

---

---

# **PROGRAMMAZIONE II**

---

---

**LAUREA TRIENNALE IN SCIENZE INFORMATICHE**

Magliani Andrea  
Perego Luca

Università degli studi di Milano-Bicocca

A.A. 2022/2023

# INDICE

<b>1 Classi, Oggetti &amp; Istanze.....</b>	<b>3</b>
1.1 Definizioni.....	3
1.2 Astrazione.....	3
<b>2 Astrazioni Strutturali.....</b>	<b>4</b>
2.1 Classificazione & Istanziamento.....	4
2.2 Generalizzazione.....	4
2.3 Associazione.....	4
2.4 Aggregazione.....	4
<b>3 Formalismi di rappresentazione.....</b>	<b>5</b>
3.1 ER [Entity-Relationship].....	5
3.2 UML [Unified Modeling Language].....	5
<b>4 UML: diagramma delle classi.....</b>	<b>6</b>
4.1 Classi.....	6
4.2 Oggetti.....	6
<b>5 Associazioni.....</b>	<b>7</b>
5.1 Definizione.....	7
5.2 Nome.....	7
5.3 Ruolo.....	7
5.4 Cardinalità.....	7
5.5 Associazioni multiple.....	8
5.6 Associazioni cappio.....	8
5.7 Navigabilità delle associazioni.....	8
<b>6 Aggregazioni.....</b>	<b>9</b>
6.1 Definizione.....	9
<b>7 Composizioni.....</b>	<b>10</b>
7.1 Definizione.....	10
<b>8 Generalizzazione.....</b>	<b>11</b>
8.1 Definizione.....	11
<b>9 Object Oriented Programming.....</b>	<b>12</b>
9.1 OOP e Oggetti.....	12
9.2 Caratteristiche di un oggetto.....	12
9.3 Classi.....	13
<b>10 Gestione della memoria.....</b>	<b>14</b>
10.1 Gestione di memoria.....	14
10.2 Dot notation.....	14
10.3 Debug.....	14
10.4 Array di oggetti.....	15

# Classi, Oggetti & Istanze

## 1.1 Definizioni

Classe: Raccolta di proprietà comuni ad un insieme di istanze;

Istanza: Concretizzazione degli attributi e i comportamenti descritti da una classe, è sinonimo di oggetto;

## 1.2 Astrazione

L'astrazione è il processo concettuale tramite il quale si definisce un concetto generico partendo da una sua istanza specifica ed estraendo solamente gli aspetti generali.

# Astrazioni Strutturali

## 2.1 Classificazione & Istanziamento

Si dice **istanziamento** l'azione tramite la quale si crea un oggetto rappresentato astrattamente da una classe.

Si dice **classificazione** l'azione tramite la quale si crea una classe rappresentando le caratteristiche e comportamenti di una particolare istanza.

La classificazione si indica con la keyword **instance\_of**.

## 2.2 Generalizzazione

La generalizzazione lega una **superclasse** (classe padre) ad un o più **sottoclassi** (classi figlie). Ogni sottoclasse possiede tutte le caratteristiche della sua superclasse.

La generalizzazione si indica con la keyword **is\_a**.

## 2.3 Associazione

L'associazione definisce una connessione logica tra 2 oggetti, senza definire un'interpretazione.

La generalizzazione si indica con la keyword **has\_a**.

## 2.4 Aggregazione

L'aggregazione è un caso particolare dell'associazione: lega un classe "aggregato" con un insieme di classi "parti".  
Ogni aggregato è costituito da parti.

L'aggregazione si indica con la keyword **part\_of**.

# Formalismi di rappresentazione

## 3.1 ER [Entity-Relationship]

1. Entità
2. Relazioni

Il formalismo ER è nato nell'area "Basi di dati"

## 3.2 UML [Unified Modeling Language]

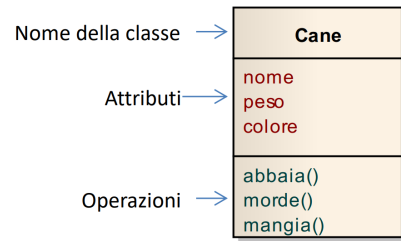
1. Classi
2. Proprietà
3. Relazioni
4. Comportamenti
5. ...

Il formalismo UML è nato nell'area "Programmazione"

# UML: diagramma delle classi

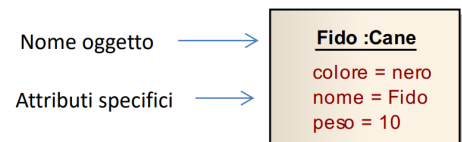
## 4.1 Classi

Identificazione delle classi e delle relazioni fra le stesse. Attività fondamentale in fase di **analisi** e di **progetto**.



## 4.2 Oggetti

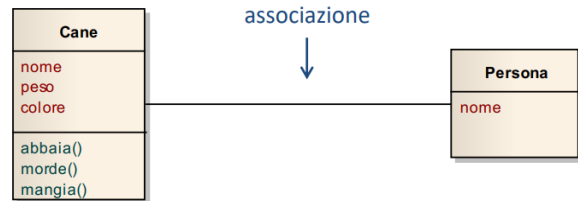
Identifica gli oggetti del sistema e i loro stati durante la loro vita.



# Associazioni

## 5.1 Definizione

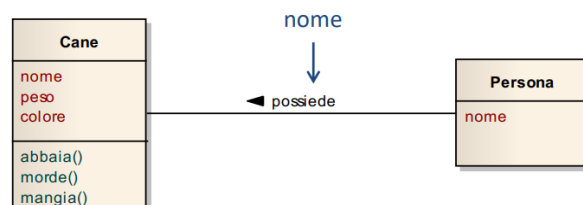
Un'associazione è una linea che collega le classi coinvolte.



## 5.2 Nome

Esprime il significato dell'associazione.

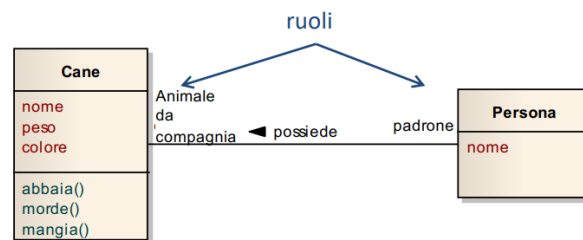
1. Spesso è un verbo
2. Opzionale [ma consigliato]



## 5.3 Ruolo

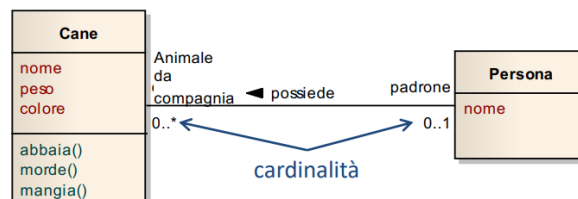
Esprime il ruolo giocato dal partner.

1. Spesso è un sostantivo o un aggettivo.
2. Opzionale [ma consigliato]



## 5.4 Cardinalità

Esprime quante istanze della classe possono essere associate all'altra classe.



## 5.5 Associazioni multiple

L'associazione multipla esplicita più associazioni tra una coppia di classi per arricchirne la descrizione.

Le associazioni tra una coppia di classi possono variare in:

1. Ruolo
2. Direzione
3. Nome

Teoricamente, ogni istanza è rappresentata da un'unica associazione tra quelle disponibili della coppia di classi, ma non è obbligatorio seguire questa norma.



## 5.6 Associazioni cappio

Esistono associazioni cappio, ovvero classi poste in relazione a loro stesse.

## 5.7 Navigabilità delle associazioni

Le associazioni hanno una proprietà detta navigabilità, se hanno una freccia che esplicita la modalità in cui viene realizzata.

Una classe con un'associazione in ingresso non ha accesso ai dati della classe a cui è collegata.

L'assenza di frecce implica un'associazione con **navigabilità bidirezionale**.



# Aggregazioni

## 6.1 Definizione

Le aggregazioni sono relazioni **poco forti**:

1. Si rappresenta un diamante vuoto.
2. Le parti possono esistere indipendentemente dal tutto.  
*a. Ex. La stampante esiste anche senza computer*
3. Il tutto può in alcuni casi esistere indipendentemente dalle parti, ma in altri casi no.  
*a. Ex. Il computer può esistere anche senza stampante*
4. Il tutto è in qualche modo incompleto se mancano alcune delle sue parti.  
*a. Ex. Azienda senza lavoratori*
5. E' possibile che più aggregati condividano una stessa parte.  
*a. Ex. Stampante condivisa*

# Composizioni

## 7.1 Definizione

Le composizioni sono relazioni **molto forti**:

1. Si rappresenta un diamante pieno.
2. Ogni parte può appartenere ad un solo tutto per volta.  
*a. Ex. Il motore appartiene ad una sola auto*
3. Il tutto è l'unico responsabile di tutte le sue parti: questo vuol dire che è responsabile della loro creazione e distruzione.
4. Il tutto può rilasciare una sua parte, a patto che un altro oggetto si prenda la relativa responsabilità.  
*a. Ex. Il motore viene montato su un'altra auto*
5. Se il tutto viene distrutto, devo distruggere tutte le sue parti o cederne la responsabilità a qualche altro oggetto.  
*a. Ex. L'auto si rottama tutta, oppure si cedono le parti ad altre auto*

# Generalizzazione

## 8.1 Definizione

La generalizzazione è rappresentata da triangolo vuoti (puntati verso la superclasse).

La superclasse è detta anche **classe padre**.

Le classi a lei collegate sono anche dette **classi figlie**.

# Object Oriented Programming

## 9.1 OOP e Oggetti

Java è un linguaggio di programmazione ad oggetti.

Un oggetto è un contenitore che incapsula dati ed algoritmi.

I dati contenuti sono gli elementi che determinano lo stato dell'oggetto, mentre gli algoritmi contenuti sono incapsulati nei metodi della classe generatrice dell'oggetto

Tramite un oggetto è possibile rappresentare un'entità reale o astratta tramite un processo di astrazione.

## 9.2 Caratteristiche di un oggetto

### Stato:

Una delle condizioni in cui può trovarsi l'oggetto.

Lo stato di un oggetto è definito dalle sue variabili e attributi.

Gli attributi di un oggetto possono avere qualsiasi tipo, primitivo o non primitivo.

### Comportamento:

Determina la risposta di un oggetto a una determinata richiesta.

Il comportamento di un oggetto è il modo in cui risponde alle richieste esterne.

Gli oggetti comunicano scambiandosi chiamate attraverso le loro interfacce.

Le chiamate attivano i metodi, i quali infine determinano il comportamento dell'oggetto stesso.

## Identità:

Gli oggetti dispongono di un'identità univoca: due oggetti che si trovano nello stesso stato e hanno lo stesso comportamento sono comunque 2 entità distinte.

Ogni oggetto dispone di un Object Identifier [OID], un codice univoco e invariante nel tempo.

## **9.3 Classi**

Tutti gli oggetti che condividono proprietà e comportamenti possono essere classificati assieme.

Una classe rappresenta una e una sola astrazione dalla quale è possibile creare oggetti.

La classe è una struttura che definisce le caratteristiche delle sue istanze.

Una classe specifica l'interfaccia che ogni sua istanza offre agli altri elementi con cui interagisce, essa:

- Definisce comandi: nome, return e parametri formali
- Implementa: attributi, metodi interni e operazioni di questi

## Information Hiding

Le classi dispongono anche del principio detto information hiding: questa caratteristica permette di impostare la visibilità dei suoi elementi, rendendoli invocabili dall'esterno o invisibili.

Questa tecnica permette modularità nella programmazione e inoltre lascia cambiare l'implementazione di una classe senza farlo notare agli elementi che interagiscono esternamente.

# Gestione della memoria

## 10.1 Gestione di memoria

Le dichiarazioni delle variabili vengono salvate nella **memoria stack** e inizializzate con valore undefined.

Le assegnazioni vengono salvate nella **memoria heap**.

Le dichiarazioni di un oggetto viene gestita come le variabili, ma l'inizializzazione di questo avviene con i valori standard di ogni tipo (int = 0, char = "", ..),

In Java l'operatore di confronto permette di verificare l'**identità** di due elementi:

1. Nel caso di variabili primitive, l'identità è uguale se entrambe contengono lo stesso valore.
2. Nel caso degli oggetti l'identità confronta l'**area di memoria** legata alla dichiarazione di questi.

## 10.2 Dot notation

Per accedere ad un attributo di un oggetto, si utilizza la scrittura: oggetto.attributo.

Questo metodo è detto **dot notation** e permette di confrontare gli attributi degli oggetti invece dei loro indirizzi di memoria.

## 10.3 Debug

Il sistema di debug permette di controllare il codice senza scrivere output al suo interno.

Tramite gli IDE è possibile posizionare **breakpoint**, ovvero punti in cui il sistema di debug si ferma permettendo di controllare il codice un'istruzione alla volta.

Il debug permette di visualizzare identificativo e attributi di un oggetto, aggiornando il loro stato in tempo reale.

Tramite gli identificativi è possibile capire se due istanze fanno riferimento a istanze diverse.

## 10.4 Array di oggetti

Gli array di oggetti contengono le **reference** agli oggetti in questione, in quanto vengono memorizzati gli indirizzi di memoria degli oggetti.

Gli array di oggetti si dichiarano nel seguente modo:

```
Oggetto [] nomeArray;
```

Questi array sono di fatto array di reference.

I valori di inizializzazione di ogni cella è null (in quanto è il valore di inizializzazione degli oggetti).