Modellazione e progettazione con UML

Eduard Roccatello 3D GIS Specialist
<eduard.roccatello@3dgis.it>
www.roccatello.it

Object Oriented Analysis and Design

- Consente di modellare un sistema attraverso l'interazione tra *gruppi di oggetti*.

- Si tratta di un modo intuitivo di rappresentare la realtà, per mezzo di
  - **OGGETTI**
  - Scambio di **MESSAGGI**
  - **RESPONSABILITÀ**

Oggetti

- Un oggetto è l'elemento base della programmazione ad orientata agli oggetti (OOP).

- Può essere definito come:
  - Contenitore di **attributi** (le variabili) e **comportamenti** (i metodi), che agiscono sugli attributi o su altri oggetti.

- Gli attributi possono essere tipi di dato fondamentali, altri oggetti o strutture dati.

Vantaggi della OOAD

- Facilita la realizzazione di modelli “eccellenti” grazie a
  - Incapsulamento (information hiding)
  - Astrazione ed ereditarietà (generalizzazione)
  - Modularizzazione
  - Riutilizzo

- Grazie alla OOAD, il sistema da automatizzare viene modellato ad oggetti, così come viene programmato il software che lo automatizza.

Progettazione

- Prima della realizzazione di un qualsiasi sistema complesso è necessario progettare il software
  - I linguaggi di programmazione servono solo all'implementazione.
- E' necessario utilizzare un **linguaggio di modellazione** che consenta di modellare e progettare software complessi.
- NB: Si progetta prima di implementare.

Approcio visuale alla progettazione

- Chi progetta un qualsiasi tipo di costruzione o artefatto
  - utilizza sempre **figure, schemi, diagrammi** per svolgere la propria attività:
    - ingegneri
    - architetti
    - ... ma anche stilisti
- Si utilizzano diagrammi e figure per visualizzare i propri progetti

Approccio visuale / 2

- Anche i progettisti e gli analisti di sistemi informativi utilizzano figure e diagrammi per visualizzare il risultato del loro lavoro:
  - Modellazione dati
  - Modellazione software

- **Progettazione visuale nonostante il prodotto finale risultante non sia necessariamente visuale.**

Vantaggi approccio visuale

- Il progettista di un sistema sofware ha la necessità di rappresentare i **diversi aspetti** del progetto
  - Diagrammi differenti, focalizzati sui vari aspetti

- Analogamente al progetto di un edificio
  - **Visione completa da più punti di vista**
    - es. Collegamento tra le componenti software
  - **Visione di dettaglio e particolari**
    - es. Sequenze di comunicazione

Generazione del modello

- Tramite il concetto di **astrazione** una realtà anche molto complessa viene rappresentata semplificandola in un ***modello***.

- Durante analisi e progettazione vengono generati modelli che consentono di **identificare ed isolare le caratteristiche** di un sistema.
  - Il progettista sceglie le caratteristiche rilevanti e definisce le **relazioni** tra queste.

Tipi di relazione

- Devono essere considerati diversi tipi di relazione
  - **Strutturali**
    - es. Associazioni tra classi
  - **Temporali**
    - es. Sequenza di messaggi
  - **Causa-Effetto**
    - es. Diagramma di stato
  - **Organizzative**
    - es. Raggruppamento degli elementi
  - **Evolutive**
    - es. Dipendenze tra i diagrammi

UML

- *Unified Modeling Language (UML)*
  - È un linguaggio di modellazione visuale

- **Consente ad analisti e progettisti di sistemi orientati agli oggetti di modellare, rappresentare e documentare sistemi software.**

- Non è un linguaggio di programmazione.
  - Tuttavia esistono software per convertire diagrammi UML in elementi di software da espandere.

UML / 2

- Non è una metodologia di sviluppo del software.
- ***UML è un linguaggio, un insieme di elementi e regole, di specifica formale.***
- Gli *elementi* sono forme grafiche che rappresentano ciò che si sta modellando.
- Le *regole* spiegano come combinare gli elementi.

UML / 3

- **Tre tipi di regole**
  - Sintassi astratta
    - Diagrammi e linguaggio naturale
  - Regole sintattiche
    - Object Constraint Language (OCL) e linguaggio naturale
  - Semantica
    - Linguaggio naturale con supporto di diagrammi

UML / 4

- UML è il linguaggio standard di modellazione **più diffuso** nello sviluppo di software a livello industriale.

- E' uno standard in evoluzione riconosciuto dall'Organizzazione Internazionale per la Standardizzazione (International Organization for Standardization, ISO).

- **Versione attuale: 2.2**
- Sviluppato da Object Management Group (www.omg.org)

Processo unificato

- Lo scopo di UML è di definire in modo formale un sistema.

- I progettisti di UML hanno definito un modo per procedere nel processo di sviluppo utilizzando UML.

- ***Processo Unificato per lo Sviluppo del Software***
  - *Unified Software Development Process (USDP)*
  - Coinvolge persone, progetti, strumenti, processi e prodotti

Processo unificato / 2

- Nel processo unificato:
  - *i partecipanti e gli sviluppatori coinvolti nel progetto di sviluppo di un sistema, seguono un determinato processo, utilizzando strumenti di ausilio nello sviluppo, generando prodotti software.*

- Il processo parte dai requisiti dell'utente, che vengono raccolti nei cosiddetti **casi d'uso**, delle sequenze di esecuzione del sistema in grado di fornire un valore all'utente.

Caratteristiche del PU

- Le caratteristiche del processo unificato:
  - **Architetto-centrico**
    - l'architettura del sistema viene sviluppata in modo da soddisfare i requisiti dei casi d'uso più importanti.
  - **Iterativo**
    - il progetto viene scomposto in sottoprogetti che costruiscono parti del sistema finale
  - **Incrementale**
    - il sistema viene costruito incrementalmente unendo le singole parti sviluppate nei sottoprogetti.

Processo unificato / 3

- Gli sviluppatori, partendo dai casi d'uso raccolti, producono i modelli del sistema e le implementazioni che li realizzano.

- **Fasi del processo**
  - Inizio
  - Elaborazione
  - Costruzione
  - Transizione

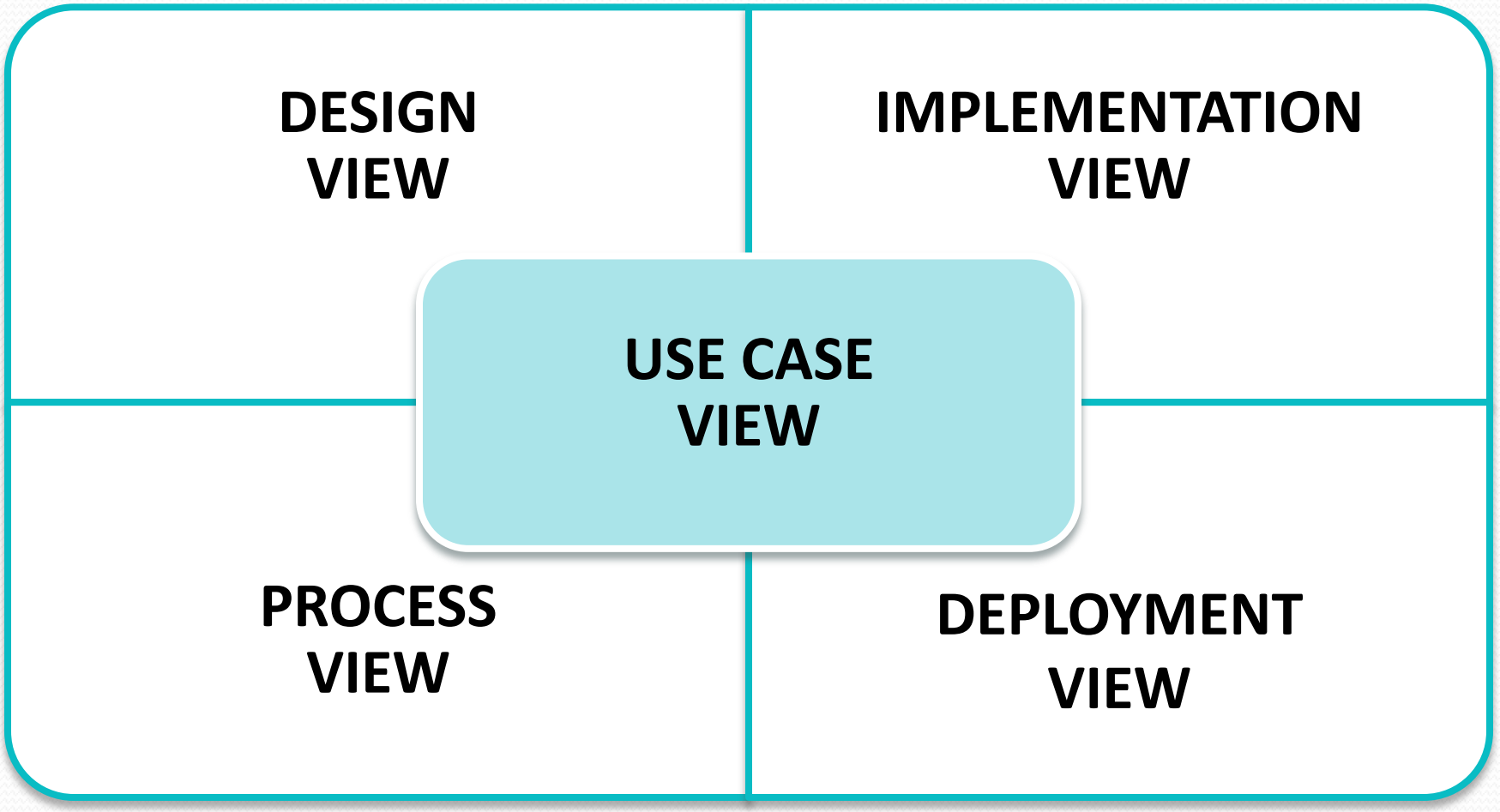
Modelli del processo unificato

- I prodotti intermedi del processo sono i modelli (astrazioni delle caratteristiche del sistema).
- Ogni modello definisce il sistema da un certo punto di vista, i principali:
  - dei Casi d'Uso
  - di Analisi
  - di Progetto
  - di Deployment
  - di Implementazione
  - di Test

Struttura di UML

- Composto da
  - **Viste**
    - Mostrano i differenti aspetti di un sistema attraverso la realizzazione di un certo numero di diagrammi.
    - Sono astrazioni, ognuna delle quali modella il sistema con un'ottica diversa (funzionale, non funzionale, organizzativa).
    - La somma delle viste fornisce il quadro d'insieme.
  - **Diagrammi**
    - Esprimono le viste tramite diagrammi.
  - **Elementi**
    - Sono i concetti che permettono di realizzare i diagrammi (attori).

Le viste di UML



Le viste di UML / 2

- **Use case view**
  - Analizza i ***requisiti utente***
    - Cosa dovrà fare il sistema?
  - Vista ad alto livello di importanza fondamentale:
    - Guida lo sviluppo delle rimanenti.
    - Stabilisce le funzionalità che il sistema dovrà realizzare.
  - Le funzionalità saranno individuate *assieme al cliente* grazie all'ausilio di tale vista, utile al reperimento dei requisiti.
    - A questo livello è necessario **concentrarsi sul cosa fare**, **astraendosi** il più possibile dal come verrà implementato.

Le viste di UML / 3

- **Design view**
  - Descrive come realizzare le funzioni
    - Si analizza il sistema dall'interno.
  - Si trovano
    - **struttura statica** del sistema
      - diagramma delle classi
      - diagramma degli oggetti
    - **collaborazione dinamica**
      - interazioni tra gli oggetti del sistema

Le viste di UML / 4

- **Implementation view**
  - Descrive l'aggregazione in moduli, chiamati package, e le relative interdipendenze.

- **Process view**
  - Analizza gli aspetti non funzionali del sistema
    - Individuazione dei processi
  - Serve a massimizzare l'utilizzo efficiente delle risorse
    - Stabilendo l'esecuzione parallela di determinate operazioni
    - Consentendo la gestione di eventi asincroni

Le viste di UML / 5

- **Deployment view**
  - Mostra l'architettura fisica del sistema
  - La distribuzione e l'ubicazioni delle componenti software all'interno della struttura stessa.

Diagrammi UML

- Sono grafici che visualizzano una particolare proiezione del sistema da modellare.
  - **Use case diagram**
  - **Class diagram**
  - Object diagram
  - Sequence diagram
  - Collaboration diagram
  - **State diagram**
  - Activity diagram
  - Component diagram
  - Deployment diagram

Casi d'uso

- **Visione d'insieme del sistema** che si sta analizzando
  - Fondamentali nelle fasi iniziali del progetto

- Rappresentano il sistema visto dall'utilizzatore
  - Modellano il dialogo tra l'utilizzatore e il sistema stesso
    - Descrivono l'interazione tra attori e sistema
      - Non la logica propria della funzione (sono astratti)
    - Espressi in forma testuale (comprensibili anche ai non addetti)
    - Diversi livelli di definizione (sistema complessivo o sue parti)
    - Rappresentano le modalità d'uso del sistema da parte di uno o più attori

Casi d'uso / 2

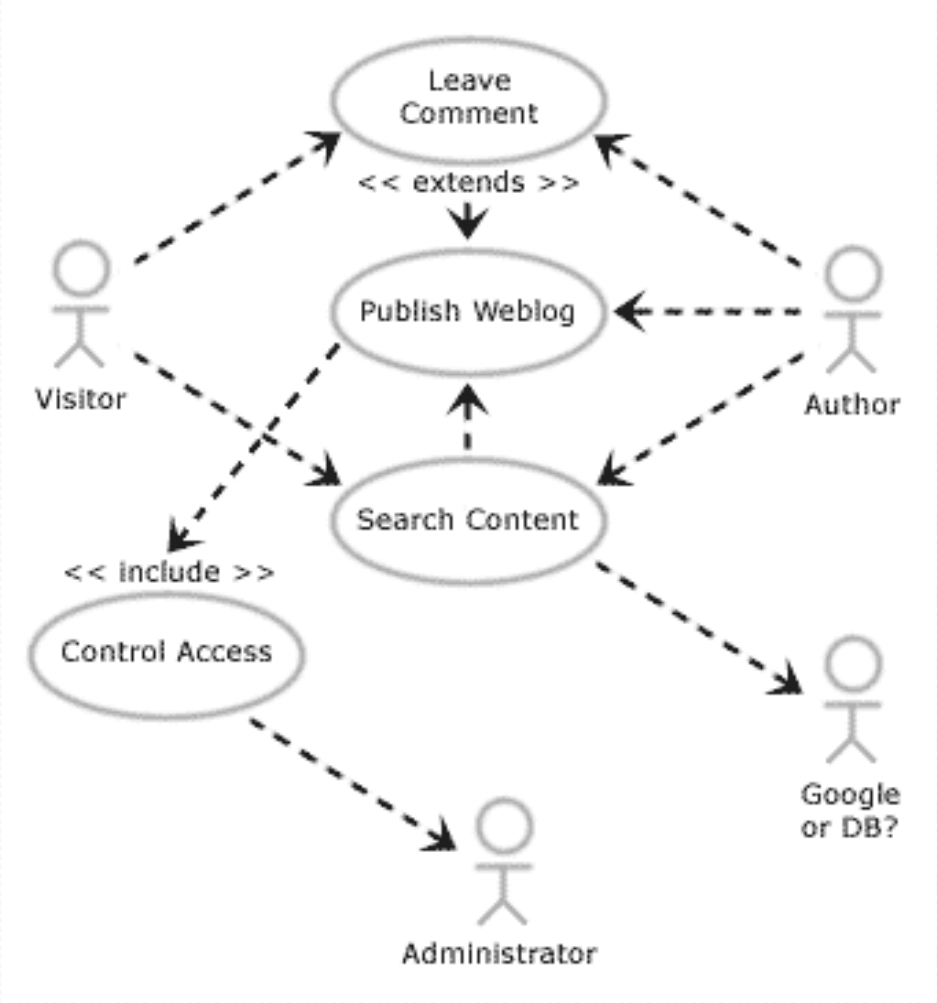
- Consentono di scoprire i **requisiti funzionali**
  - Rappresentano un valido ausilio nel dialogo con l'utente
  - Fondamentali con esponenti non tecnici

- Vengono rappresentati da un diagramma.

Diagramma dei casi d'uso

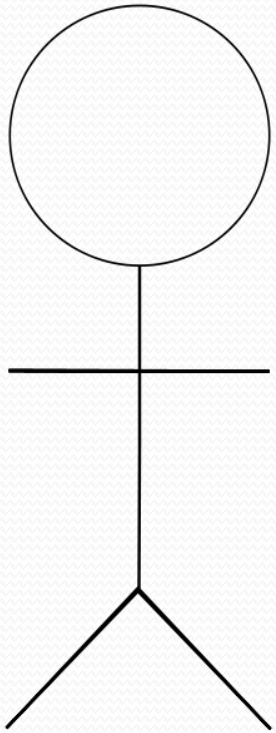
- Rappresenta i casi d'uso, gli attori e le associazioni che li legano.
- Visualizzano le *modalità d'utilizzo* del del sistema da parte di uno o più utilizzatori, che vengono definiti attori.
- I singoli casi d'uso descrivono le iterazioni tra sistema e attori (non la logica delle funzioni).
- Un **attore** è un elemento esterno al sistema che fornisce un'input a cui il sistema deve rispondere.

Esempio

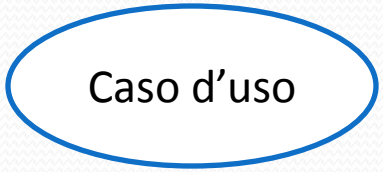

Leave Comment
<< extends >>
Publish Weblog
Visitor
Author
Search Content
<< include >>
Control Access
Google or DB?
Administrator

Attore

- **Utilizzatore del sistema** che interagisce con i casi d'uso
  - Può essere umano oppure un altro sistema
- Rappresenta un ruolo di un oggetto
  - Un oggetto fisico (classe) può assumere ruoli diversi e quindi essere modellato da più attori.
  - Nel caso di attori umani, il nome di un attore identifica il ruolo che l'attore svolge in quel contesto, non il suo incarico o mansione.

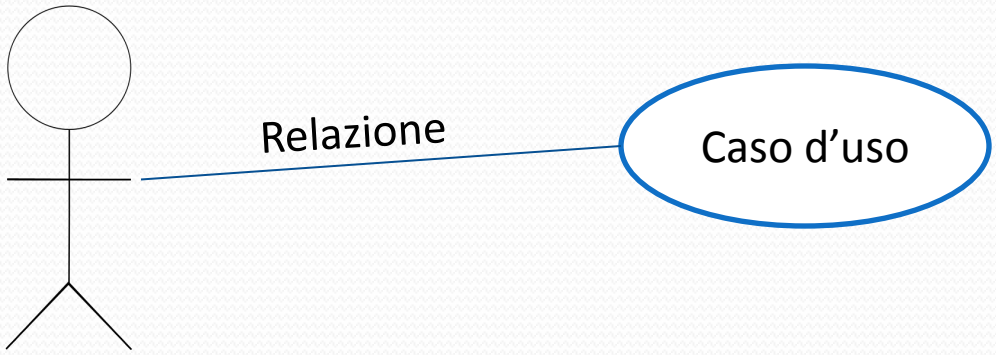


Caso d'uso



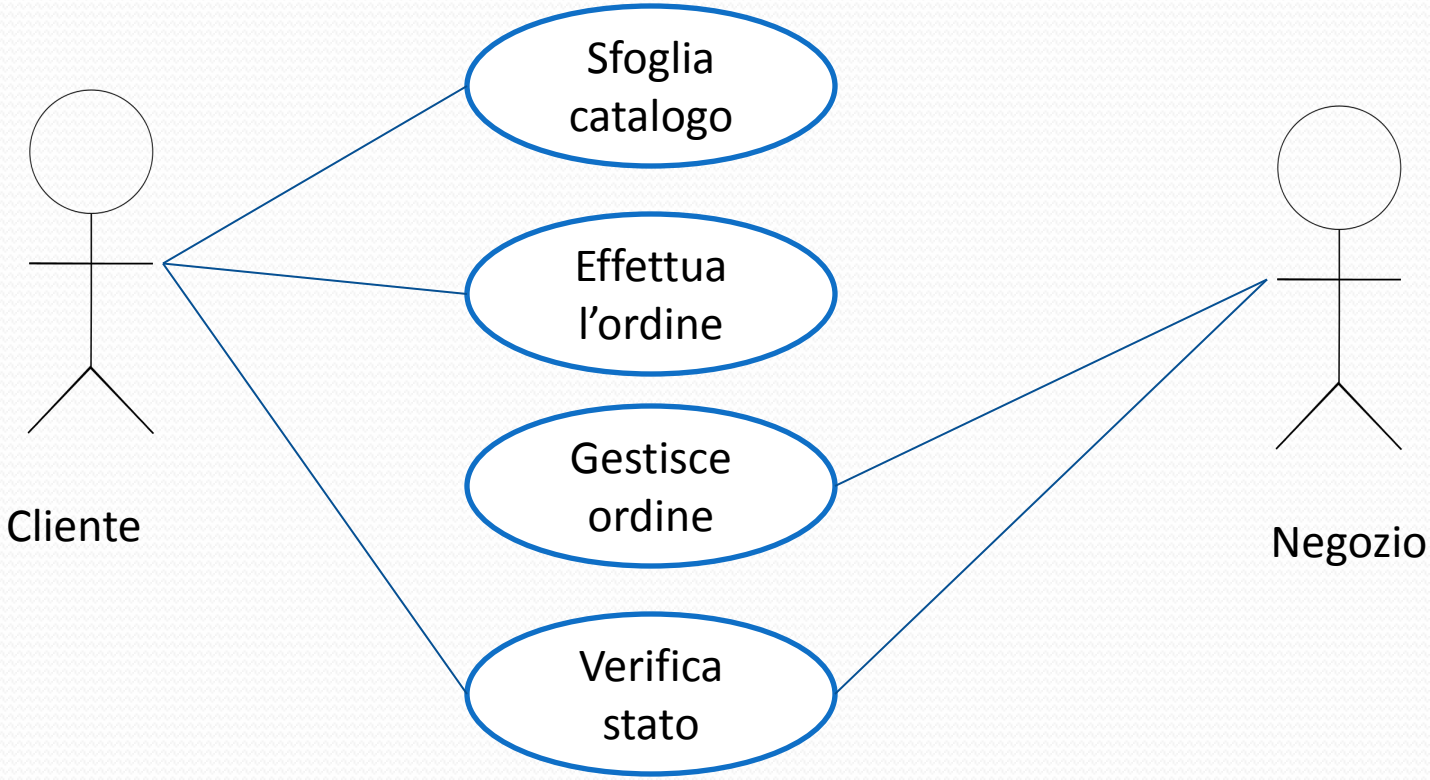
- Rappresenta una particolare modalità d'utilizzo del sistema.
- Rappresenta alcune delle funzioni visibili all'attore.
- Consente all'attore di raggiungere alcuni obiettivi
  - Manipolazione di informazione
  - Esecuzione di funzione
  - *es. archivia documento*

Relazioni tra attori e use case



- La relazione indica l'esistenza di un'associazione tra un attore ed un caso d'uso.
  - *Quindi una particolare persona (o sistema) che si trova in un certo ruolo comunica con specifiche istanze del caso d'uso, partecipando agli eventi rappresentati dal caso.*

Esempio: e-commerce



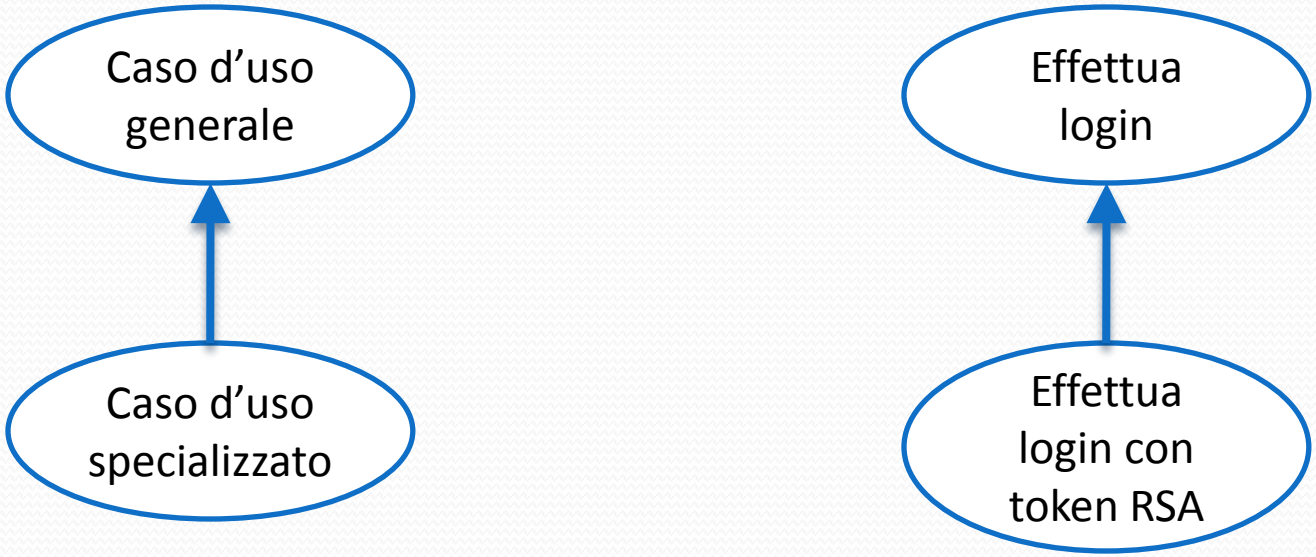
Rappresentazioni accessorie

- Con i diagrammi dei casi d'uso si possono inoltre rappresentare:
  - Generalizzazioni tra casi d'uso
  - Generalizzazioni tra attori
  - Relazioni di inclusione tra casi d'uso (include)
  - Relazioni di estensione tra casi d'uso (extend)

Generalizzazione tra casi d’uso

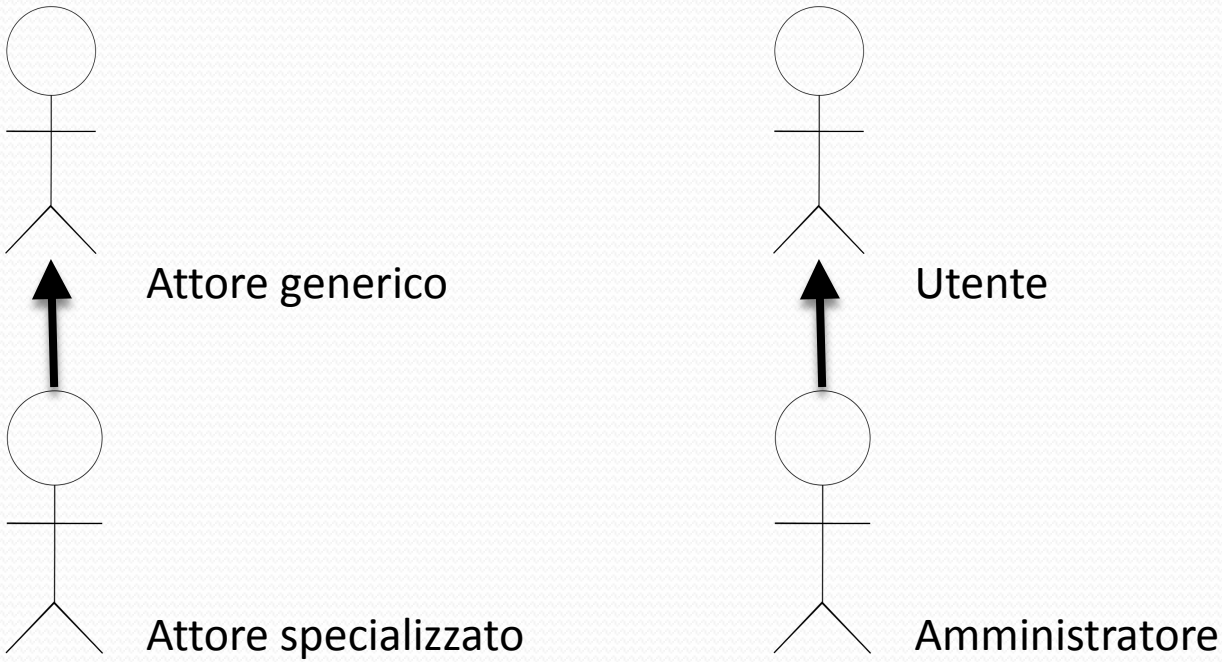
- Può esistere più di una versione di un caso d’uso, aventi ognuna alcune azioni in comune ed altre uniche per ciascun caso.
- Nel diagramma si associano i casi d’uso con una freccia puntata verso il caso d’uso più generale.
- Il caso d’uso specializzato eredita parte delle funzionalità del caso d’uso generico

Esempio generalizzazione UC



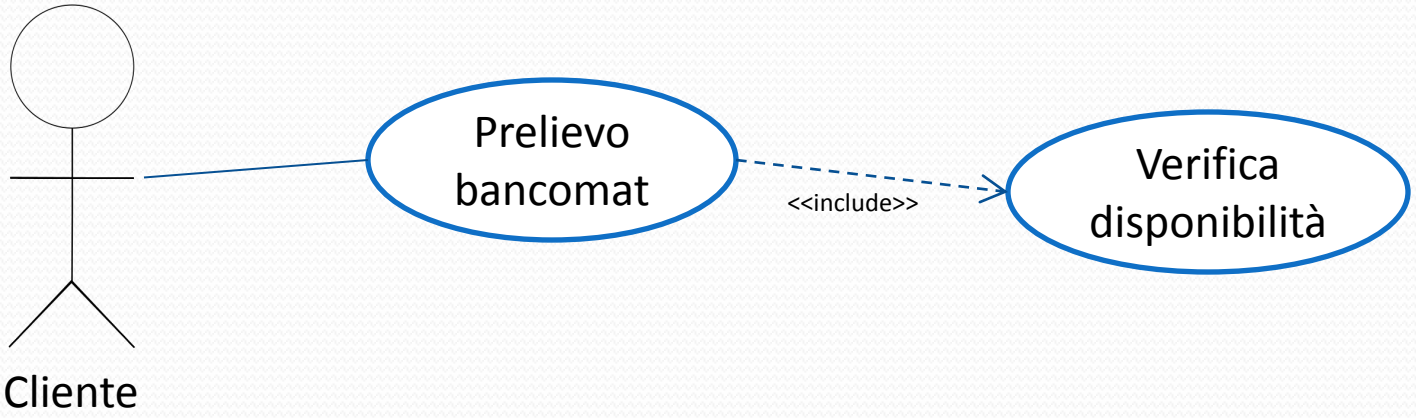
Generalizzazione tra attori

- L'attore specializzato eredita parte delle caratteristiche dell'attore generico.



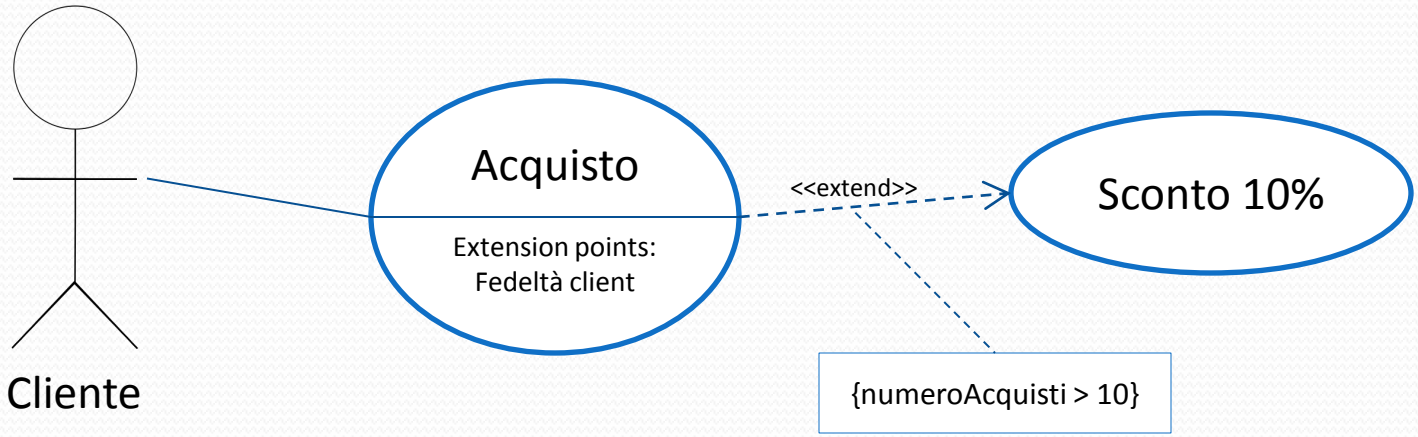
Inclusione tra casi d'uso

- Un caso d'uso **comprende le funzionalità** di un altro caso.
  - Una relazione di inclusione da un caso d'uso A(base) ad un caso d'uso B(incluso) indica che le funzionalità del caso d'suo B saranno *sempre presenti* in un istanza del caso d'uso A.



Estensione di un caso d'uso

- Un caso d'uso può essere **esteso nella sua funzionalità** ad un altro caso
  - Una relazione di estensione da un caso d'uso A ad un caso d'uso B indica che un istanza del caso d'uso B può essere *maggiorata, sotto determinate condizioni specificate* nell'estensione, dal comportamento di A.



Differenze tra extend e include

- Una relazione di inclusione (A include B)
  - indica che ogni istanza del caso d'uso A **include sempre** il comportamento di B

- Una relazione di estensione (A estende B)
  - indica che **al verificarsi di opportune condizioni** (condizioni di guardia) il comportamento di B può essere esteso dal comportamento di A.

Specifica comportamentale

- Ogni caso d'uso costituisce una sequenza di attività che generano un risultato per l'attore che con esso interagisce.

- La sequenza di attività viene descritta in una **specifica comportamentale**
  - Può essere un diagramma di attività, di sequenza, di stato o una descrizione informale.
    - Generalmente si sceglie quest'ultima.

Descrizione del caso d'uso

- Ogni descrizione di un caso d'uso è caratterizzata dalla descrizione di informazioni di base che permettono di specificare il contesto, chi utilizza il caso d'suo, cosa succede allo stato del sistema prima e dopo l'attivazione del caso d'uso ed uno o più scenari di interazione.

- A loro volta gli scenari di interazione si suddividono in:
  - Base, quando il caso d'uso esegue il suo compito correttamente andando opportunamente a modificare lo stato del sistema.
  - Alternativi, quando il caso d'uso termina con esiti positivi, con complicazioni o va incontro ad errori, cioè fallisce il suo compito.

Tabella descrizione Use Case

Caso d'uso	<codice>
Data	
Nome	
Versione	<per revisioni>
Descrizione	
Priorità	
Durata	
Punti di estensione	
Estende	
Include	
Attore primario	
Attori secondari	

Tabella descrizione Use Case / 2

Precondizioni	<cose necessarie per l'avvio del caso d'uso>
Postcondizioni	<condizioni da verificare per successo / insuccesso del caso d'uso>
Innesco	<cosa da avvio>
Scenario principale	
Scenario alternativo I	
Scenario alternativo II (insuccesso)	
Scenario alternativo N	
Note	
Riferimento	

Creare il diagramma Use Case

- Si parte da appunti e trascrizioni di interviste e colloqui con il cliente.

- Si procede per fasi
  - **Definizione attori e casi d'uso**
    - Chi utilizzerà il sistema per il data entry?
    - Chi è il destinatario delle informazioni fornite dal sistema?
    - Quali altri sistemi devono interagire con questo?
  - **Organizzare secondo priorità i casi d'uso individuati**
    - Sviluppare prima i più importanti

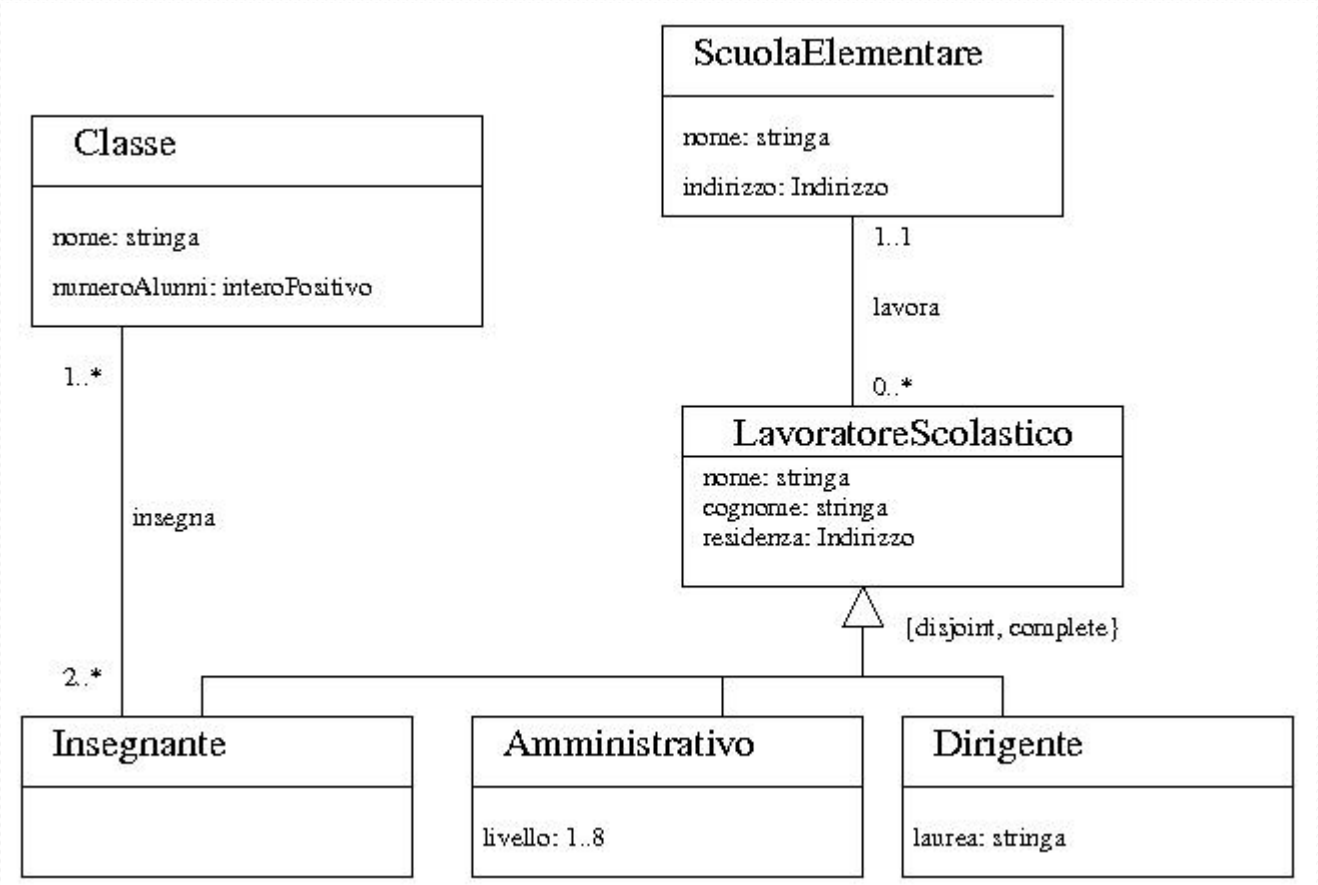
Creare il diagramma Use Case / 2

- Sviluppare ciascun caso d'uso
  - Creare la specifica dettagliata di ogni caso d'uso
- Strutturare il diagramma dei casi d'uso
  - Aggiunta di attori e casi d'uso
  - Generalizzazioni
  - Inclusioni
  - Estensioni
  - Raggruppamento in unità logiche (package)

Diagramma delle classi

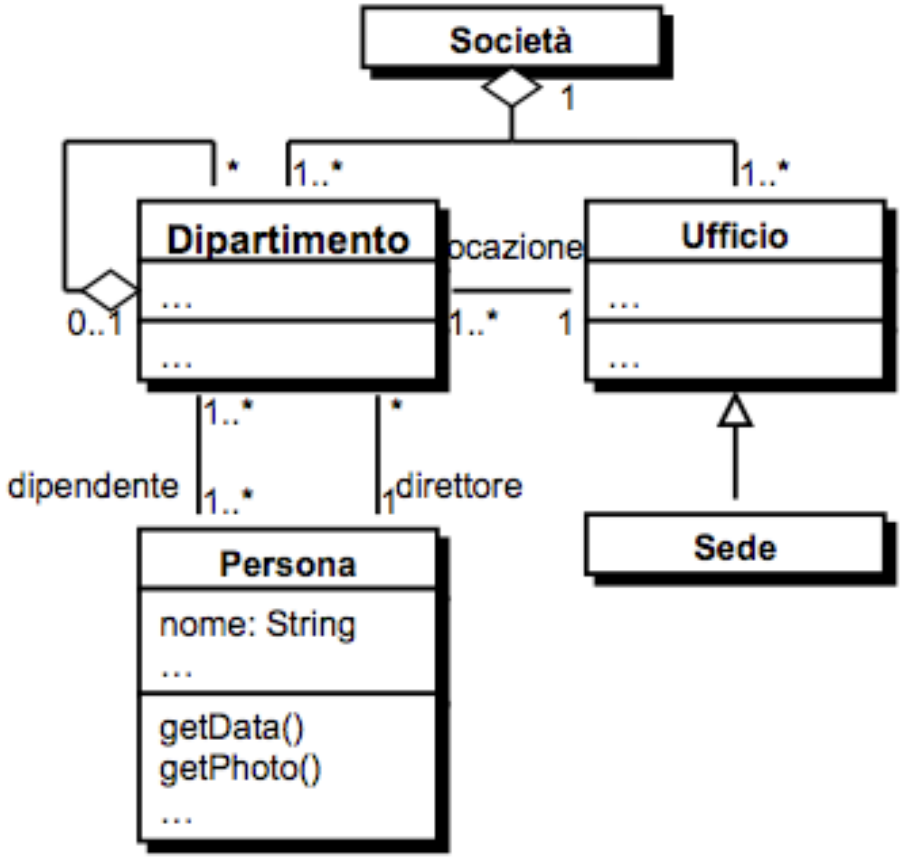
- I diagrammi delle classi rappresentano le classi e gli oggetti, con i relativi attributi ed operazioni, che compongono il sistema.
  - Rappresenta graficamente la parte **statica** del sistema.
- Specifica mediante associazioni, i vincoli che legano le varie classi del sistema.
- Può essere redatto a molteplici livelli di dettaglio (analisi, progettazione di dettaglio)

Esempio


ScuolaElementare
nome: stringa
indirizzo: Indirizzo
1..1
lavora
0..*
Classe
nome: stringa
numeroAlunni: interoPositivo
1..*
insegna
2..*
LavoratoreScolastico
nome: stringa
cognome: stringa
residenza: Indirizzo
{disjoint, complete}
Insegnante
Amministrativo
livello: 1..8
Dirigente
laurea: stringa

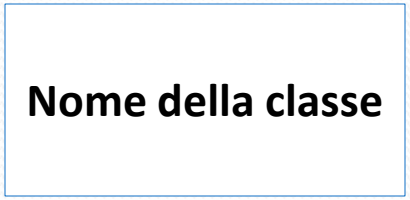
Esempio

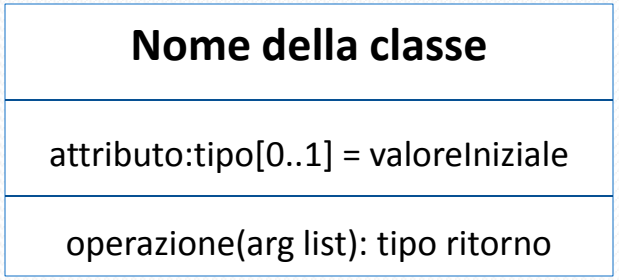
- Una società è formata da dipartimenti e uffici
- Un dipartimento ha un direttore e più dipendenti
- Un dipartimento è situato in un ufficio
- Esiste una struttura gerarchica dei dipartimenti
- Le sedi sono uffici



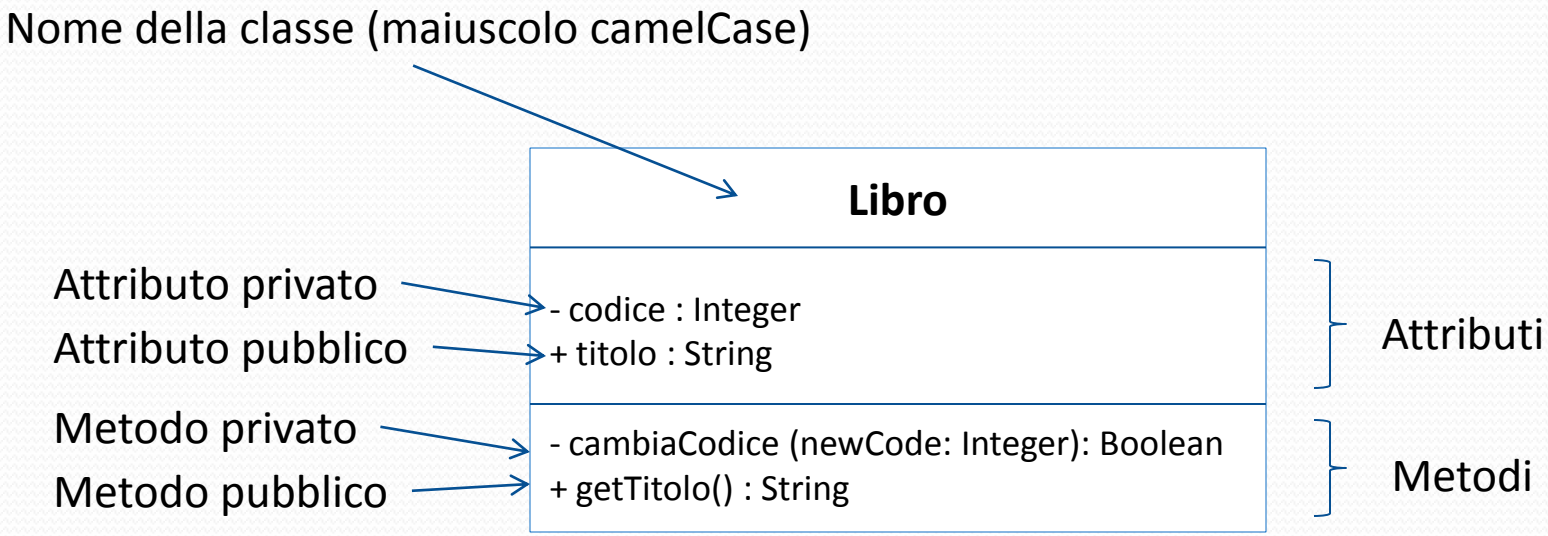
Classe

- Rappresentazione di una classe
  - Nome
  - Attributi
  - Operazioni (metodi)





Classe / 2


Nome della classe (maiuscolo camelCase)
Libro
Attributo privato
Attributo pubblico
- codice : Integer
+ titolo : String
Attributi
Metodo privato
Metodo pubblico
- cambiaCodice (newCode: Integer): Boolean
+ getTitolo() : String
Metodi

Attributi

```
visibilità nome: tipo molteplicità = default {proprietà}
```

- **Visibilità** attributo pubblico (+) o privato (-)
- **Nome** nome della variabile
- **Tipo** tipo di dato (String, Integer, Classe, ...)
- **Molteplicità** numero di elementi (vedi slide successive)
- **Default** valore iniziale dell'attributo
- **Proprietà** proprietà aggiuntive (const, readOnly, ...)

Operazioni (firma)

```
visibilità nome (nomeParametro:tipo,...): tipoRestituito
```

- **Visibilità** pubblico (+) o privato (-)
- **Nome** nome del metodo (camelCase)
- **Parametri** elenco nome e tipo di dato
- **TipoRestituito** tipo di dato restituito

- Ogni operazione di una classe deve avere una firma univoca (signature).

Visibilità

- Un elemento è visibile all'esterno del namespace o della classe che lo contiene a seconda della sua visibilità.
  - Concetto fondamentale della OOP: incapsulamento o information hiding.

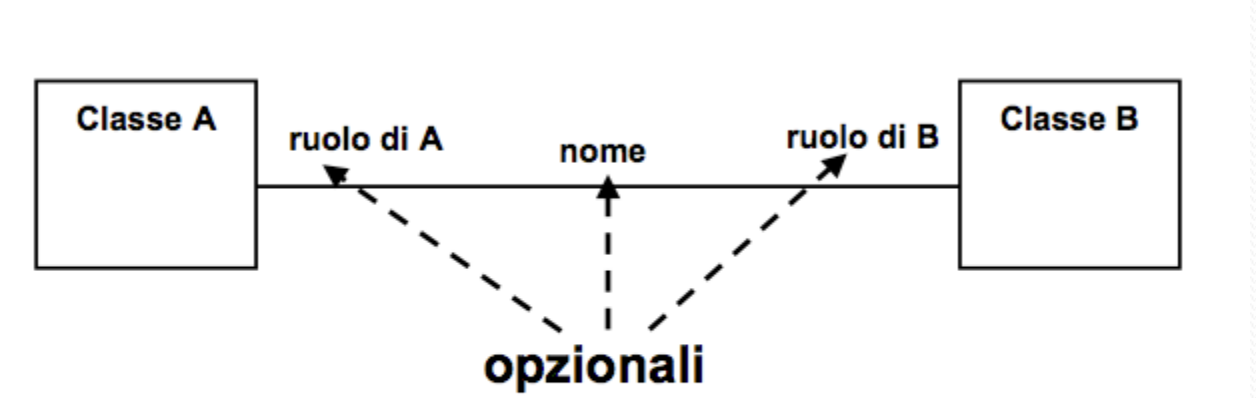
+	**public**	altri elementi possono vedere ed usare
#	**protected**	solo dai suoi discendenti (ereditarietà)
-	**private**	solo dagli elementi nell'elemento stesso
~	**package**	solo dagli elementi nello stesso package

Relazioni

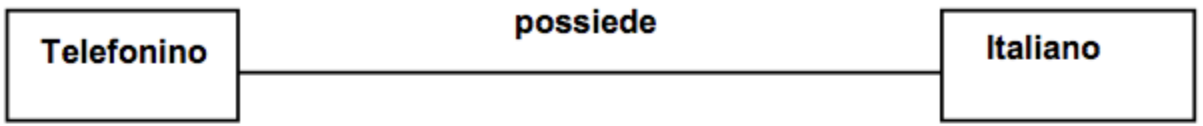
- Una relazione rappresenta un legame tra due oggetti
  - Normalmente tra istanze di classi diverse (non necessariamente)

- UML fornisce rappresentazioni grafiche per
  - **Associazione**
    - Aggregazione
    - Composizione
  - **Generalizzazione**
  - **Dipendenza**
    - Uso
    - Realizzazione

Associazione binaria



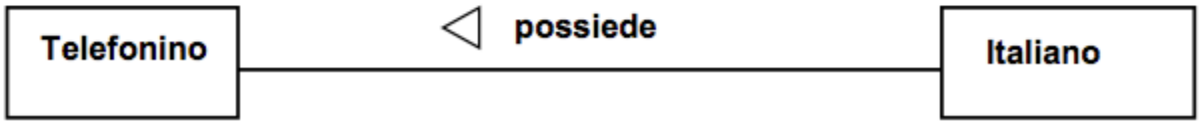
Associazione binaria / 2



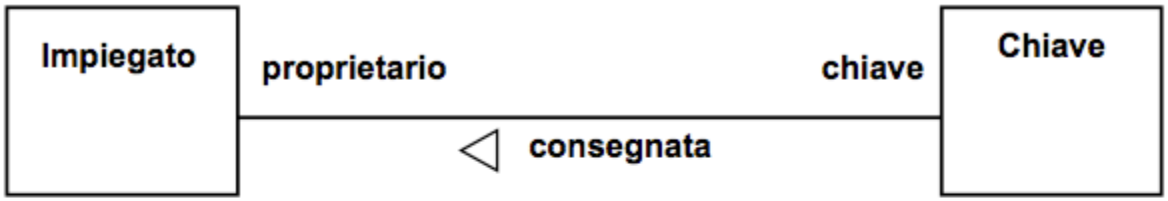
- **Potrebbe essere letto:**

  (ogni) telefonino possiede un italiano [Altan]

- **Uso una freccia di *direzione di lettura* per disambiguare**



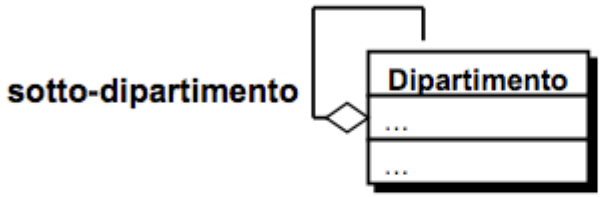
Associazione: ruoli



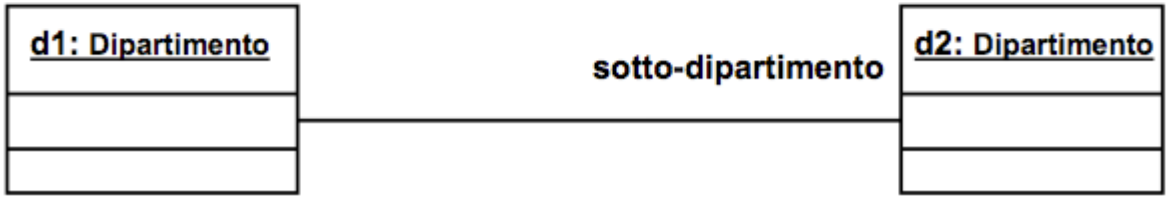
- se i è un Impiegato, e c una Chiave
- e c è stata consegnata ad i,
- si può dire che i è il *proprietario* di c
  e che c è la *chiave* di i

Associazioni e gerarchia

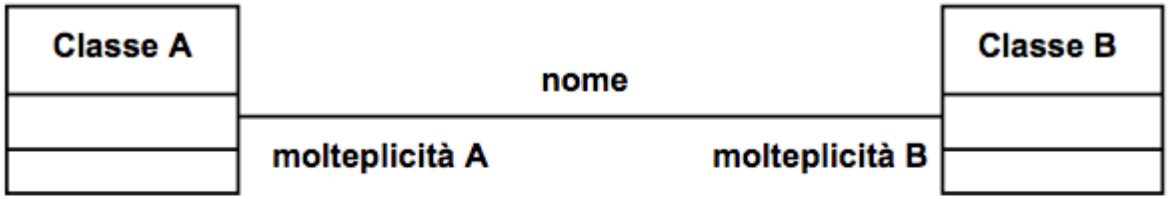
- **Autorelazione nel diagramma delle classi**



- **I ruoli discriminano tra oggetti dello stesso tipo**



Associazioni: molteplicità



Numero di oggetti coinvolti nell'associazione in un dato istante

- **1** un solo oggetto
- **0..2** nessun oggetto, un oggetto o 2 oggetti
- **1..*** più di un oggetto

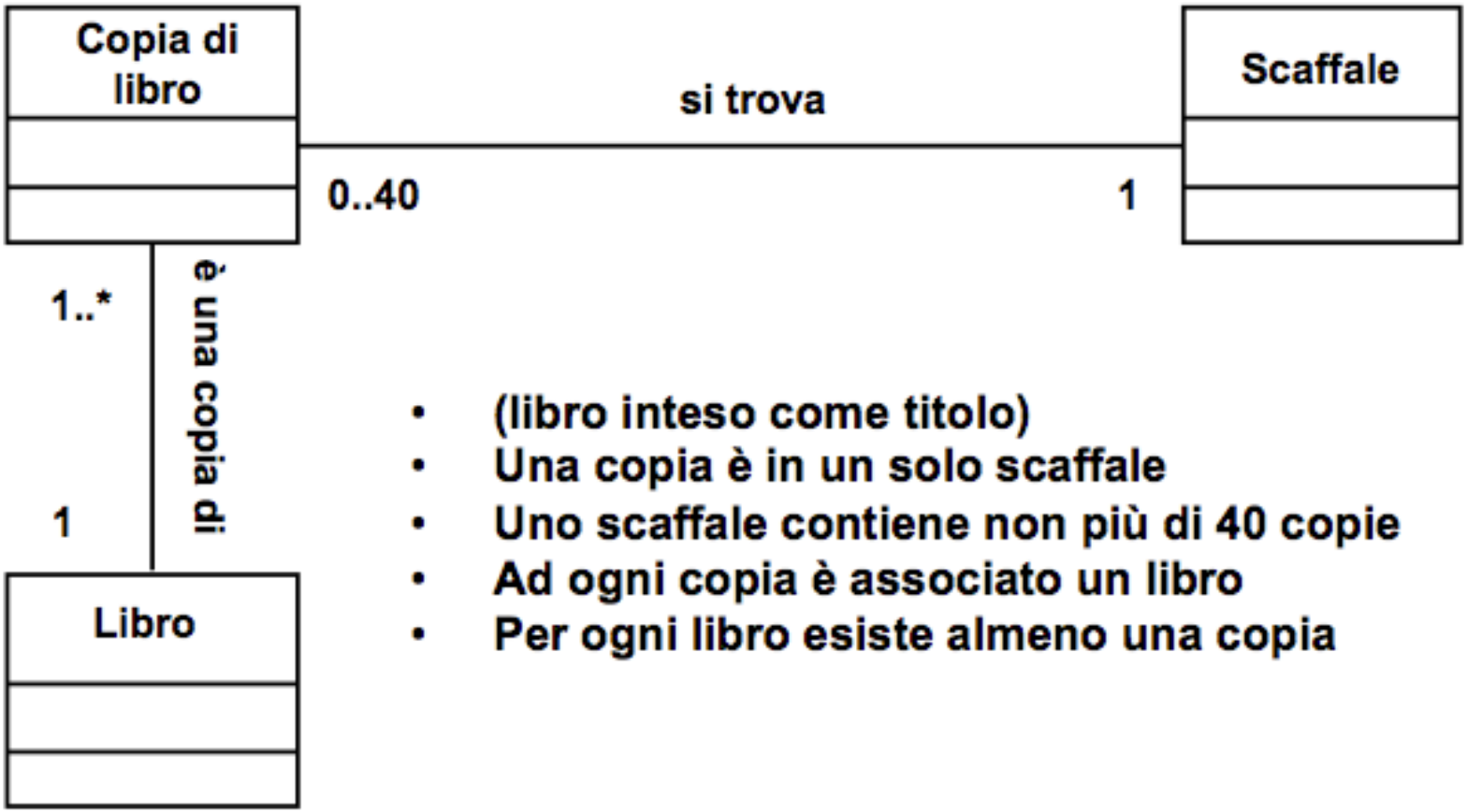
Molteplicità

- Nelle associazioni e nelle dichiarazioni UML consente di specificare la molteplicità (nota anche come cardinalità).
  - Specificata da Limite inferiore..Limite Superiore (N..M)
    - **Opzionale** Limite inferiore = 0
    - **Obbligatorio** Limite inferiore = 1
    - **Un solo valore** Limite superiore = 1
    - **Più di un valore** Limite superiore > 1
    - **Zero o più** *
    - **Uno o più** +
  - Ad esempio 1, 0..1, N..M, +, *

Molteplicità

- Un oggetto Società può essere in relazione con molti oggetti Persona
- Un oggetto persona può essere in relazione con un solo oggetto Società

Esempio



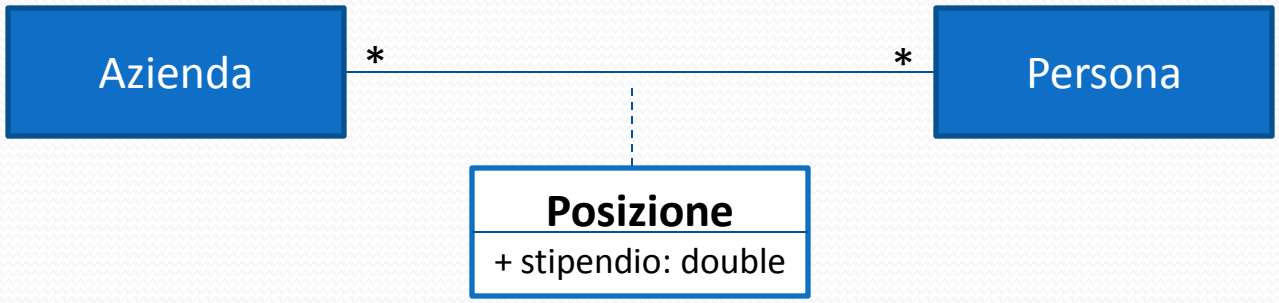
- (libro inteso come titolo)
- Una copia è in un solo scaffale
- Uno scaffale contiene non più di 40 copie
- Ad ogni copia è associato un libro
- Per ogni libro esiste almeno una copia

Classi di associazione

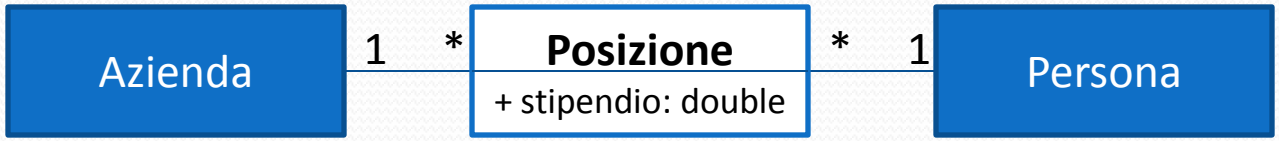
- Data questa associazione, aggiungiamo la regola per cui ogni **Persona** percepisce uno stipendio da ogni **Azienda** in cui è impiegata.

- L'attributo stipendio è proprio dell'associazione tra Azienda e Persona.

Classi di associazione / 2



- La classe associazione che connette due classi e definisce un insieme di caratteristiche proprie della associazione stessa



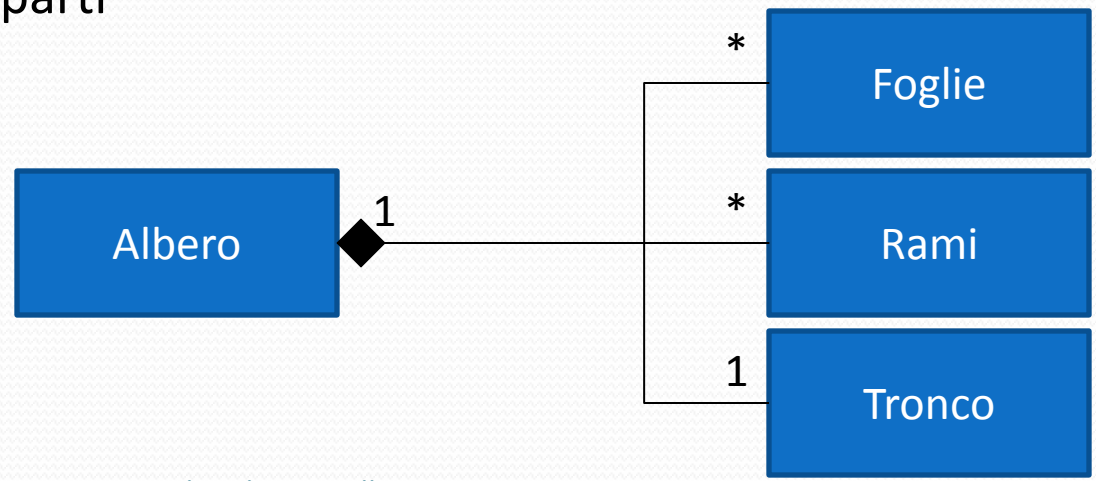
Aggregazione

- Speciale forma di associazione
  - Relazione tra l'aggregato e le parti che lo compongono
  - Relazione debole: le parti esistono anche senza il tutto

- L'aggregato può in alcuni casi esistere indipendentemente dalle parti, ma in altri casi no.
- Più aggregati possono condividere la stessa parte
- Transitiva e asimmetrica

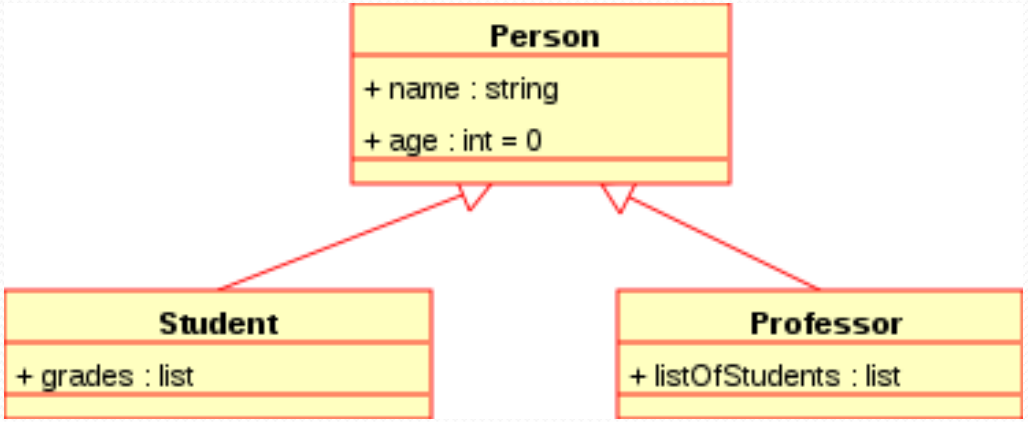
Composizione

- Speciale forma di associazione
  - Relazione molto forte: le parti dipendono dall'aggregato.
  - Ogni parte può appartenere ad un solo composito per volta
    - Il composito è l'unico responsabile di tutte le sue parti: questo vuol dire che è responsabile della loro creazione e distruzione
    - Se il composito viene distrutto, devono essere distrutte anche le sue parti

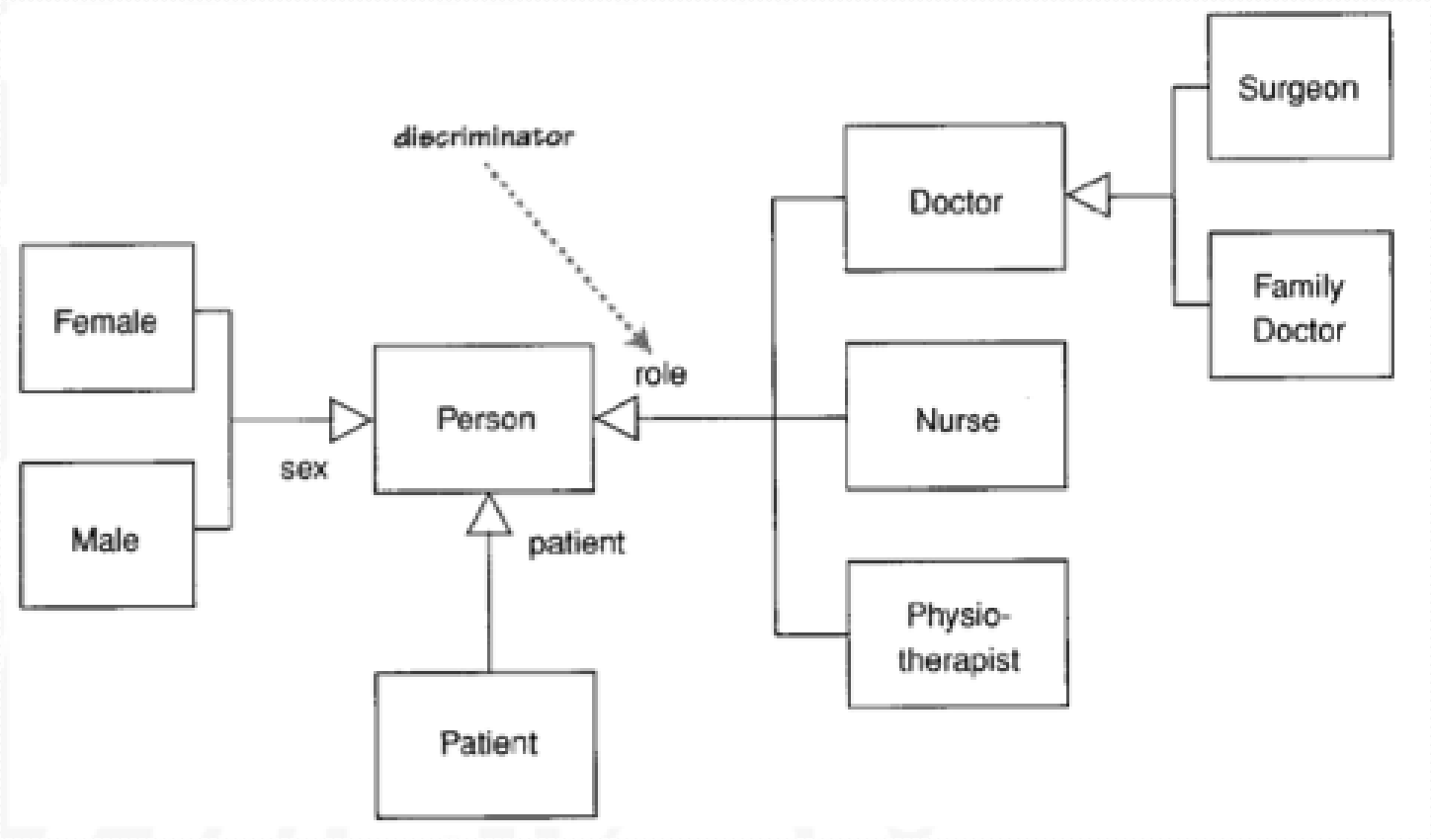


Generalizzazione

- La relazione di generalizzazione indica che una delle due classi considerate è sottotipo (specializzazione) di dell'altra (supertipo).
  - Ogni istanza del sottotipo è anche istanza del supertipo (polimorfismo).
- In UML la generalizzazione viene indicata con una freccia con la testa vuota.

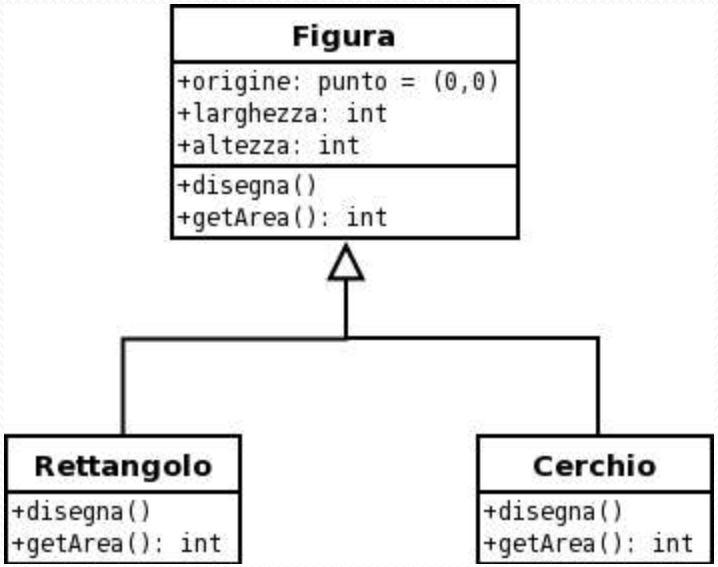


Generalizzazione



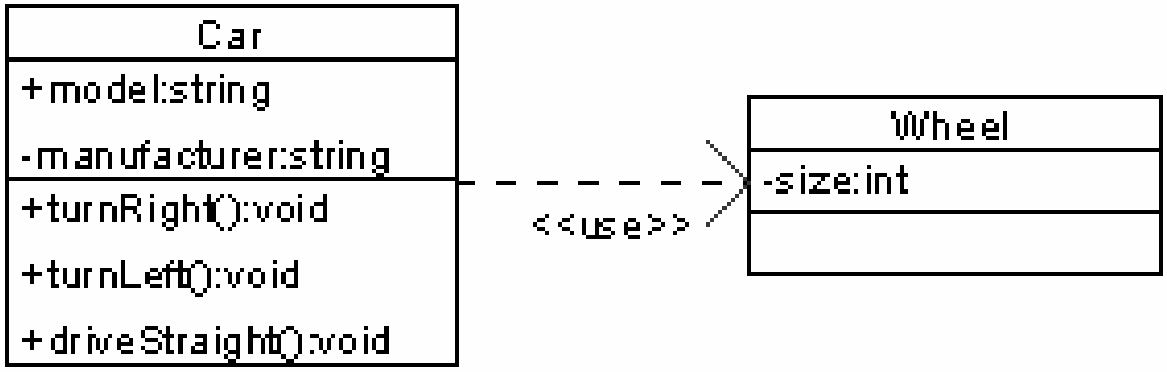
Ereditarietà e sovrascrittura

- Per ridefinire una operazione della superclasse, una sottoclasse deve avere una propria operazione con identica *firma*.
- Questa operazione è nota come overriding.

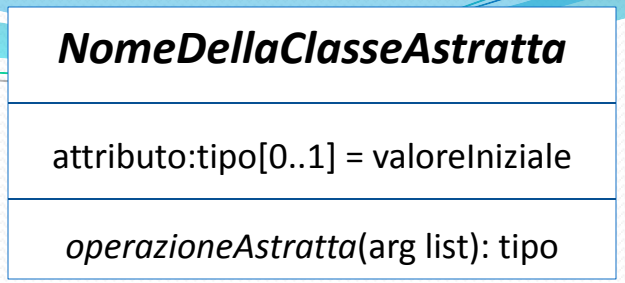


Dipendenza

- Forma debole di relazione
- Sta ad indicare che una classe dipende da un'altra, in quanto quest'ultima viene utilizzata dalla prima.
  - Esiste dipendenza se una classe è un parametro o una variabile di un metodo di un'altra classe.



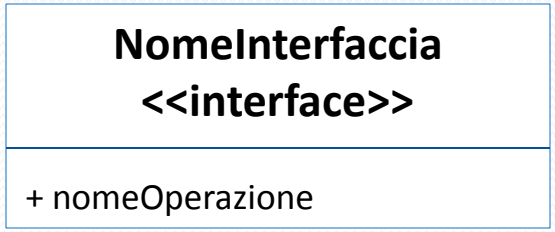
Classi astratte



- Una classe astratta è una classe che non può essere istanziata direttamente.
  - Per poterla istanziare è necessario estenderla e realizzarne quindi una classe concreta.
- Analogamente esistono le operazioni astratte, che definiscono la firma ma non l'implementazione.
- Le operazioni e le classi astratte sono indicate scrivendo il nome in corsivo.

Interfacce

- Insieme dei metodi visibili dall'esterno di un oggetto.
- L'interfaccia è intesa come astrazione per la fruizione dall'esterno.
  - Questo consente di separare la dichiarazione delle firme dall'implementazione reale, favorendo generalizzazione e polimorfismo.
- Rappresentata come una classe con lo sterotipo <<interface>>.



Realizzazione

- Una relazione di realizzazione tra una classe o un componente e una o più interfacce, mostra che la classe realizza le operazioni offerte dall'interfaccia.
  - Nella modellazione UML la relazione, nella quale un elemento realizza il comportamento specificato dall'elemento fornitore, viene indicata da una freccia vuota con linea tratteggiata.
  - A partire da UML 2.0 è possibile visualizzare la relazione utilizzando anche i cosidetti "lecca-lecca", specificando il nome dell'interfaccia implementata.

Nome interfaccia

Realizzazione / 2

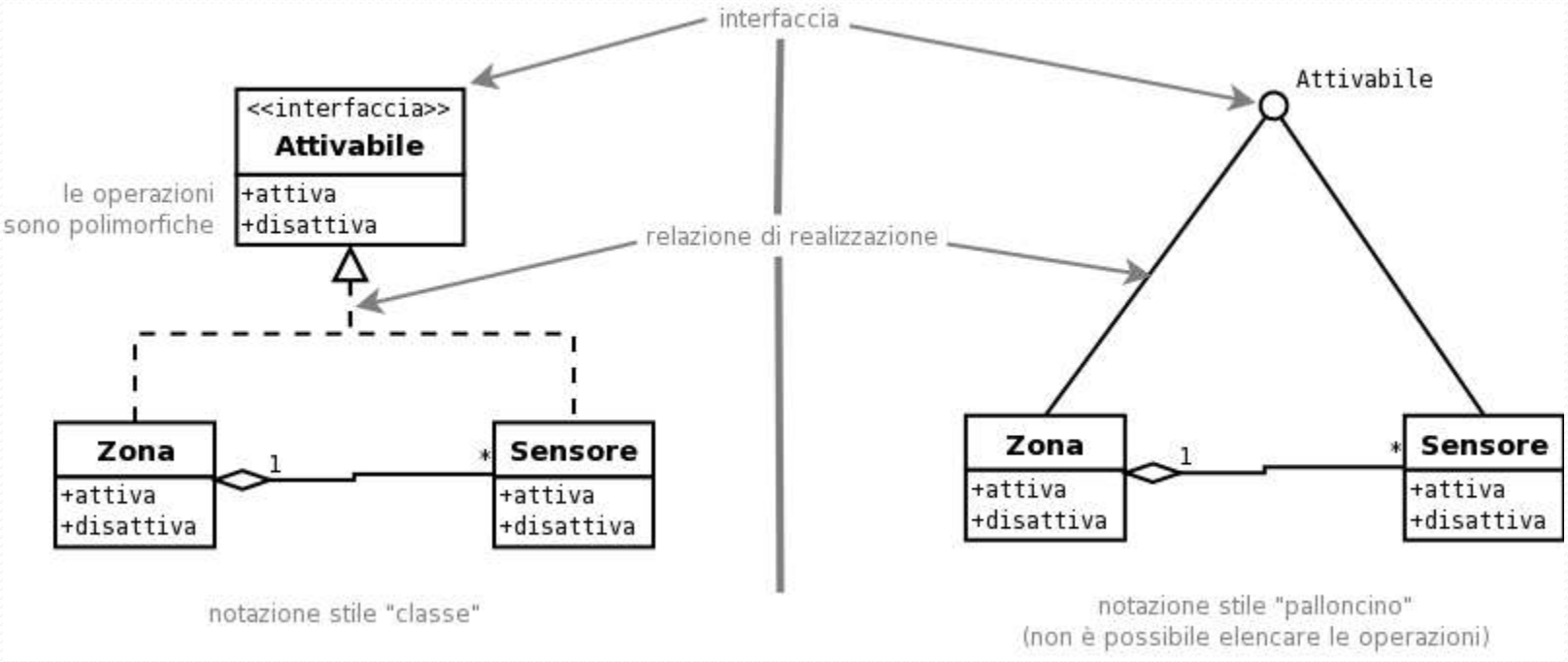


Diagramma di Stato

- Sono diagrammi di flusso orientati agli oggetti.
- Consentono di modellare un processo tramite
  - Stati
  - Transizioni tra stati
- Usati per modellare il ciclo di vita di una entità reattiva.

Diagramma di Stato / 2

- Descrive la macchina a stati di una singola entità

- Modella il comportamento dinamico di:
  - Classi
  - Casi d'uso
  - Sottosistemi
  - Sistemi interi

Stati e classi

- Per ogni classe può esistere una macchina a stati che modella tutte le transizioni di stato di tutti gli oggetti di quella classe, in risposta a diversi tipi di evento

- Gli eventi sono messaggi
  - Inviati da altri oggetti
  - Generati internamente

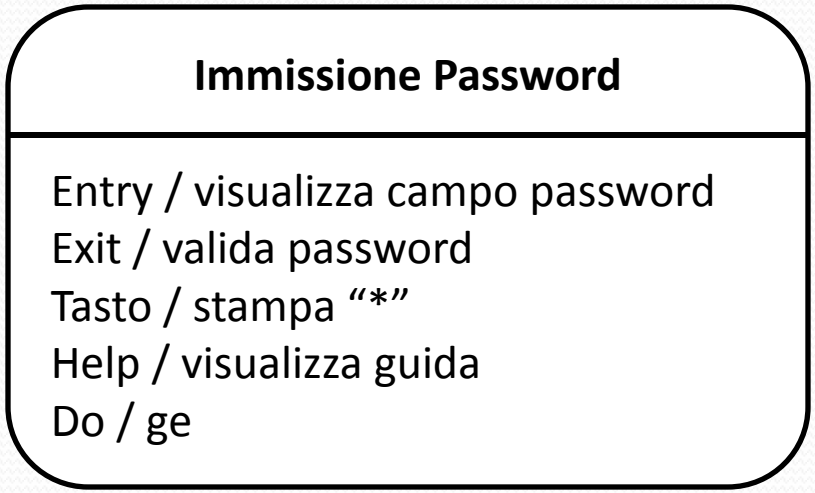
- La macchina a stati di una classe modella il comportamento degli oggetti della classe in modo trasversale per tutti i casi d'uso interessati

Sintassi

- Stato 
- Stato iniziale 
- Stato finale 
- Transizioni 
  - Gli eventi sono istantanei. Il nome dell'evento è scritto sulla freccia della transizione.

Stato

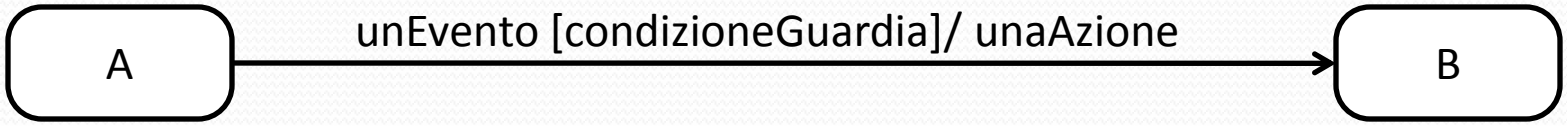
- Una condizione o una situazione della vita di un oggetto durante la quale tale oggetto soddisfa una condizione, esegue un‘attività o aspetta un qualche evento.
  - Deve esistere tra tali stati una differenza semantica



Azioni dello stato

- Uno stato può attivare delle azioni
  - **Entry** Attivata all'ingresso nello stato
  - **Do** Si svolge nel ciclo di vita dello stato
  - **Evento** Attivata in risposta ad un evento
  - **Exit** Attivata prima dell'uscita dallo stato
  - **Include** Chiama una macchina a stati inclusa (submachine)

Transizioni



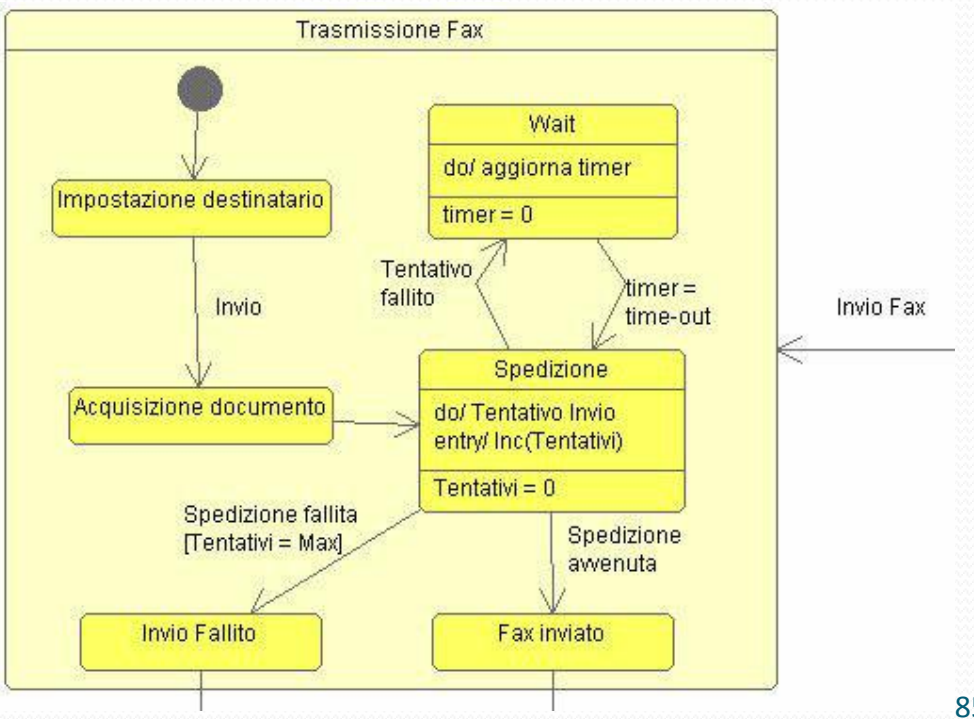
- Quando avviene *unEvento*, se la *condizioneGuardia* risulta vera, allora esegue *unaAzione*, quindi entra immediatamente nello stato B

Eventi

- Un **evento** è la specifica di un'occorrenza di interesse che ha una collocazione nel tempo e nello spazio.
  - **Evento di chiamata**
    - Richiesta di esecuzione di una serie di azioni
  - **Evento di segnale**
    - Ricezione di un segnale
  - **Evento di variazione**
    - Se si verificano particolari condizioni
  - **Evento temporale**
    - In momenti determinati

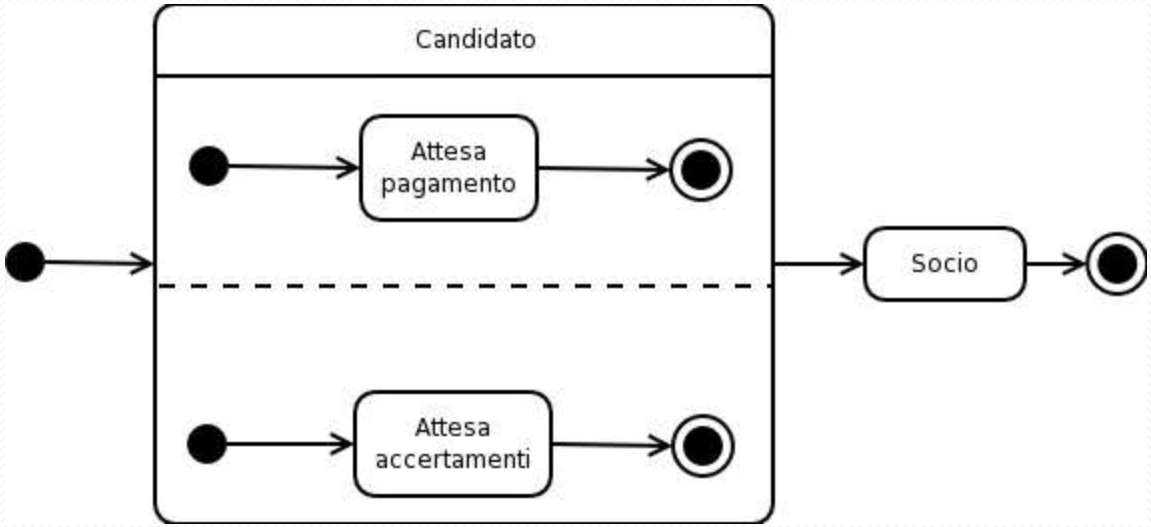
Stati composti

- Uno stato composto è uno stato che può essere ulteriormente spezzato in sottostati.
  - Ognuno dei sottostati può essere rappresentato da un diagramma di stato.



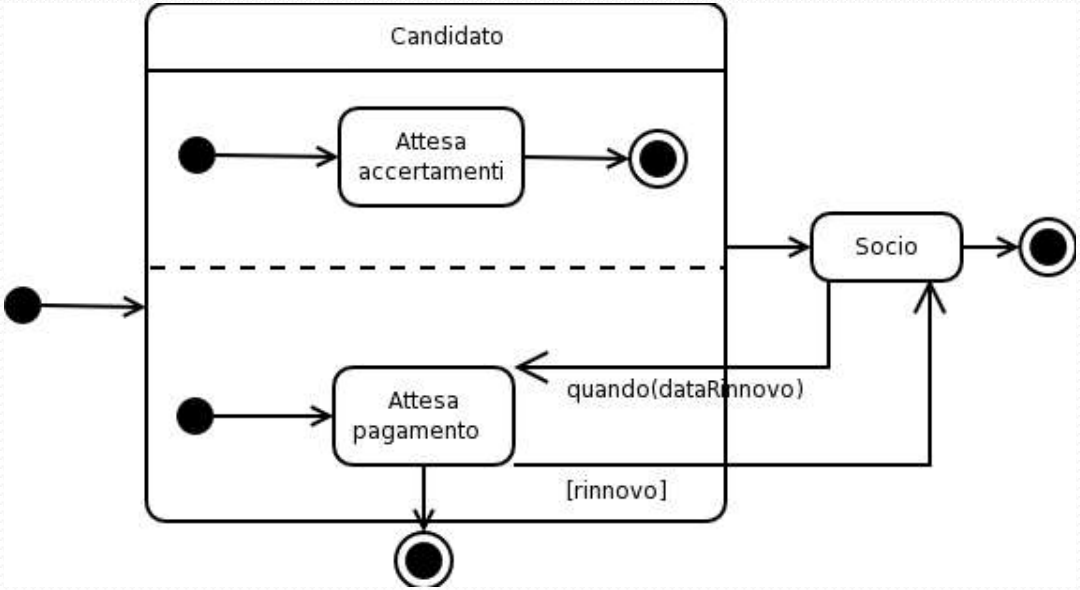
Stati concorrenti

- È possibile che uno stato composto possieda più stati **concorrenti**.



Stati concorrenti / 2

- Per indicare che, in fase di rinnovo, non si ha la necessità di ulteriori accertamenti legali, si modella una transizione con condizione verso lo stato interno relativo al pagamento.



Diagrammi di attività

- I diagrammi di attività sono un tipo speciale di diagramma di stato
  - Ogni stato ha un'azione di ingresso che specifica una procedura o funzione da eseguire quando si entra nello stato
  - Contengono stati di azione e stati di sottoattività
    - Gli stati di azione rappresentano attività che non possono essere scomposte in sottoattività
  - È possibile associare un diagramma di attività a qualunque elemento di modellazione, al solo fine di modellarne il comportamento.

Question time!