UNIVERSITÀ
DEGLI STUDI DI TRIESTE

dipartimento
di ingegneria
e architettura

Corso di Laurea in Tecniche di Radiologia Medica per immagini e Radioterapia

Informatica Medica

2CFU – 20 ore

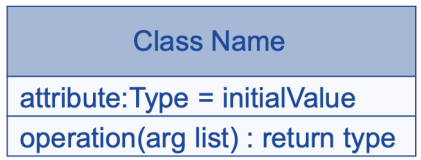
CENNI DI UNIFIED MODELING LANGUAGE

Prof. Sara Renata Francesca Marceglia

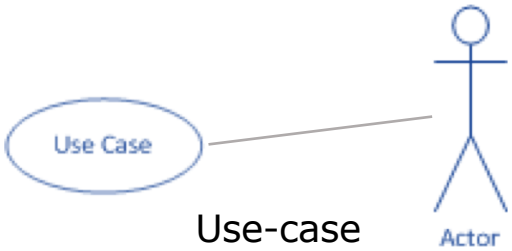
CHE COS'È UML

"The Unified Modeling Language (UML) is a graphical language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems".

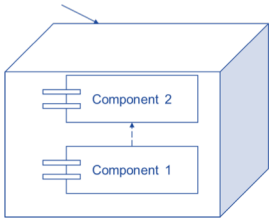
STATIC VIEWS



Class Diagram



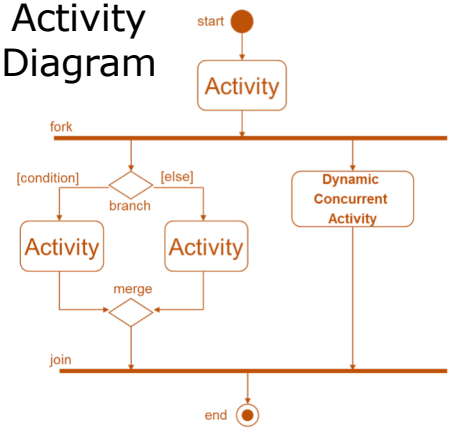
Use-case Diagram

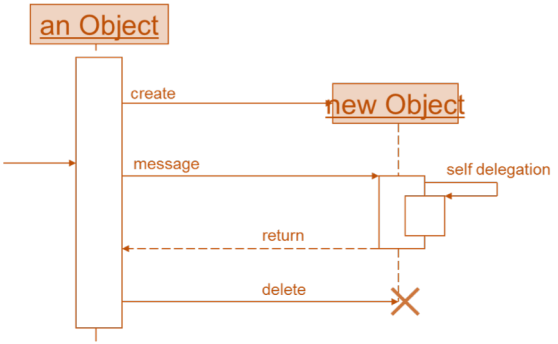


Component Diagram

DYNAMIC VIEWS

Activity Diagram





Sequence Diagram

OBIETTIVI DI UML

- UML è un linguaggio e non un metodo
- UML è costituito da un insieme di linguaggi che consentono di dare descrivere e modellare diversi aspetti di un sistema:
  - Aspetti statici
  - Aspetti dinamici
  - Interazioni tra le diverse componenti

- UML facilita la **comunicazione**→
  - tra analisti e esperti di dominio
  - Tra analisti e progettisti
  - Tra progettisti e sviluppatori

- Linguaggio grafico ma formale, preciso e non ambiguo
- Dà una visione concisa del sistema da diversi punti di vista
- Non necessita di conoscenza del codice
- UML è basato sulla logica della **programmazione a oggetti**

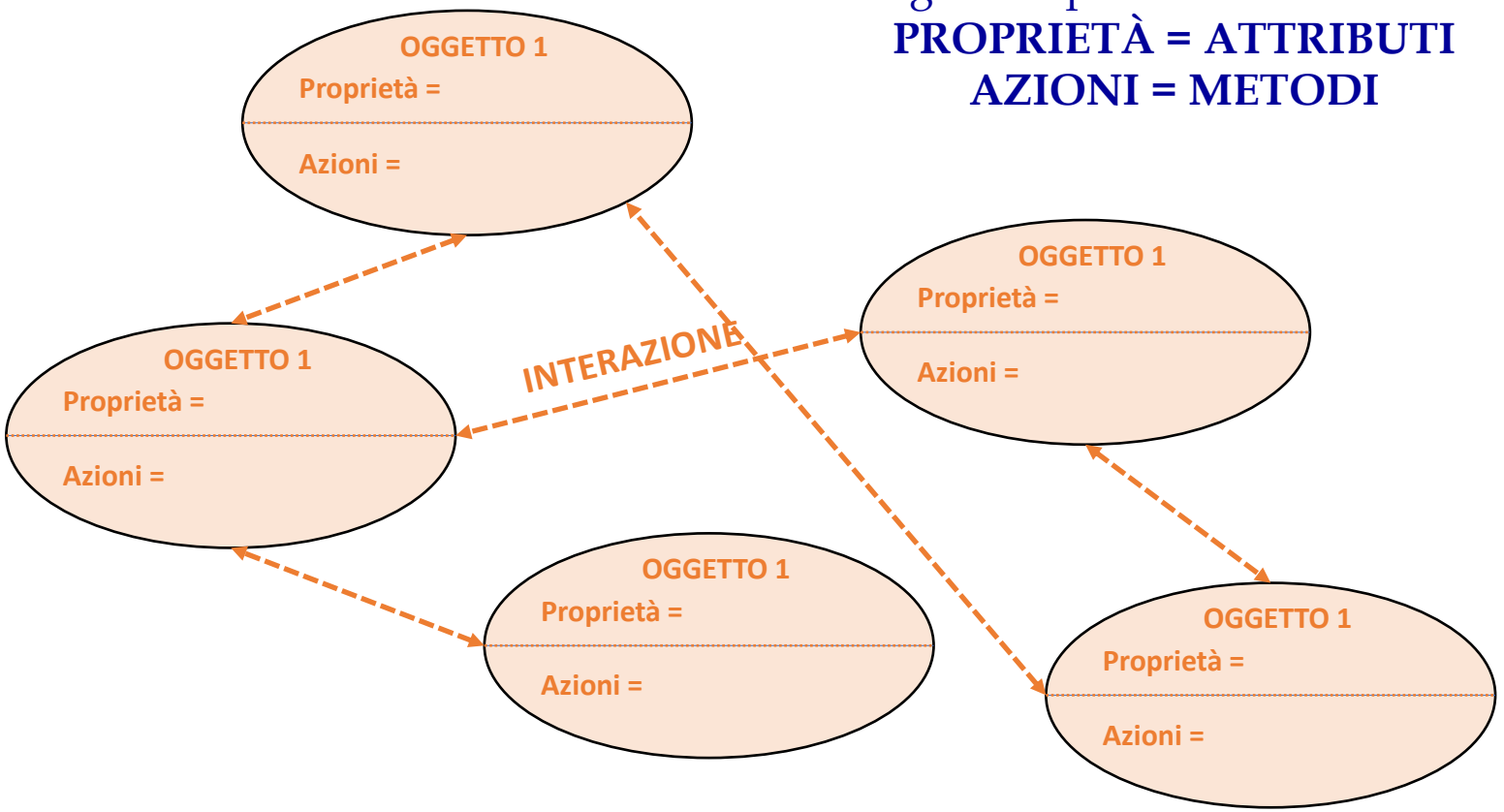
PERCHÈ SI USA UML

- Si deve operare con esperti di dominio
- Si devono catturare gli aspetti più importanti senza perdersi nei dettagli
- La notazione di UML è flessibile → molti diagrammi rappresentano aspetti diversi dei concetti
- La riunione di tutti i diagrammi prodotti può essere facilmente utilizzata come rappresentazione del sistema intero → scheletro del sistema

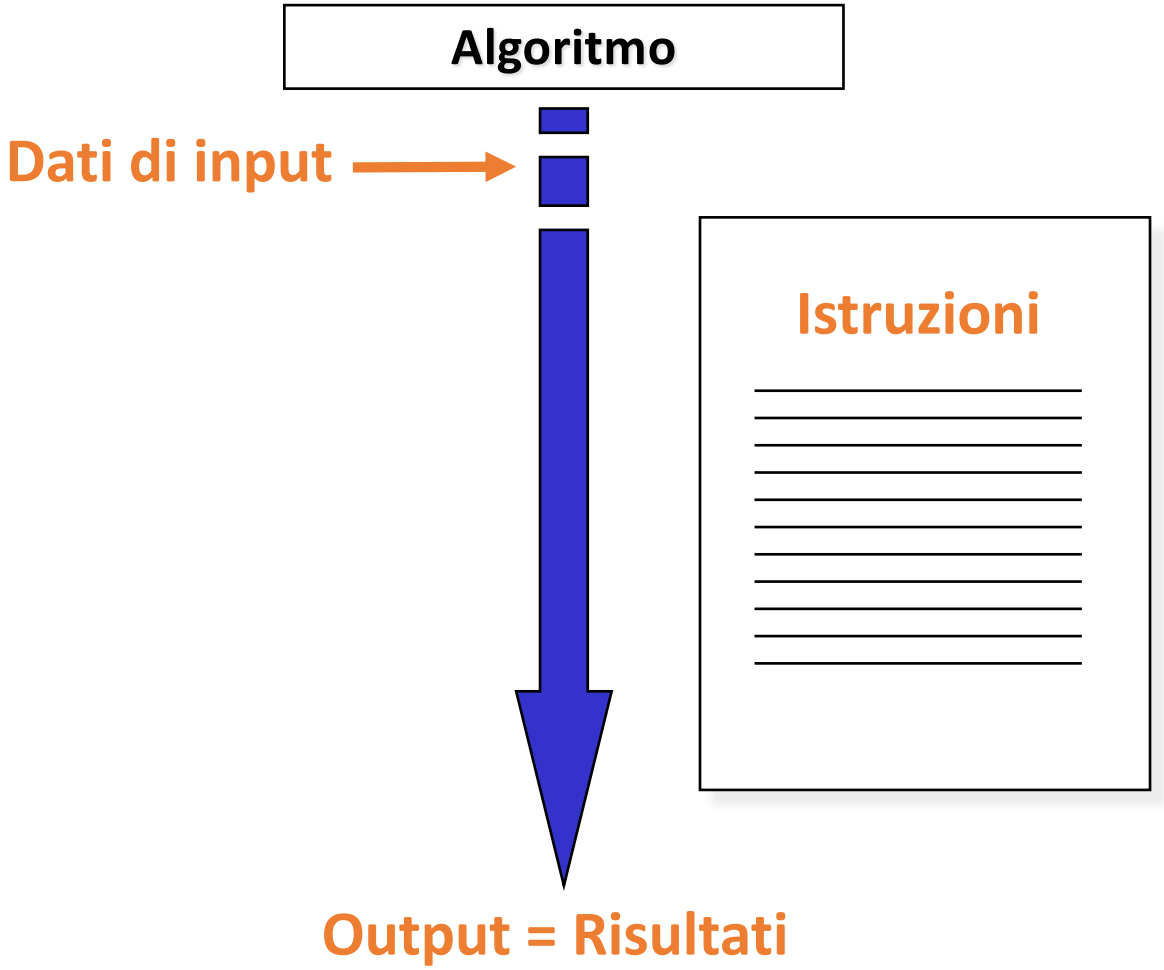
PROGRAMMAZIONE A OGGETTI

SISTEMA

Un sistema è un insieme di
entità interagenti = OGGETTI
in cui ogni componente è caratterizzato da
PROPRIETÀ = ATTRIBUTI
AZIONI = METODI



PROGRAMMAZIONE ALGORITMICA



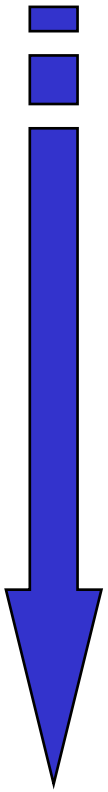
PROGRAMMAZIONE ALGORITMICA (2)

DATO UN VETTORE IN INGRESSO
[2 77 1 935 11 19 773 15 3]

VOGLIO IN USCITA IL VETTORE ORDINATO

Cerco il minimo
Lo metto da parte
Lo elimino dal vettore
Cerco il minimo nel vettore rimanente

VETTORE IN USCITA
[1 2 3 11 15 19 77 773 935]



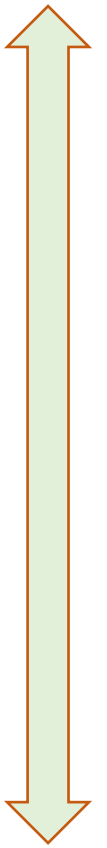
ALGORITMI vs OGGETTI

PROGRAMMAZIONE ALGORITMICA

- Sequenza di azioni
- Basato si DATI e FUNZIONI = PROGRAMMI
- Obiettivo: risolvere un PROBLEMA

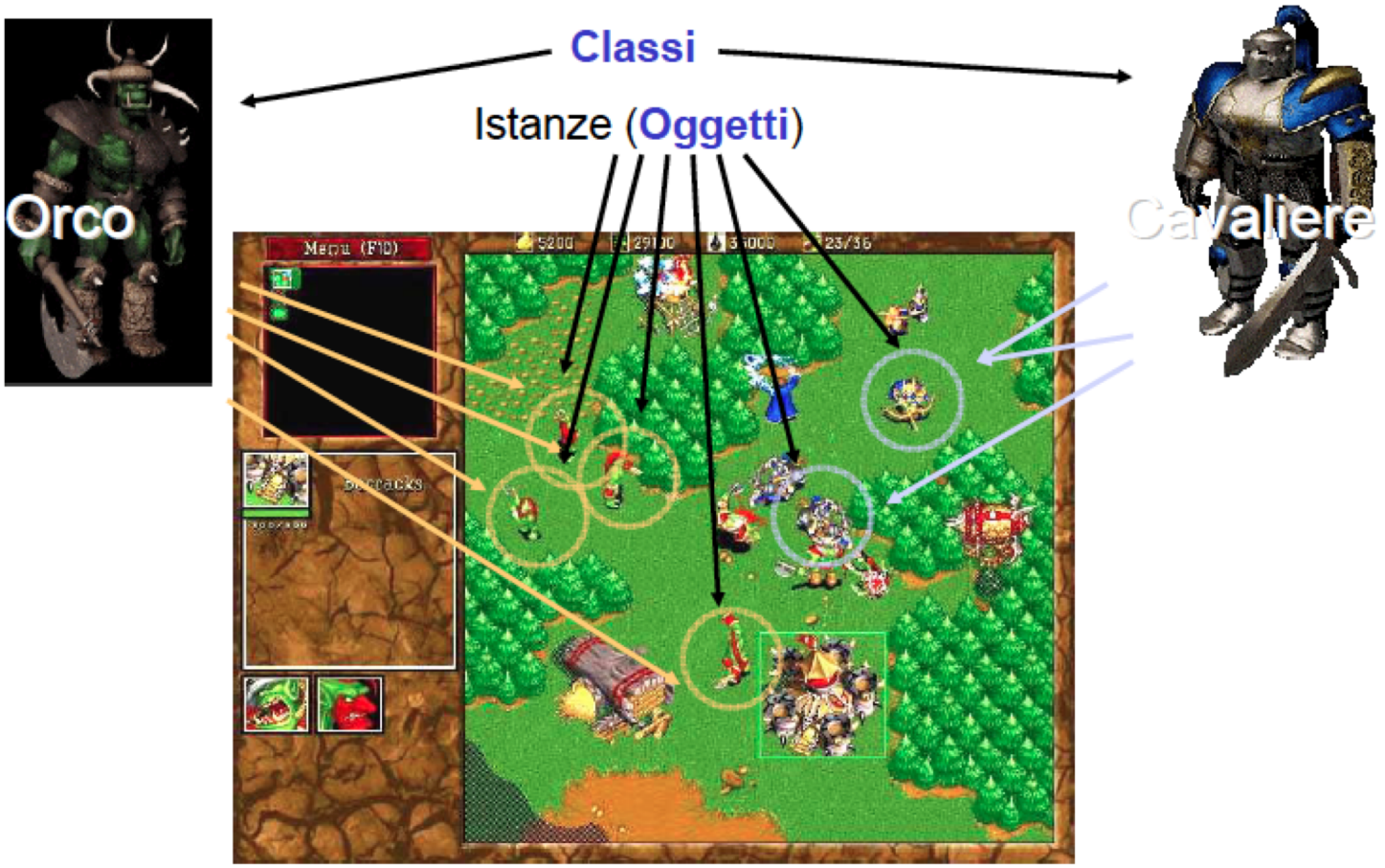
PROGRAMMAZIONE A OGGETTI

- Sistema = Insieme di oggetti
- Basato su OGGETTI fatti da AZIONI e ATTRIBUTI
- Obiettivo: gestire un SISTEMA



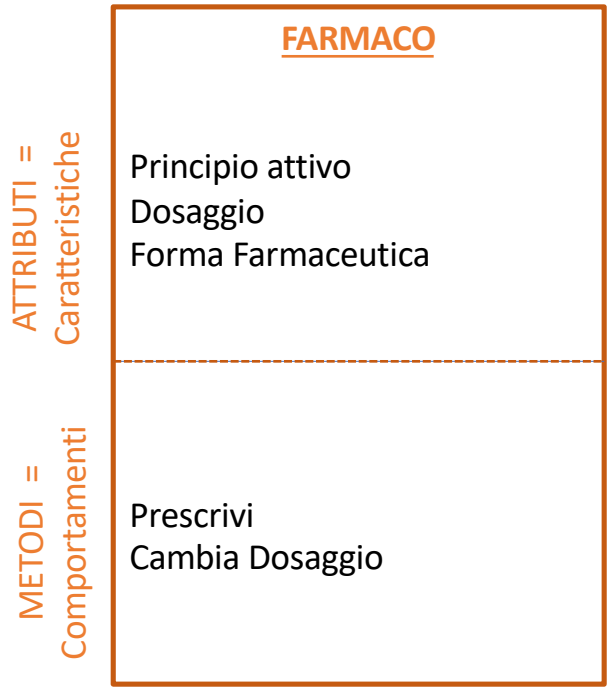
OGGETTI E CLASSI

OGGETTI E CLASSI



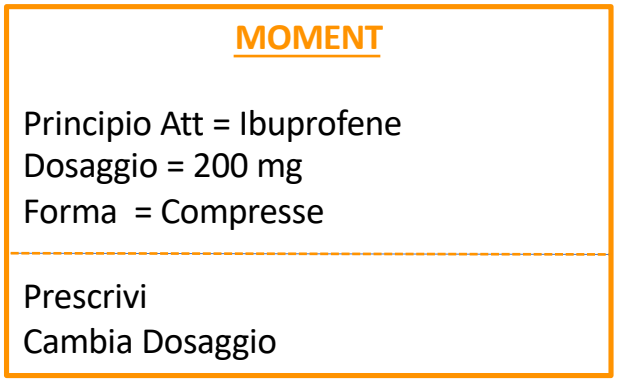
ESEMPIO

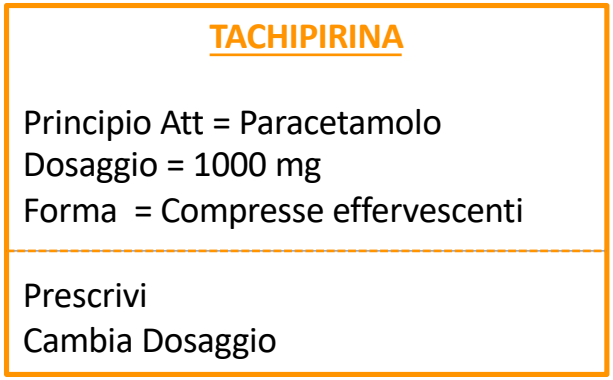
CLASSE



OGGETTO =
ISTANZA DI UNA CLASSE

OGGETTI





ESEMPIO: OSSERVAZIONI

MOMENT

Principio Att = Ibuprofene
Dosaggio = 200 mg
Forma = Compresse

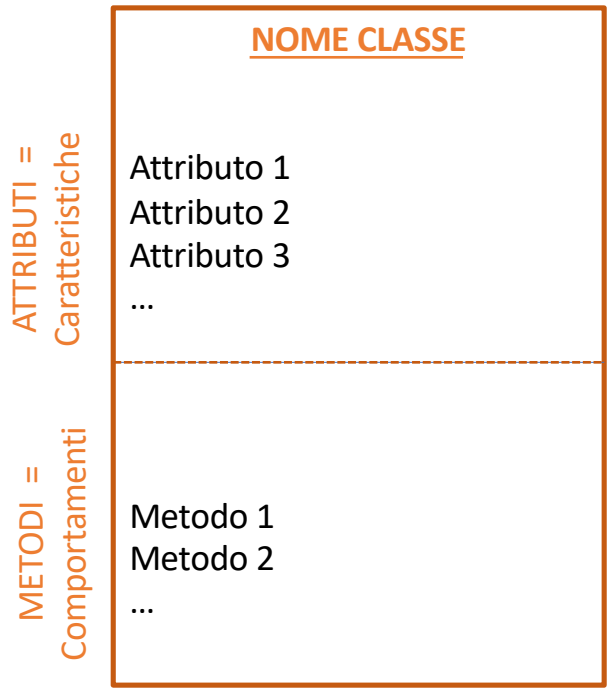
Prescrivi
Cambia Dosaggio

I VALORI DEGLI ATTTRIBUTI SONO SPECIFICI DELL'OGGETTO ISTANZIATO (ogni oggetto ha il suo insieme di valori)

LE AZIONI SONO COMUNI A TUTTE LE ISTANZE DELLA CLASSE

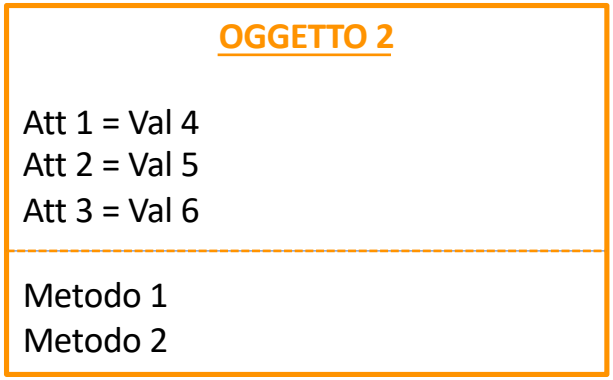
CLASSI E OGGETTI: DEFINIZIONE

CLASSE



OGGETTO =
ISTANZA DI UNA CLASSE

OGGETTI



ATTRIBUTI E METODI

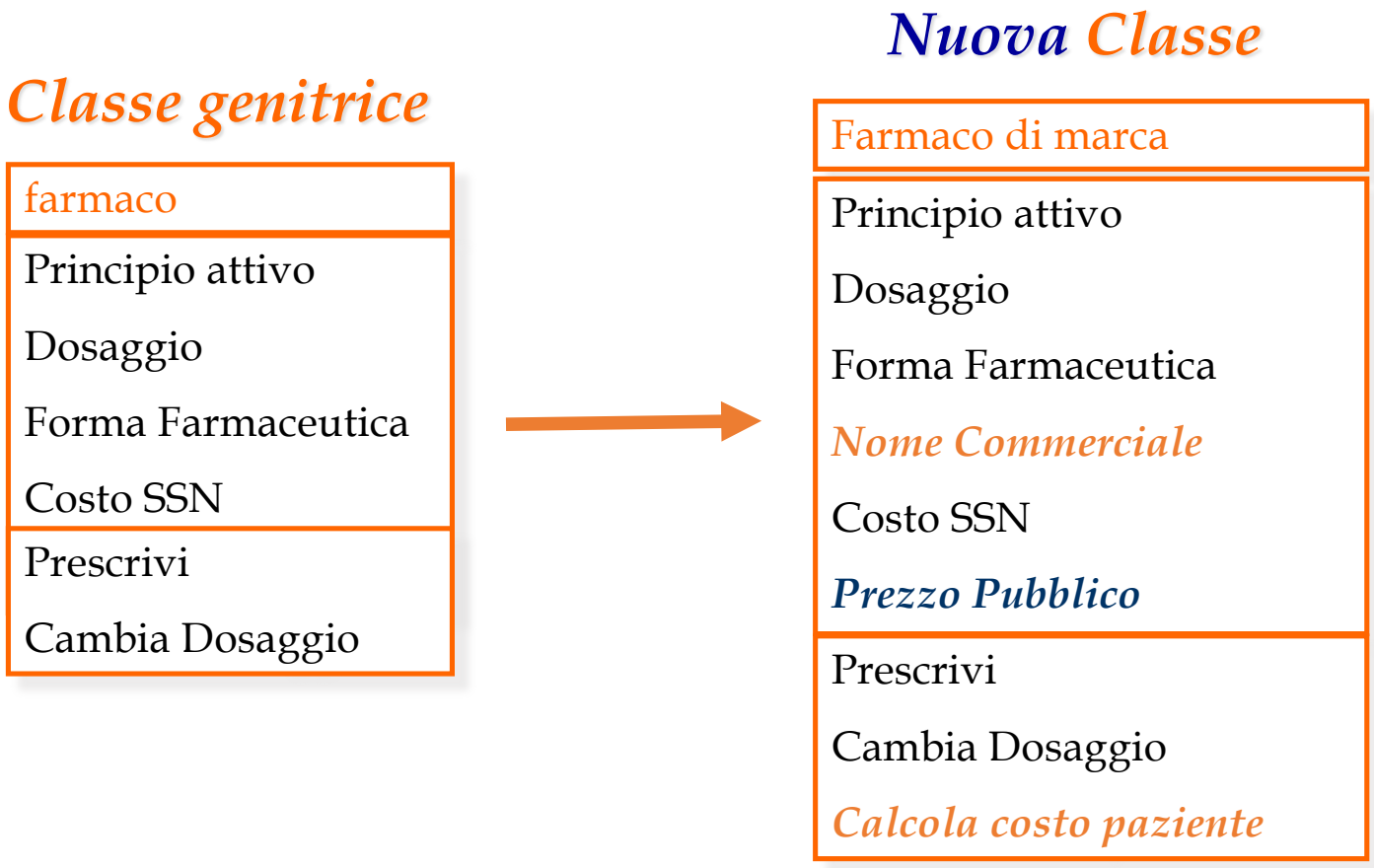
ATTRIBUTI

- Descrivono le proprietà **statiche** dell'oggetto
- Nella **programmazione** gli attributi vengono realizzati attraverso l'uso delle **variabili** utilizzate dall'oggetto per memorizzare i dati

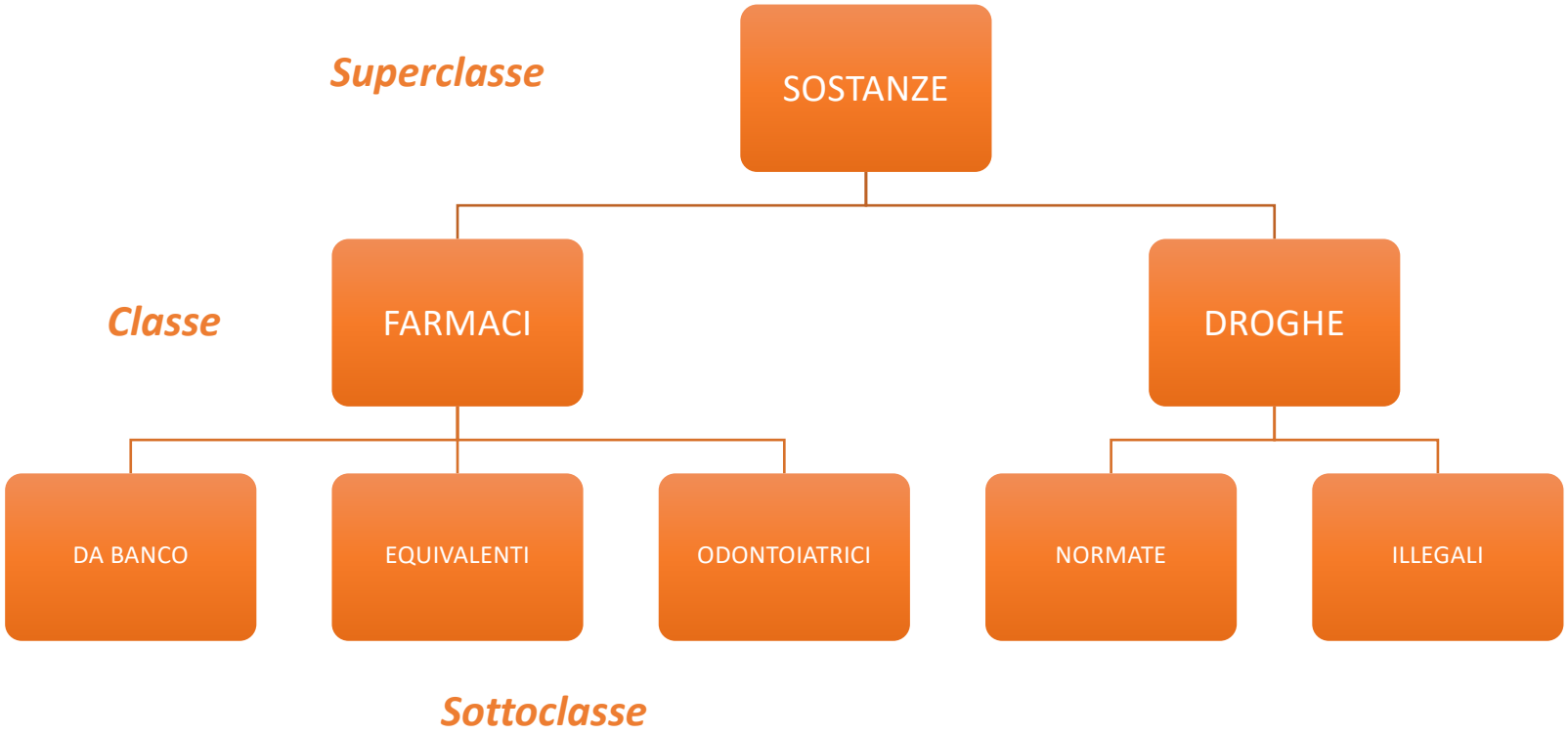
METODI

- Descrivono le proprietà **dinamiche** dell'oggetto
- Nella **programmazione** i metodi vengono realizzati attraverso la scrittura di codice (**procedure** e **funzioni**) che implementano le operazioni dell'oggetto

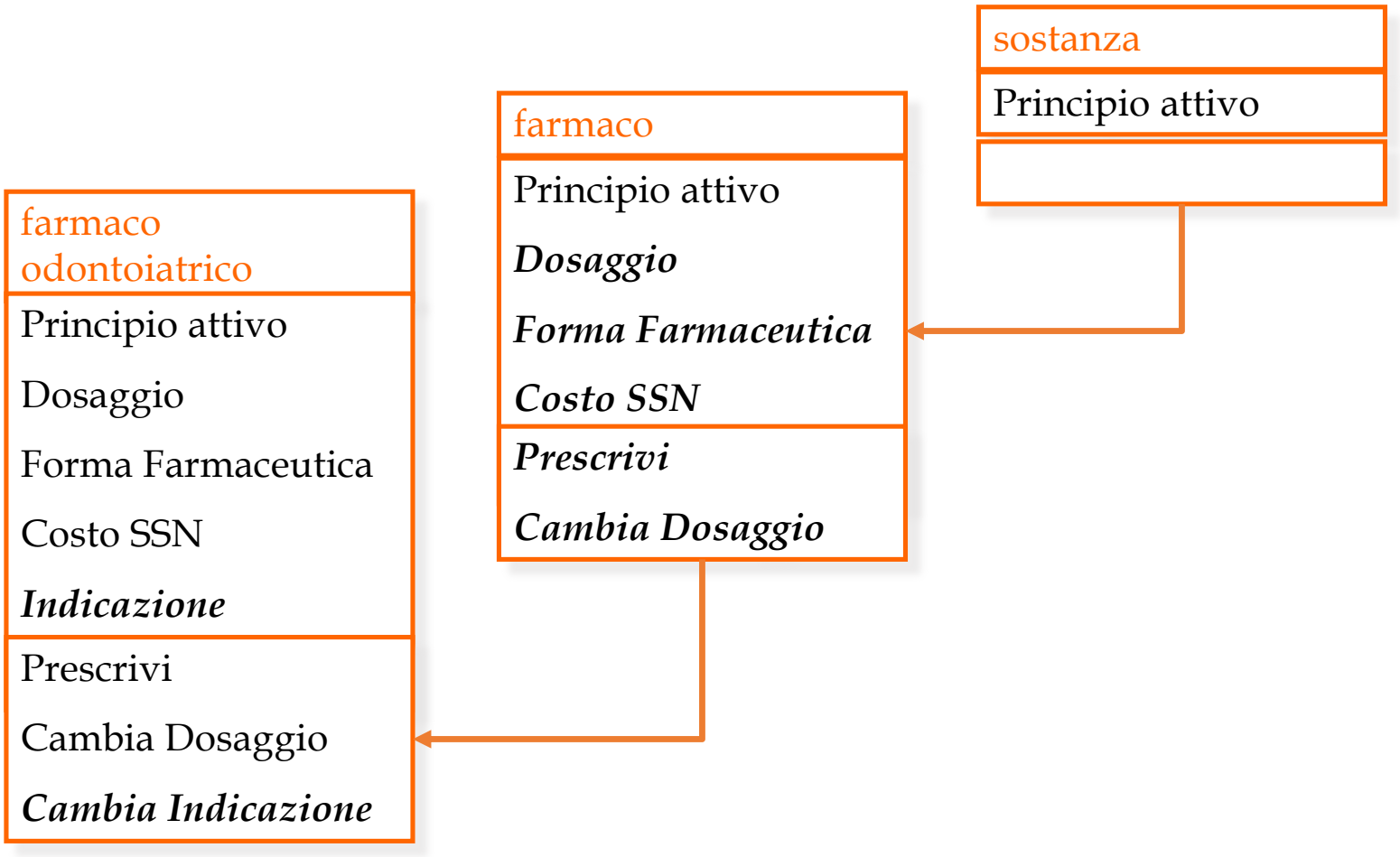
PROPRIETÀ DELLE CLASSI: EREDITARIETÀ



GERARCHIE DI CLASSI

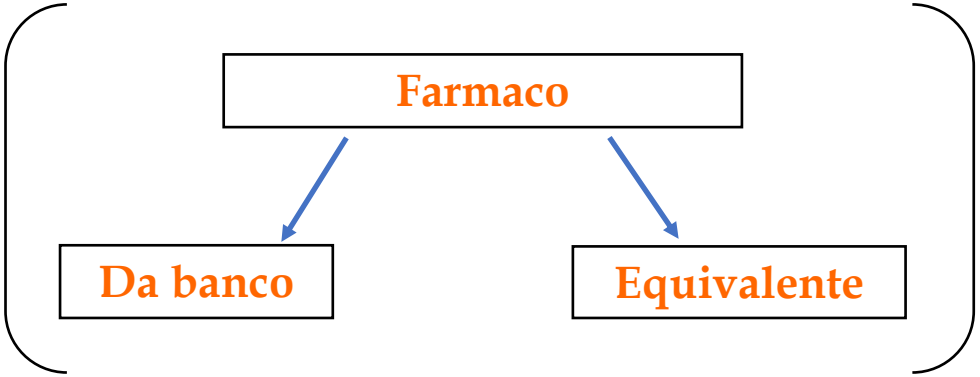


ESEMPIO

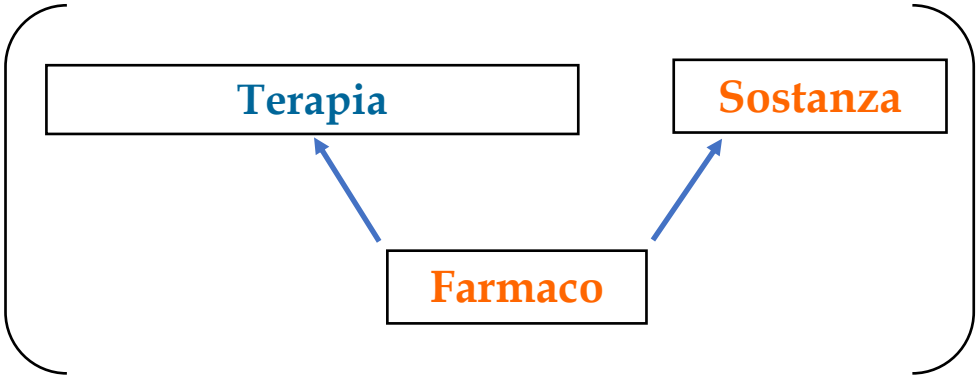

sostanza
Principio attivo
farmaco
Principio attivo
Dosaggio
Forma Farmaceutica
Costo SSN
Prescrivi
Cambia Dosaggio
farmaco odontoiatrico
Principio attivo
Dosaggio
Forma Farmaceutica
Costo SSN
Indicazione
Prescrivi
Cambia Dosaggio
Cambia Indicazione

TIPI DI EREDITARETÀ

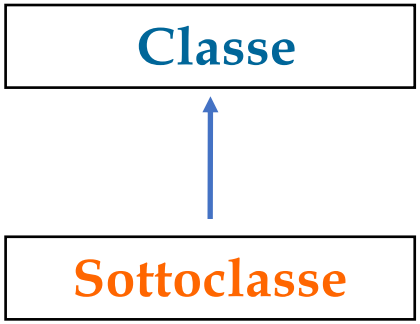
Ereditarietà singola



Ereditarietà multipla



PROPRIETÀ DELL'EREDITARIETÀ



ESTENSIONE

la sottoclasse AGGIUNGE NUOVI METODI/ATTRIBUTI

RIDEFINIZIONE

la sottoclasse RIDEFINISCE I METODI

OVERRIDING = riscrittura del codice del metodo

POLIMORFISMO

OVERRIDING = i ***metodi*** possono ***assumere forme diverse*** (cioè *implementazioni diverse)* all'interno della gerarchia delle classi

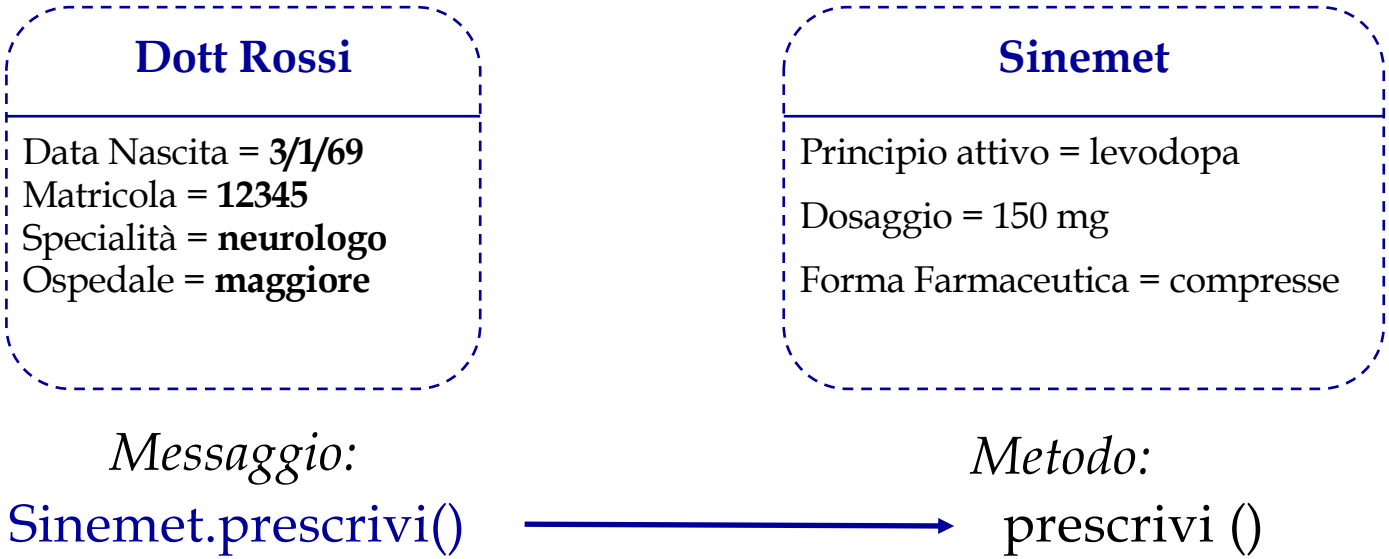
ES. IL METODO "PRESCRIVI" SARÀ IMPLEMENTATO DIVERSAMENTE NEL FARMACO DA BANCO E NEL FARMACO GENERICO

OVERLOADING = i ***metodi*** possono ***assumere forme diverse*** (cioè *implementazioni diverse)* all'interno della stessa classe

ES. IL METODO "CAMBIA DOSAGGIO" Può RICHIEDERE O DI CAMBIARE IL NUMERO DI ASSUNZIONI O DI CAMBIARE IL NUMERO DI DOSI PER ASSUNZIONE

INTERAZIONE TRA OGGETTI: MESSAGGI

- Un programma ad oggetti è caratterizzato dalla presenza di tanti oggetti che **interagiscono fra loro attraverso il meccanismo dello scambio di messaggi**
- I messaggi possono:
  - Richiedere un'informazione su un oggetto
  - Modificare lo stato di un oggetto



DIAGRAMMI UML

VISTE STATICHE

- Use case diagrams
- **Class diagram**
- Component diagram
- Deployment diagram
- Object diagram

RAPPRESENTAZIONE DELLE CLASSI

VISTE DINAMICHE

- **Sequence diagrams**
- Collaboration diagrams
- State chart diagrams
- Activity diagrams

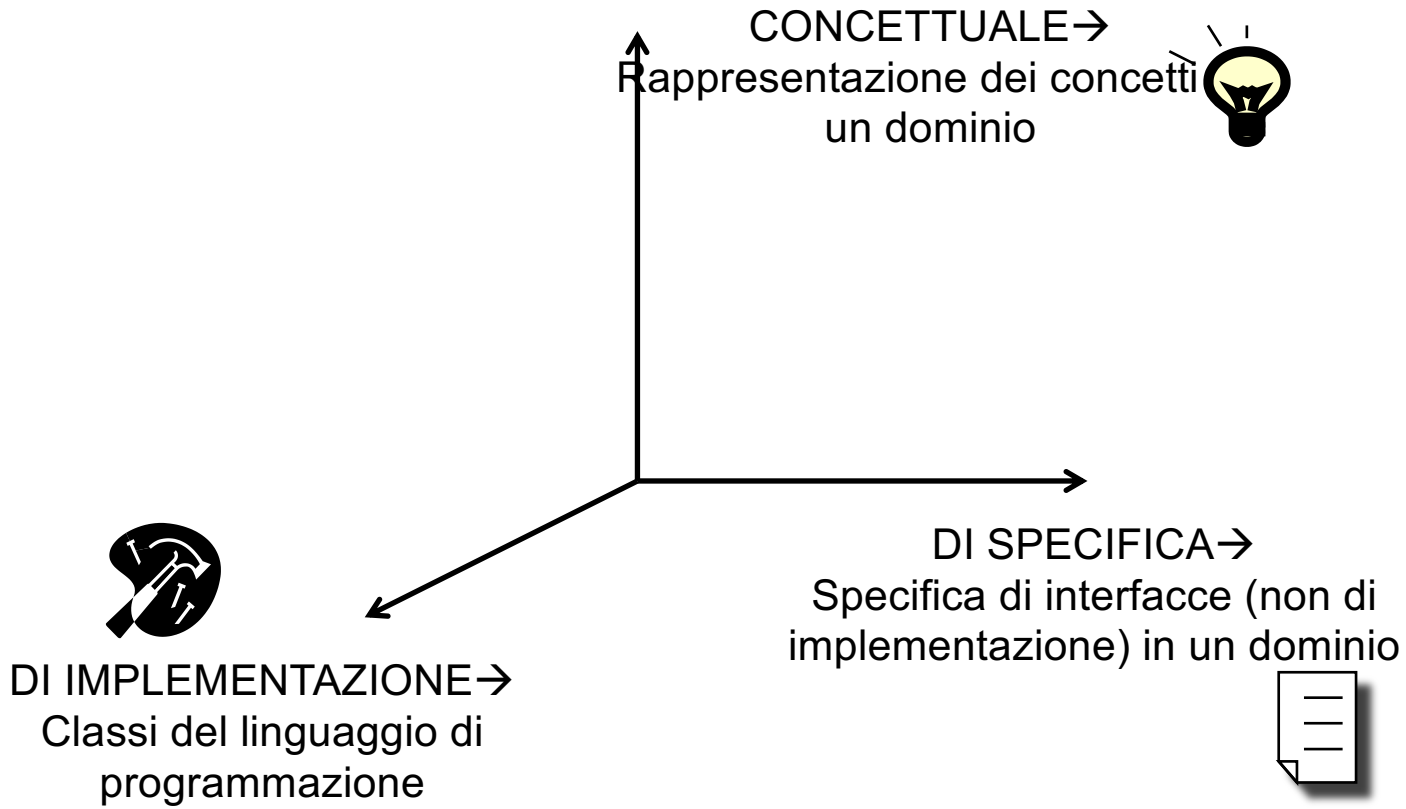
RAPPRESENTAZIONE DELLO SCAMBIO DI MESSAGGI

DIAGRAMMI E OBIETTIVI

Diagramma	Obiettivo
Use case diagram	Estrae i requisiti dagli utenti in modo sinettico La progettazione della fase di costruzione avviene introno a ciascun use case (per ogni iterazione) Base del testing del sistema
Class diagram	Rappresenta la struttura statica dei concetti, tipi e classi Concetti: entità che rappresentano il dominio Tipi: interfacce di componenti software Classi: implementazioni dei componenti software
Activity diagram	Rappresenta il comportamento controllato Può rappresentare un solo oggetto in più casi d'uso o più oggetti in un singolo caso d'uso
Interaction diagram	Rappresenta come diversi oggetti interagiscono/collaborano in un singolo use case
Deployment diagram	Rappresenta le componenti fisiche su nodi hardare

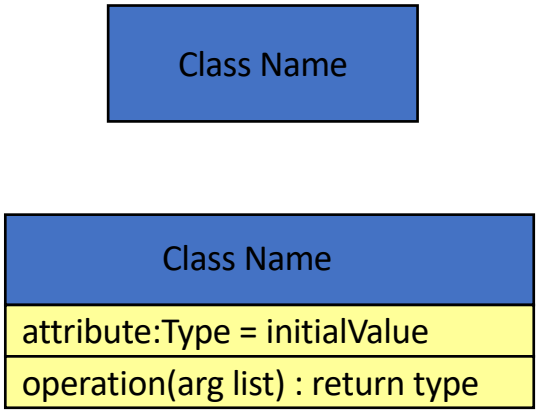
CLASS DIAGRAMS

- Le classi rappresentano OGGETTI e CATEGORIE con significato diverso



CLASS DIAGRAM: NOTAZIONE GRAFICA

- Ogni classe ha un'intestazione, degli attributi e delle operazioni
- Una classe viene rappresentata graficamente da un rettangolo contenente il nome della classe e, opzionalmente, l'elenco degli attributi e delle operazioni.



ATTRIBUTI

<visibilita‘> <nome> <molteplicita‘> : <tipo> = <val-iniziale>

- Nome → obbligatorio
- Tipo:
  - tipi fondamentali predefiniti dall’UML (corrispondenti a quelli usati comunemente nei linguaggi di programmazione)
  - tipo definito in un linguaggio di programmazione
  - classe definita (in UML) dallo sviluppatore;
- Visibilità: privata, protetta, pubblica, package; quest’ultimo livello di visibilità significa che l’attributo è visibile da tutte le classi appartenenti allo stesso package
  - + pubblica
  - # protetta
  - ~ package
  - - privata
- Molteplicità: ´ındica se l’attributo può essere replicato, cioè avere più valori (array). Si ındica con un numero o un intervallo numerico fra parentesi quadre
  - [3] tre valori
  - [1..4] da uno a quattro valori
  - [1..*] uno o pi´u valori
  - [0..1] zero o un valore (attributo opzionale)
- Valore iniziale: valore assegnato all’attributo quando si crea l’oggetto

OPERAZIONI

<visibilita‘> <nome> (<lista-parametri>) : <tipo>

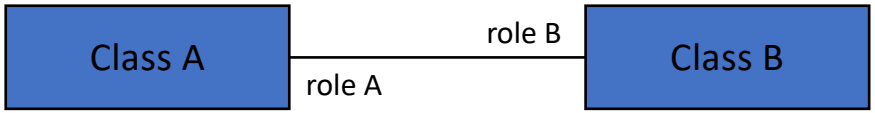
Per ogni parametro:

<direzione> <nome> : <tipo> = <val-default>

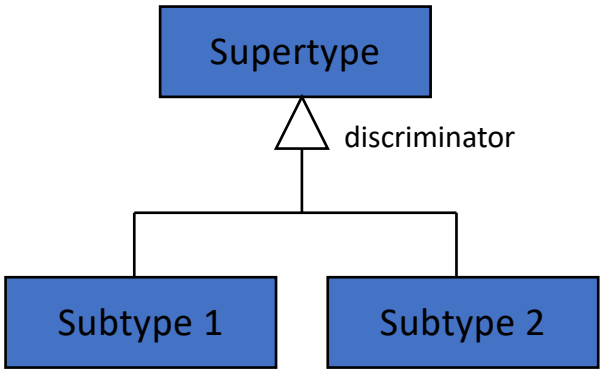
- PARAMETRI
  - Direzione: ingresso (in), uscita (out), ingresso e uscita (inout);
  - Tipo;
  - Valore default: valore passato al metodo che implementa la funzione.
  - Solo il nome del parametro è obbligatorio
  - Parametri separati da virgole
- OPERAZIONI
  - Visibilità (come attributi)
  - Tipo del valore restituito

CLASS DIAGRAM: NOTAZIONE GRAFICA

Association

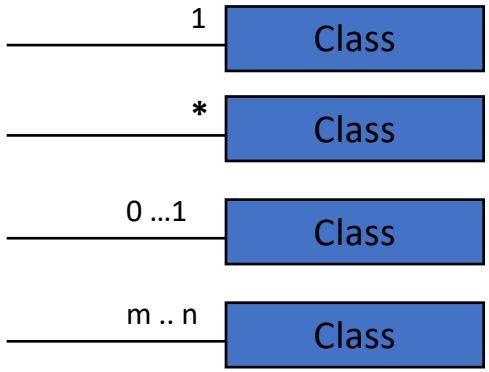


Generalization

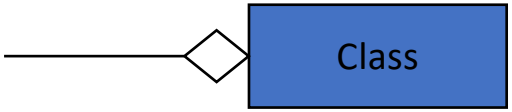


CLASS DIAGRAM: NOTAZIONE GRAFICA

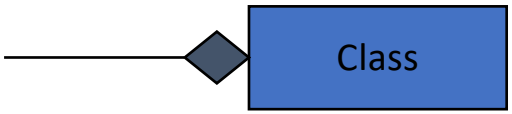
Multiplicities



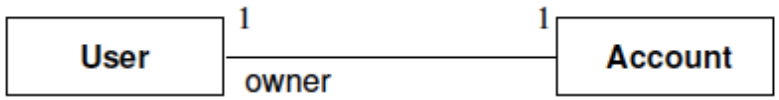
Aggregation



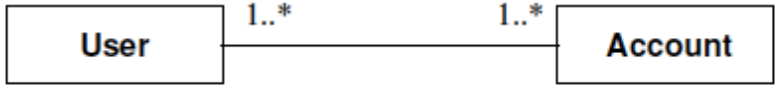
Composition



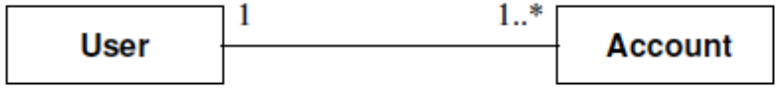
ASSOCIAZIONI E MOLTEPLICITÀ



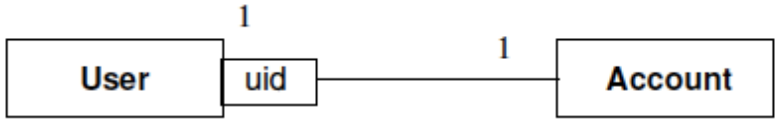
uno a uno



molti a molti



uno a molti



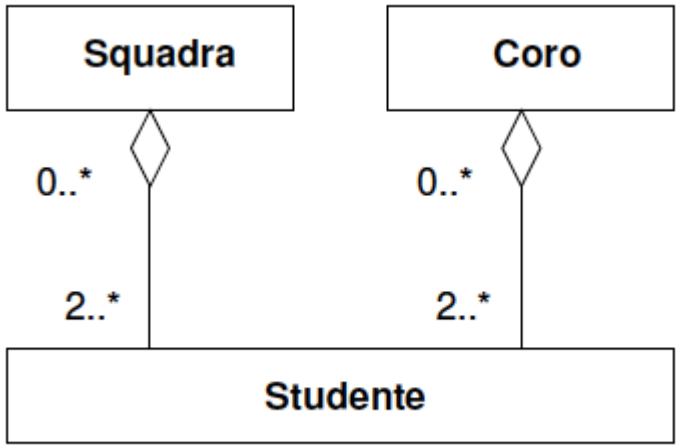
uno a molti con qualificatore

Uid è il qualificatore dell'account che rende la relazione 1..1 → uno user può avere più account ma se considero un certo account con un certo uid, avrà uno ed un solo utente associato

AGGREGAZIONI

AGGREGAZIONE = associazione che lega un'entità complessa (aggregato) alle proprie parti componenti.

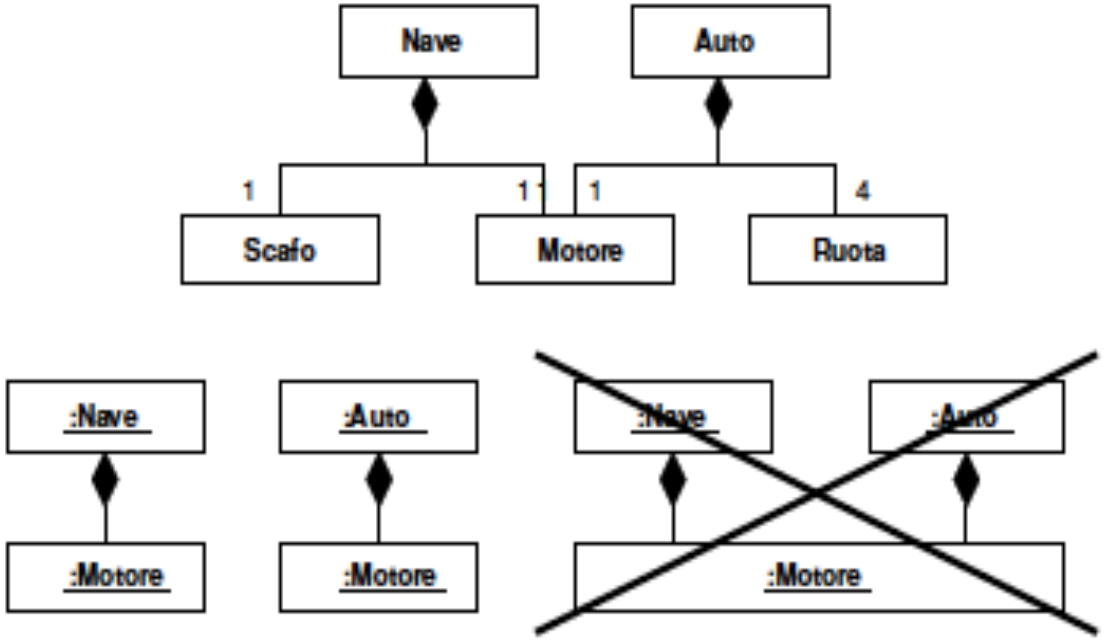
I clienti non fanno parte della loro banca, mentre i giocatori fanno parte della squadra



Uno studente può far parte sia del coro (che è formato da studenti) che della squadra (che è formata da studenti)

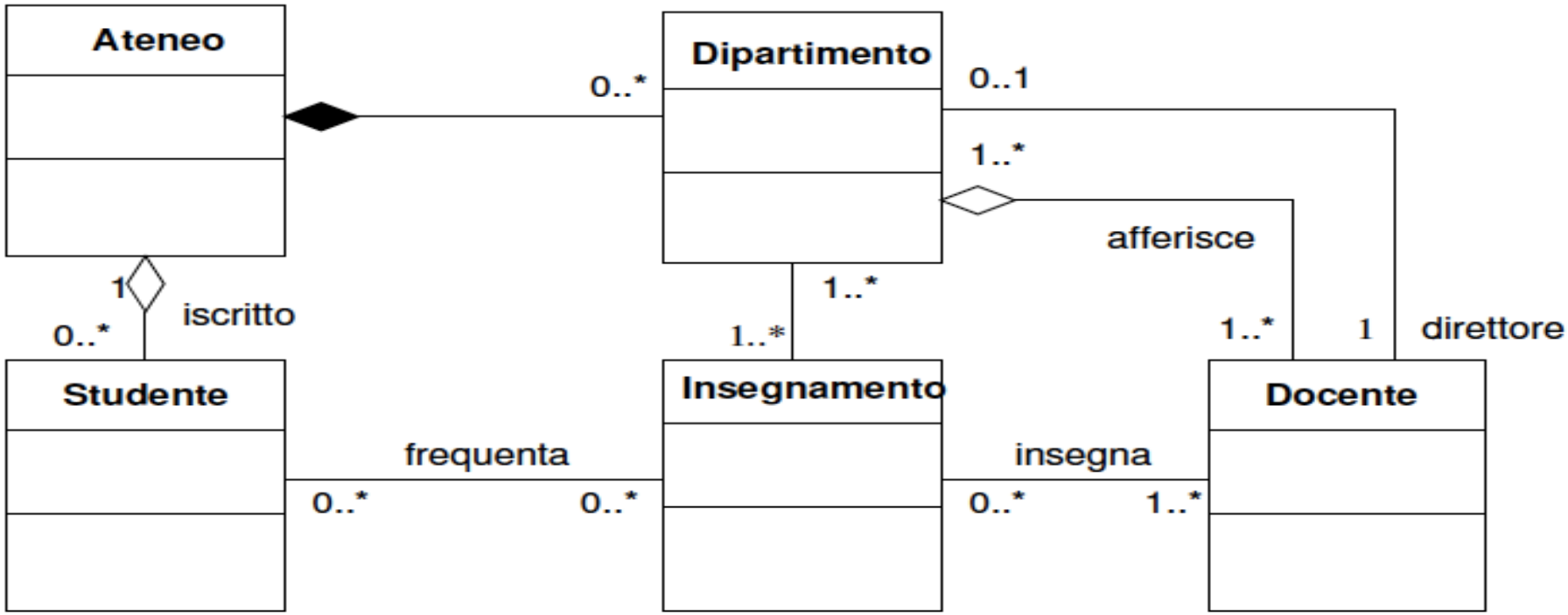
COMPOSIZIONE

COMPOSIZIONE = subordinazione strutturale rigida in cui l'aggregato ha il completo e unico controllo delle parti componenti.

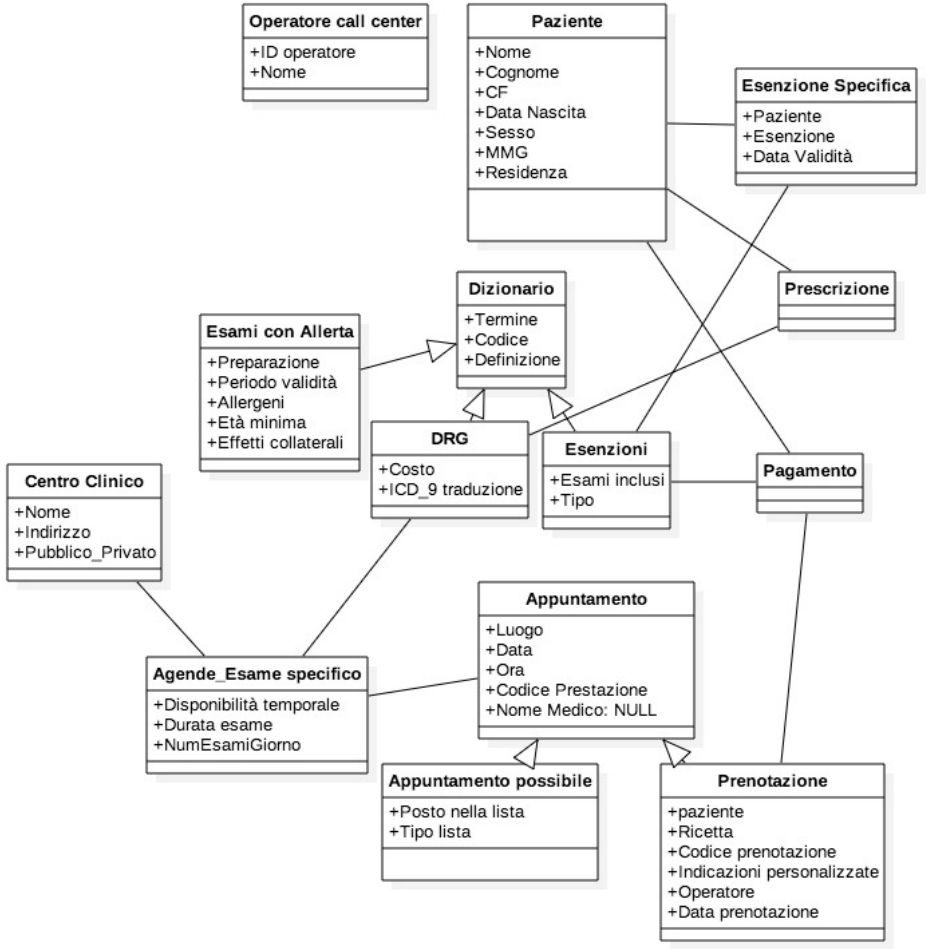


- Una classe può appartenere come componente a più di una composizione
- Un'istanza di una classe componente può appartenere ad una sola istanza di una classe composta

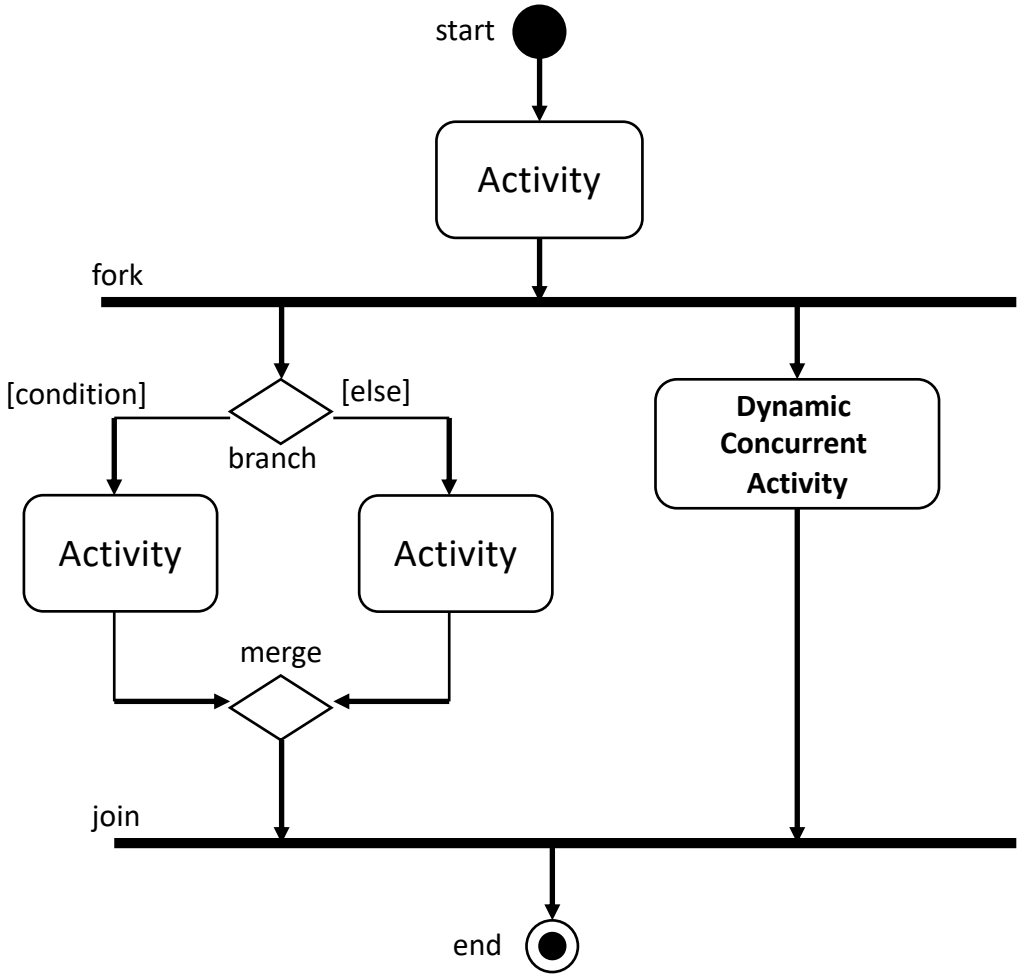
AGGREGAZIONE E COMPOSIZIONE



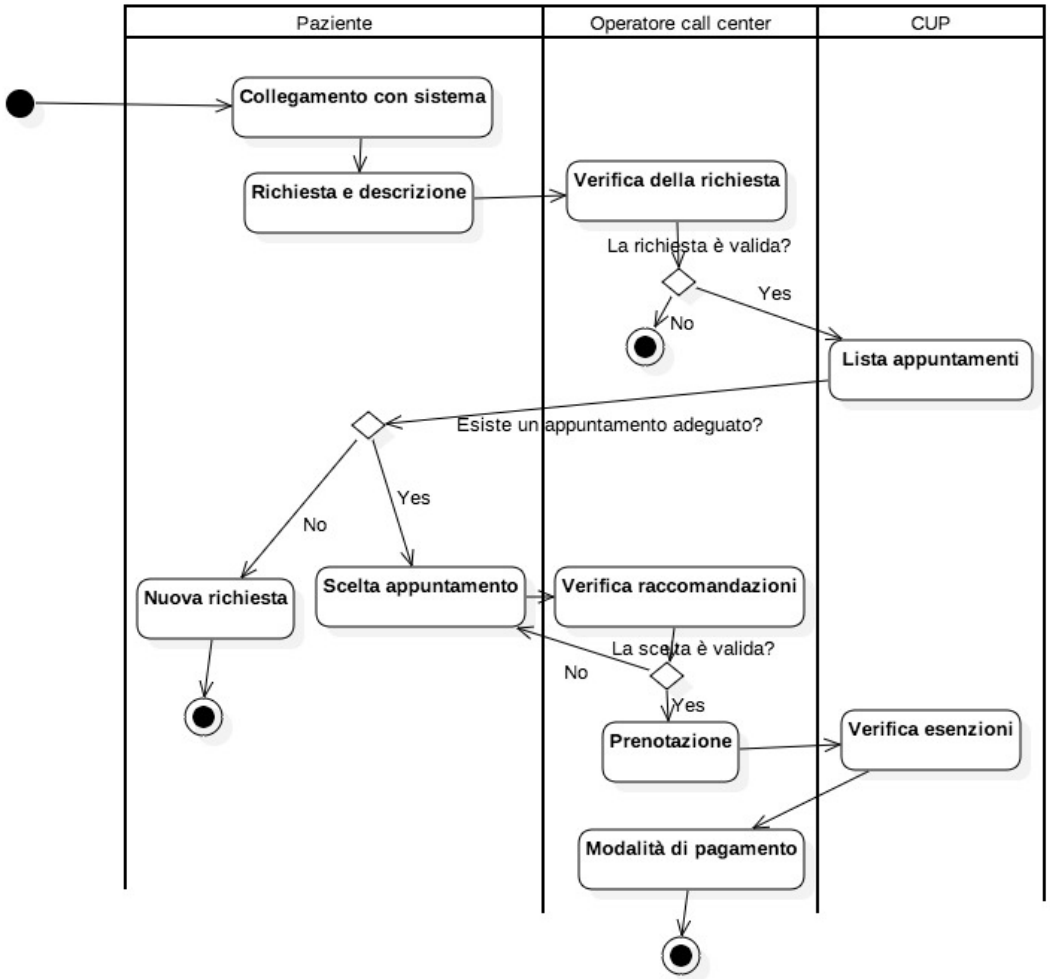
CLASS DIAGRAM ESEMPIO



ACTIVITY DIAGRAM: NOTAZIONE GRAFICA



ACTIVITY DIAGRAM ESEMPIO



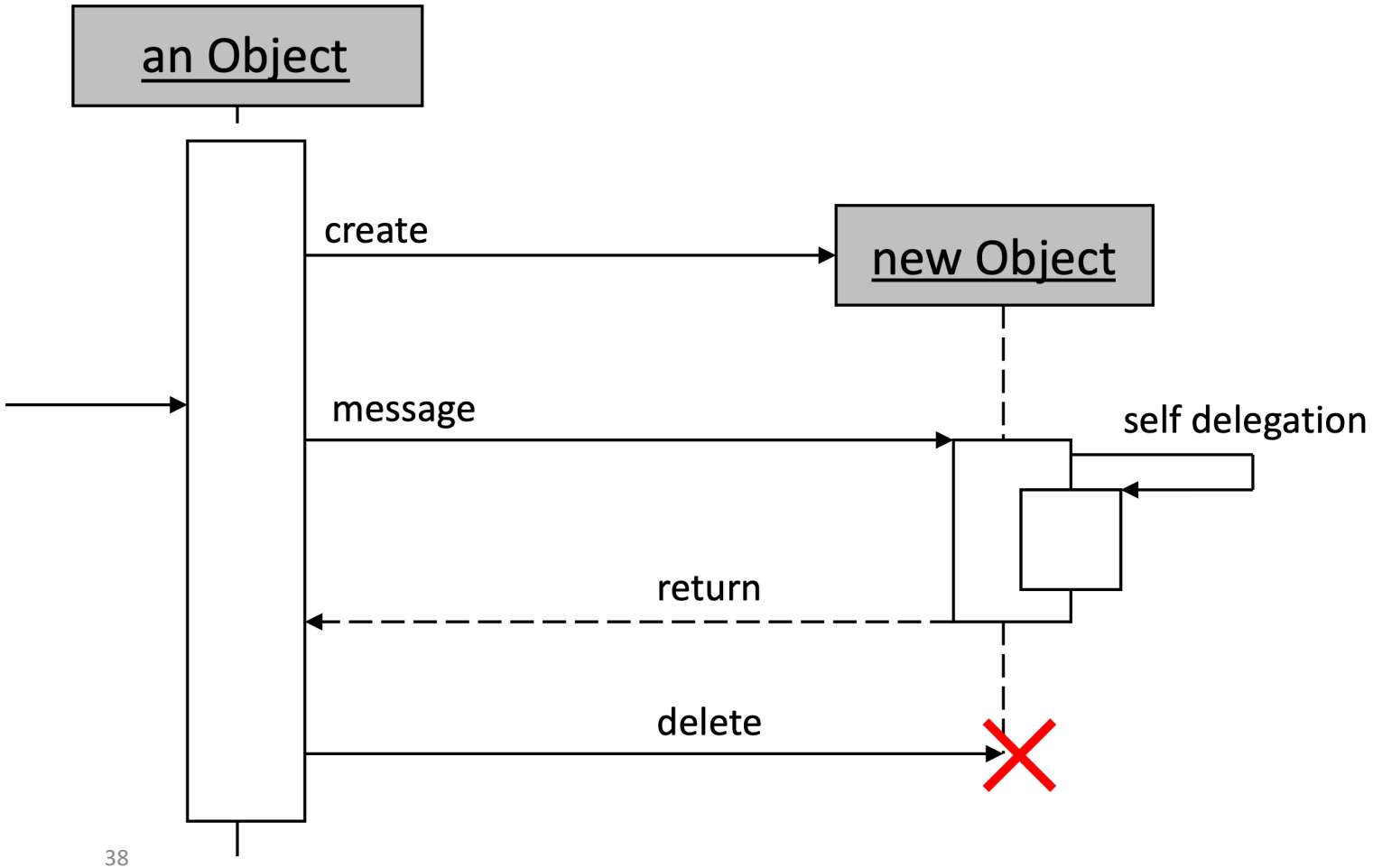
SWIMLANE = definizione delle attività che ricadono sotto uno specifico attore

SEQUENCE DIAGRAMS

- Diagrammi di interazione
- Rappresentano i messaggi che gli oggetti si scambiano in determinate situazioni

- Oggetti → rettangoli posti all'inizio di linee verticali tratteggiate
- Linea della vita dell'oggetto →
  - Linea verticale tratteggiata
  - Rettangolo verticale sulla linea della vita: momento in cui l'oggetto è attivo e scambia messaggi
- Messaggi →
  - Frecce tra le linee della vita di due oggetti
  - L'ordine va dall'alto verso il basso
  - Selfcall: messaggio che l'oggetto scambia con se stesso

SEQUENCE DIAGRAM: NOTAZIONE GRAFICA



SEQUENCE DIAGRAM: ESEMPIO