



DIAGRAMMI DELLE CLASSI

INGEGNERIA DEL SOFTWARE

Università degli Studi di Padova

Dipartimento di Matematica

Corso di Laurea in Informatica

rcardin@math.unipd.it

SOMMARIO

- Introduzione
- Proprietà e Operazioni
- Concetti base e avanzati
- Diagrammi degli oggetti

Ingegneria del software

Riccardo Cardin

2

SOMMARIO

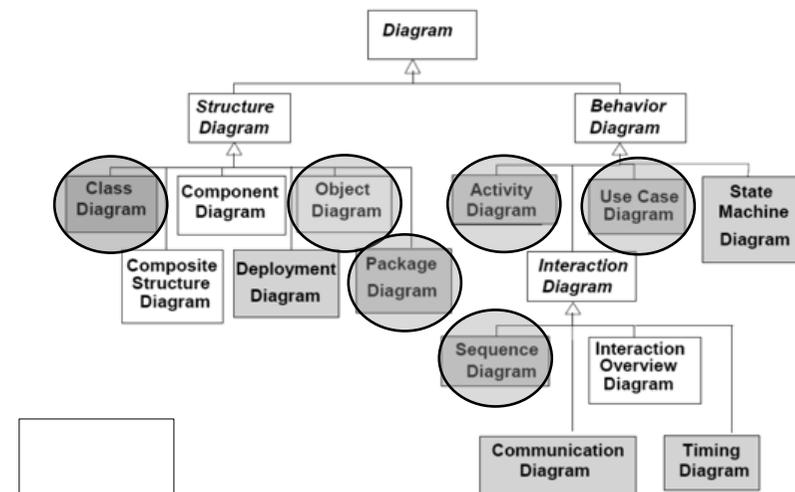
- Introduzione
- Proprietà e Operazioni
- Concetti base e avanzati
- Diagrammi degli oggetti

Ingegneria del software

Riccardo Cardin

3

DIAGRAMMI DELLE CLASSI



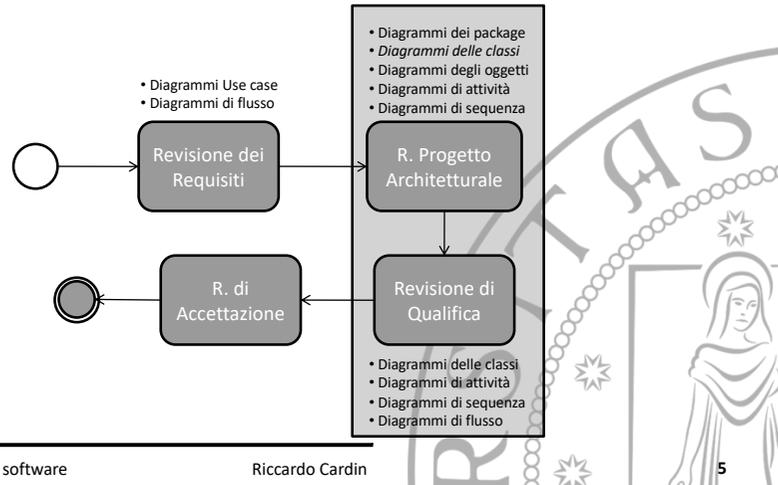
Ingegneria del software

Riccardo Cardin

4

DIAGRAMMI DELLE CLASSI

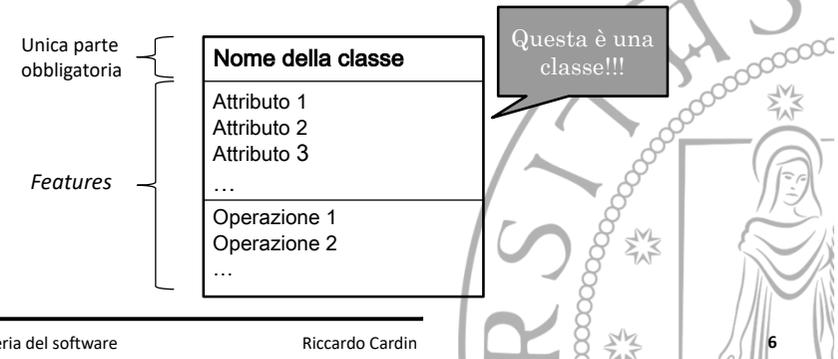
o Specifica Tecnica, Definizione di Prodotto



DIAGRAMMI DELLE CLASSI

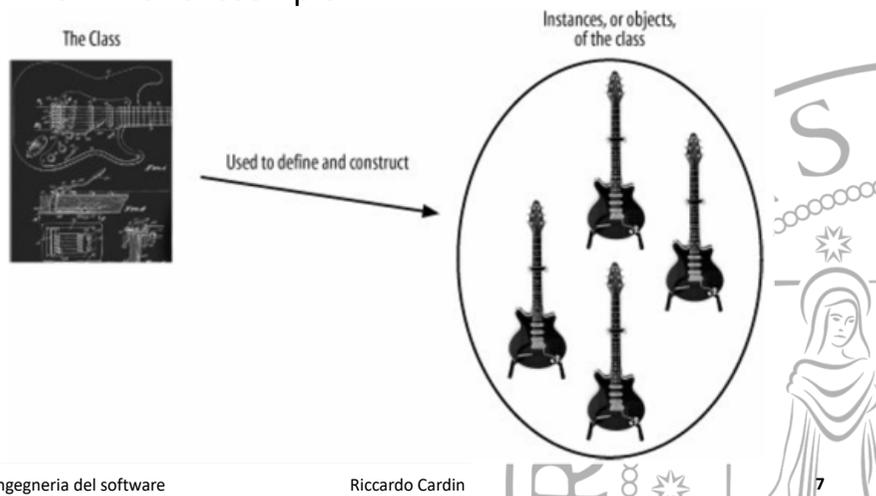
o Definizione

- Descrizione del tipo degli oggetti che fa parte di un sistema
 - o Relazioni statiche fra i tipi degli oggetti



DIAGRAMMI DELLE CLASSI

o Definizione: esempio



DIAGRAMMI DELLE CLASSI

o Esempio principale

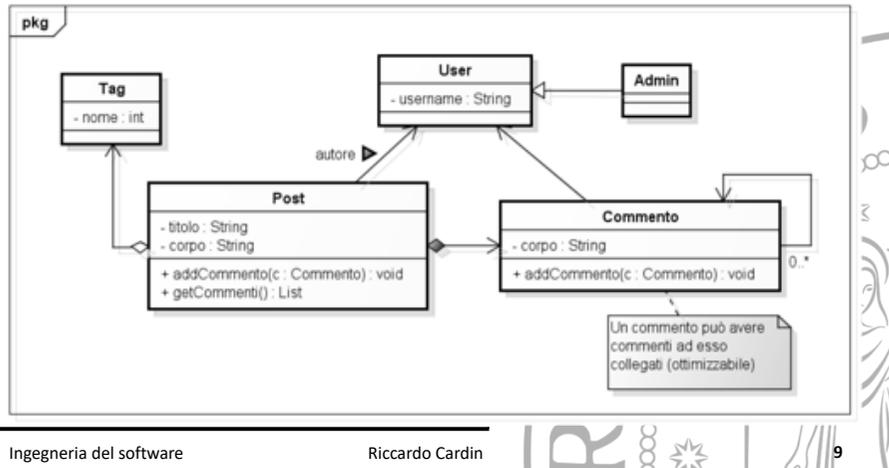
Esempio

È richiesto lo sviluppo di un'applicazione che permetta la gestione di un semplice blog. In particolare devono essere disponibili almeno tutte le funzionalità base di un blog: deve essere possibile per un utente inserire un nuovo post e successivamente per gli altri utenti deve essere possibile commentarlo. Queste due operazioni devono essere disponibili unicamente agli utenti registrati all'interno del sistema. La registrazione avviene scegliendo una username e una password. La username deve essere univoca all'interno del sistema.

The footer includes 'Ingegneria del software' and 'Riccardo Cardin'.

DIAGRAMMI DELLE CLASSI

o Esempio principale



SOMMARIO

- o Introduzione
- o Proprietà e Operazioni
- o Concetti base e avanzati
- o Diagrammi degli oggetti

PROPRIETÀ

o Caratteristiche strutturali

- Attributo

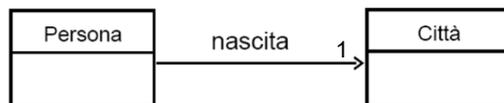
Definizione

Visibilità nome : tipo [molteplicità] = default {proprietà aggiuntive}

- o Visibilità: + pubblica, - privata, # protetta

• Associazione

- o Linea continua e orientata fra due classi



o Molteplicità

- o Quanti oggetti possono far parte dell'associazione

- 1, 0..1, 0..*, *, ...

- o Spesso interscambiabile con un attributo: quando usarla?

PROPRIETÀ

o Visibilità

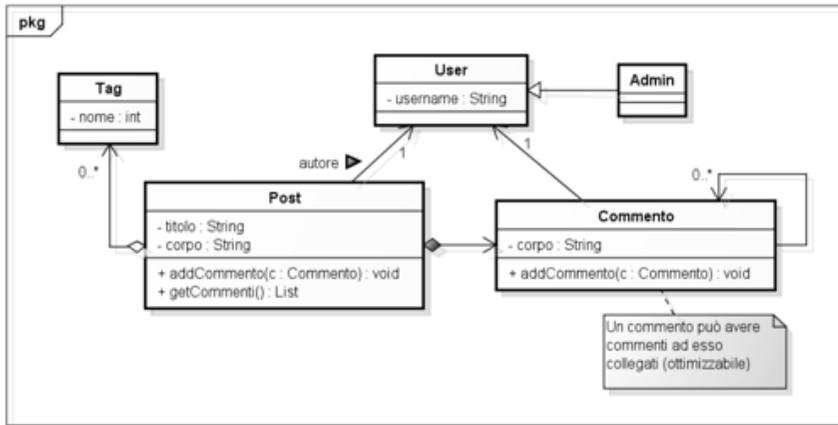
Name (Notation)	Public (+)	Protected (#)	Package (~)	Private (-)
-----------------	------------	---------------	-------------	-------------

More accessible to other parts of the system

Less accessible to other parts of the system

PROPRIETÀ

o Esempio 1



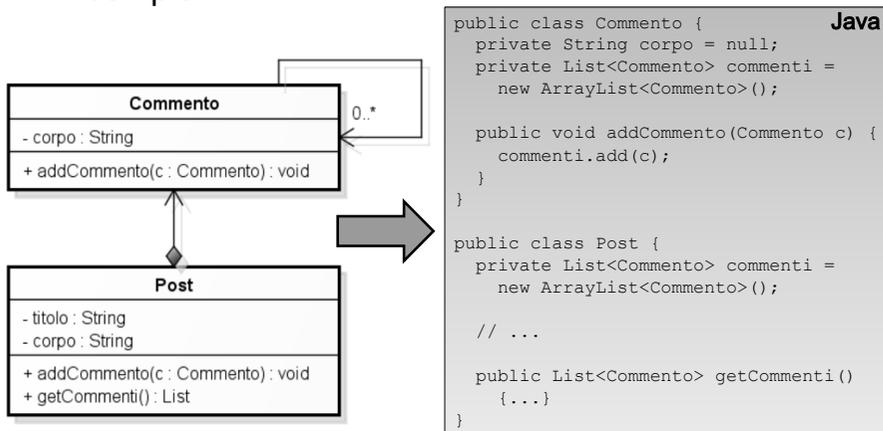
PROPRIETÀ

o ...nel linguaggio di programmazione

- **Attributi**
 - o Membri di classe (privati, se possibile)
 - o Proprietà aggiuntive
 - o Se *ordered*: Array o vettori
 - o Se *unordered*: insiemi
 - o Convenzioni dei gruppi di programmazione
 - o Esempio: *Getter* e *setter* per ogni attributo
- **Associazioni**
 - o Anche se etichettata con verbo, meglio renderla con un nome
 - o Evitare le associazioni bidirezionali
 - o Di chi è la responsabilità di aggiornare la relazione?

PROPRIETÀ

o Esempio 2



OPERAZIONI

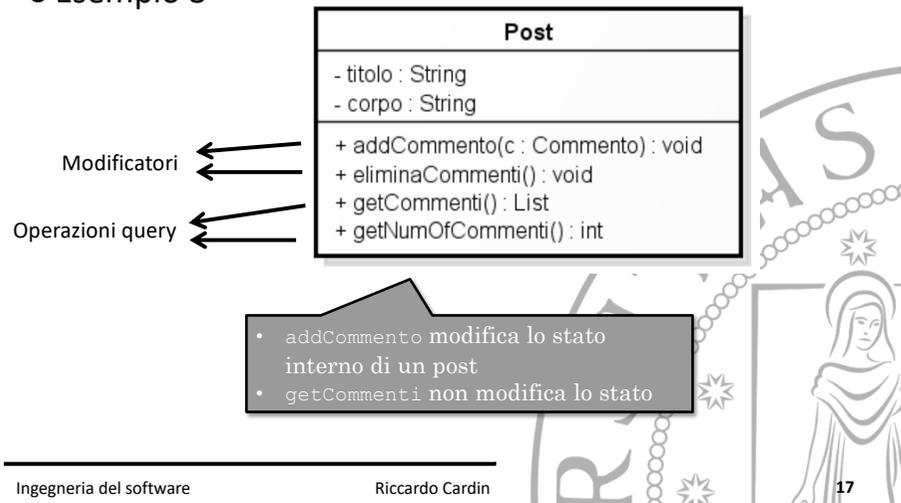
o Le azioni che la classe "sa eseguire"

- Aspetto comportamentale
 - Servizio che può essere richiesto ad ogni istanza della classe
- Definizione**
- Visibilità nome (lista-parametri) : tipo-ritorno** {proprietà aggiuntive}

Lista-parametri := direzione nome : tipo = default
- o Direzione: **in, out, inout** (*default in*)
 - o Visibilità: **+** pubblica, **-** privata, **#** protetta
 - **Query**
 - **Modificatori**
 - **Operazione ≠ metodo**
 - o Concetti differenti in presenza di polimorfismo

OPERAZIONI

o Esempio 3



SOMMARIO

o Introduzione

o Proprietà e Operazioni

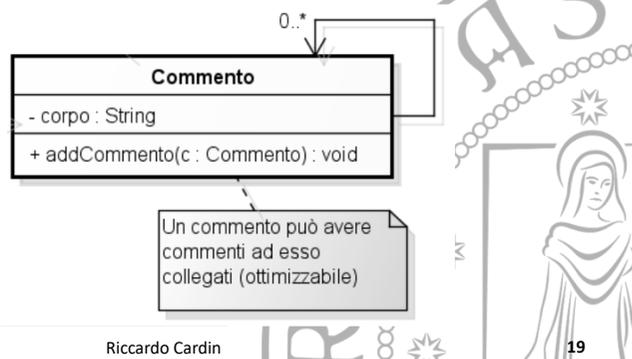
o Concetti base e avanzati

o Diagrammi degli oggetti

COMMENTI E NOTE

o Informazioni aggiuntive

- Singole e solitarie
- Legate a qualsiasi elemento grafico
 - o Linea tratteggiata
- Esempio 5



RELAZIONE DI DIPENDENZA

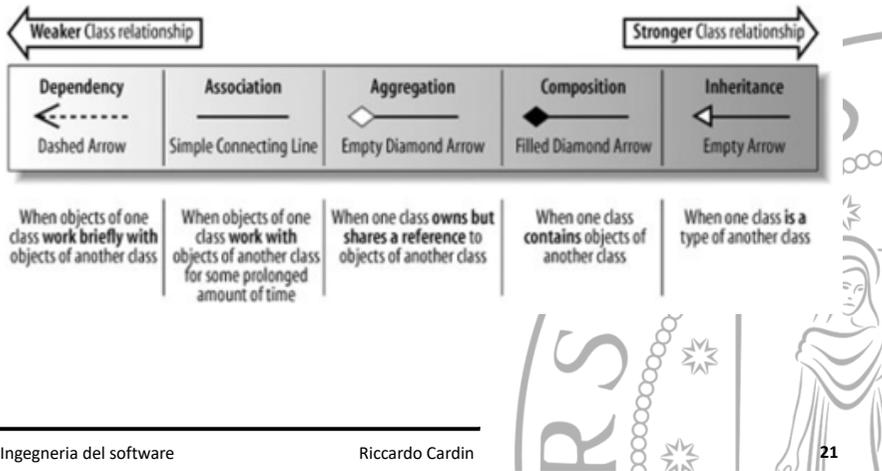
o Definizione

Si ha dipendenza tra due elementi di un diagramma se la modifica alla definizione del primo (fornitore) può cambiare la definizione del secondo (client)

- UML permettere di modellare ogni sorta di dipendenza
 - o Non è una proprietà transitiva!
- Le dipendenze vanno minimizzate!
 - o *Loose coupling*
- Da inserire solo quando danno valore aggiunto
 - o Troppe dipendenze creano confusione nel diagramma

RELAZIONE DI DIPENDENZA

o Definizione



RELAZIONE DI DIPENDENZA

o Definizione

Maggiore è la quantità di codice condiviso fra due tipi, maggiore è la dipendenza fra essi.

- La dipendenza tra due tipi è direttamente proporzionale alla probabilità di modificare entrambi

$$\delta_{A \rightarrow B} \propto P(B_{mod} | A_{mod})$$

- La dipendenza è quindi una funzione di numero SLOC condivise e di ampiezza dello scope del codice condiviso

$$\delta_{A \rightarrow B} = \frac{\varphi_{S_{A|B}}}{\varphi_{S_{totB}}} \epsilon_{A \rightarrow B}$$

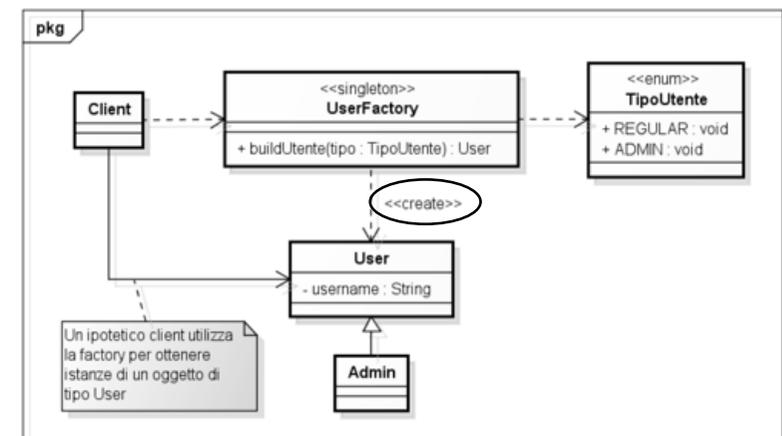
RELAZIONE DI DIPENDENZA

o Dipendenze UML

Parola chiave	Significato
«call»	La sorgente invoca un'operazione della classe destinazione.
«create»	La sorgente crea istanze della classe destinazione.
«derive»	La sorgente è derivata dalla classe destinazione
«instantiate»	La sorgente è una istanza della classe destinazione (meta-classe)
«permit»	La classe destinazione permette alla sorgente di accedere ai suoi campi privati.
«realize»	La sorgente è un'implementazione di una specifica o di una interfaccia definita dalla sorgente
«refine»	Raffinamento tra differenti livelli semantici.
«substitute»	La sorgente è sostituibile alla destinazione.
«trace»	Tiene traccia dei requisiti o di come i cambiamenti di una parte di modello si colleghino ad altre
«use»	La sorgente richiede la destinazione per la sua implementazione.

RELAZIONE DI DIPENDENZA

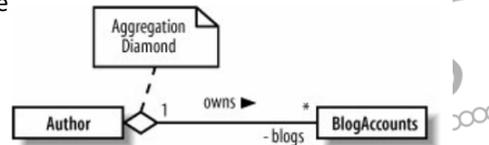
o Esempio 6



AGGREGAZIONE E COMPOSIZIONE

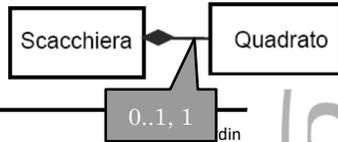
o Aggregazione

- Relazione di tipo "parte di" (*part of*)
 - o Gli aggregati possono essere condivisi



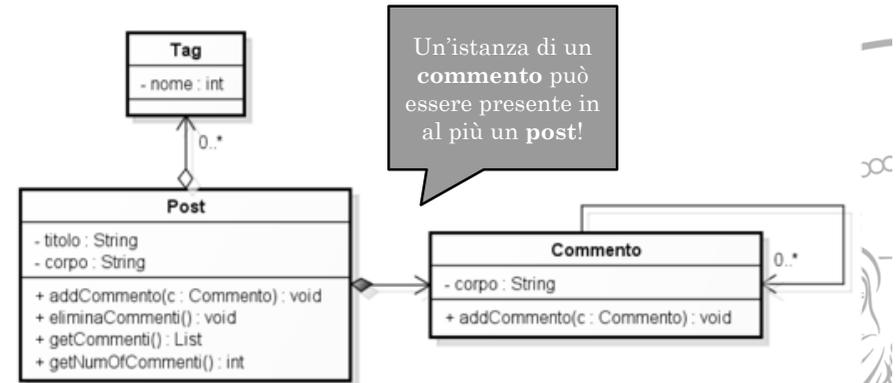
o Composizione

- Come aggregazione, ma:
 - o Gli aggregati appartengono ad un solo aggregato
 - o Solo l'oggetto intero può creare e distruggere le sue parti



AGGREGAZIONE E COMPOSIZIONE

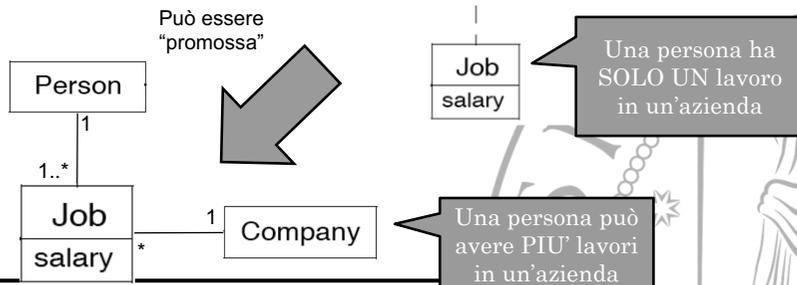
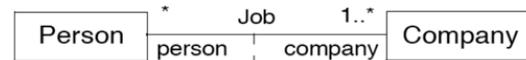
o Esempio 7



CLASSI DI ASSOCIAZIONE

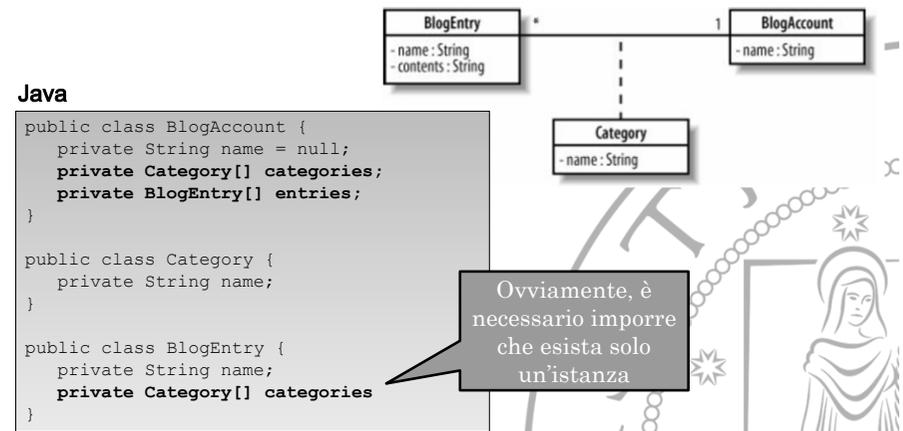
o Aggiungono attributi e operazioni alle associazioni

- Esiste solo una istanza della classe associazione fra i due oggetti



CLASSI DI ASSOCIAZIONE

o Traduzione in linguaggio di programmazione



Java

```
public class BlogAccount {
    private String name = null;
    private Category[] categories;
    private BlogEntry[] entries;
}

public class Category {
    private String name;
}

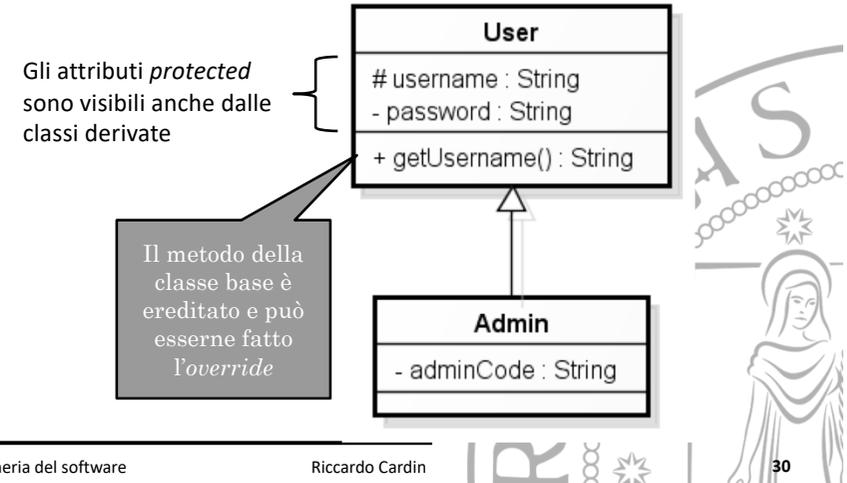
public class BlogEntry {
    private String name;
    private Category[] categories
}
```

GENERALIZZAZIONE

- A generalizza B, se ogni oggetto di B è anche un oggetto di A
 - Equivale all'ereditarietà dei linguaggi di programmazione
 - Ereditarietà multipla supportata, ma da **NON USARE!**
 - Le proprietà della superclasse non si riportano nel diagramma della sottoclasse
 - A meno di *override*
 - Sostituibilità
 - Sottotipo ≠ sottoclasse
 - Interfacce / implementazione

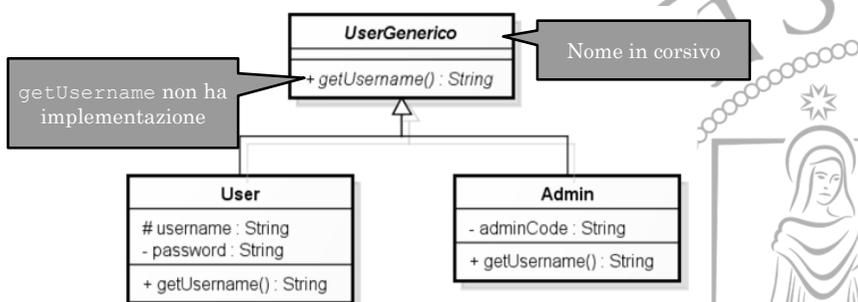
GENERALIZZAZIONE

- Esempio 4



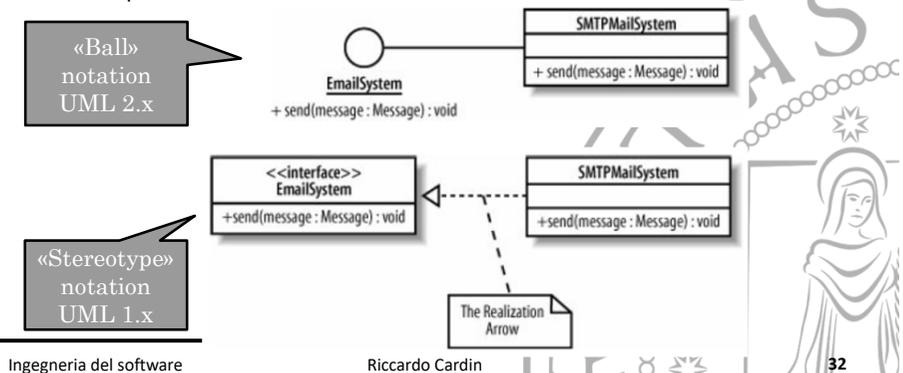
CLASSI ASTRATTE

- Classe Astratta {*abstract*}
- Classe che non può essere istanziata
 - Operazione astratta non ha implementazione
 - Altre operazioni possono avere implementazione



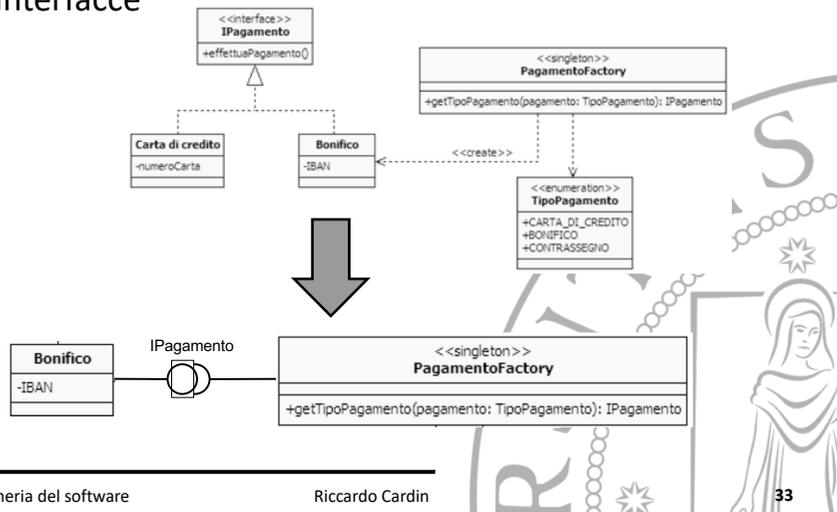
INTERFACCE

- Interfaccia «*interface*»
 - Classe priva di implementazione
 - Una classe realizza un'interfaccia se ne implementa le operazioni



INTERFACCE

o Interfacce



CLASSIFICAZIONE E GENERALIZZAZIONE

o Sottotipo ≠ "è un" (IS A)

Generalizzazione

- *Un Border Collie è un cane*
- *I cani sono animali*
- *I cani sono una specie*

Classificazione

- *Shep è un Border Collie*
- *Border Collie è una razza*

• Generalizzazione

- o Proprietà transitiva
- o La classificazione non lo è!

• Classificazione

- o Dipendenza `<instantiated>`

CARATTERISTICHE VARIE

o Operazioni e attributi statici

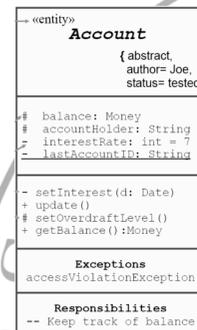
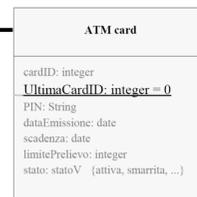
- Applicabili alla classe, non all'oggetto
- o Sottolineati sul diagramma

o Parole chiave

- Estensione della semantica UML
- o Costrutto simile + parola chiave!
- o `<<interface>>`
- o `{abstract}`

o Responsabilità

- Funzionalità offerte
- Aggiunta alla classe con commento



CARATTERISTICHE VARIE

o Proprietà derivate

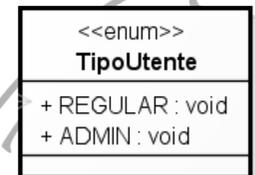
- Possono essere calcolate a partire da altri valori
- o Definiscono un vincolo fra valori
- o Si indicano con `"/"` che precede il nome della proprietà

o Proprietà *read only* e *frozen*

- `{readOnly}`
- o Non vengono forniti i servizi di scrittura
- `{frozen}`
- o Immutabile, non può variare nel suo ciclo di vita

o Enumerazioni

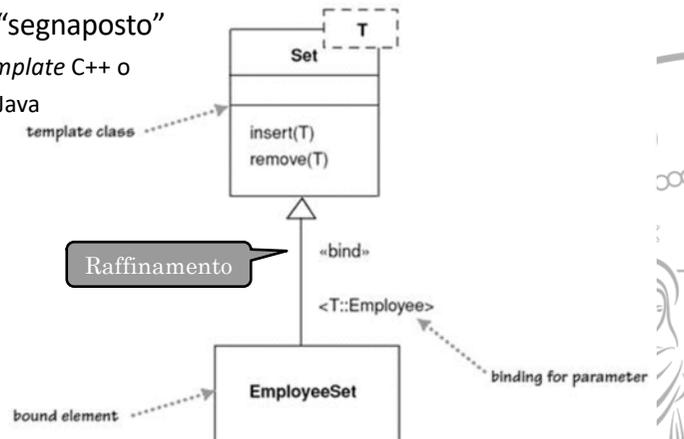
- Insiemi di valori che non hanno altre proprietà oltre il valore simbolico
- `<<enumeration>>`



CARATTERISTICHE VARIE

o Classi Parametriche

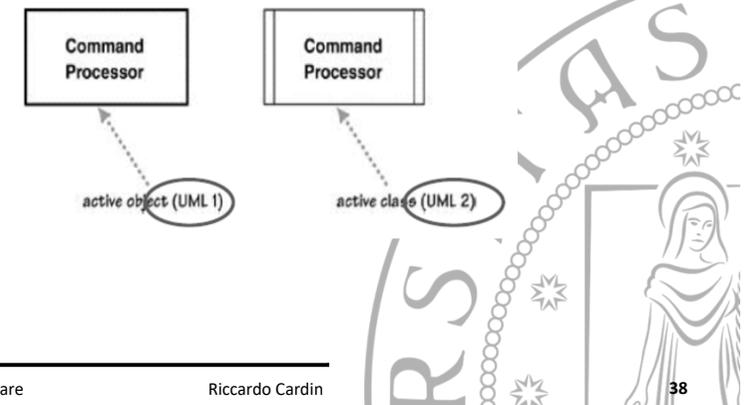
- T è detto “segnaposto”
 - o Come *template* C++ o *generics* Java



CARATTERISTICHE VARIE

o Classi Attive

- Eseguono e controllano il proprio *thread*



CONSIGLI UTILI

o Diagrammi molto ricchi di concetti

- Non cercare di utilizzare tutte le notazioni disponibili
 - o Cominciare dapprima con i concetti semplici
- Una prospettiva concettuale permette di esplorare il linguaggio di un particolare *business*
 - o Mantenere la notazione semplice e non introdurre concetti legati al *software*
- Concentrarsi nel disegno dei diagrammi delle parti più importanti
 - o Disegnare ogni cosa è sinonimo di diagrammi non fondamentali che diventano obsoleti molto presto!

SOMMARIO

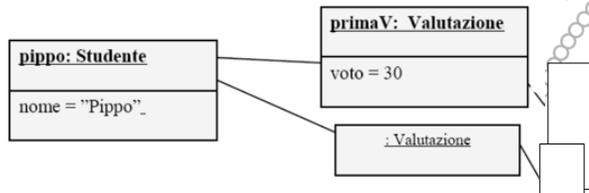
- o Introduzione
- o Proprietà e Operazioni
- o Concetti base e avanzati
- o Diagrammi degli oggetti

DIAGRAMMI DEGLI OGGETTI

- o Grafo delle istanze, comprensivo di associazioni e valori delle proprietà

```
nome dell'istanza : nome della classe
```

- Fotografia degli oggetti che compongono un sistema
- Non ci sono parti obbligatorie
- Specifica di istanza
 - o Anche di classi astratte, omissione dei metodi, ecc...



RIFERIMENTI

- o OMG Homepage – www.omg.org
- o UML Homepage – www.uml.org
- o UML Distilled, Martin Fowler, 2004, Pearson (Addison Wesley)
- o Learning UML 2.0, Kim Hamilton, Russell Miles, O'Reilly, 2006
- o Dependency - <http://rcardin.github.io/programming/oop/software-engineering/2017/04/10/dependency-dot.html>

GITHUB REPOSITORY



<https://github.com/rcardin/swe>