



UNIVERSITÀ DEGLI STUDI DI MILANO
DIPARTIMENTO DI INFORMATICA

Corso di Progettazione model-driven del software (Ing SW - Mod 2)

Laboratorio 2 - Class e Sequence Diagram

`chiara.braghin@unimi.it`

1

Introduzione al laboratorio

2

Organizzazione dei laboratori (1)

24 ore in 6 settimane => 4 ore settimanali

- 2 hr su **Zoom** (solo io): ripasso, "pillole" di teoria e esercizio caso di studio svolto insieme
 - **Lezione SINCRONA, REGISTRATA (ricordatemelo!!)**
 - **ORARIO:** giovedì, ore 11:30-13:30 (si inizia puntuali)
 - Registrazioni disponibili su Ariel
 - Link: <https://zoom.us/j/94255671991?pwd=TkFzSmpHNG5uSXd6akZ5Y1RnNDFOU09>
- 2 hr su **Discord** (con i tutor): esercizi che gli studenti svolgono individualmente o in gruppo
 - **Lezione SINCRONA, NON REGISTRATA**
 - **ORARIO:** giovedì, ore 14:30-16:30 (si inizia puntuali)

3

Organizzazione dei laboratori (2)

Settimana	Argomenti
WEEK 1 – 9-13/11/20	Diagrammi dei casi d'uso per l'analisi dei requisiti (scenari e storie), specifica dei requisiti
WEEK 2 – 16-20/11/20	Diagrammi di sequenza vs diagrammi di attività
WEEK 3 – 23-27/11/20	Macchine di stato vs diagrammi attività
WEEK 4 – 30/11- 4/12/20	Java e persistenza dei dati (JDBC, Hibernate), Java Swing
WEEK 5 – 7-11/12/20	Design pattern MVC e DAO
WEEK 6 – 14-18/12/20	Altri design pattern Diagramma delle componenti e di deployment Dai modelli al codice

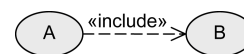
4

ANCORA DIAGRAMMI DEI CASI D'USO

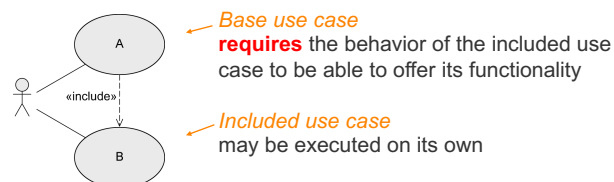
5

Relationships between Use Cases

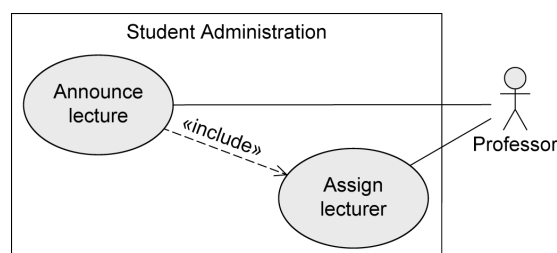
«include» - Relationship



- The behavior of one use case (included use case) is integrated in the behavior of another use case (base use case)



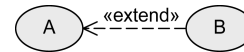
- Example:



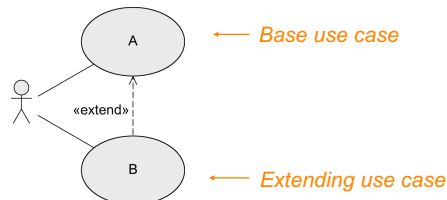
6

Relationships between Use Cases

«extend» - Relationship



- The behavior of one use case (extending use case) may be integrated in the behavior of another use case (base use case) but does not have to.
- Both use cases may also be executed independently of each other.



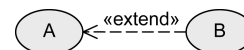
- A decides if B is executed.
- Extension points define at which point the behavior is integrated.
- Conditions define under which circumstances the behavior is integrated.



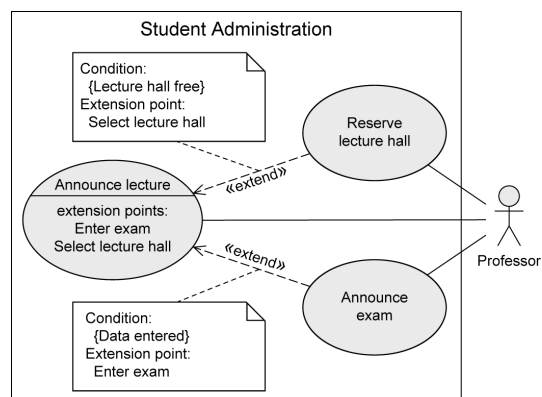
7

Relationships between Use Cases

«extend» - Relationship: Extension Points



- Extension points are written directly within the use case.
- Specification of multiple extension points is possible.
- Example:



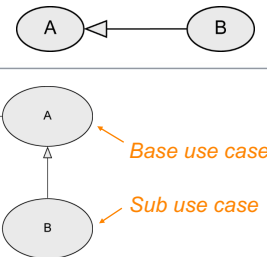
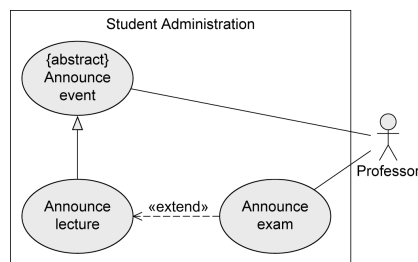
8

Relationships between Use Cases

Generalization of Use Cases

- Use case **A** generalizes use case **B**.
- B** inherits the behavior of **A** and may either extend or overwrite it.
- B** also inherits all relationships from **A**.
- B** adopts the basic functionality of **A** but decides itself what part of **A** is executed or changed.
- A** may be labeled **{abstract}**
 - Cannot be executed directly
 - Only **B** is executable
- Example:

BIG © BIG / TU Wien



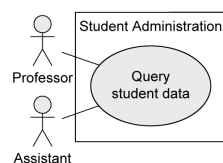
9

Relationships between Actors

Generalization of Actors

- Actor **A** inherits from actor **B**.
- A** can communicate with **X** and **Y**.
- B** can only communicate with **Y**.
- Multiple inheritance is permitted.
- Abstract actors are possible.

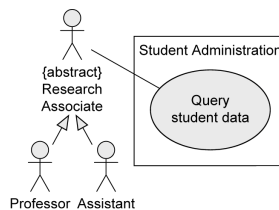
- Example:



Professor AND Assistant needed for executing **Query student data**

BIG © BIG / TU Wien

≠



Professor OR Assistant needed for executing **Query student data**

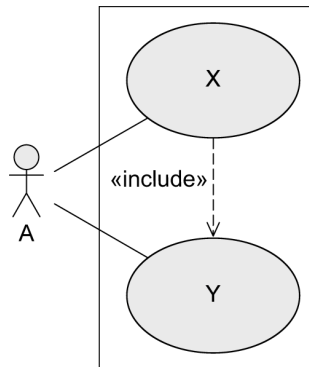


10

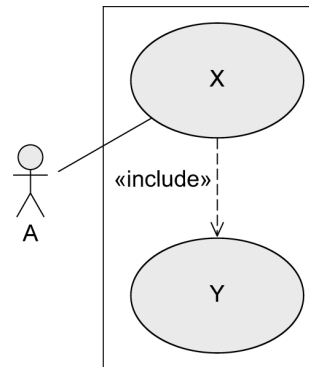
Best Practices

«include»

UML standard



Best practice



© BIG / TU Wien

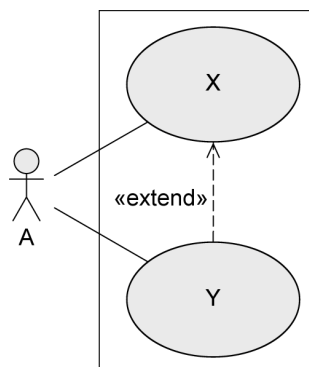


11

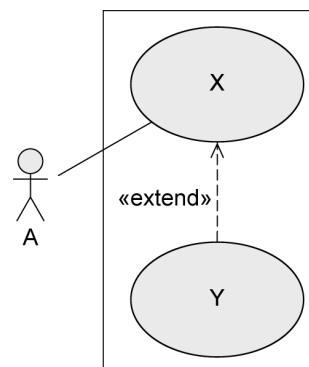
Best Practices

«extend»

UML standard



Best practice



© BIG / TU Wien

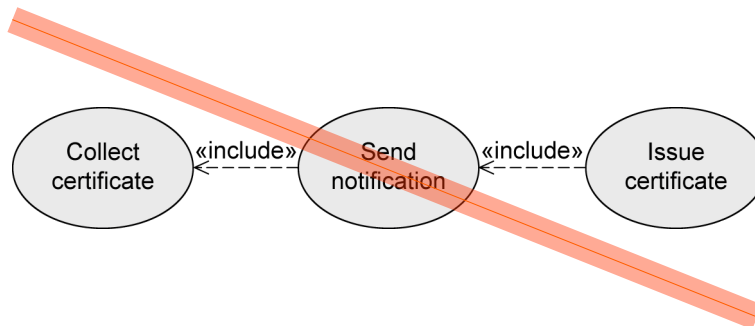


12

Best Practices

Typical Errors To Avoid (1/5)

- Use case diagrams do not model processes/workflows!

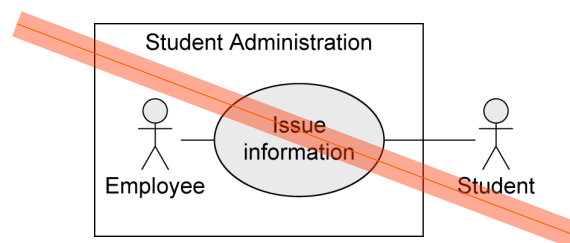


13

Best Practices

Typical Errors To Avoid (2/5)

- Actors are not part of the system, hence, they are positioned outside the system boundaries!

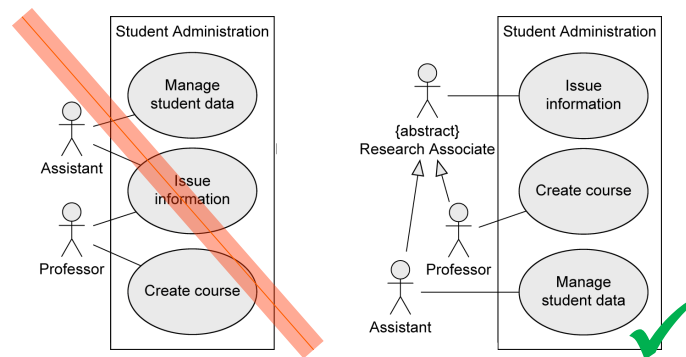


14

Best Practices

Typical Errors To Avoid (3/5)

- Use case **Issue information** needs **EITHER** one actor **Assistant** **OR** one actor **Professor** for execution



© BIG / TU Wien

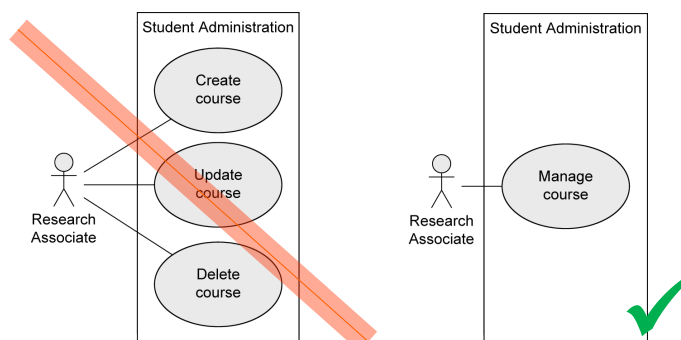


15

Best Practices

Typical Errors To Avoid (4/5)

- Many small use cases that have the same objective may be grouped to form one use case



© BIG / TU Wien

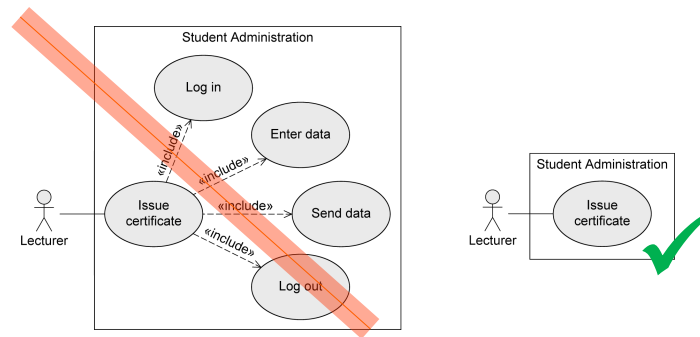


16

Best Practices

Typical Errors To Avoid (5/5)

- The various steps are part of the use cases, not separate use cases themselves! -> NO functional decomposition



17

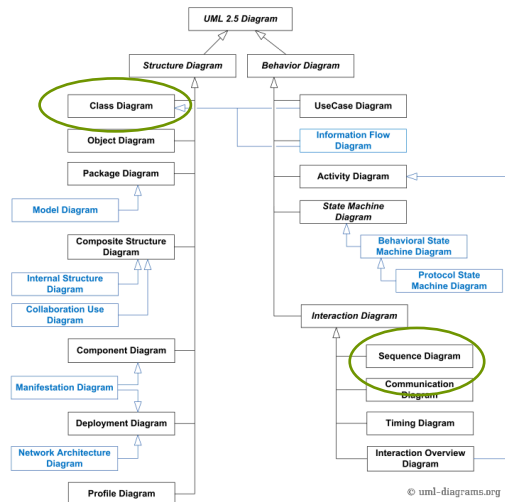
LAB 2

DIAGRAMMI DELLE CLASSI E DI SEQUENZA

18

Dove siamo?

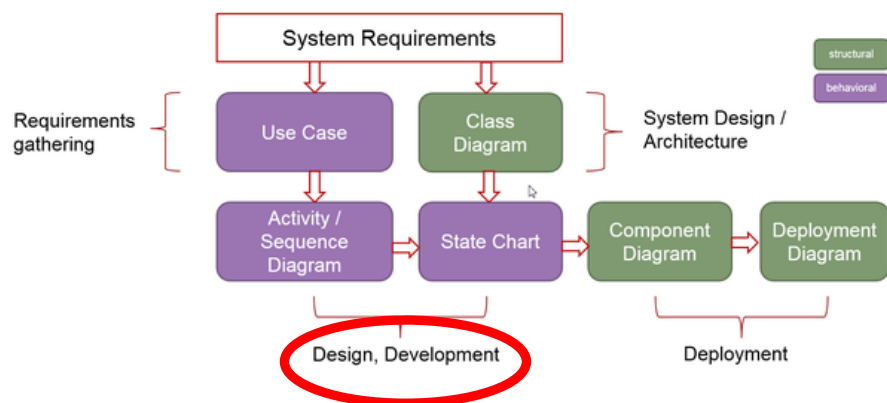
UML a supporto di uno sviluppo del SW model-driven



19

Ripasso (2)

Models: Which and when?



20

DIAGRAMMA DELLE CLASSI

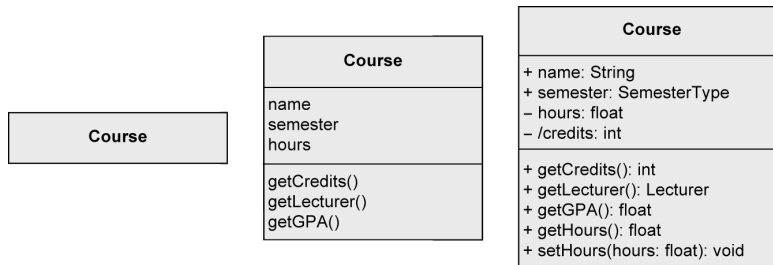
21

Diagramma delle classi

- Rappresenta le **classi** che compongono il sistema, cioè le collezioni di oggetti, ciascuno con il proprio **stato** e **comportamento** (attributi ed operazioni)
- Specifica, mediante **associazioni**, le relazioni fra le classi

22

Specification of Classes: Different Levels of Detail



© BIG / TU Wien



23

Come estrarre le classi? (1)

FASE 1: MODELLO DI DOMINIO (classi di analisi)

- Una *classe di analisi* modella un concetto o entità del problema: se la specifica dei casi d'uso è buona i concetti basilari sono già in evidenza
- Di sicuro non è possibile estrarre classi, attributi e metodi in modo completo (e automatico) da un testo
- **NB:** Le classi di analisi non sopravvivranno necessariamente alla progettazione.

24

Come estrarre le classi? (2)

Metodi:

- **Analisi nome-verbo:**
 - Si analizza tutta la documentazione disponibile, selezionando *nomi*, *aggettivi* e *verbi*:
 - **Nomi:** spesso potenziali candidati per **classi**
 - **Aggettivi:** spesso corrispondono ad **attributi**
 - **Verbi:** spesso corrispondono a **metodi/operazioni o responsabilità di classe**
- **Analisi CRC (Class-Responsibilities-Collaborators):**
 - post-it divisi in tre sezioni per ognuna delle 3 categorie
 - si individuano:
 - i nomi delle classi
 - un insieme ristretto di responsabilità (cose che la classe sa/fa)
 - Un insieme di classi collaboratori (alle quali viene richiesto comportamento/informazione).

25

Come estrarre le classi? (3)

FASE 2: DIAGRAMMA DELLE CLASSI

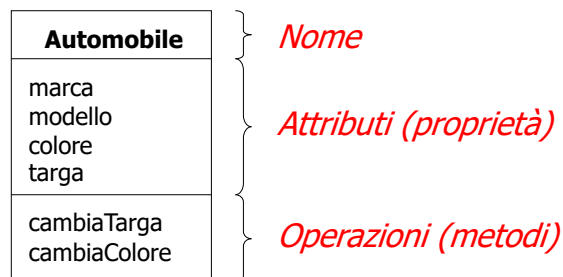
- **Identificare le classi**
- **Inserire gli attributi della classe indicando *nome*, *tipo* e *se*:**
 - Pubblico +
 - Privato –
 - Protetto #
 - Ristretto al package ~
- **Inserire i metodi indicando il *nome*, i *parametri* e il *tipo* di ritorno**
- **Aggiungere le associazioni indicando il *nome* e la *molteplicità*.**
 - Anche *ruolo*, *visibilità* e *navigabilità*.

26

Class diagram: classi (1)

- Una classe è una tipologia di oggetti, con propri attributi e operazioni

Rappresentazione di una classe in UML:



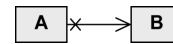
27

Class diagram: classi (2)

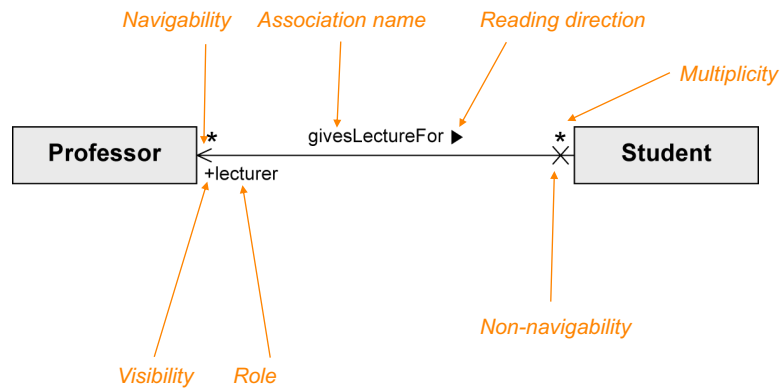
- **Nome:** inizia con una lettera maiuscola, non è sottolineato e non contiene underscore
- **Attributi:** proprietà i cui valori identificano un oggetto istanza della classe e ne costituiscono lo stato; iniziano con una lettera minuscola
- **Operazioni/Metodi:** insieme di funzionalità che esprimono il comportamento di un oggetto, ovvero ciò che ogni oggetto di questa classe può fare

28

Binary Association



- Connects instances of two classes with one another

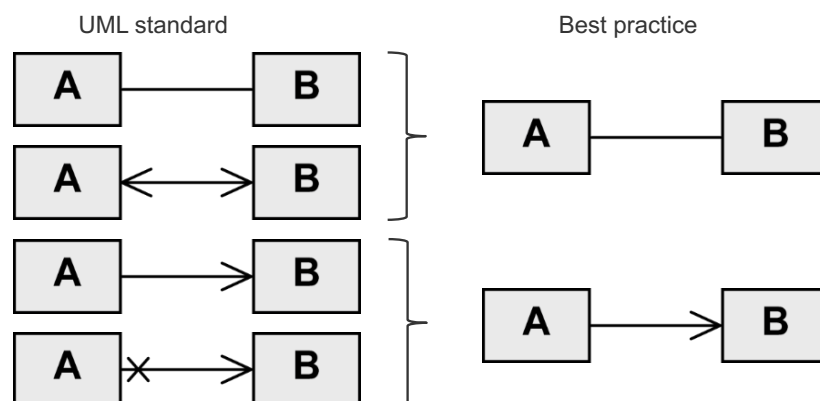


© BIG / TU Wien



29

Navigability – UML Standard vs. Best Practice



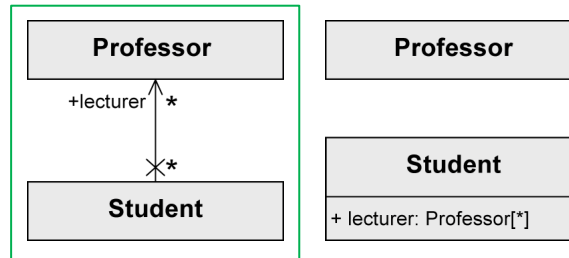
© BIG / TU Wien



30

Binary Association as Attribute

Preferable



- Java-like notation:

```

class Professor {...}

class Student{
    public Professor[]
    lecturer;
    ...
}
  
```



© BIG / TU Wien



31

Class diagram: aggregazione

- **Aggregazione:** tipo particolare di associazione
 - esprime concetto "è parte di" (*part of*), che si ha quando un insieme è relazionato con le sue parti

32

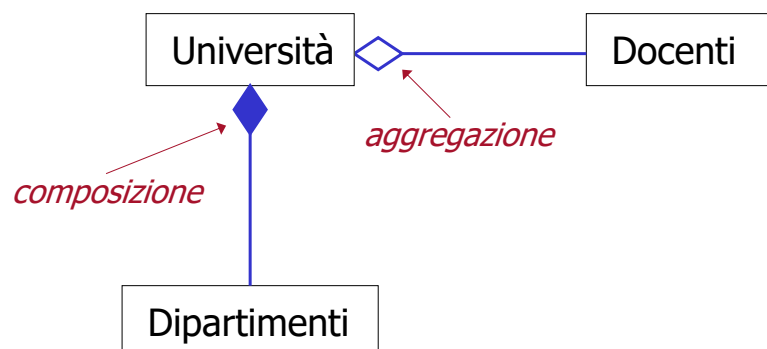
Class diagram: composizione

E' un caso particolare di aggregazione in cui:

- la parte (componente) **non** può esistere da sola, cioè senza la classe composto
- una componente appartiene ad un solo composto

33

Class diagram: aggregazione/composizione



34

Diagrammi di interazione

- Diagrammi che descrivono la *cooperazione tra un insieme di oggetti* che collaborano per realizzare un fine comune

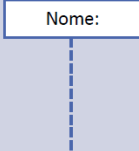
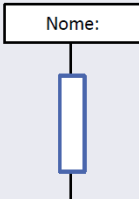
DIAGRAMMI DI SEQUENZA

Scopi:

- *strumento di progettazione*: metodo alternativo per documentare il comportamento **di un singolo scenario** (**uno stesso caso d'uso**).
- *strumento di convalida*: realizzati utilizzando le classi per convalidare la corretta identificazione

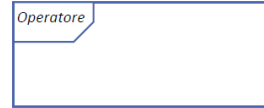
35

Elementi: Nodi

Costruttore	Descrizione	Sintassi
Linea di vita / Lifeline	Esistenza dell'oggetto ad un particolare istante	
Barra di attivazione	Periodo durante il quale l'oggetto esegue un'azione	

36

Elementi: cicli, condizioni,...



Operatore	Significato
alt	Identifica due blocchi eseguibili in alternativa
opt	Il blocco viene eseguito solo se la condizione è vera
par	Ogni frammento è eseguito in parallelo
loop	Il blocco viene seguito più volte in base alla guardia
region	Il blocco può essere eseguito da un solo thread alla volta
neg	Il blocco mostra un'iterazione non valida (non deve mai essere eseguito)
ref	Si riferisce a un'iterazione definita in un altro diagramma
sd	Racchiude un intero diagramma di sequenza

37

Elementi: Messaggi

Costruttore	Descrizione	Sintassi
Messaggio Sincrono	Procedure call o altro tipo di flusso di controllo annidato. Deve aspettare la risposta prima di procedere.	
Messaggio Asincrono	Comunicazione Asincrona. Non c'è bisogno di aspettare la risposta per procedere.	
Risposta	Ritorna una risposta	

38

Elementi: Messaggi

Costruttore	Descrizione	Sintassi
Messaggio di creazione	Crea un nuovo nodo	
Messaggio di distruzione	Distrugge un nodo	

39

Diagrammi di interazione (2)

Quando usare i diagrammi di sequenza?

- *Per descrivere interazione tra attori/oggetti/classi*
 - *A volte sono di aiuto per identificare le classi*
- *NON per descrivere comportamento (strutture di controllo)*
 - *Activity diagram*

40