Ingegneria del Software
6. Classi e oggetti

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

classi e oggetti

- Una *classe* cattura
  - un concetto nel dominio del problema o della realizzazione
- Una classe descrive
  - un insieme di oggetti con caratteristiche simili (lo stesso *tipo*!)
- Un *oggetto* è un'entità caratterizzata da
  - un'identità
  - uno stato
  - un comportamento

classificatori e istanze

- Le classi sono *classificatori*
- Gli oggetti sono *istanze*

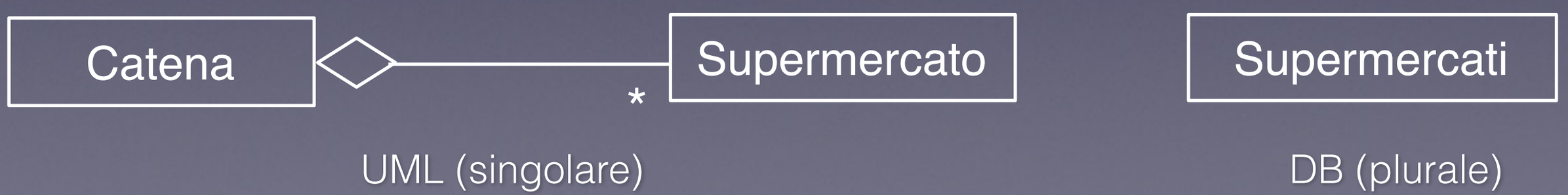
- Modellare a livello dei classificatori significa vincolare i modelli a livello di istanza

ancora classi e oggetti

- In teoria i modelli a livello di istanza possono esistere solo in un ambiente definito dai modelli a livello dei classificatori
  - Le variabili in un linguaggio fortemente tipato possono vivere solo in un ambiente nel quale il tipo è definito
- In pratica UML è più flessibile
  - Si possono introdurre oggetti senza definirne le classi
  - UML tollera le inconsistenze

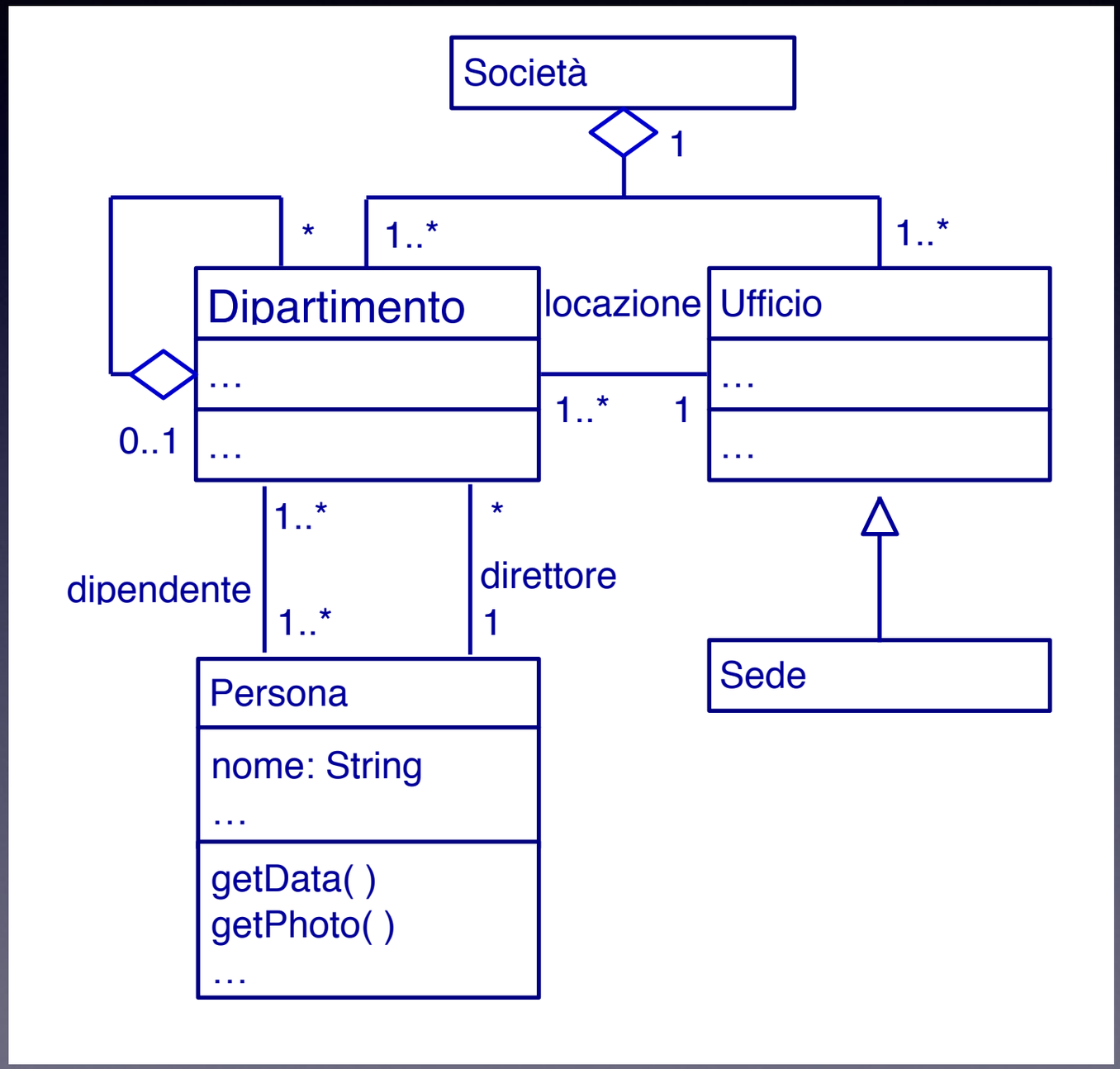
classi UML vs. entità (DB)

- DB: le classi sono intese come collezioni! Resta sottinteso che
  - ci sono più istanze
  - ci sono operazioni per visitare tutte le istanze
- La differenza è significativa più in prospettiva di progettazione che di descrizione del dominio
  - In progettazione OO e quindi in UML uso “ListaDiQcosa” come aggregato di “Qcosa”



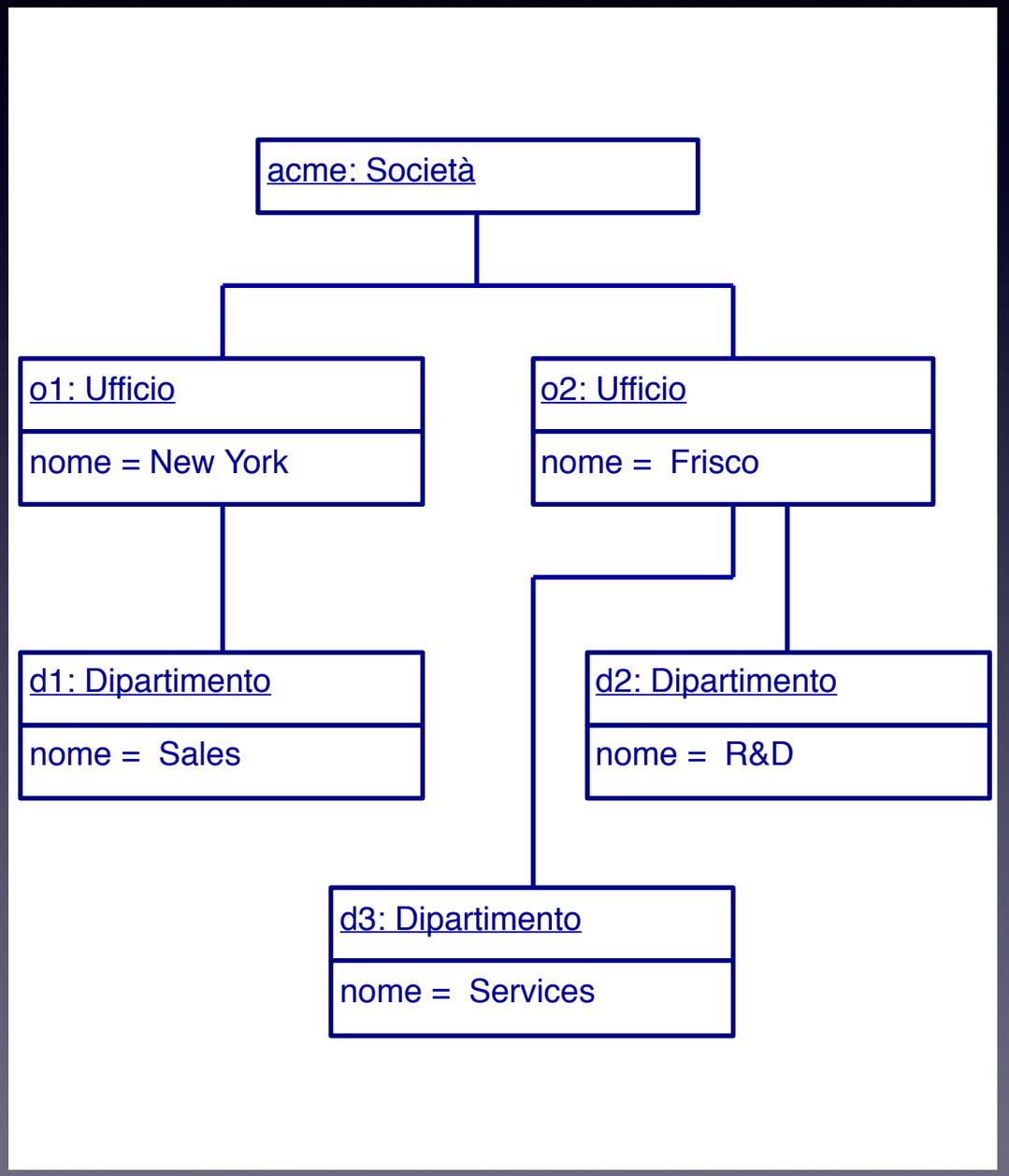
esempio: classi

- Una società è formata da dipartimenti e uffici
- Un dipartimento ha un direttore e più dipendenti
- ed è situato in un ufficio
- Esiste una struttura gerarchica dei dipartimenti
- Le sedi sono uffici

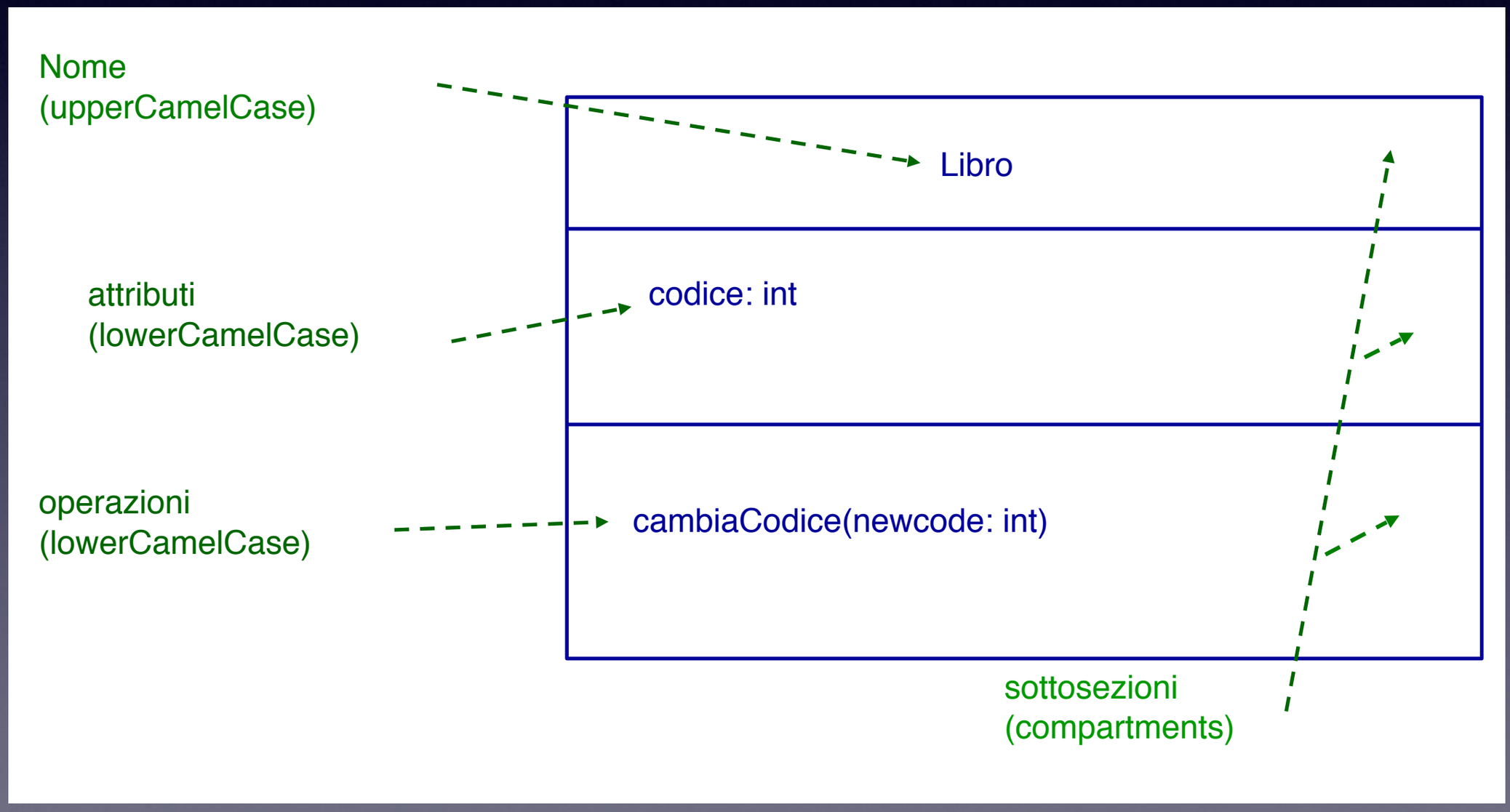


esempio: oggetti

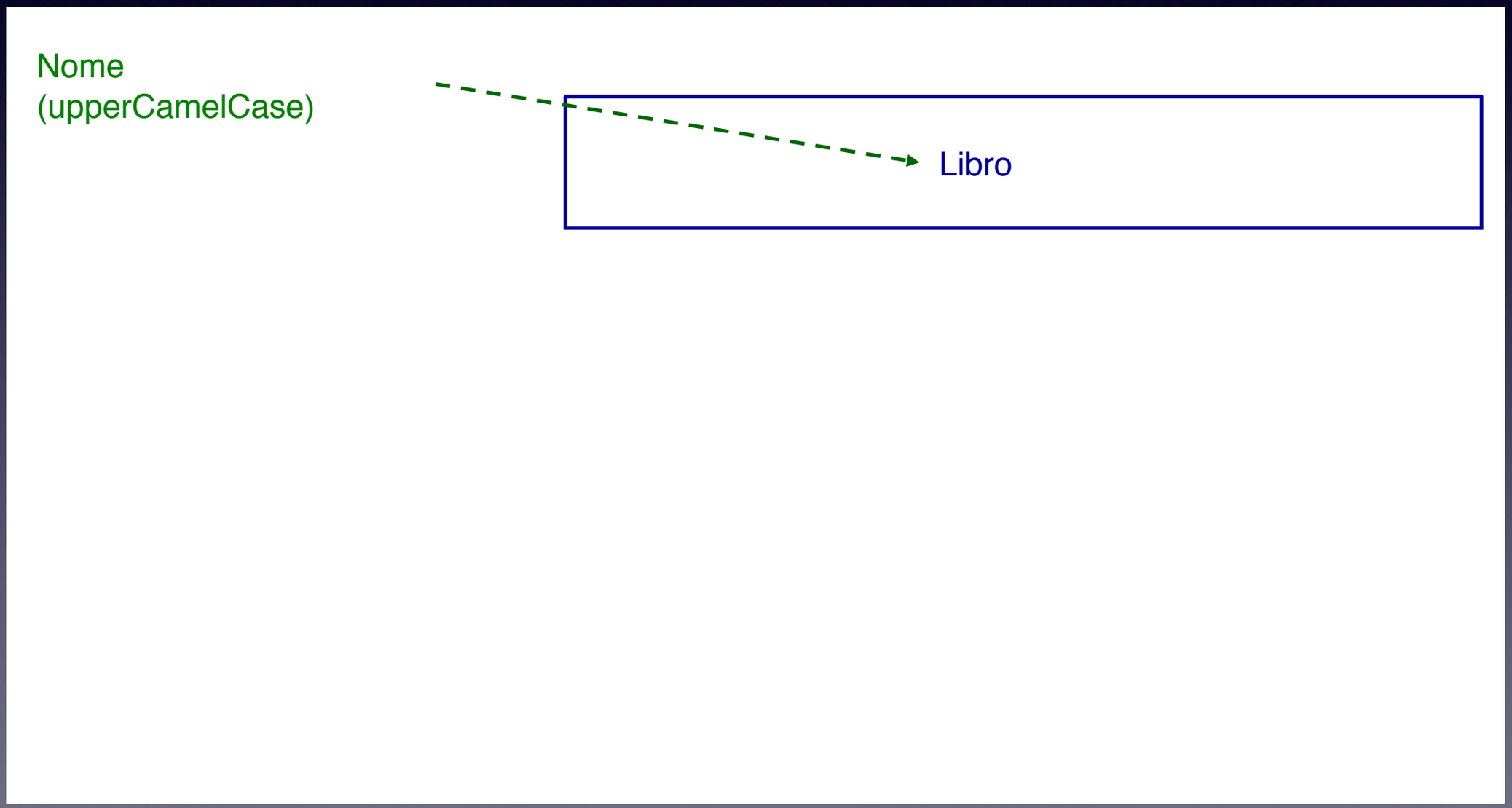
- Una società, ACME,
- ha due Uffici:
- il dipartimento Vendite è a New York,
- i dipartimenti Ricerca&Sviluppo e Servizi sono a San Francisco



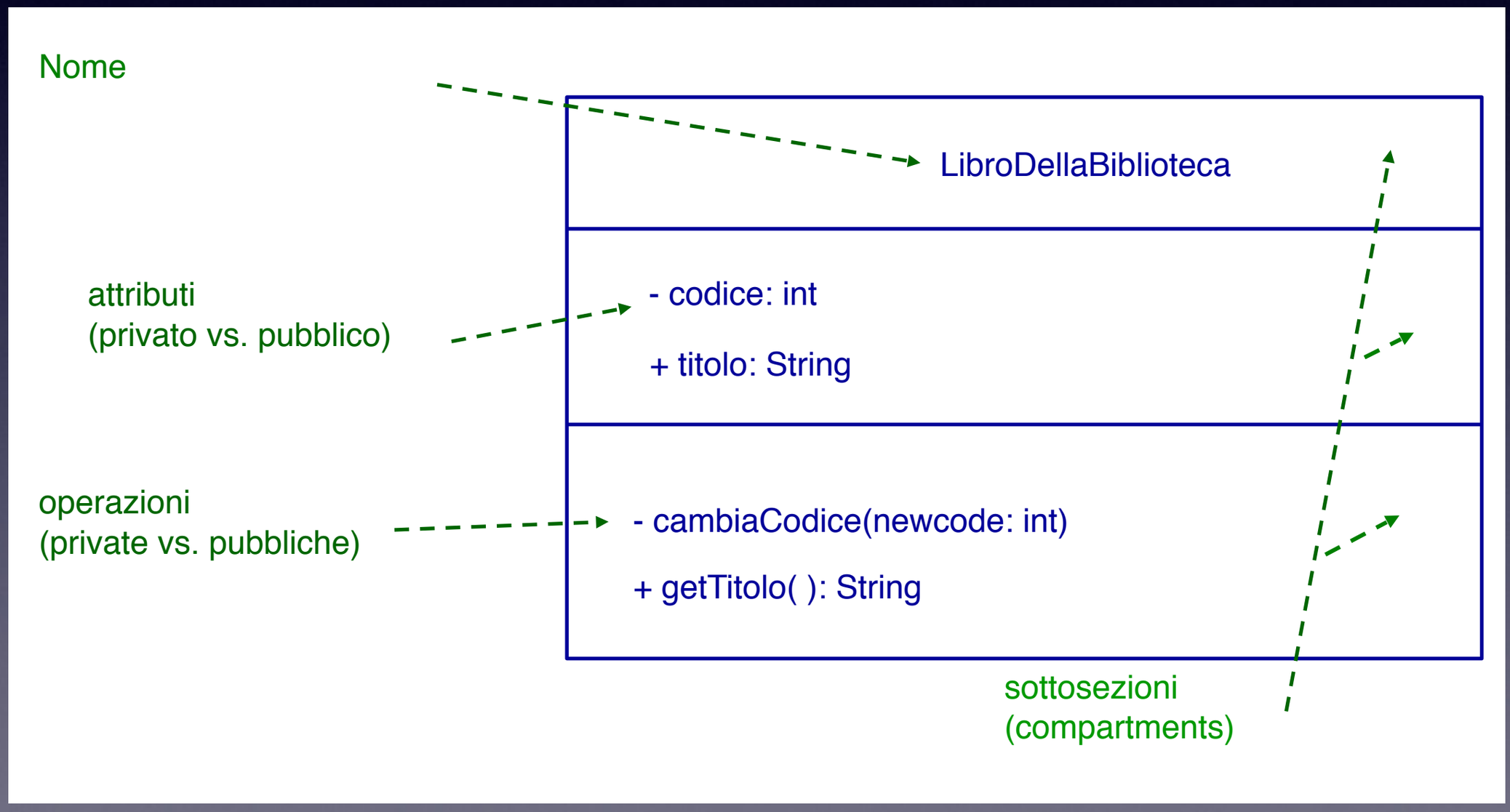
sintassi della classe



o astraendo…



sintassi della classe



semantica intesa

- Un *oggetto* è un'entità caratterizzata da
  - un'identità, uno stato, un comportamento
- L'*identità* si vede a livello di istanza
- Gli *attributi* definiscono lo stato dell'oggetto
- Le *operazioni* definiscono il suo comportamento

spazio dei nomi

- Un classificatore è uno spazio di nomi
  - gli elementi contenuti hanno un nome unico
- Per esempio un Package…
  - è un costrutto di strutturazione
  - può contenere package annidati e altri elementi (classi, diagrammi, …)

visibilità

- Un elemento è visibile all’esterno dello spazio di nomi che lo contiene, in accordo con il suo tipo di visibilità
  - \+ public: tutti
  - \# protected: i discendenti
  - \- private: solo nell’elemento stesso
  - ~ package: nello stesso package

sintassi attributi

visibilità nome: tipo [molteplicità] = valoreIniziale {proprietà}

obbligatorio

molteplicità:
array di valori

colore: Saturazione [3]

nome: String [0..1] 0 per permettere valore null

[1] può essere omesso

proprietà

{ordered}

{>=0}

visibilità attributi e operazioni

- In modo simile alla visibilità della classe
  - \+ public: accessibile a ogni elemento che può vedere e usare la classe
  - \# protected: accessibile a ogni elemento discendente
  - \- private: solo le operazioni della classe possono vedere e usare l'elemento
  - ~ package: accessibile solo agli elementi dichiarati nello stesso package

esempio

- numero intero
  - n: Integer
- numero intero positivo
  - n: Integer {>= 0}
- numero intero positivo, inizialmente a 0
  - n: Integer = 0 {>= 0}

esempio

- Punti del quadrante positivo, pubblico
  - + puntiQuadPos: Integer [2] {>= 0}
- sequenza ordinata di 10 interi compresi tra 3 e 33, privato
  - - seq: Integer[10] {>= 3, <= 33, ordered}

sintassi operazioni

visibilità nome (listaParametri): tipoRitorno {proprietà}

obbligatori

può essere vuota

listaParametri

default

direzione nome: tipo = default

in, out, inout

valore assegnato al parametro
in assenza di argomento

esempio

- Metodo pubblico che restituisce la somma di 2 interi
  - + sum (a: Integer, b: Integer): Integer
- sum con 10 valore di default del secondo parametro
  - + sum (a: Integer, b: Integer = 10): Integer
- Metodo privato che restituisce un oggetto di tipo Grafo
  - - gr( ): Grafo

attributi e operazioni statiche

- sottolineati !!

esempio

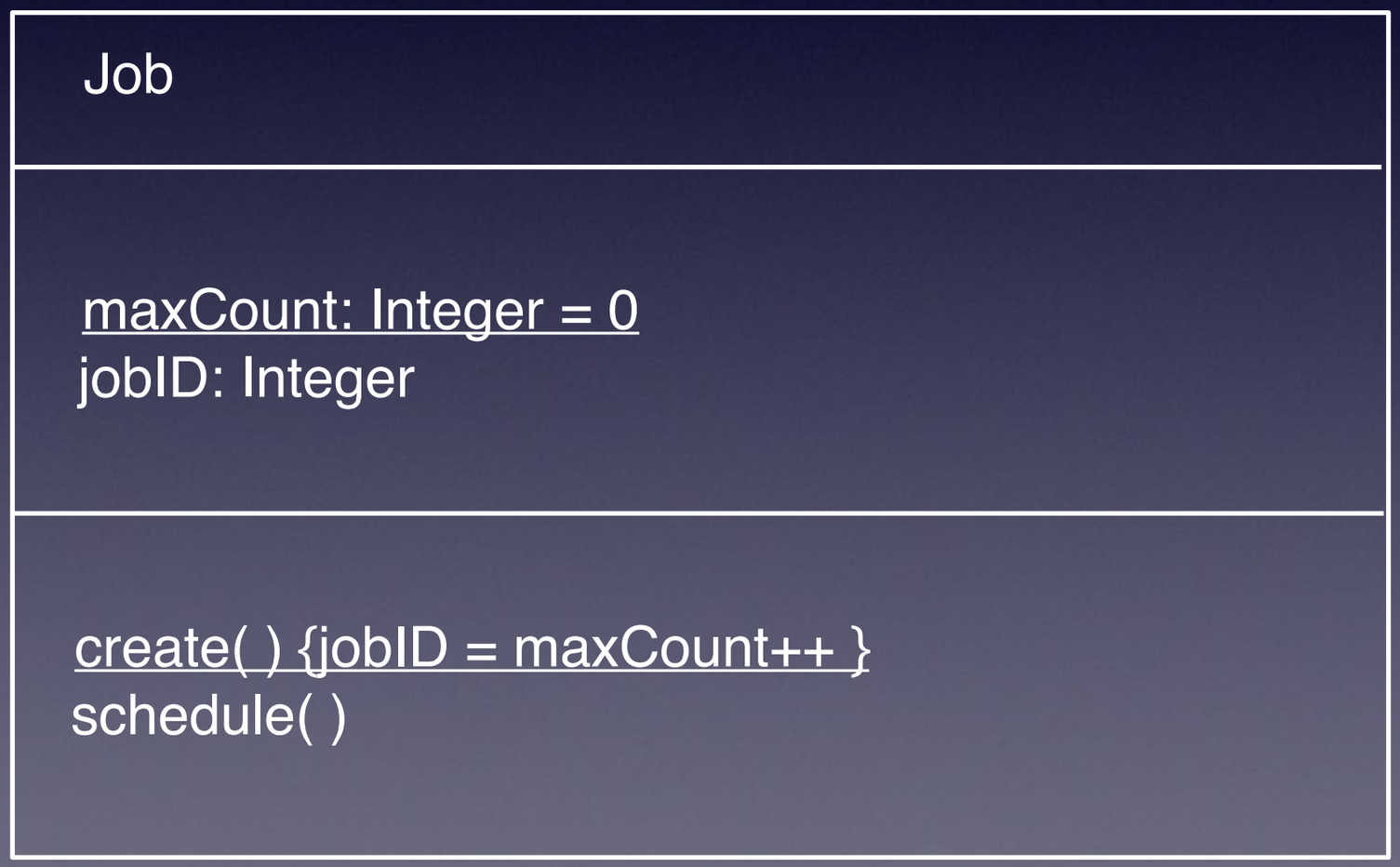
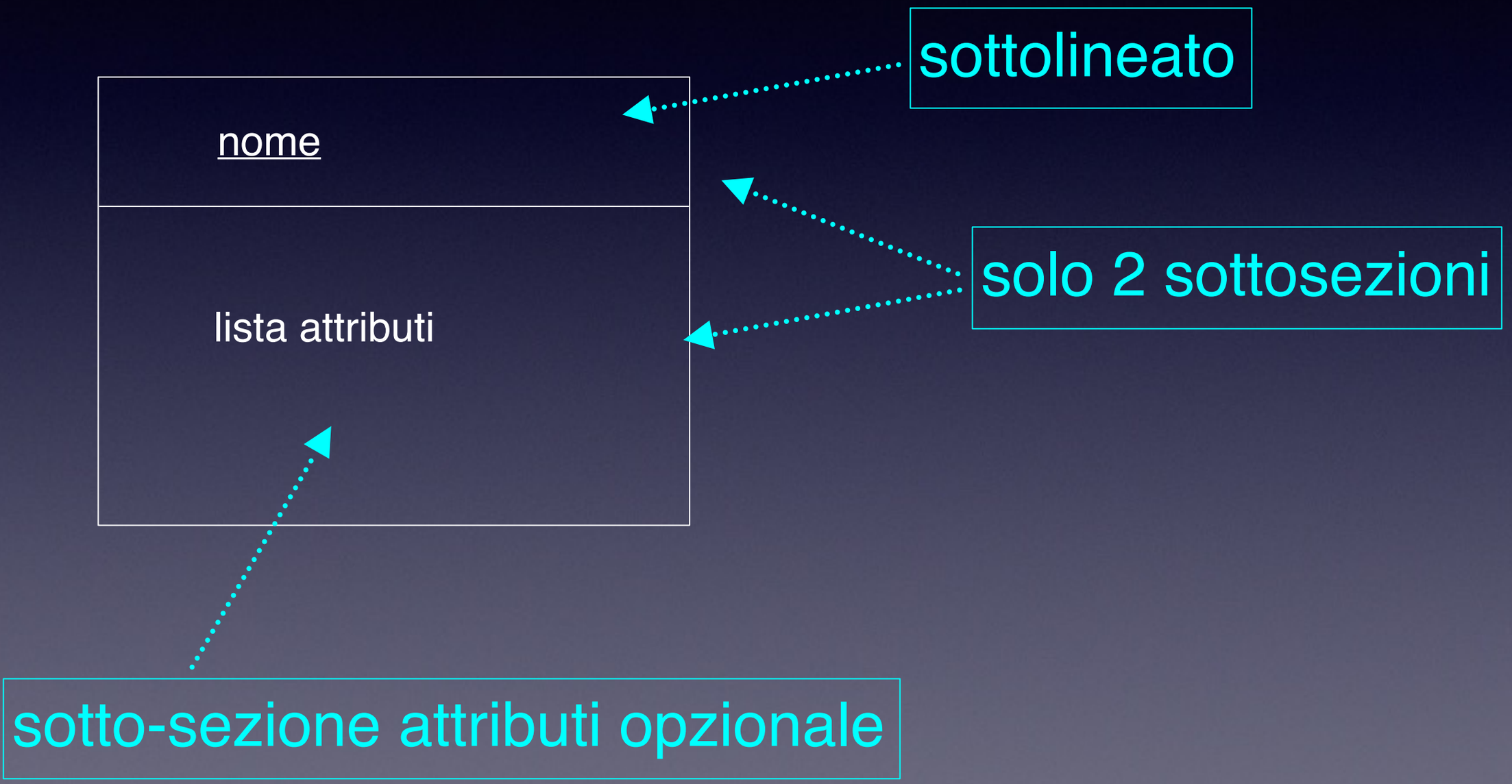

Job
maxCount: Integer = 0
jobID: Integer
create( ) {jobID = maxCount++ }
schedule( )

diagramma degli oggetti



diagrammi degli oggetti: nome

nomeoggetto: Nomeclasse

minou: Gatto

nomeoggetto

minou

: Nomeclasse

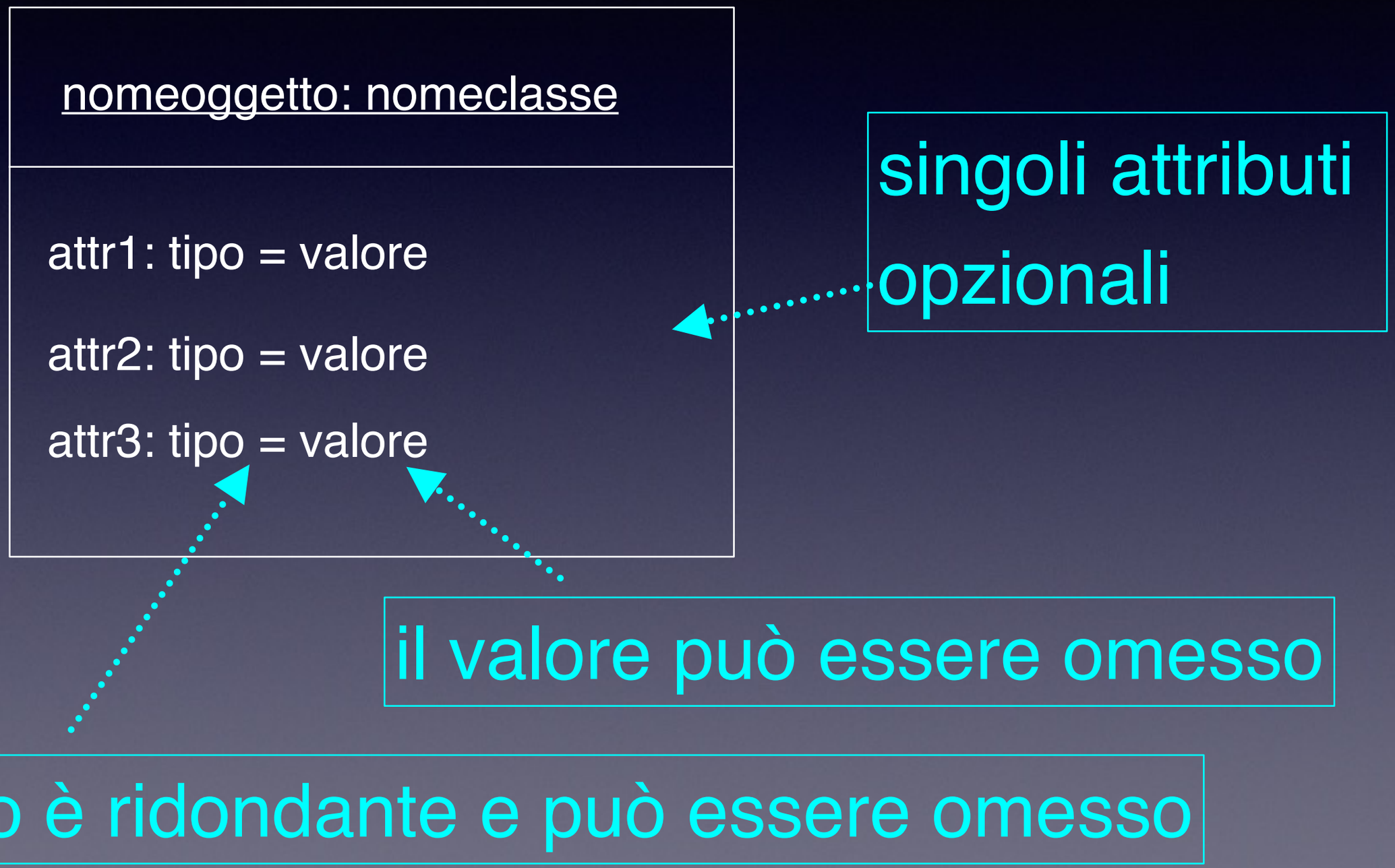
: Gatto

nomeoggetto: Nomeclasse, Nomeclasse

minou: Gatto,Cantante

[ereditarietà multipla!]

diagrammi degli oggetti: attributi



classi e oggetti

Punto
x: Real y: Real colore: Saturazione [3]

p1: Punto
x = 3,14 y = 2,78

p2: Punto
x = 1 y = 2

individuare le classi di analisi

- Cosa sono le classi di analisi?
- Come sono fatte?
- Tecniche per individuarle
  - Nome-verbo
  - CRC

cosa sono le classi di analisi

- Corrispondono a concetti concreti del dominio
  - Per esempio i concetti descritti nel glossario
  - Normalmente, ciascuna classe di analisi sarà raffinata in una o più classi di progettazione
- Evitare di introdurre delle classi di progettazione

classi di analisi: caratteristiche

- Astrazione di uno specifico elemento del dominio
- Numero ridotto di responsabilità (funzionalità)
- Evitare le classi “onnipotenti”
  - Attenzione quando si chiamano “sistema”, “controllore”, ….
- Evitare funzioni travestite da classi
- Evitare gerarchie di ereditarietà profonde (>=3)
- Coesione e disaccoppiamento
  - Tenere responsabilità simili in una classe
  - Limitare interdipendenze tra classi

classi di analisi: livello di dettaglio

- Operazioni e attributi solo quando veramente utili
- Le classi di analisi dovrebbero contenere attributi e operazioni ad “alto livello”
- Limitare la specifica di tipi, valori, etc.
- Non inventare mai niente!

identificazione delle classi

- Problema classico delle prime fasi di sviluppo
- Approccio data driven
  - Si identificano tutti i dati del sistema e si dividono in classi (ad esempio mediante identificazione dei sostantivi)
- Approccio responsibility driven
  - Si identificano le responsabilità e si dividono in classi (ad esempio mediante CRC Cards)

analisi nome-verbo

- Sostantivi → classi o attributi
- Verbi → responsabilità o operazioni

- Passi

1. Individuazione delle classi
2. Assegnazione di attributi e responsabilità alle classi
3. Individuazione di relazioni tra le classi

analisi nome-verbo

- Problemi ricorrenti

1. Tagliare le classi inutili

    1.1. Trattare i casi di sinonimia

2. Individuare le classi nascoste (le classi implicite del dominio del problema che possono non essere menzionate esplicitamente)

    2.1. In un sistema di prenotazione di una compagnia di viaggi si potrebbe parlare di prenotazione, richiesta, ma tralasciare il termine ordine

tagliare le classi inutili

- Elenco dei sostantivi
- Eliminazione dei sostantivi riconosciuti come
  - **Sinonimi**
  - **Eventi o operazioni**
  - **Metalinguaggio** **(sistema)**
  - **Inutili** **(estranei al sistema)**
  - **Attributi** **(titolo del libro....)**

quidditch

- Il preside di Hogwarts, Albus Silente, chiede la realizzazione di un sistema per la visualizzazione e l'archiviazione dei punti segnati durante le partite del campionato di Quidditch del collegio. Il sistema deve permettere di gestire i seguenti eventi: inizio partita, goal (con punti realizzati e nome del realizzatore), cattura del boccino (con punti realizzati e nome del cercatore), fine partita. Vengono inseriti dall'aiuto arbitro. Siccome si possono portare le bacchette magiche in campo il sistema dovrà garantire opportuna sicurezza

CRC cards

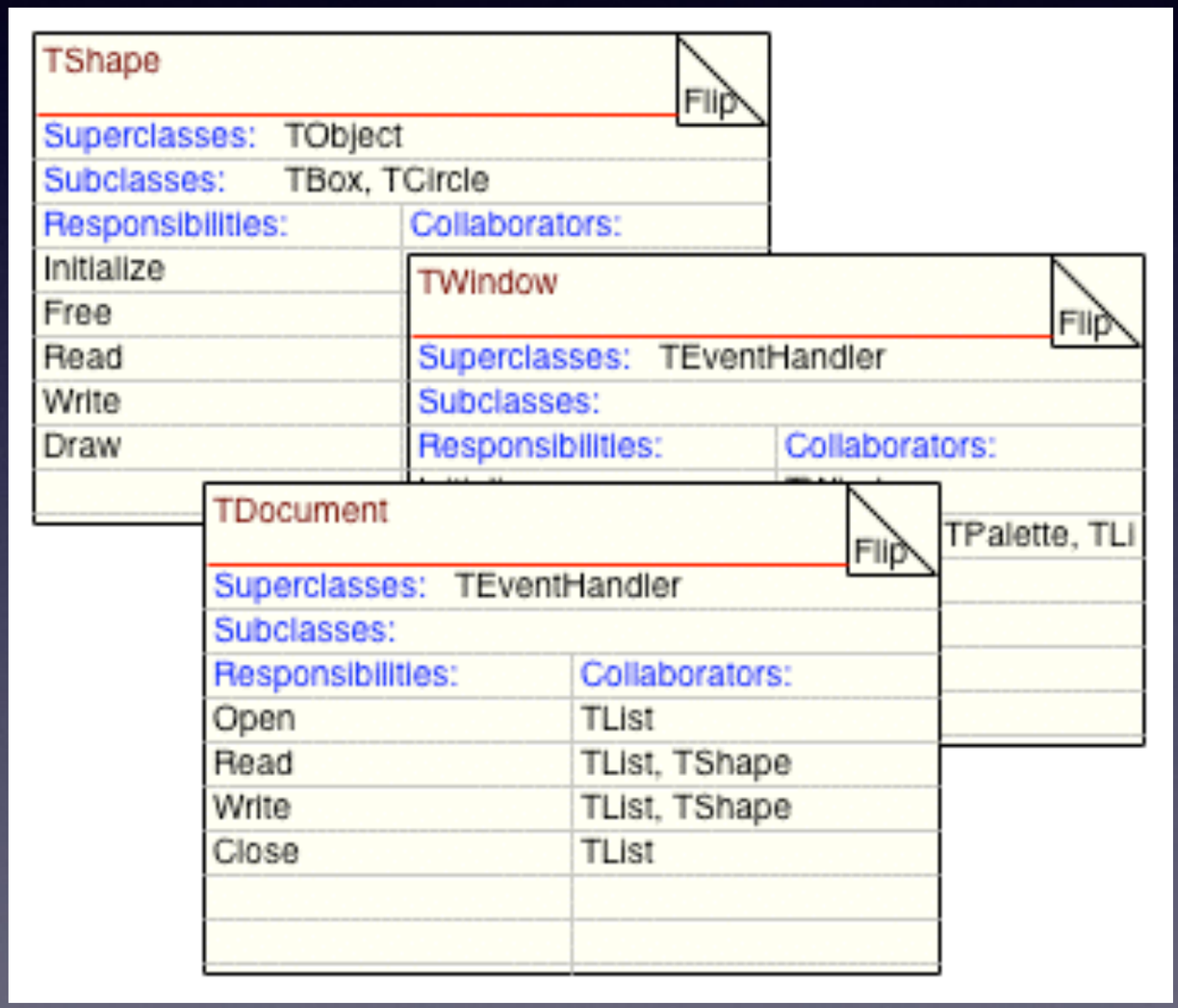
- Class-Responsibility-Collaboration Cards
- Ideate da Ward Cunningham e Kent Beck (Smalltalk) come una tecnica per insegnare a programmatori con poca esperienza in linguaggi OO a pensare in termini di oggetti

  [oopsla 1989]
- Non fanno parte di UML, ma sono utili per realizzare i diagrammi UML

  [check wiki]

CRC cards

- Responsabilità
  - Una funzionalità che la classe deve realizzare
- Collaboratore
  - Le classi che partecipano alla realizzazione di una data responsabilità



struttura delle CRC cards

- La notazione usata per rappresentare le CRC cards è la seguente: fare una tabella con il nome della classe in cima, sotto a sinistra le responsabilità e a destra, per ogni responsabilità, i collaboratori

Copia di libro	
Responsabilità	Collaboratori
Mantenere lo stato di una particolare copia di un libro	
Informare il libro su prestiti e restituzioni	Libro