

# Ingegneria del Software

## 16. Progettazione dettaglio

Dipartimento di Informatica  
Università di Pisa  
A.A. 2014/15

# classificatore strutturato

- È un classificatore di cui si vede la struttura interna, data in termini di *parti, porti e connettori*
- Un classificatore strutturato definisce l'implementazione di un classificatore
  - Così come le interfacce del classificatore definiscono *cosa* deve fare
  - la sua struttura interna definisce *come* viene fatto il lavoro
- I tipi che definiscono le parti contenute in un classificatore strutturato possono a loro volta essere strutturati, ricorsivamente

# parte

- Una parte ha un nome, un tipo e una molteplicità (anche se sono tutti facoltativi)
  - nomeParte : Tipo [molt]
- Una parte  $p:T$  descrive il ruolo che una istanza di  $T$  gioca all'interno dell'istanza del classificatore la cui struttura contiene  $p$
- La molteplicità indica quante istanze possono esserci in quel ruolo
- Un'istanza di  $p$  è un'istanza di  $T$  (e quindi di un qualunque sottotipo)

# connettore

- Un connettore può essere di due tipi
  - di assemblaggio (assembly connector)
    - esprime un legame che permette una comunicazione tra due istanze nei ruoli specificati dalla struttura
  - di delega (delegation connector)
    - identifica l'istanza che realizza le comunicazioni attribuite a un porto

# notazione connettori

- Assemblaggio
  - collega i porti delle due parti le cui istanze devono comunicare
  - si usa la notazione lollipop
- Delega
  - si usa una semplice linea tra il porto della parte e il porto della struttura composita

# diagrammi di struttura composita

- Mostrano la struttura di dettaglio (o struttura interna) di un classificatore a tempo di esecuzione
  - di solito di un componente, ma anche di una classe
- Sono definiti in termini di
  - parti (eventualmente con molteplicità)
  - porti (eventualmente con interfacce fornite e/o richieste) per mostrare il comportamento visibile all'esterno
  - connettori
- Esplicitano le interazioni fra le parti e i punti di interazione (porti) con l'esterno

# dalla specifica all'implementazione...

- ...passando per le strutture composite!!

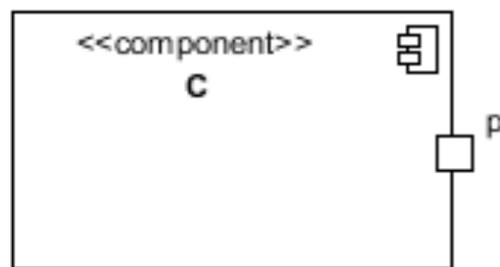


Fig. 1

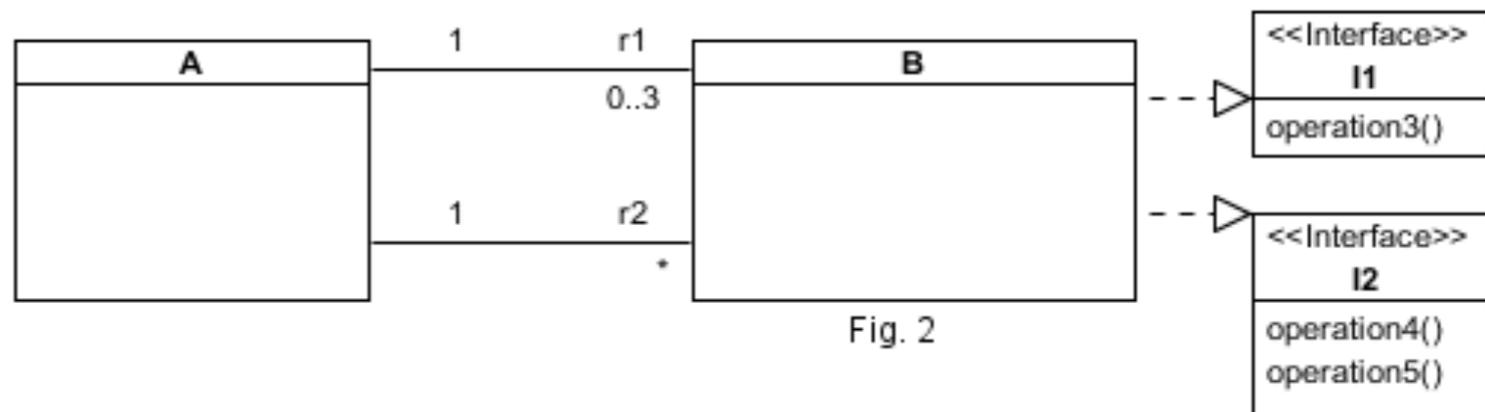


Fig. 2

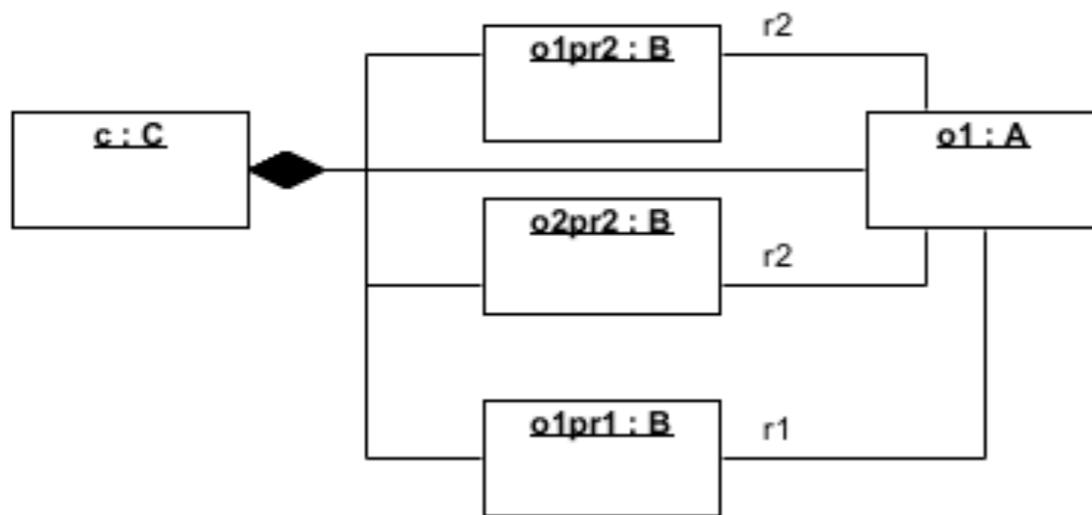


Fig. 3

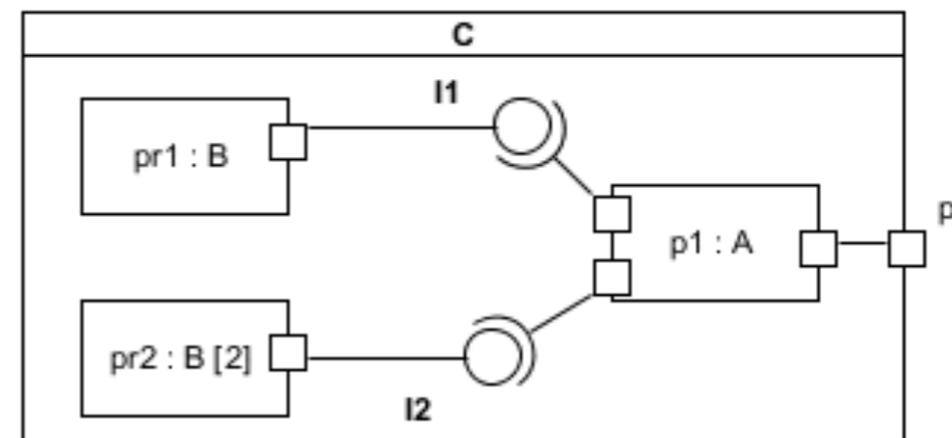


Fig. 4

# legenda

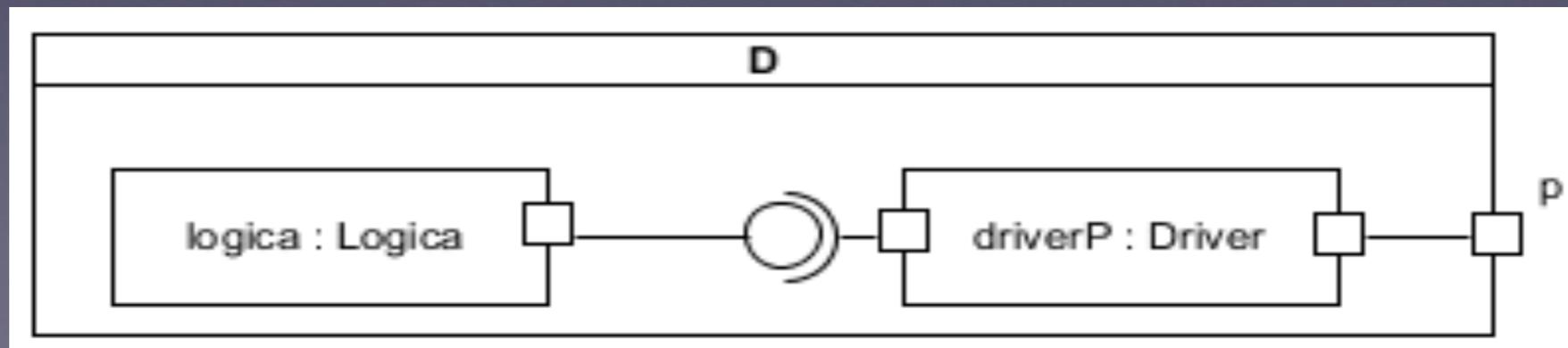
- In Fig. 1 si mostra la componente C
- In Fig. 2 si specificano le classi A e B, con associazioni, ruoli e interfacce. Si assume che un'istanza di A usi l'interfaccia I1 (I2) di B quando connessa con un'istanza di B in ruolo r1 (r2 risp.)
- In Fig. 3 si mostra un diagramma degli oggetti conforme al diagramma in Fig. 2
- In Fig. 4 si mostra la struttura di C, che fornisce una astrazione del diagramma in Fig. 3. Introduce vincoli rispetto al diagramma delle classi: dice che uso istanze di A e di B, collegate in un certo modo, con dato ruolo nel contesto di C, e date molteplicità

# come strutturare una componente

- Un modo conveniente di strutturare una componente prevede di separare gli aspetti di comunicazione da quelli di realizzazione delle funzionalità richieste
  - favorisce modificabilità e comprensibilità della componente

# diagramma minimo, 1

- Supponiamo di avere una componente D con un porto p
- La struttura di D dovrà avere almeno due parti
  - driverP, che realizza la parte di comunicazione richiesta per implementare il porto
  - logica, che realizza la funzionalità richiesta alla componente
- e due connettori
  - di delega tra driverP e P
  - di assemblaggio tra driverP e logica (il verso del lollipop non è fissato a priori)



# diagramma minimo, 2

- In generale, data una componente con  $n$  porti, la sua struttura minima dovrà prevedere
  - $n$  parti col ruolo di driver
    - collegate ognuna a un porto, con un connettore di delega
  - una parte che realizza la logica della componente
    - collegata a tutti i driver con connettori di assemblaggio
- I nomi “driver” e “logica” per le parti di una componente non sono standard di UML: sono usati per scopi didattici, per aiutare a distinguere le loro responsabilità

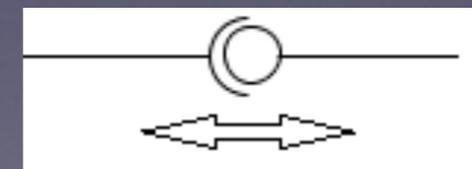
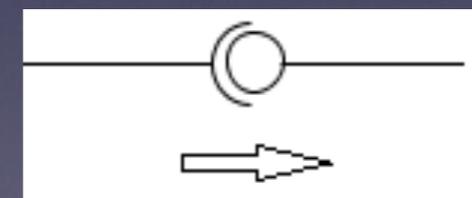
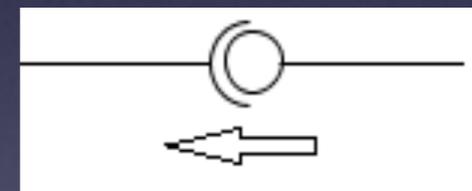
# diagramma non minimo

- È ovviamente possibile, quando necessario, dettagliare maggiormente la struttura di una componente
  - raffinando la “logica” in un insieme di parti interconnesse
    - con connettori di assemblaggio
    - con dipendenze
  - introducendo dei “proxy”, per realizzare comunicazioni con sistemi remoti o chiamate al sistema operativo
    - connessi alle parti che descrivono la logica, con connettori di assemblaggio
    - non sono collegati ai porti, perché non realizzano la comunicazione con altre componenti del sistema che si sta progettando
- Il nome “proxy” è introdotto per scopi didattici, per distinguere il ruolo da quello di un driver: non è corretto pensare sia simile al proxy di RMI

# verso del lollipop, verso dell'informazione

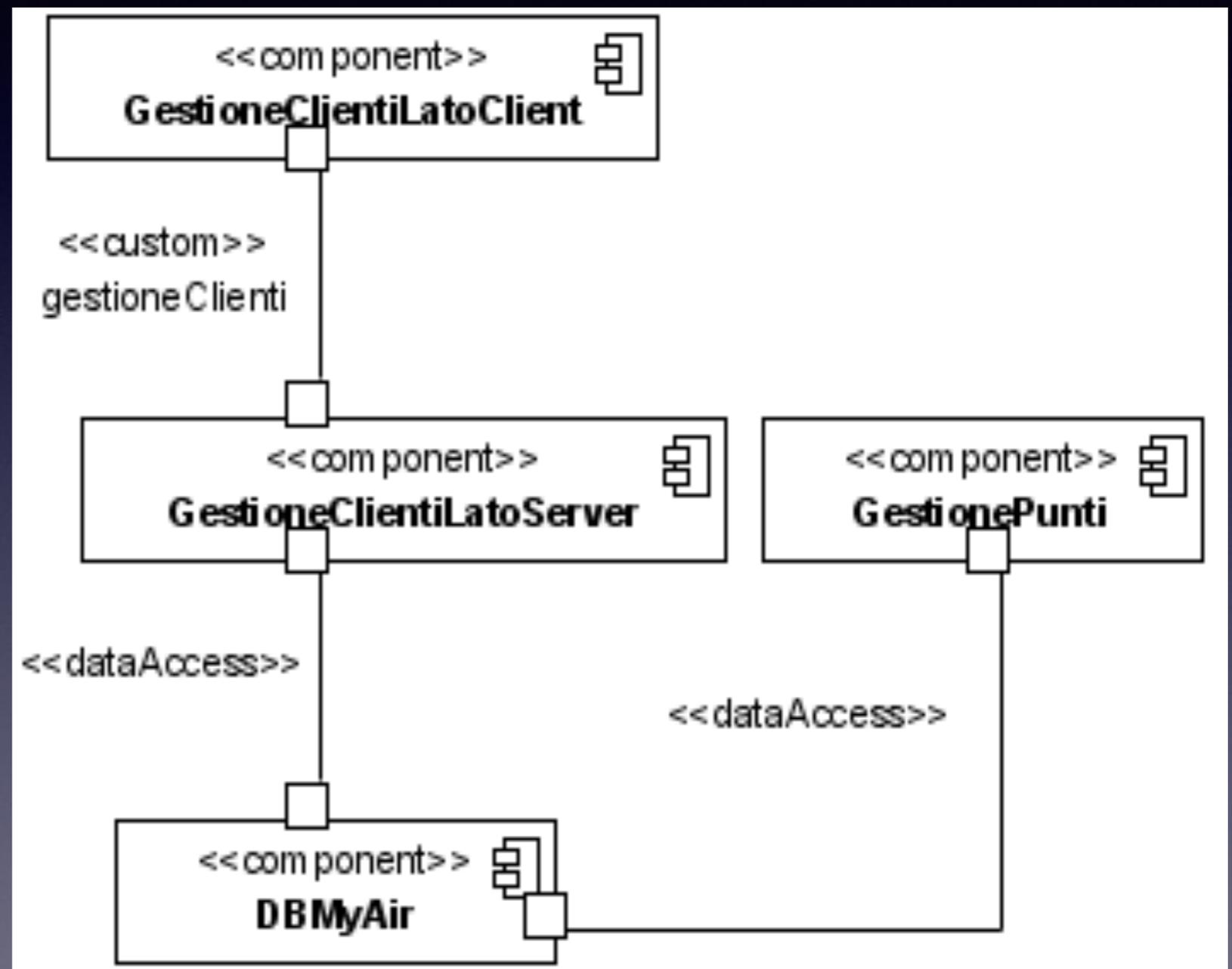
- Il verso del lollipop non ha alcun legame con il verso in cui viaggiano i dati: ha solo a che vedere con chi ha il controllo e con chi, interrogato, risponde

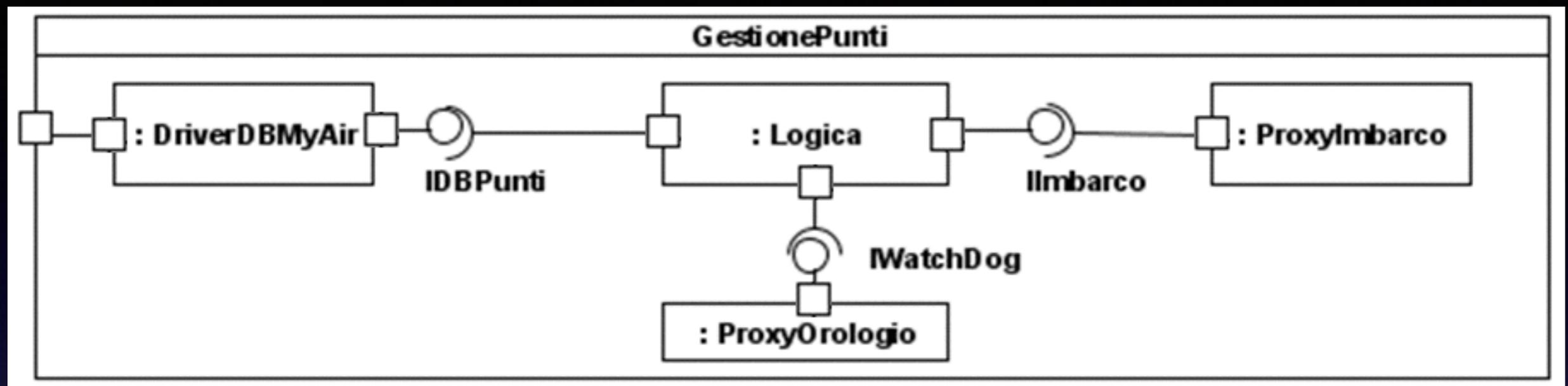
- un'interfaccia con solo operazioni di read
- un'interfaccia con solo operazioni di write
- un'interfaccia con operazioni di read e write



# esercizio myAir

- Vista C&C: la componente GestionePunti realizza i casi d'uso AccumuloPunti e Aggiornamento Annuale





| Parte         | Responsabilità  |
|---------------|---|
| DriverDBMyAir | Realizza l'interfaccia IDBPunti che permette a Logica di accedere tramite il solo porto della componente al DBMyAir |
| ProxyImbarco  | Realizza la connessione con il sistema Imbarco, passando alla Logica la lista degli imbarcati                       |
| ProxyOrologio | Sveglia la logica alla data e ora richiesta tramite IWatchDog   |
| Logica        | Realizza i casi d'uso, sfruttando le interfacce introdotte  |