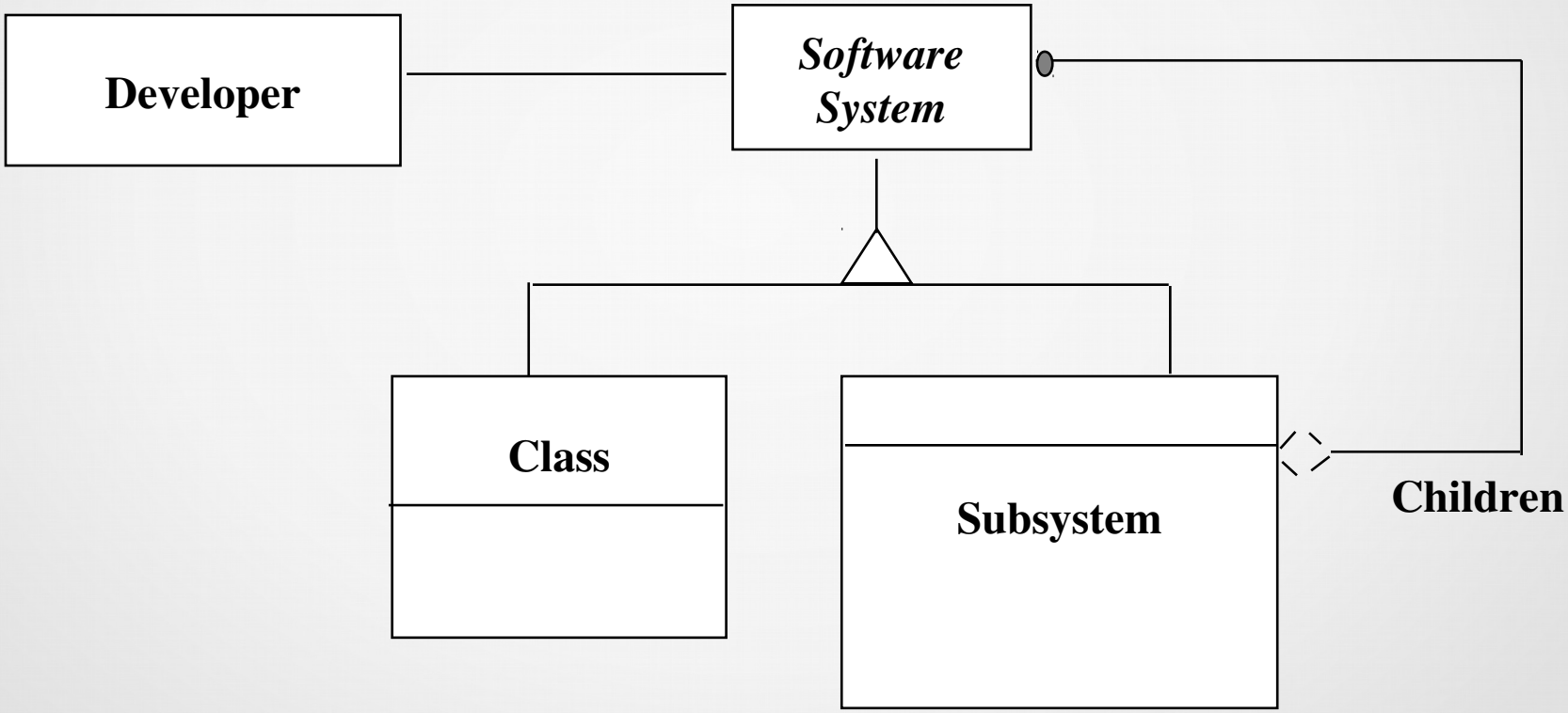
- ciclo di vita del SW
  - ogni attivita’ e’ un compito specifico (foglia) oppure una collezione di altre attivita’ (nodo intermedio)
- Sistema SW
  - un sistema e’ una collezione di sottosistemi a loro volta scomponibili in ulteriori sottosistemi (nodi intermedi) o classi elementari (foglie)

schema composto del ciclo di vita del SW



schema composto di un sistema sw

fate voi qualche esempio

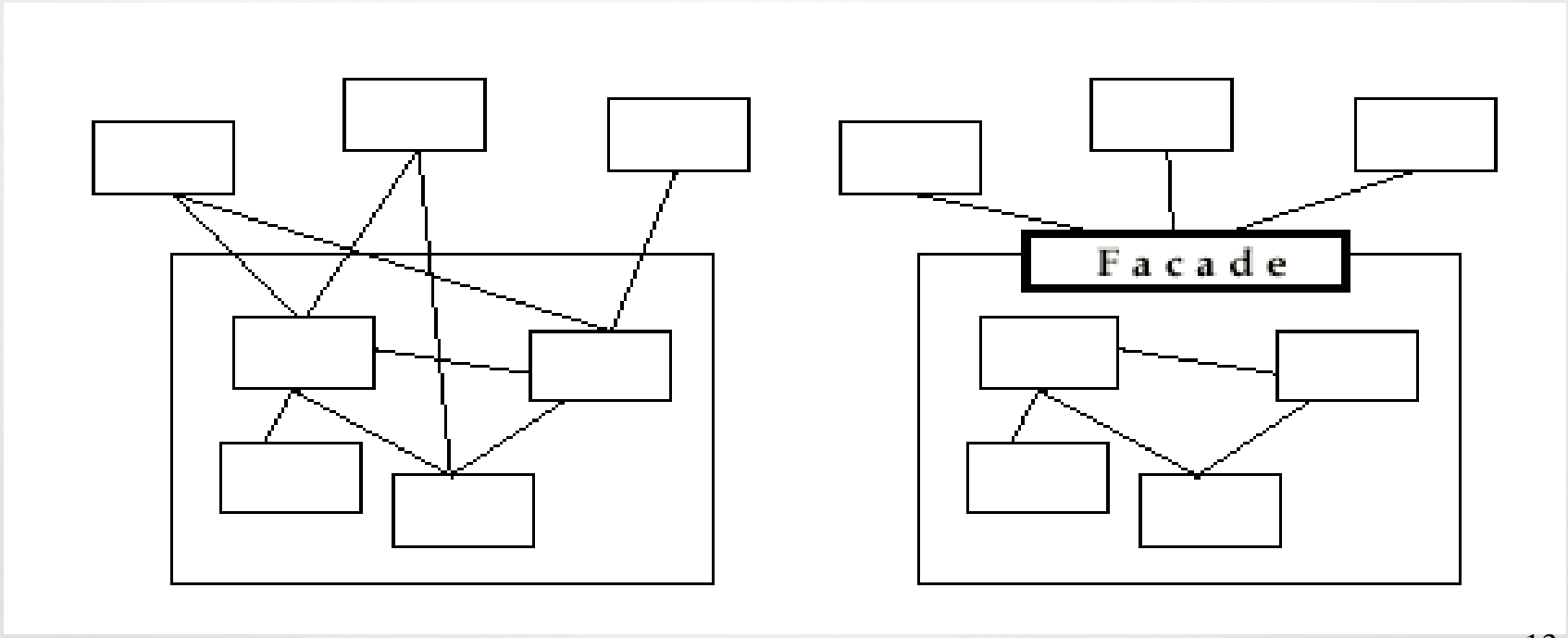


struttura di un sottosistema: facciata ,adattatore, ponte

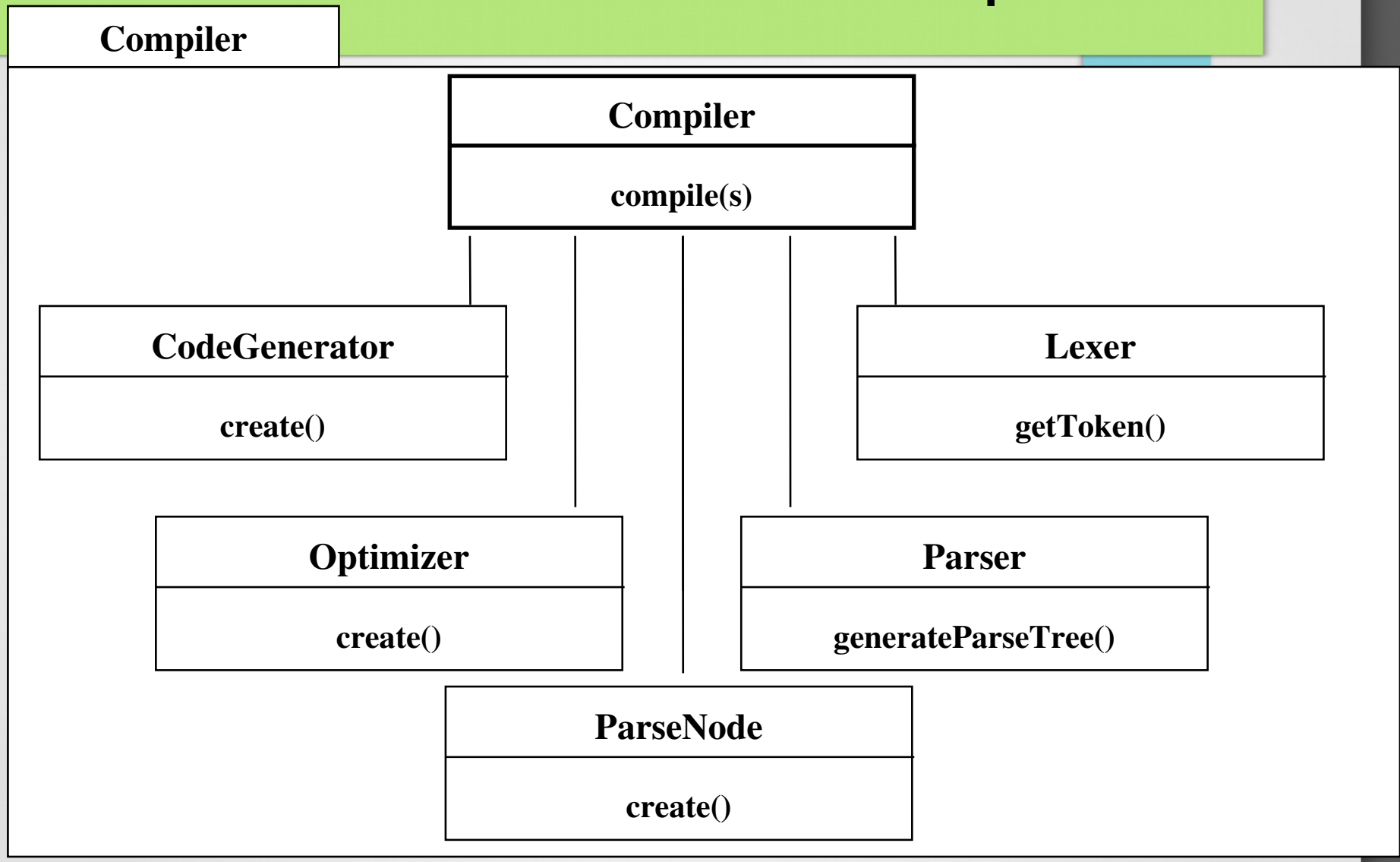
- un sottosistema consiste di
  - un oggetto interfaccia
  - un insieme di oggetti del dominio modellanti entita' reali o sistemi esistenti, alcuni dei quali sono interfacce ad altri sistemi
  - degli oggetti di controllo
- realizzazione della interfaccia: facciata
  - fornisce l'interfaccia al sottosistema
- interfaccia a sottosistemi esistenti: ponte o adattatore
  - forniscono interfaccia a sistemi preesistenti (legacy) che non sono necessariamente ad oggetti!

schema a facciata

- interfaccia unificata ad insieme di oggetti del sottosistema
- interfaccia che rende piu' facile l'uso del sottosistema
- consente di realizzare un'architettura chiusa



schema a facciata in un compilatore

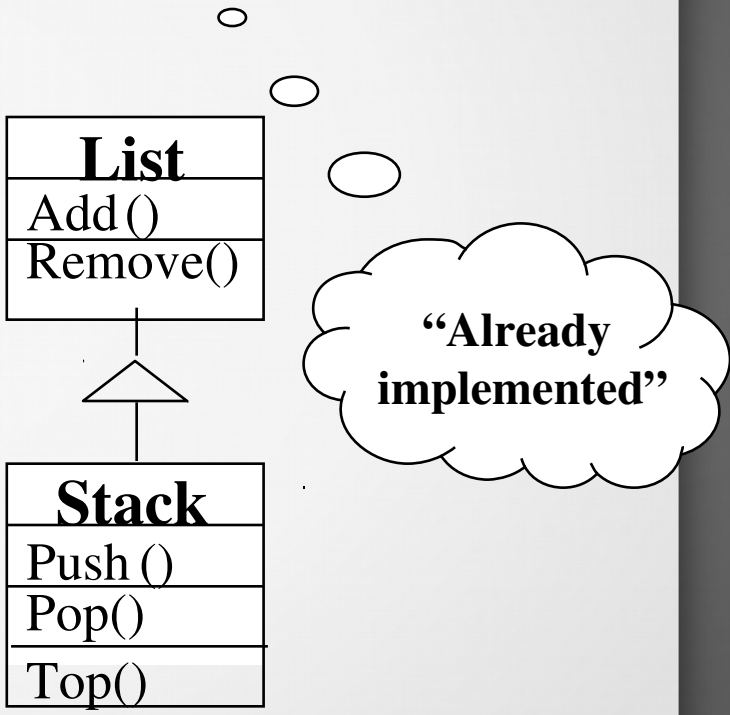


riuso: ereditare l'implementazione

- esiste gia' una classe quasi identica a quella che serve (utile se la classe ha molte proprieta')
- esempio: una classe pila che eredita da lista

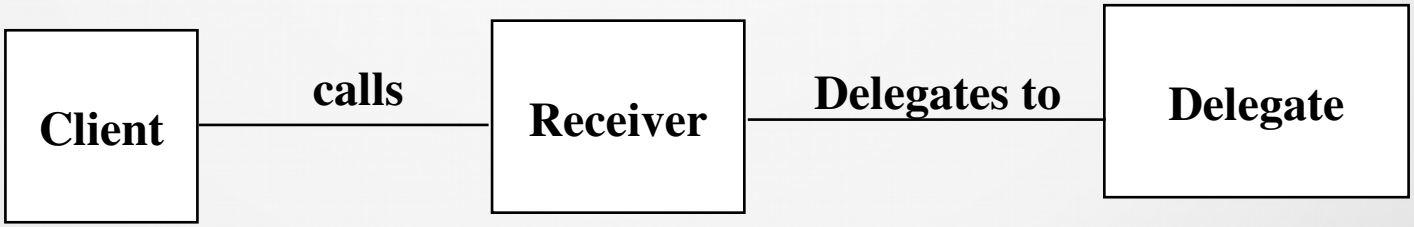
e' equivalente anziche' pop()

richiamare Remove()?



delegazione

- due oggetti coinvolti:
- l’oggetto ricevente delega l’operazione ad un secondo che la realizza affettivamente
- lo sviluppatore puo’ controllare l’accesso al delegato



di nuovo: ereditare o delegare ?

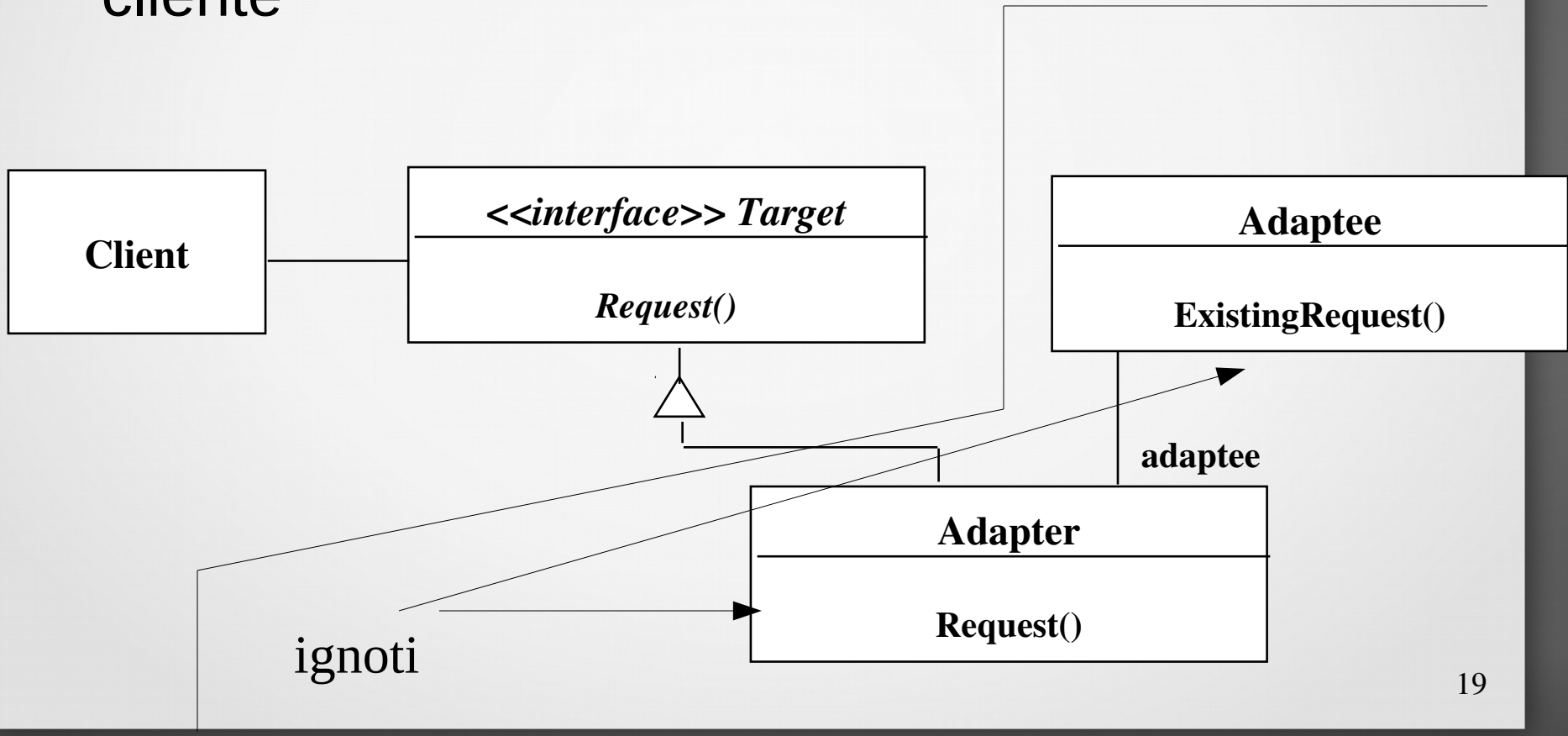
- delegare
  - pro: flessibilita': l'oggetto delegato puo' essere sostituito a runtime e cambiare comportamento
  - contro: inefficienza, controllo difficile
- ereditare
  - pro: uso facile, supportato dai linguaggi
  - contro: cambi all'antenato possono alterare le sottoclassi esistenti

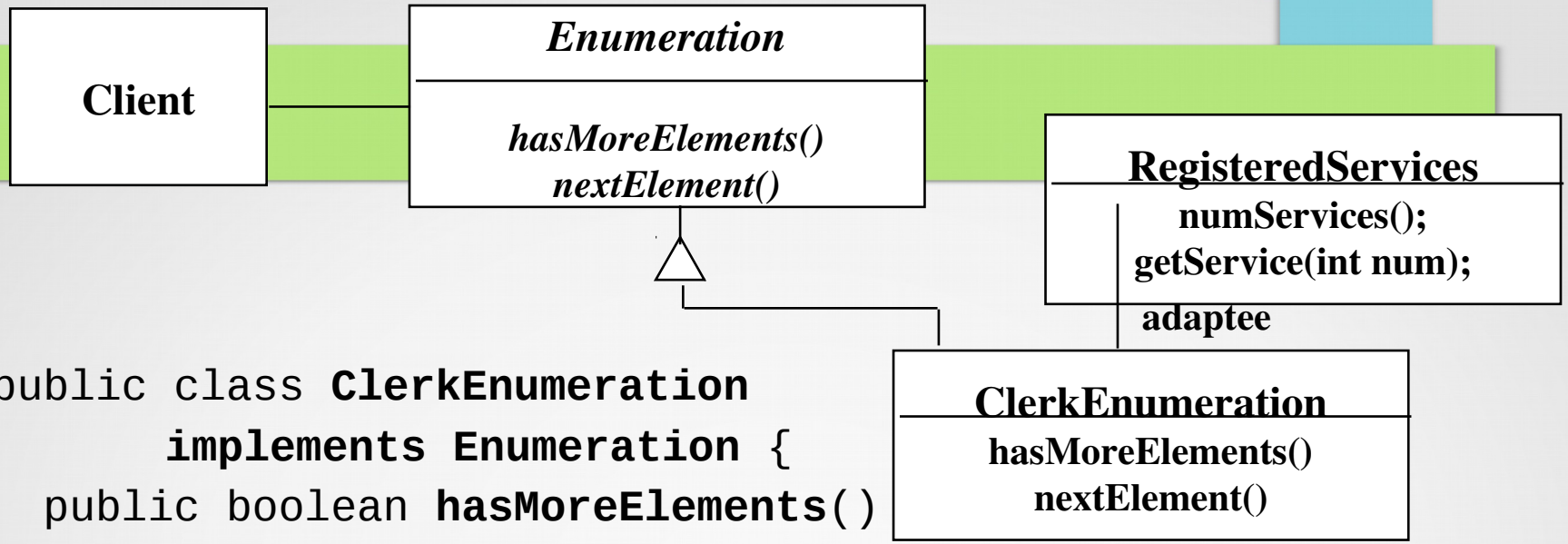
schema “Adattatore”

- converte l’interfaccia fornita da una classe in un’altra compatibile con il cliente utilizzatore
- fornita per sistemi legacy
- detta anche “wrapper”

schema “Adattatore”

- la delegazione e’ usata per legare l’ adattatore all’ oggetto preesistente
- l’adattatore realizza l’interfaccia invocata dal cliente

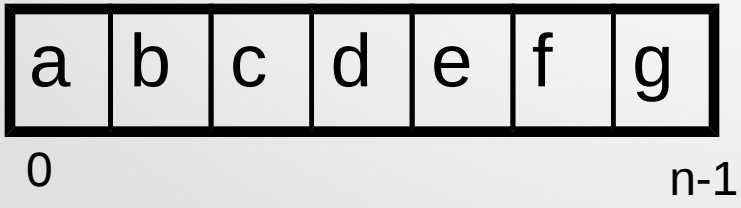


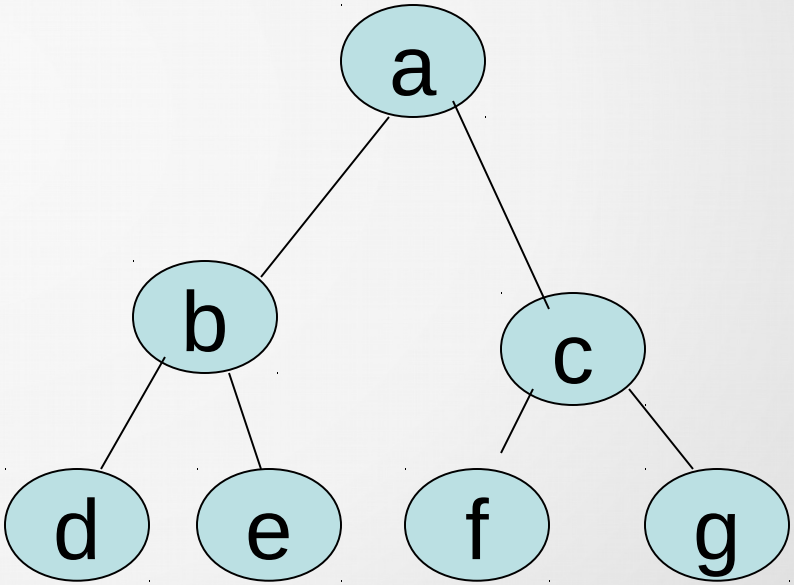



```
public class ClerkEnumeration
       implements Enumeration {
  public boolean hasMoreElements() {
    return this.clerkIdx <= adaptee.numClerk();
  }
  public Object nextElement() {
    if (!this.hasMoreElements()) {
       throw new NoSuchElementException();
    }
    return adaptee.getClerk(this.currentClerkIdx++);
  }
```

in questo modo..

la struttura degli impiegati potrebbe essere realizzata in molti modi diversi
ma l'adattatore fornisce una "vista" che le rende uguali rispetto alle operazioni dell'interfaccia. Differenze fra le due strutture?





schema “Ponte”

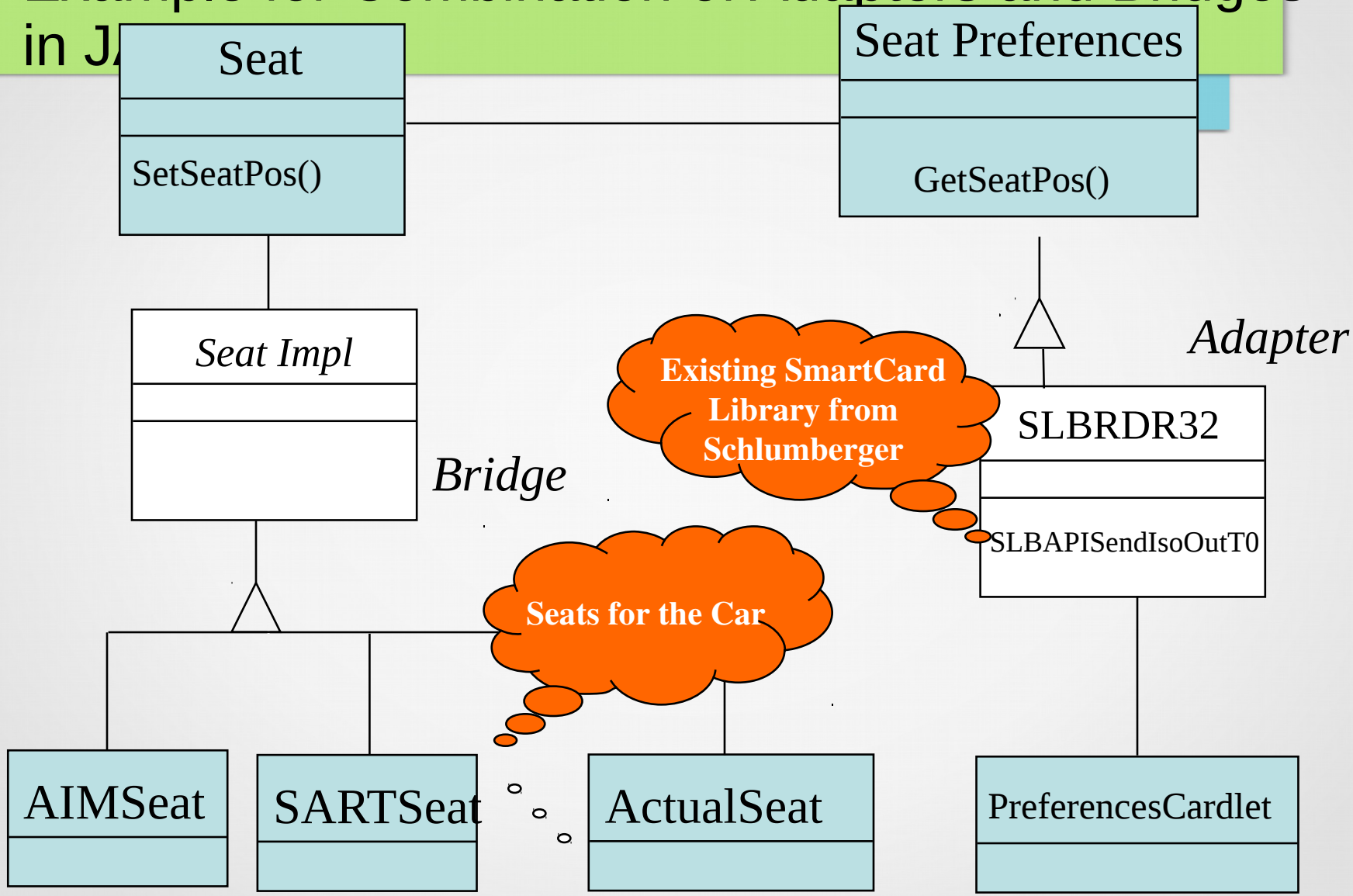
Usare un ponte per sconnettere una astrazione dall’implementazione, per poterle evolvere indipendentemente

Permette di decidere dinamicamente quale implementazione di una certa astrazione venga utilizzata

confronto tra ponte ed adattatore

- entrambi usati per nascondere i dettagli implementativi
- l’adattatore connette sistemi disparati (legacy)
- il ponte e’ progettato a priori per consentire di estendere un sistema in futuro con componenti ignoti
- una volta che si connette al nuovo componente, diventa un adattatore

Example for Combination of Adapters and Bridges in J



Il pattern dei pattern

- in tutti questi pattern, lo schema e' sempre piu'o meno lo stesso:
  - 1 il cliente invoca operazioni su una interfaccia
  - 2 un oggetto mediatore (che realizza l'interfaccia) le riceve, e le traduce in 1..N invocazioni a 1..N oggetti target.
  - 3 svolge eventualmente ulteriori elaborazioni, assembla la risposta e restituisce il risultato al cliente
  - Esempio connessione vari clienti

esempi di pattern

- http://www.dofactory.com/

interfacce e sistemi fisici

- come detto N volte, e’ fondamentale progettare un sistema SW in cui i componenti siano il piu’ possibile indipendenti.
- fate il paragone con un impianto idraulico
  - volete che i vari componenti abbiano il minimo numero di tubi che li connettano
  - che siano di tipo standard (se cambiate da una caldaia all’altra ad es.)
  - esiste un unico punto di ispezione per ogni gruppo di tubi

interfacce

- per realizzare un sistema modificabile occorre sempre introdurre una indirettezza per disaccoppiare i vari componenti
- a volte a forza di indirettezza e generalizzazioni non si capisce piu' nulla
- le interfacce sono piu' utili nella parti fluide del sistema, in quelle piu' stabili meno
- le interfacce sono sempre piu' preferite ad ereditarieta' di implementazione anche perche' ormai si **programma sempre piu' tramite configurazione e composizione**