

10 ~~00~~. Verso la progettazione ad oggetti: modellazione dinamica e statica con UML

Sviluppo di Applicazioni Software

Matteo Baldoni

a.a. 2023/24

Università degli Studi di Torino - Dipartimento di Informatica

Attenzione!



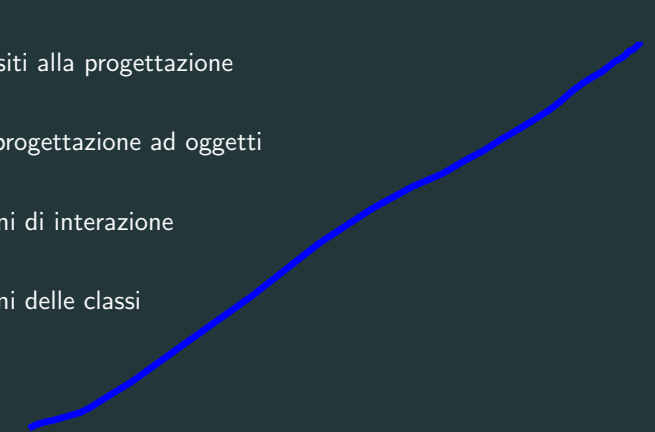
©2024 Copyright for this slides by Matteo Baldoni. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

<https://creativecommons.org/licenses/by/4.0/>

Si noti che

questi lucidi sono basati sul libro di testo del corso “C. Larman, *Applicare UML e i Pattern*, Pearson, 2016” e parzialmente sul materiale fornito da Viviana Bono, Claudia Picardi e Gianluca Torta dell’Università degli Studi di Torino.

Table of contents

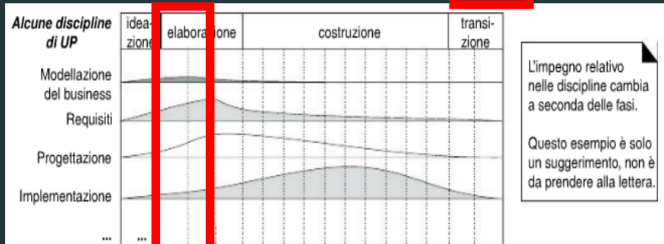
- 
1. Dai requisiti alla progettazione
 2. Verso la progettazione ad oggetti
 3. Diagrammi di interazione
 4. Diagrammi delle classi

Dai requisiti alla progettazione

UP maps

Tabella 2.1 Scenario di Sviluppo di esempio (i – inizio; r – raffinamento)

Disciplina	Pratica	Elaborato Iterazione →	Ideazione I1	Elaboraz. E1..En	Costr. C1..Cn	Transiz. T1..T2
Modellazione del business	modellazione agile workshop requisiti	Modello di Dominio		i		
Requisiti	workshop requisiti esercizio sulla visione votazione a punti	Modello dei Casi d'Uso	i	r		
		Visione Specifica	i	r		
		Supplementare	i	r		
Progettazione	modellazione agile sviluppo guidato dai test	Modello di Progetto		i	r	
		Documento dell'Architettura Software		i		
		Modello dei Dati		i	r	
	da test programmazione a coppie integrazione continua standard di codifica					
Gestione del progetto	gestione del progetto agile riunioni Scrum giornaliere	...				
...						



Dove siamo?

- Il 10% dei requisiti è stato esaminato durante l'ideazione
- Un'indagine un poco più approfondita è stata iniziata nella prima iterazione dell'elaborazione

Progettazione

Nei *requisiti* il fuoco è “*fare la cosa giusta*” ovvero capire alcuni degli obiettivi preminenti per i casi di studio e le relative regole e vincoli. Nella *progettazione* si pone l'accento sul “*fare la cosa bene*”, ovvero sul progettare abilmente una soluzione che soddisfa i requisiti per l'iterazione corrente.

Dove siamo?

- Il 10% dei requisiti è stato esaminato durante l'ideazione
- Un'indagine un poco più approfondita è stata iniziata nella prima iterazione dell'elaborazione

Progettazione

Nei *requisiti* il fuoco è “*fare la cosa giusta*”, ovvero **capire alcuni degli obiettivi preminenti per i casi di studio e le relative regole e vincoli**. Nella *progettazione* si pone l'accento sul “*fare la cosa bene*”, ovvero sul **progettare abilmente una soluzione che soddisfa i requisiti per l'iterazione corrente**.

Nota: **è naturale scoprire e modificare alcuni requisiti durante il lavoro di progettazione e di implementazione, soprattutto nelle iterazioni iniziali**. La programmazione dall'inizio, i test e le demo aiutano a provocare presto questi cambiamenti inevitabili. È lo scopo dello sviluppo iterativo.

Verso la progettazione ad oggetti

Progettare a oggetti

Come progettare a oggetti?

- **Codifica.** Progettare mentre si codifica.
- **Disegno, poi codifica.** Disegnare alcuni diagrammi UML, poi pasare alla codifica.
- **Solo disegno.** Lo strumento genera ogni cosa dai diagrammi.

Progettare a oggetti

Come progettare a oggetti?

- **Codifica**a. Progettare mentre si codifica.
- **Disegno, poi codifica**. Disegnare alcuni diagrammi UML, poi pasare alla codifica.
- **Solo disegno**. Lo strumento genera ogni cosa dai diagrammi.

Disegno leggero, con "disegno, poi codifica". Il costo aggiuntivo dovuto al disegno dovrebbe ripagare lo sforzo impiegato.

Modellazione agile: ridurre il costo aggiuntivo del disegno e modellare per comprendere e **comunicare, anziché per documentare**. Pratiche:

- Modellare insieme agli altri, modellazione in gruppo
- Creare diversi modelli in parallelo (sia *dinamici* che *statici*)

Modellazione statica e dinamica

Ci sono due tipi di modelli per gli oggetti:

- **dinamici**
- **statici**

Modellazione statica e dinamica

Ci sono due tipi di modelli per gli oggetti:

- **dinamici**
- **statici**

Modelli dinamici (es., diagrammi di interazione UML)

Rappresentano il comportamento del sistema, la collaborazione tra oggetti software per realizzare (uno/più scenari di) un caso d'uso, i metodi di classi software.

La modellazione a oggetti dinamica più comune è quella con i diagrammi di sequenza di UML.

Modellazione statica e dinamica

Ci sono due tipi di modelli per gli oggetti:

- **dinamici**
- **statici**

Modelli dinamici (es., diagrammi di interazione UML)

Rappresentano il comportamento del sistema, la collaborazione tra oggetti software per realizzare (uno/più scenari di) un caso d'uso, i metodi di classi software.

La modellazione a oggetti dinamica più comune è quella con i diagrammi di sequenza di UML.

Modelli statici (es., diagrammi di classe UML)

Servono per definire i package, i nomi delle classi, gli attributi, le firme di operazioni.

La modellazione a oggetti statica più comune è quella con i diagrammi delle classi di UML.

Modellazione statica e dinamica

Ci sono due tipi di modelli per gli oggetti:

- **dinamici**
- **statici**

Modelli dinamici (es., diagrammi di interazione UML)

Rappresentano il comportamento del sistema, la collaborazione tra oggetti software per realizzare (uno/più scenari di) un caso d'uso, i metodi di classi software.

La modellazione a oggetti dinamica più comune è quella con i diagrammi di sequenza di UML.

Modelli statici (es., diagrammi di classe UML)

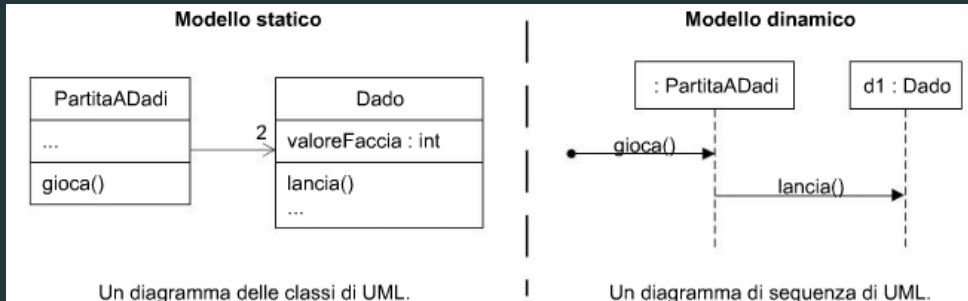
Servono per definire i package, i nomi delle classi, gli attributi, le firme di operazioni.

La modellazione a oggetti statica più comune è quella con i diagrammi delle classi di UML.

Sono tra loro relazionati ed è per questa ragione che si consiglia di crearli in parallelo.

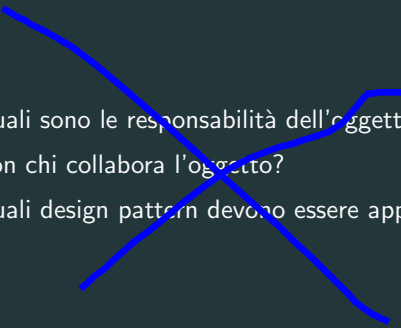
Modellazione statica e dinamica

- I messaggi nel diagramma di sequenza indicano operazioni nelle classi che ricevono il messaggio del diagramma delle classi.
- Le linee di vita nel diagramma di sequenza rappresentano oggetti di classi del diagramma delle classi



- La maggior parte del lavoro di progettazione difficile, interessante e utile avviene mentre si disegnano i diagrammi di interazione, che rappresentano una vista dinamica
- Durante la modellazione a oggetti dinamica si pensa in modo dettagliato e preciso a quali oggetti devono esistere e come questi collaborano attraverso messaggi e metodi
- Durante la modellazione dinamica si applica la progettazione guidata dalle **responsabilità** e i principi **GRASP**

Progettazione a oggetti e UML

- 
- Quali sono le responsabilità dell'oggetto?
 - Con chi collabora l'oggetto?
 - Quali design pattern devono essere applicati?

- Quali sono le responsabilità dell'oggetto?
- Con chi collabora l'oggetto?
- Quali design pattern devono essere applicati?

La progettazione a oggetti richiede soprattutto la conoscenza di:

- **principi di assegnazione di responsabilità**
- **design pattern**

Diagrammi di interazione

Diagrammi di interazione e modellazione dinamica

UML comprende i **diagrammi di interazione** per illustrare il modo in cui gli oggetti interagiscono attraverso lo scambio di messaggi.

I diagrammi di interazione sono utilizzati per la **modellazione dinamica** degli oggetti.

Interazione

Un'interazione **è una specifica di come alcuni oggetti si scambiano messaggi nel tempo** per eseguire un compito nell'ambito di un certo contesto.

Il termine diagramma di interazione è una generalizzazione dei tipi più specifici di diagrammi UML di **sequenza e di comunicazione**. Noi ci concentreremo su quelli di sequenza.

Un'interazione è motivata dalla necessità di eseguire un determinato compito.



Interazione

Un'interazione è motivata dalla necessità di eseguire un determinato compito.

Un compito è rappresentato da un messaggio che dà inizio all'interazione (messaggio trovato).

Interazione

Un'interazione è motivata dalla necessità di eseguire un determinato compito.

Un compito è rappresentato da un messaggio che dà inizio all'interazione (messaggio trovato).

Il messaggio è inviato a un oggetto designato come responsabile per questo compito.

Un'interazione è motivata dalla necessità di eseguire un determinato compito.

Un compito è rappresentato da un messaggio che dà inizio all'interazione (messaggio trovato).

Il messaggio è inviato a un oggetto designato come responsabile per questo compito.

L'oggetto responsabile collabora/interagisce con altri oggetti (partecipanti) per svolgere il compito.

Interazione

Un'interazione è motivata dalla necessità di eseguire un determinato compito.

Un compito è rappresentato da un messaggio che dà inizio all'interazione (messaggio trovato).

Il messaggio è inviato a un oggetto designato come responsabile per questo compito.

L'oggetto responsabile collabora/interagisce con altri oggetti (partecipanti) per svolgere il compito.

Ciascun partecipante svolge un proprio ruolo nell'ambito della collaborazione.

Un'interazione è motivata dalla necessità di eseguire un determinato compito.

Un compito è rappresentato da un messaggio che dà inizio all'interazione (messaggio trovato).

Il messaggio è inviato a un oggetto designato come responsabile per questo compito.

L'oggetto responsabile collabora/interagisce con altri oggetti (partecipanti) per svolgere il compito.

Ciascun partecipante svolge un proprio ruolo nell'ambito della collaborazione.

La collaborazione avviene mediante scambio di messaggi (interazione).

Un'interazione è motivata dalla necessità di eseguire un determinato compito.

Un compito è rappresentato da un messaggio che dà inizio all'interazione (messaggio trovato).

Il messaggio è inviato a un oggetto designato come responsabile per questo compito.

L'oggetto responsabile collabora/interagisce con altri oggetti (partecipanti) per svolgere il compito.

Ciascun partecipante svolge un proprio ruolo nell'ambito della collaborazione.

La collaborazione avviene mediante scambio di messaggi (interazione).

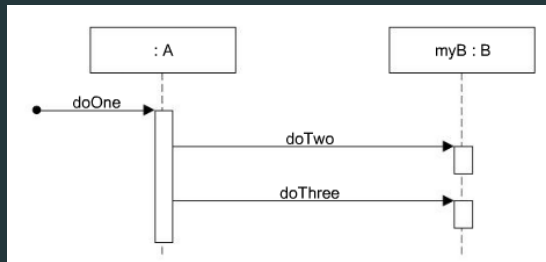
Ciascun messaggio è una richiesta che un oggetto fa a un altro oggetto di eseguire un'operazione.

Diagrammi di sequenza

I **diagrammi di sequenza** mostrano le interazioni in una specie di formato a stecchino, in cui gli oggetti che partecipano all'interazione sono mostrati in alto, uno a fianco all'altro.

Vantaggi: mostrano chiaramente la sequenza dell'ordinamento temporale dei messaggi.

Svantaggi: costringono a estendersi verso destra quando si aggiungono nuovi oggetti.



©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

```
public class A {
    private B myB = new B();

    public void doOne() {
        myB.doTwo();
        myB.doThree();
    }
    ...
}
```

©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

Diagrammi di sequenza

1. Il messaggio *makeCashPayment* viene inviato a un'istanza di *Register*. Il mittente non è identificato.
2. L'istanza di *Register* invia il messaggio *makeCashPayment* a un'istanza di *Sale*.
3. L'istanza di *Sale* crea un'istanza di *CashPayment*



©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

```
public class Sale {
    private CashPayment payment;

    public void makeCashPayment( Money cashTendered ) {
        payment = new CashPayment( cashTendered );
        ...
    }
    ...
}
```

©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

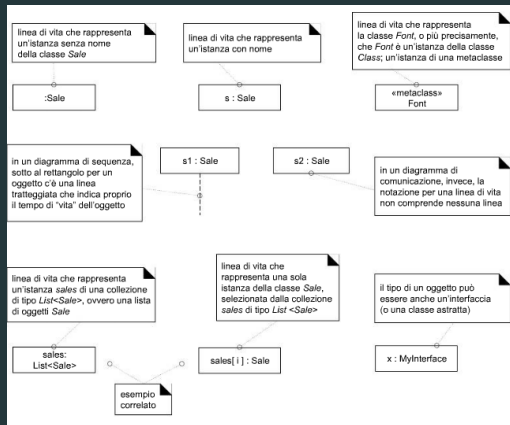
Design Sequence Diagram (DSD)

Il diagramma di sequenza di **progetto** è un diagramma di sequenza utilizzato da un punto di vista software o di progetto.

In UP, l'insieme di tutti i DSD fa parte del **Modello di Progetto** che comprende anche i diagrammi delle classi.

Diagrammi di sequenza: partecipanti

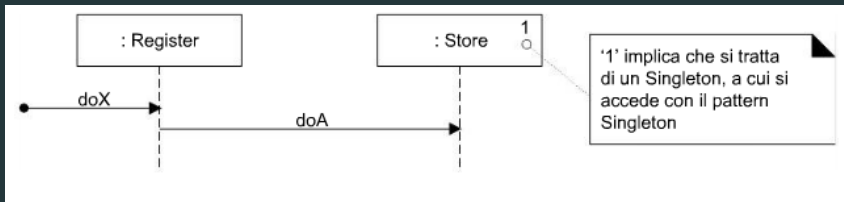
I rettangoli sono chiamati linee di vita (*lifeline*). Rappresentano i partecipanti all'interazione, ovvero le parti correlate definite nel contesto di un qualche diagramma strutturale.



Diagrammi di sequenza: singleton

Singleton: pattern nel quale da una classe viene istanziata una sola istanza, mai due o più.

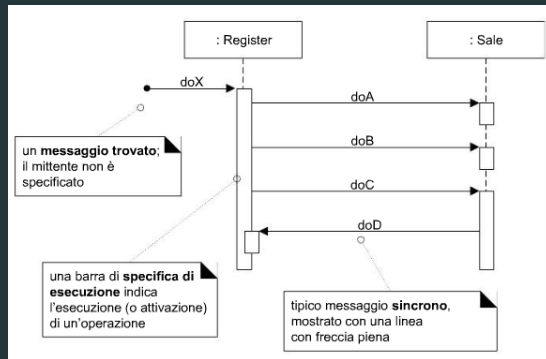
In un diagramma di interazione, un tale oggetto “singleton” è **contrassegnato da un '1'** nell'angolo superiore destro della linea di vita.



©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

Diagrammi di sequenza: linee di vita e messaggi

- Una **linea di vita** comprende sia un rettangolo che una linea verticale che si estende sotto di esso
- Ogni messaggio (di solito **sincrono**) tra gli oggetti è rappresentato da un'**espressione messaggio mostrata su una linea continua con una freccia piena tra le linee di vita verticali** (la punta sottile, non piena, indica un **messaggio asincrono**)
- Il messaggio iniziale è chiamato **messaggio trovato**
- Una barra di **specifica di esecuzione (o barra di attivazione o attivazione)** mostra l'esecuzione di un'operazione da parte di un oggetto

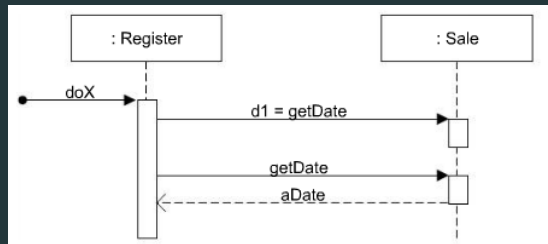


©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

Diagrammi di sequenza: risposte o ritorni

Ci sono due modi per mostrare il risultato di ritorno:

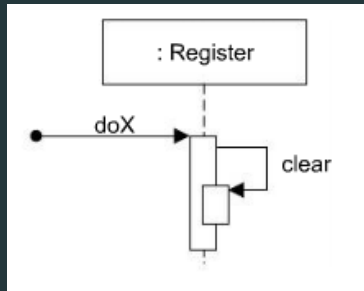
- utilizzando la sintassi *returnVar = message(parametri)*
- utilizzando una linea di messaggio di risposta (o ritorno) alla fine della barra di specifica di esecuzione



©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

Diagrammi di sequenza: self o this

È possibile mostrare un messaggio inviato da un oggetto a se stesso utilizzando una barra di specifica di esecuzione annidata.

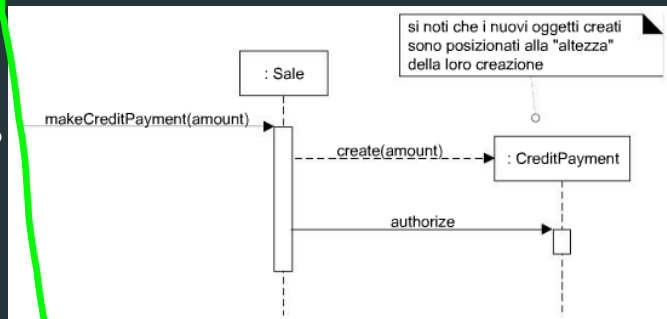


©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

Diagrammi di sequenza: creazione di istanze

È possibile mostrare un messaggio inviato da un oggetto a se stesso utilizzando una barra di specifica di esecuzione annidata.

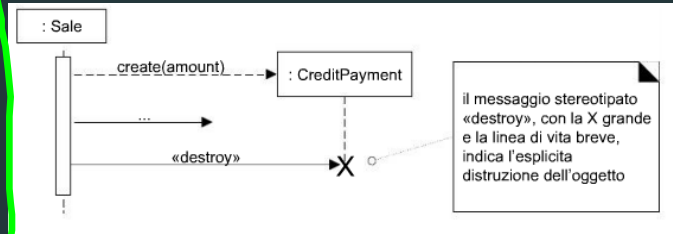
Si noti che i nuovi oggetti creati sono posizionati all'“altezza” della loro creazione.



©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

Diagrammi di sequenza: distruzione di oggetti

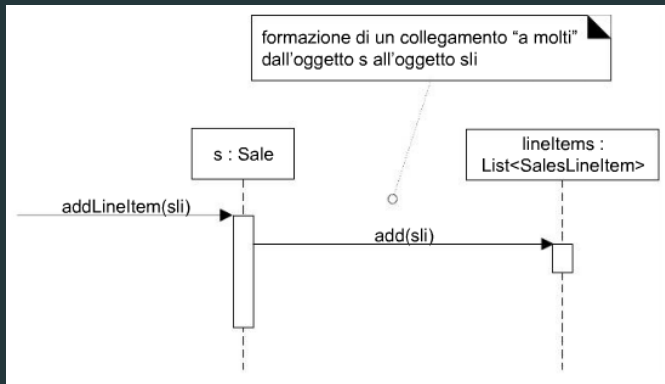
Una distruzione esplicita di un oggetto.



©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

Diagrammi di sequenza: formazione di collegamenti

Formazione di un collegamento di un'associazione "a molti".

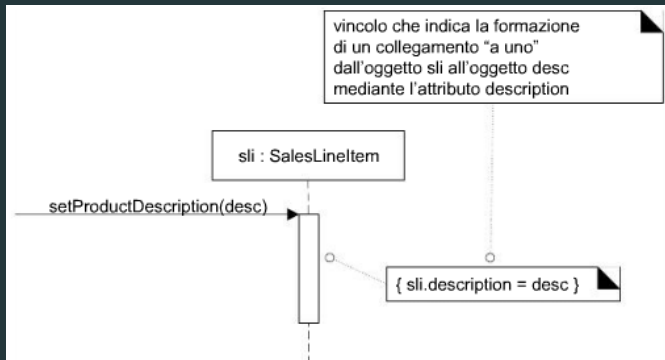


©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

Diagrammi di sequenza: formazione di collegamenti

Formazione di un collegamento di un'associazione “a uno”.

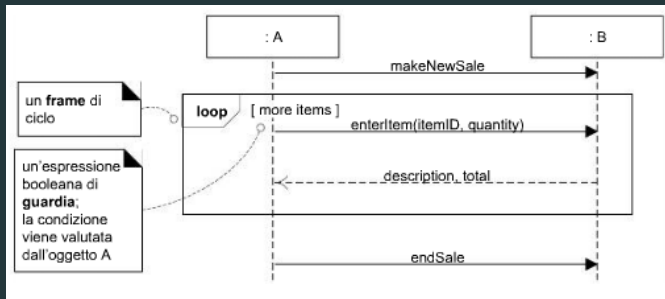
Il vincolo va interpretato come una post-condizione dell'operazione, ovvero una condizione che deve risultare vera al termine della sua esecuzione.



Diagrammi di sequenza: frame

Come supporto alle “istruzioni” di controllo condizionali e di ciclo si utilizza i frame.

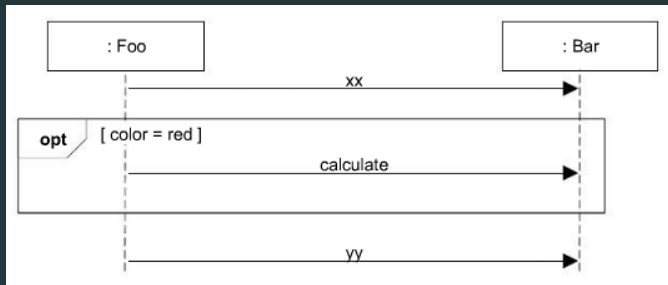
I frame sono regioni o frammenti dei diagrammi, hanno un operatore (etichetta) e una guardia (condizione o test booleano) che va posizionata sopra la linea di vita che deve valutare tale condizione.



©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

Diagrammi di sequenza: messaggi condizionali

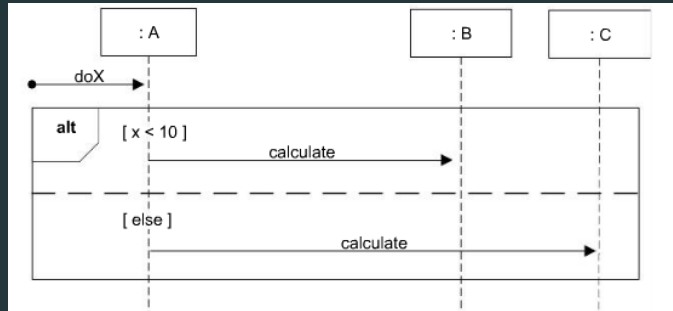
Un frame **OPT** è posizionato attorno a uno o più messaggi.



©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

Diagrammi di sequenza: messaggi condizionali mutuamente esclusivi

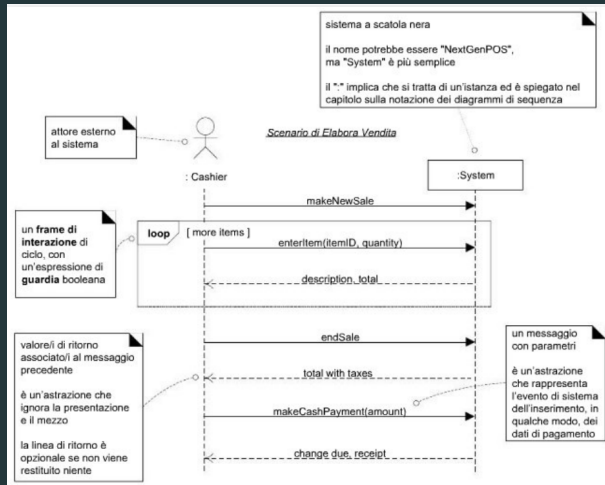
Un frame **ALT** è posizionato attorno alle alternative mutuamente esclusive.



©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

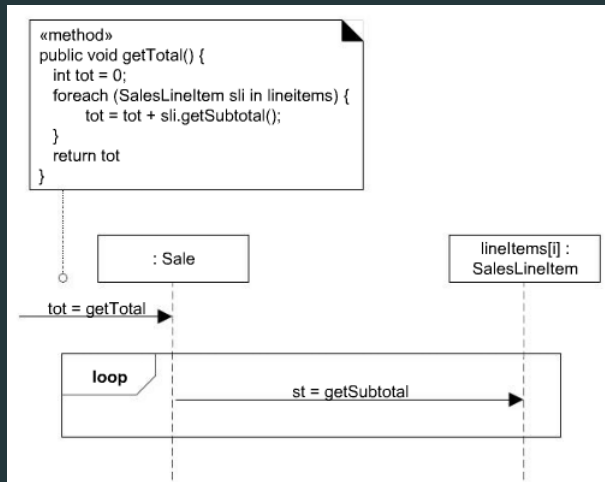
Diagrammi di sequenza: iterazione

Un frame **LOOP** è posizionato attorno a uno o più messaggi da iterare.



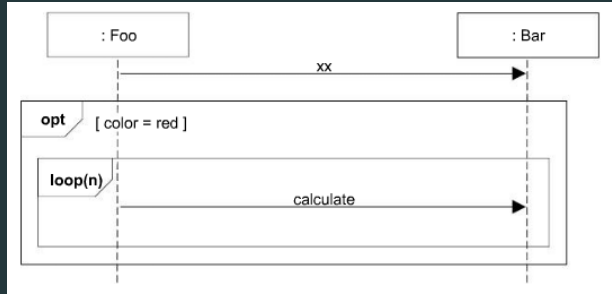
Diagrammi di sequenza: iterazione su una collezione

Un frame **LOOP** è posizionato attorno a uno o più messaggi da iterare.



Diagrammi di sequenza: annidamento di frame

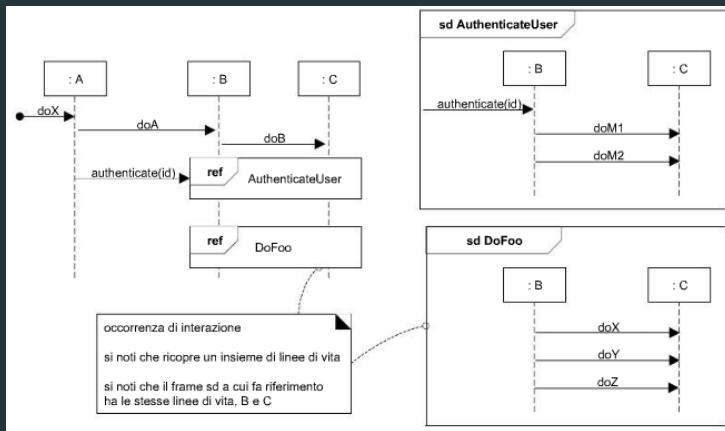
I frame possono essere annidati.



©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

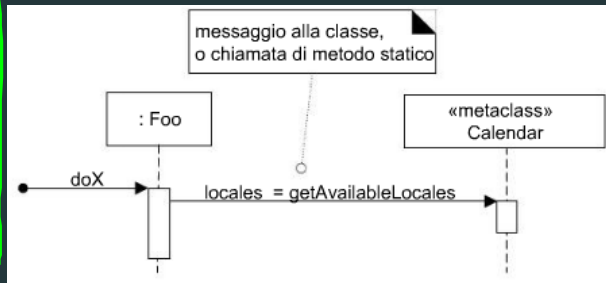
Diagrammi di sequenza: correlare diagrammi di interazione

Una occorrenza di interazione (o uso di interazione) è un riferimento a un'interazione all'interno di un'altra interazione che permette di correlare e collegare i relativi diagrammi.



Diagrammi di sequenza: invocare metodi statici

L'oggetto ricevente è una classe o, più precisamente, un'istanza di una **meta-classe**.



©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

Diagrammi di sequenza: chiamate sincrone e asincrone

Nota: la differenza tra le frecce è sottile. Non si dia per scontato che la forma della freccia sia corretta.

Oggetto attivo:

ciascuna istanza è eseguita nel proprio thread di esecuzione e lo controlla.

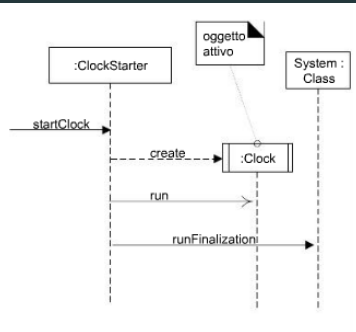
una freccia non piena indica una **chiamata asincrona** (diversamente da una freccia piena, che invece rappresenta una più comune chiamata sincrona)

in Java, per esempio, una chiamata asincrona può avvenire come segue:

```
// Clock implementa l'interfaccia Runnable  
Thread t = new Thread( new Clock() );  
t.start();
```

la chiamata di *start* implica poi una chiamata asincrona del metodo *run* dell'oggetto *Runnable* (*Clock*)

per semplificare il diagramma, l'oggetto *Thread* e il messaggio *start* si possono evitare (sono un "overhead" standard); al contrario, la chiamata asincrona è implicata dai dettagli essenziali della creazione di *Clock* e dal messaggio *run*



Diagrammi delle classi

Diagrammi delle classi e modellazione statica

UML comprende i **diagrammi delle classi** per illustrare le classi, le interfacce e le relative associazioni

I diagrammi delle classi sono utilizzati per la **modellazione statica degli oggetti**.

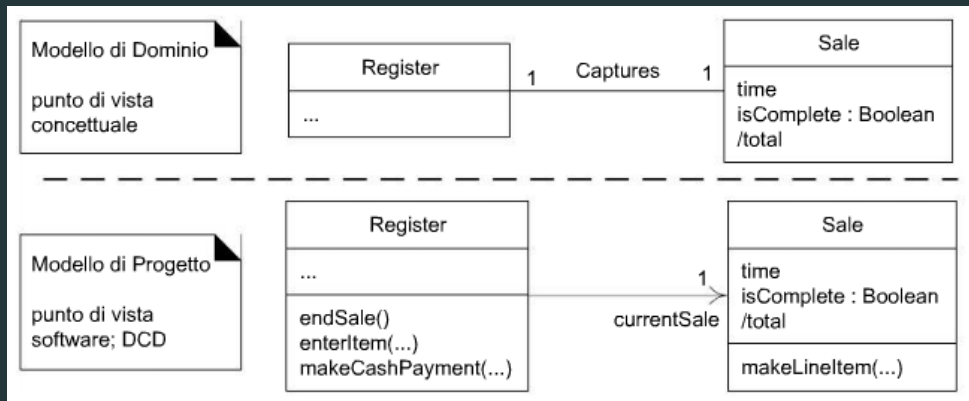
I diagrammi delle classi sono stati utilizzati, da un punto di vista concettuale, per **visualizzare un modello di dominio**.

Design Class Diagram (DCD)

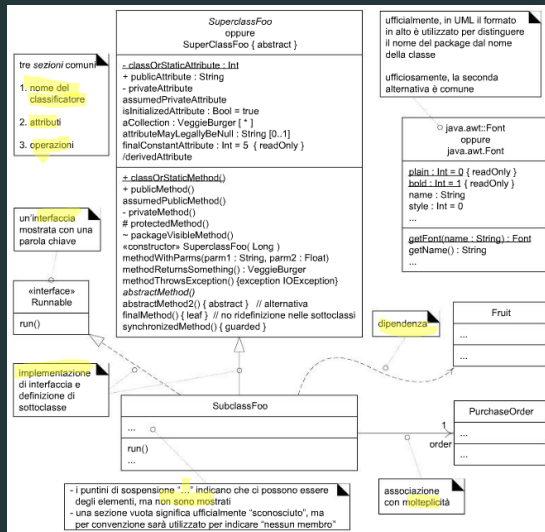
Il diagramma delle classi di progetto è **un diagramma delle classi utilizzato da un punto di vista software o di progetto**.

In UP, l'insieme di tutti i **DCD fa parte del Modello di Progetto che comprende anche i diagrammi di interazione**.

Diagrammi delle classi di UML dai due punti di vista

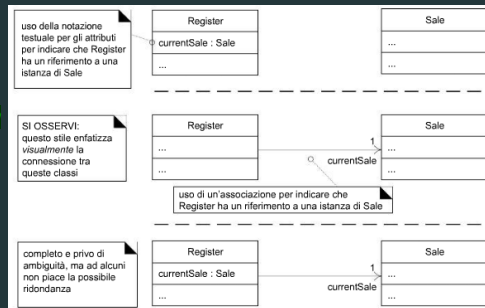


Notazione comune dei diagrammi delle classi di UML



Diagrammi delle classi: notazioni per attributi come associazioni

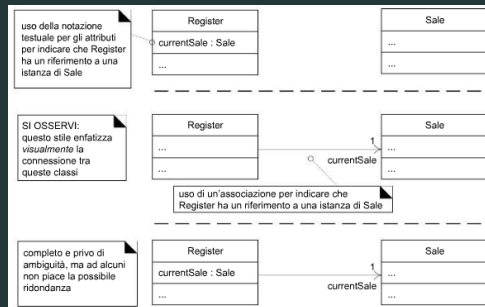
- Notazione testuale per un attributo (in alto)
- Notazione con una linea di associazione (in mezzo)
- Entrambe le notazioni, insieme (in basso): *non consigliata!*



©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

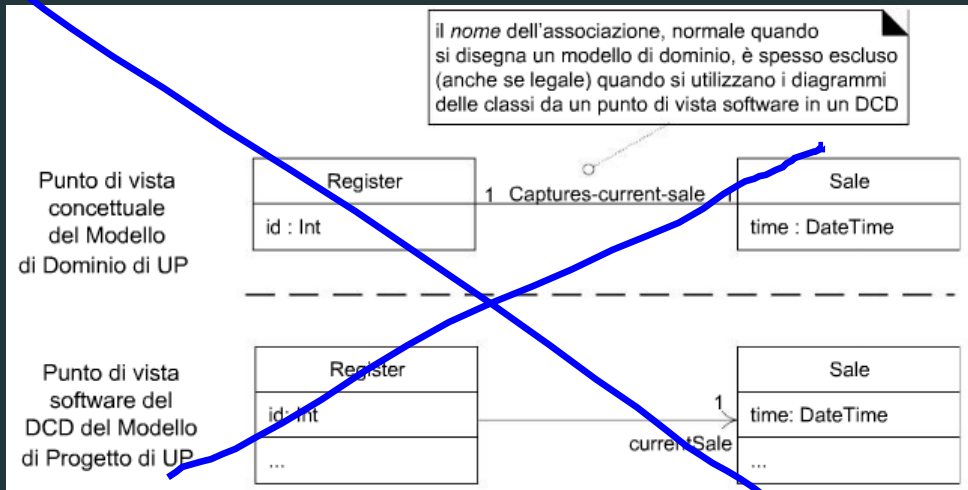
Diagrammi delle classi: notazioni per attributi come associazioni

- Se non viene indicata alcuna visibilità, solitamente si ipotizza che gli attributi siano **privati**
- Una freccia di navigabilità rivolta dalla classe sorgente alla classe destinazione dell'associazione, che indica che un oggetto della classe sorgente **ha un attributo di tipo della classe destinazione**
- Una molteplicità all'**estremità vicina alla destinazione**, ma non all'estremità vicina alla sorgente
- Un nome di ruolo solo all'**estremità vicina alla destinazione**, per indicare il nome dell'attributo
- **Nessun nome per l'associazione**



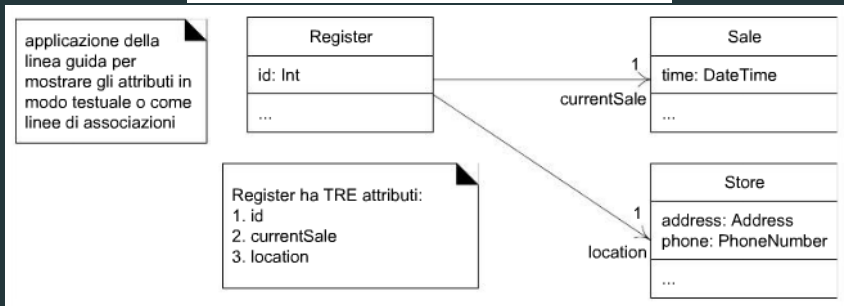
©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

Diagrammi delle classi: modello di dominio VS DCD



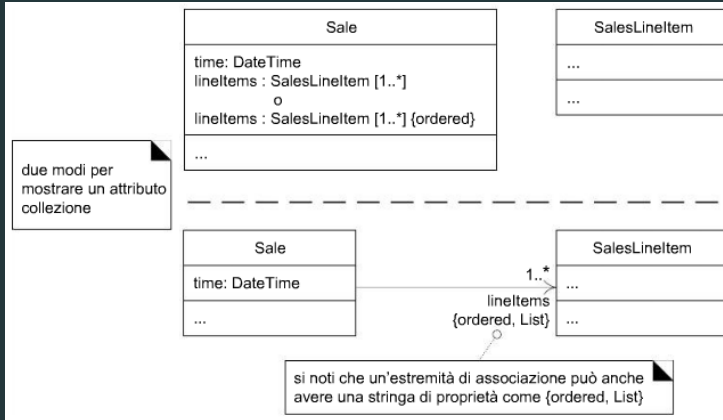
Diagrammi delle classi: associazioni e attributi

```
public class Register {  
    private int id;  
    private Sale currentSale;  
    private Store location;  
    ...  
}
```



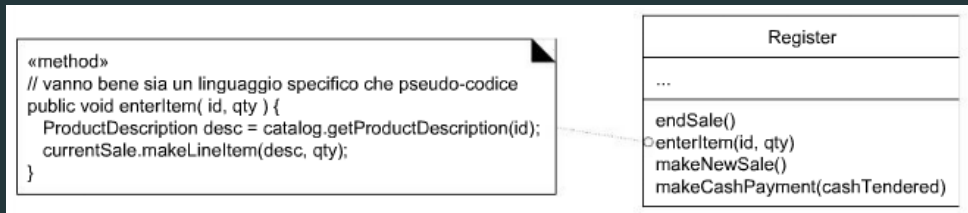
Diagrammi delle classi: attributi collezione e note

```
public class Sale {  
    private List<SalesLineItem> lineItems = new ArrayList<>();  
    ...  
}
```



Diagrammi delle classi: operazioni e metodi

- Un'operazione è una dichiarazione di un metodo, sintassi:
`visibility name (parameter-list) : return-type { property-string }`
- Per default, **le operazioni hanno visibilità pubblica**
- Nei diagrammi di classe vengono solitamente indicate le operazioni (**signature** – nomi e parametri)
- Nei diagrammi di interazione vengono modellati i metodi, come sequenze di messaggi



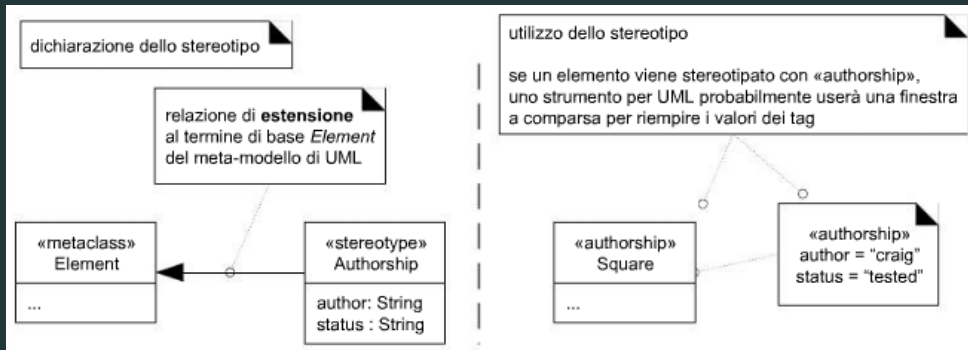
Diagrammi delle classi: parole chiave

- Decoratori testuali per classificare un elemento di un modello

Parola chiave	Significato	Esempio di uso
«actor»	il classificatore è un attore	nei diagrammi delle classi, sopra al nome di un classificatore
«interface»	il classificatore è un'interfaccia	nei diagrammi delle classi, sopra al nome di un classificatore
{abstract}	l'elemento è astratto; non può essere istanziato nome di un'operazione	nei diagrammi delle classi, dopo il nome di un classificatore o il
{ordered}	un insieme di oggetti ha un ordine predefinito	nei diagrammi delle classi, a un'estremità di associazione

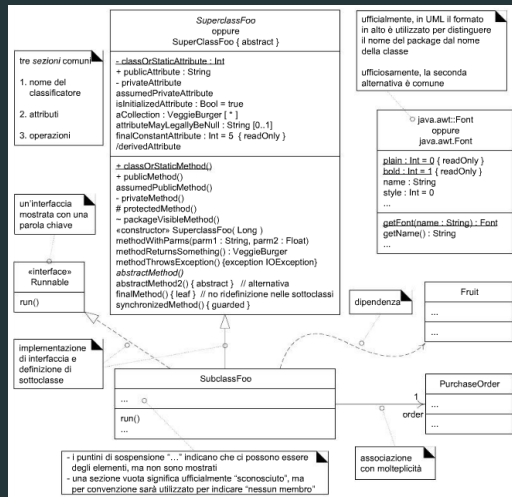
Diagrammi delle classi: stereotipi

- Gli stereotipi rappresentano un raffinamento di un concetto di modellazione esistente, ed è definito all'interno di un profilo UML (un profilo è una collezione di stereotipi)



Diagrammi delle classi: generalizzazione

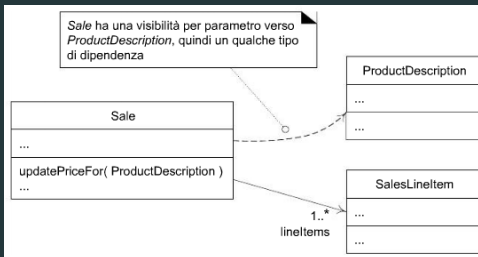
- È una relazione tassonomica tra un classificatore più generale e un classificatore più specifico. Ogni istanza del classificatore più specifico è anche un'istanza indiretta del classificatore più generale. Pertanto il classificatore più specifico possiede indirettamente le caratteristiche del classificatore più generale.
- La generalizzazione **implica l'ereditarietà nei linguaggi OO**
- Uso del tag **{abstract}** per le **classi astratte**



Diagrammi delle classi: dipendenze

- Le linee di dipendenza sono comuni nei diagrammi delle classi e dei package
- Una relazione di dipendenza indica che un elemento **cliente** è a conoscenza di un altro elemento **fornitore** e che un cambiamento nel fornitore potrebbe influire sul cliente (\equiv accoppiamento)

```
public class Sale {  
    public void updatePriceFor( ProductDescription desc ) {  
        Money basePrice = desc.getPrice();  
        ...  
    }  
    ...  
}
```

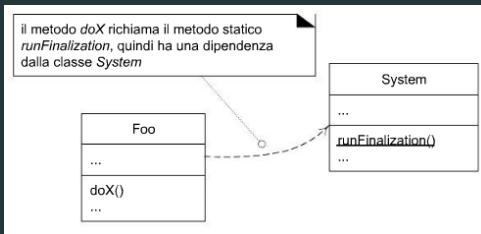


Diagrammi delle classi: dipendenze

Esistono molte tipologie di dipendenze:

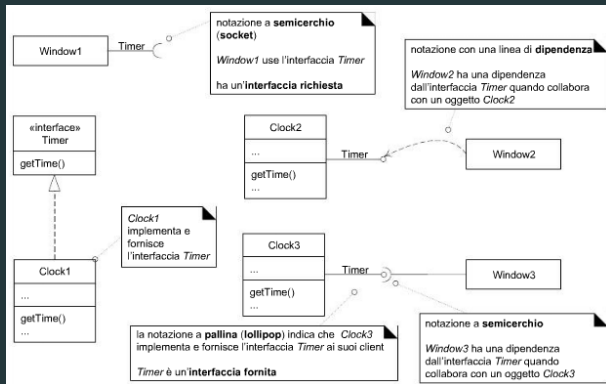
- Avere un attributo del tipo del fornitore
- Inviare un messaggio a un fornitore; la visibilità verso il fornitore potrebbe essere data da:
 - un attributo, una variabile parametro, una variabile locale, una variabile globale, o una visibilità di classe (chiamata di metodi statici o di classe)
- Ricevere un parametro del tipo del fornitore
- Il fornitore è una superclasse o un'interfaccia implementata

```
public class Foo {  
    public void doX() {  
        System.runFinalization();  
        ...  
    }  
    ...  
}
```



Diagrammi delle classi: interfacce

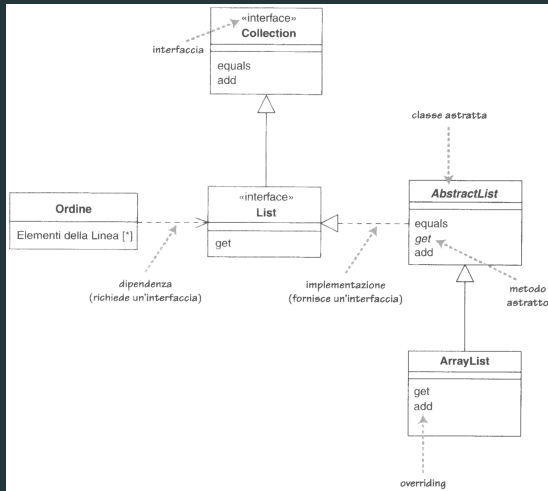
- L'implementazione di un'interfaccia viene chiamata una **realizzazione di interfaccia**
- La *notazione a pallina (lollipop)* indica che una classe X implementa (*fornisce*) un'interfaccia Y, senza disegnare il rettangolo per l'interfaccia Y
- La *notazione a semicerchio (socket)* indica che una classe X richiede (*usa*) un'interfaccia Y, senza disegnare una linea che punta all'interfaccia Y



©C. Larman. Applicare UML e i Pattern. Pearson, 2016.

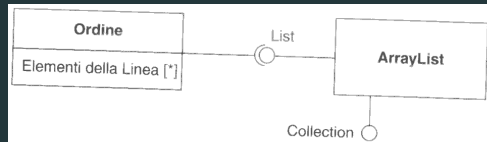
Diagrammi delle classi: interfacce

- L'implementazione di un'interfaccia viene chiamata una **realizzazione di interfaccia**
- La *notazione a pallina (lollipop)* indica che una classe X implementa (*fornisce*) un'interfaccia Y, senza disegnare il rettangolo per l'interfaccia Y
- La *notazione a semicerchio (socket)* (**socket**) indica che una classe X richiede (*usa*) un'interfaccia Y, senza disegnare una linea che punta all'interfaccia Y

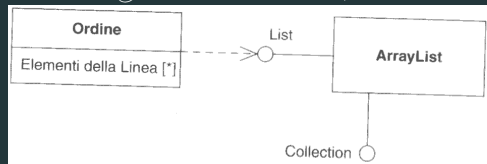


Diagrammi delle classi: interfacce

- L'implementazione di un'interfaccia viene chiamata una **realizzazione di interfaccia**
- La *notazione a pallina (lollipop)* indica che una classe X implementa (*fornisce*) un'interfaccia Y, senza disegnare il rettangolo per l'interfaccia Y
- La *notazione a semicerchio (socket)* indica che una classe X richiede (*usa*) un'interfaccia Y, senza disegnare una linea che punta all'interfaccia Y



©M. Fowler. UML Distilled. Pearson, 2018.

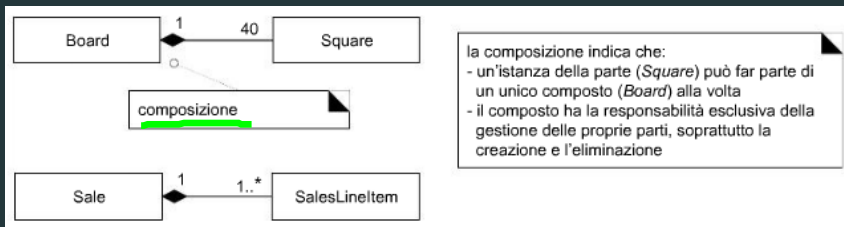


©M. Fowler. UML Distilled. Pearson, 2018.

Diagrammi delle classi: **composizione**

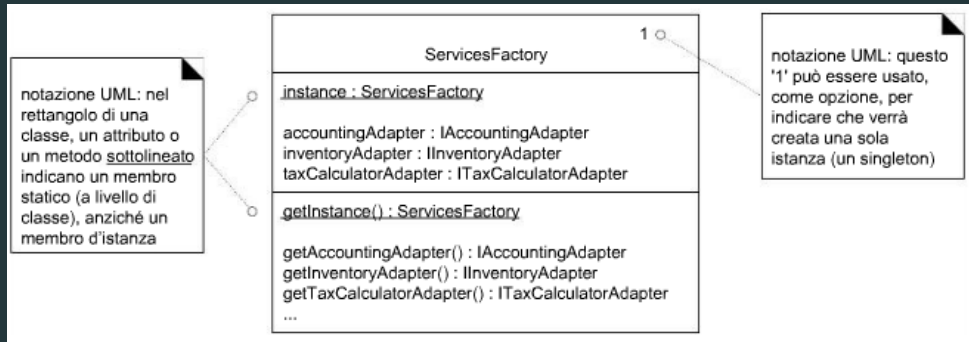
Una possibile interpretazione (dal punto di vista software) di una composizione tra le classi A e B è la seguente:

- Gli oggetti B non possono esistere indipendentemente da un oggetto A
- L'oggetto A è responsabile della creazione e distruzione dei suoi oggetti B



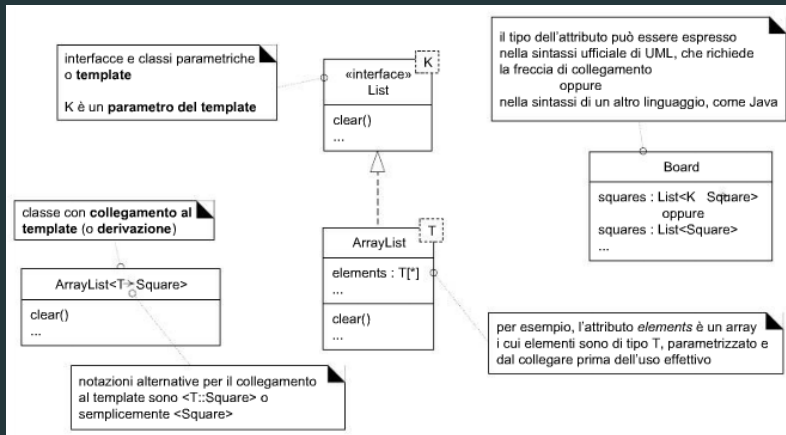
Diagrammi delle classi: singleton

Esiste **una sola istanza di una classe, mai due.**



Diagrammi delle classi: template

Molti linguaggi supportano **tipi a template, noti anche come template, tipi parametrizzati e generici**.



Diagrammi delle classi e diagrammi di interazione/sequenza

L'influenza dei diagrammi di interazione sui diagrammi delle classi.

