Unified Modeling Language
- Vista Generale -

Unified Modeling Language

- Definizione
  - Lo Unified Modeling Language (UML) è un linguaggio per specificare, visualizzare, costruire e documentare i manufatti di sistemi software-intensivi

Obiettivi

- Fornire
  - un linguaggio di modellazione visuale, "facile da usare", per sviluppare e scambiare modelli significativi tra i diversi stakeholder
  - meccanismi di estensione e specializzazione per accrescere i concetti chiavi (stereotipi).
- Supportare
  - specifiche che siano indipendenti da particolari linguaggi di programmazione e processi di sviluppo
  - concetti di sviluppo di più alto livello come componenti, collaborazioni, frameworks e patterns.

Caratteristiche

- Non è un linguaggio proprietario
  - A partire dal 1997 l'evoluzione è a carico dell'OMG (Object Management Group) ed è soggetta a procedure ben definite per ogni cambiamento
  - Alla stesura delle specifiche contribuiscono molti metodologi e alcune delle più importanti società mondiali di software
- Versione considerata
  - UML versione 1.5, Marzo 2003

Concetto chiave: Generalizzazione

- Definizione
  - una relazione tassonomica tra un elemento più generale ed uno più specifico (elemento specializzato).
- Proprietà
  - L'elemento specializzato è totalmente consistente con l'elemento generale e contiene informazioni aggiuntive.

Concetto chiave: Ereditarietà

- Definizione
  - meccanismo di condivisione delle proprietà di due elementi (casi d'uso, classi, package, etc) in una gerarchia di generalizzazione.
- Proprietà
  - la struttura ed il comportamento dell'elemento generale sono incorporati (ereditati) nell'elemento specializzato
  - gode della proprietà transitiva
  - l'elemento specializzato può contenere nuove proprietà non incluse nell'elemento generale
  - l'elemento specializzato può ridefinire le proprietà dell'elemento generale (overriding)

Concetto chiave: Package

- Definizione
  - meccanismo generale per organizzare elementi, semanticamente correlati, all'interno di un unico elemento, ossia il package
- Proprietà
  - elemento trasversale a tutti i diagrammi UML
  - gli elementi raggruppati in un package sono chiamati contenuto del Package
  - un package può a sua volta essere contenuto di un altro package stabilendo così una relazione gerarchica

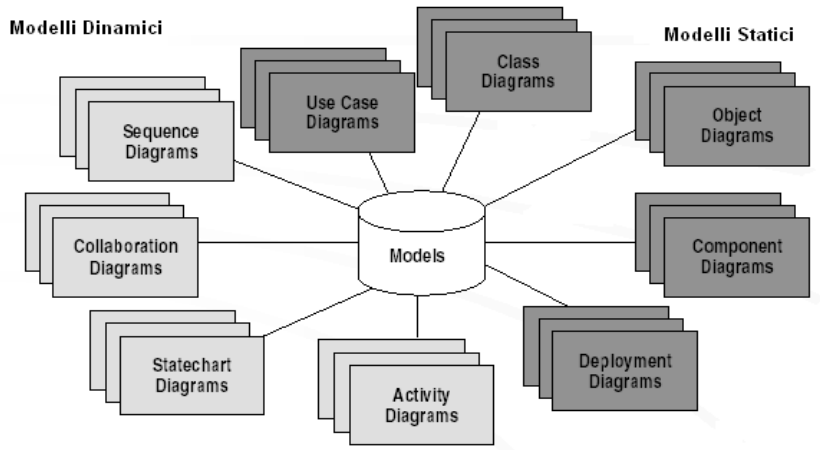
Concetto chiave: Stereotipo

- Definizione
  - un nuovo tipo di elemento che costruito su uno già esistente ne estende la semantica senza però modificarne la struttura
- Esempi di stereotipi
  - classi
    - boundary, entity, control, client page, server page, etc
  - componenti
    - subsystem
  - casi d'uso
    - business

Diagrammi UML …

- Strutturali
  - Casi d'uso (Use Case)
  - Classi (Class)
  - Oggetti (Object)
- Comportamentali
  - Stati (Statechart)
  - Attività (Activity)
  - Interazione (Interaction)
    - Sequenza (Sequence)
    - Collaborazione (Collaboration)
- Implementativi
  - Componenti (Component)
  - Configurazione/Dispiegamento (Deployment)

… Diagrammi UML …



… Diagrammi UML

- Analisi
  - Casi d'uso
- Progettazione
  - Classi
  - Oggetti
  - Transizioni di stato
  - Attività
  - Interazione
    - Sequenza
    - Collaborazione
- Implementazione
  - Componenti
  - Dispiegamento

I Casi D'Uso

Definizione (1) …

- "Una sequenza di transazioni di un sistema, il cui compito è di produrre un risultato di valore misurabile per uno o più attori del sistema" (Ivar Jacobson et al., 1995)
  - la descrizione di un caso d'uso definisce cosa accade nel sistema in seguito all'evento di attivazione

… Definizione (2)

- "Un caso d'uso rappresenta un'unità coerente di funzionalità fornita da una specifica entità* e descritta sia dalla serie di messaggi scambiati tra l'entità stessa e gli attori, sia dalla sequenze di azioni svolte" (definizione UML 1.5)

* entità: sistema, sottosistema, classe

Perché nascono?

- Per fornire delle descrizioni di utilizzo di un sistema che siano facilmente:
  - leggibili
  - comprensibili
  - validabili
- anche da chi, molto spesso il committente e gli utilizzatori, non ha alcuna competenza informatica (formalizzati nel 1987 da Ivar Jacobson)

Generalità …

- I caso d'uso
  - devono essere specificati sotto forma di testo; a corredo di tali specifiche possono esserci delle rappresentazioni grafiche
  - possono essere integrati con l'Analisi di un sistema, per aiutare a scoprire il comportamento che i suoi utenti si aspettano e quindi a rilevare eventuale lacune ed incoerenze dell'Analisi
  - modellano i requisiti di un sistema per come esso appare all'esterno (vista black-box)

… Generalità

- definiscono molto chiaramente i confini del sistema e gli attori umani e non
- sono generalmente attivati da un attore ma possono anche essere attivati dal sistema stesso (es. produzione cedolini a fine mese, ricarico automatico di un magazzino)
- corrispondono a un compito che l'attore vuole svolgere (se l'attore attiva il caso d'uso) o che il sistema deve eseguire (se il sistema attiva il caso d'uso)

Identificare i casi d'uso …

⇨ A seconda della parti interessate:
- Committente
- Commerciale
- Analista
- Progettista
- Implementatore
- Manutentore
- Utilizzatore
- …

⇨ esistono diversi modi di "guardare" un sistema. Per identificare i casi d'uso è necessario assumere il punto di vista degli utilizzatori

… Identificare i casi d'uso

⇨ Esempio: in un sistema per la gestione della posta elettronica
- protocolli TCP/IP
- algoritmi di criptaggio e decriptaggio
- meta-informazioni associate ad una email

⇨ interessano il progettista ed il manutentore, mentre
- inserire, inoltrare, cancellare un'email
- usare i filtri, creare una propria firma

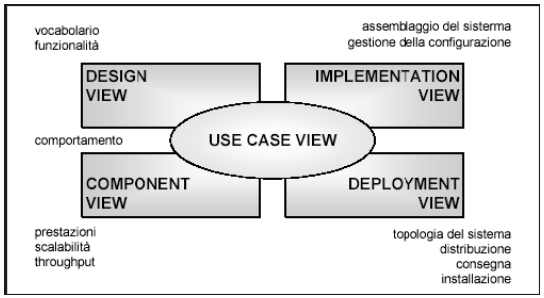
⇨ interessano l'utilizzatore

UML ed i Casi d'Uso

UML ed i Casi d'Uso

- UML prevede una divisione del sistema in viste*

*insieme di costrutti e concetti che rappresentano uno stesso aspetto di un sistema



Vista dei Casi d'uso …

- La Vista dei Casi d'Uso:
  - cattura il comportamento del sistema così come esso appare all'esterno
- e
  - partiziona le funzionalità del sistema in transazioni significative per gli attori, ossia gli utenti idealizzati del sistema.

… Vista dei Casi d'uso

- Proiezioni
  - Statica
    - fornisce una mappa degli utilizzi del sistema da realizzare senza specificare circa la dinamica di utilizzo; è rappresentata mediante i diagrammi dei Casi d'Uso
  - Dinamica
    - illustra la sequenza di attività che il sistema compie nell'esecuzione di un caso d'uso. Può essere rappresentato in forma diagrammatica: diagrammi di Stato, di Sequenza, di Collaborazione

Diagrammi dei Casi d'Uso

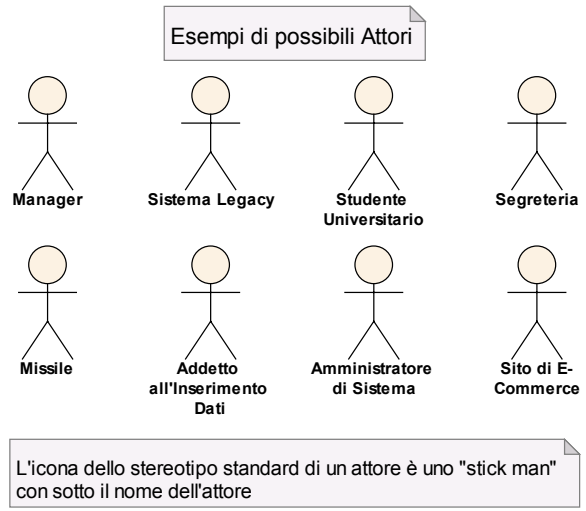
Definizione

- Semantica
  - è un diagramma che mostra attori e casi d'uso insieme alle relazioni tra questi elementi. I casi d'uso rappresentano le funzionalità di un sistema, manifestate ad attori o altre entità esterna al sistema.
- Notazione
  - è un grafico costituito da attori, casi d'uso e le relazioni tra questi elementi. Le relazioni sono associazioni tra attori e casi d'uso, generalizzazioni tra attori, generalizzazioni, estensioni e inclusioni tra casi d'uso

Attore …

- Definizione
  - un ruolo o un insieme di ruoli che qualcuno o qualcosa, esterno al sistema, svolge nell'interagire con il sistema;
- Esempi di Attori
  - una classe di persone fisiche (es. fornitore)
  - un altro sistema (es. sistema di contabilità)
  - un dispositivo hardware esterno (es. sensore)

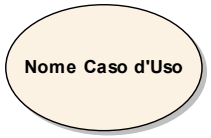
… Attore …



… Attore

- Differenza concettuale tra
  - Utente
    - soggetto/oggetto fisico che può utilizzare il sistema; ogni utente può assumere il ruolo di diversi attori per il Sistema Software
- e
  - Attore
    - ruolo che descrive una prospettiva d'uso del sistema software; lo stesso ruolo può essere svolto da uno o più utenti

Caso d'uso: Notazione



Un caso d'uso è rappresentato mediante un ellisse all'interno del quale è indicato il nome del caso d'uso.
Al di sopra del nome può essere indicato, opzionalmente, lo stereotipo del caso d'uso.
Al di sotto del nome può essere presente, opzionalmente, un elenco delle proprietà del caso d'uso.

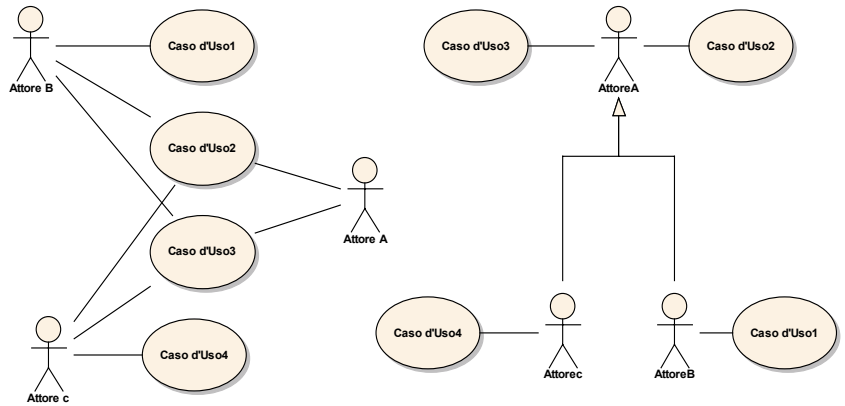
Tipi di Relazioni

Elementi relati	Relazione	Funzione	Notazione
Attori	Generalizzazione	un attore, specializzato, eredita la partecipazione a tutti i casi d'uso con i quali l'attore specializzante comunica	
Attore – Caso d'uso	Associazione	esprime la partecipazione di uno o più attori ad un caso d'uso	
Casi d'uso	Estensione	il comportamento di un caso d'uso base può, opzionalmente, essere esteso dal comportamento definito da un altro caso d'uso	«extend»
	Inclusione	il comportamento di un caso d'uso di base incorpora, sempre, il comportamento del caso d'uso di inclusione	«include»
	Generalizzazione	un caso d'uso generale ed uno o più specifici casi d'uso che ne ereditano ed aggiungono caratteristiche	

Relazione di Generalizzazione tra attori …

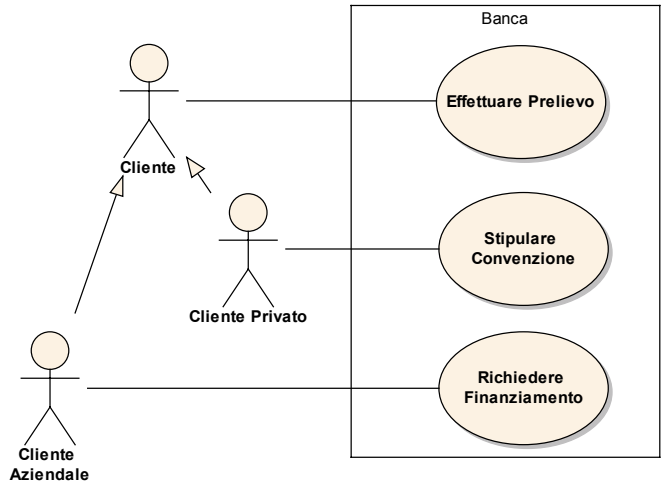
- Definizione
  - una generalizzazione tra un Attore A ed un Attore B indica che un'istanza di A può comunicare con le stesse istanze dei casi d'uso di B.
- Proprietà
  - gli attori figli possiedono, tutte le caratteristiche dell'attore padre e ne possono aggiungere delle altre.

… Relazione di Generalizzazione tra attori …

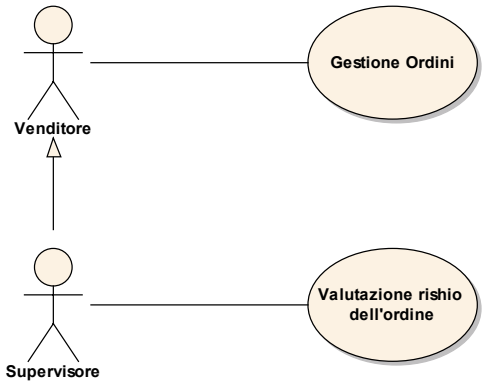


L'attore A è associato ad un sotto insieme dei casi d'uso a cui sono associati B e C
L'attore A diviene padre e B e C divengono suoi figli.
B e C saranno associati a tutti i casi d'uso a cui è associato A

… Relazione di Generalizzazione tra attori …



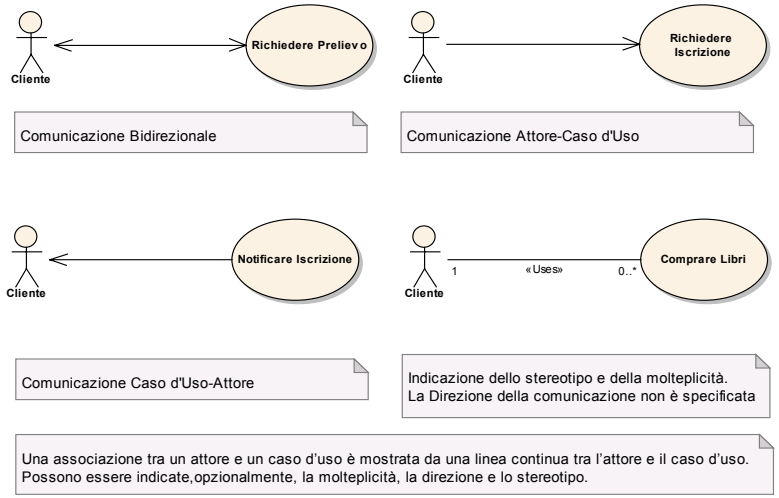
… Relazione di Generalizzazione tra attori



Relazione di Associazione …

- ⇨ Definizione
  - partecipazione di un attore ad un caso d'uso; cioè, istanze dell'attore e istanze del caso d'uso comunicano l'una con l'altra
- ⇨ Proprietà
  - è l'unica relazione tra attori e casi d'uso
  - la direzione di un'associazione può essere
    - bi-direzionale
    - uni-direzionale (da Attore a Caso d'Uso; da Caso d'Uso ad Attore)
    - non specificata

… Relazione di Associazione: Esempi



Punto di Estensione di un Caso d'Uso …

- Definizione
  - riferimento a una locazione all'interno di un caso d'uso (base) in cui sequenze di azione di un altro caso d'uso (estendente) potrebbero essere inserite.
- Proprietà
  - ogni punto di estensione è identificato da un nome (unico all'interno del caso d'uso); questo stesso nome, insieme alle azioni ad esso associate, è presente nel caso d'uso estendente
  - l'estensione avviene sotto il controllo di una condizione di guardia

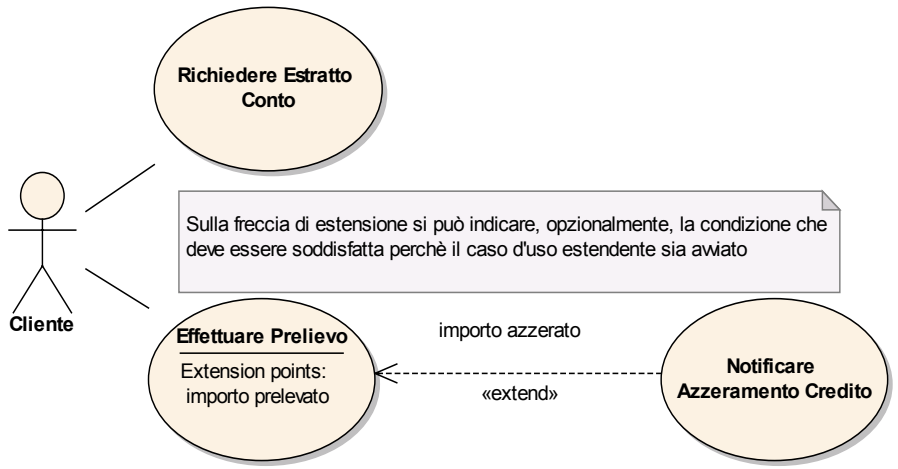
… Punto di Estensione di un Caso d'Uso

- un caso d'uso base può avere più punti di estensione appartenenti a casi d'uso estendenti diversi
- un caso d'uso estendente può avere più segmenti di estensione

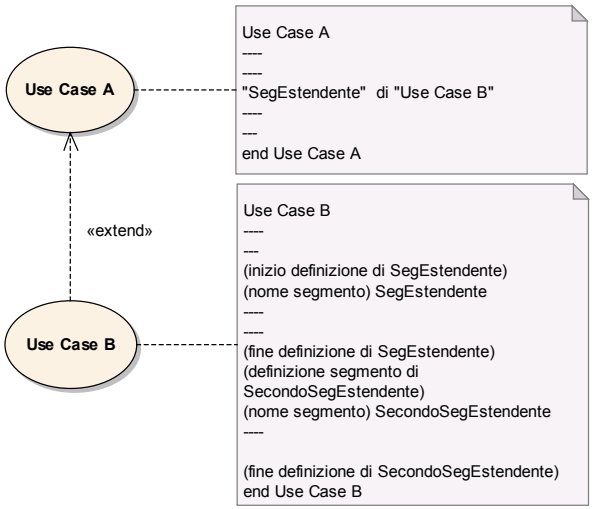
Relazione di Estensione …

- Definizione
  - una relazione di estensione da un caso d'uso A (estendente) a un caso d'uso B (base) indica che un'istanza del caso d'uso B può essere accresciuta, sotto determinate condizioni specificate nell'estensione, dal comportamento specificato in A.
- Proprietà
  - il punto di estensione in A è etichettato con un nome di estensione
  - lo stesso nome è usato nel caso d'uso estendente per racchiudere il comportamento

… Relazione di Estensione: Esempio …



… Relazione di Estensione: Flusso

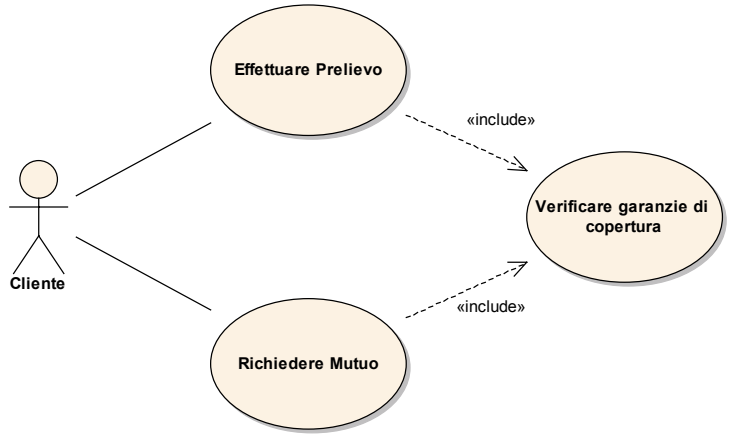


Relazione di Inclusione …

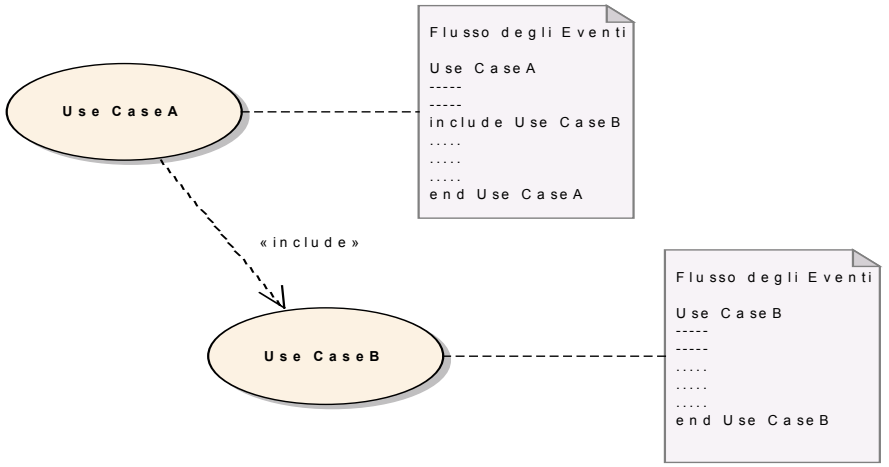
- Definizione
  - Una relazione di inclusione da un caso d'uso A (base) ad un caso d'uso B (inclusione) indica che un'istanza del caso d'uso A conterrà anche il comportamento specificato da B.
- Proprietà
  - Il comportamento di B è inserito nella locazione definita in A come di inclusione.
  - La relazione di inclusione esplicita come il caso d'uso di base incorpora il comportamento del caso d'uso di inclusione (è molto utile quando si vuole evitare il ripetersi di uno stesso flusso di sequenza)

… Relazione di Inclusione: Esempio …

Una relazione di inclusione è rappresentata da una freccia, tratteggiata e con la "testa" aperta, che punta dal caso d'uso base al caso d'uso incluso



… Relazione di Inclusione: Flusso



Differenze tra Inclusione ed Estensione

- Relazione di Inclusione
  - il caso d'uso A include il caso d'uso B
    - specifica che il comportamento di B include **sempre** il comportamento, nella sua **totalità**, di B
- Relazione di Estensione
  - caso d'uso A estende Caso d'uso B
    - specifica che il comportamento di B **potrebbe** essere esteso da una **parte o** dalla **totalità** del comportamento di B qualora si verifichi una determinata condizione (condizione di guardia)

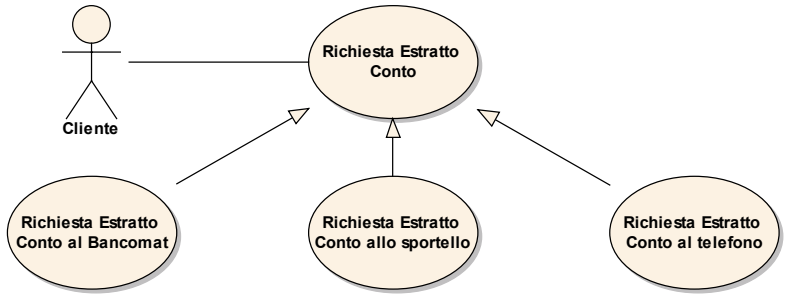
Relazione di Generalizzazione tra Casi d'Uso …

- Definizione
  - una generalizzazione da un caso d'uso B a un caso d'uso A indica che A è una specializzazione di B.
- Proprietà
  - il caso d'uso generale definisce una serie di passi, ed ha relazioni di associazione con uno o più attori
  - ogni caso d'uso specializzato eredita le caratteristiche, i passi, gli eventuali punti di estensione e le associazioni del caso d'uso generale.

… Relazione di Generalizzazione tra Casi d'Uso …

- il caso d'uso specializzato può aggiungere nuovi passi, oppure ridefinire i passi ereditati da quello generale (override)
- l'esecuzione di un caso d'uso specializzato esclude l'esecuzione di tutti gli altri casi d'uso specializzati

… Relazione di Generalizzazione tra Casi d'Uso: Esempio



Una generalizzazione è rappresentata da una freccia, continua e con la "testa" chiusa, che parte dal caso d'uso specializzato e punta al caso d'uso generale

Descrizione Testuale di un Caso d'Uso

Descrizione testuale di un caso d'uso

- Indicazioni UML
  - non fornisce alcuna specifica su come definire la descrizione testuale di un caso d'uso.
- Indicazioni generali
  - ogni descrizione di un caso d'uso è caratterizzata da un insieme di informazioni di base, che permettono di definire:
    - il contesto
    - chi comunica e perché con il caso d'uso
    - cosa deve essere vero prima e dopo l'attivazione del caso d'uso,
  - e da uno o più scenari di interazione

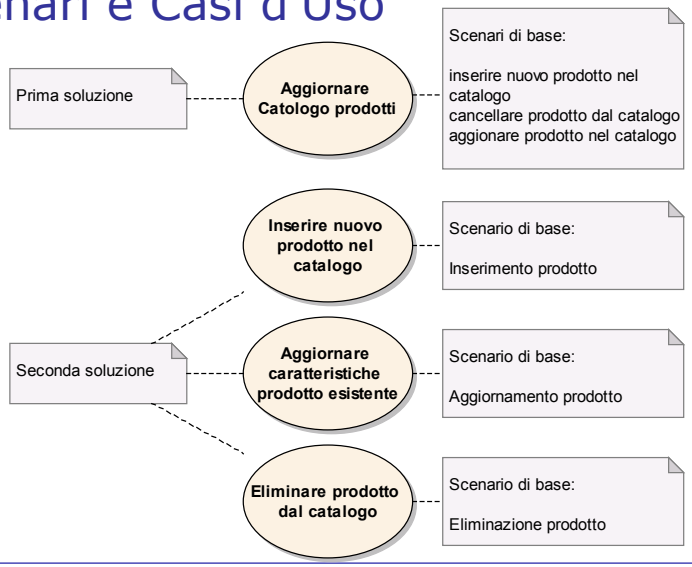
Scenari di interazione

- Definizione
  - ogni specifica esecuzione (istanza) di un caso d'uso assume il nome di scenario (di interazione)
- Descrizione di uno scenario
  - sequenza di passi, in linguaggio naturale, che specifica il dialogo, costituito da stimoli e risposte, tra il sistema e gli attori coinvolti nello scenario
- Tipi di scenari
  - base, quando il caso d'uso termina con esito positivo
  - alternativi, quando il caso d'uso termina con esito negativo (fallisce)

Scenari e Casi d'Uso …

- Distinzione
  - netta in teoria, risulta essere molto più sfumata nella pratica; se ad un caso d'uso, infatti, corrispondono più scenari di base si potrebbe rendere ognuno di questi in un caso d'uso
- Osservazione
  - qualunque sia la soluzione scelta l'insieme delle responsabilità del sistema non cambia

… Scenari e Casi d'Uso



Informazioni di base …

Descrizione:	<Descrizione del caso d'uso e del risultato atteso>
Pre-Condizioni:	<Ciò che deve essere vero prima che inizi il caso d'uso>
Post-Condizioni per Successo:	<Ciò che deve essere vero dopo che il caso d'uso si sia concluso con il raggiungimento del risultato atteso>
Post-Condizioni per Fallimento:	< Ciò che deve essere vero dopo che il caso d'uso si sia concluso senza il raggiungimento del risultato atteso>
Attore primario:	<Nome dell'attore interessato in modo primario all'esito del caso d'uso; è possibile che non sia l'attore che inizia il caso d'uso stesso>
Evento scatenante:	<L'azione di stimolo nei confronti del sistema che dà avvio al caso d'uso; potrebbe essere un evento periodico>
Estende il caso d'uso (opzionale)**:**	<Nome del caso d'uso "base" che può essere esteso dal presente caso d'uso, e punto di estensione (definito nel caso d'uso "base") a cui il presente caso d'uso si andrà ad agganciare>
Specializza il caso d'uso (opzionale)**:**	<Nome del caso d'uso "generico" del quale il presente caso d'uso costituisce una specializzazione >

… Informazioni di base: Esempio

Compilare Questionario	
Descrizione:	Inserimento nel sistema dei dati di un questionario compilato, del compilatore e dell'addetto all'inserimento. I dati memorizzati saranno usati per ottenere informazioni sui servizi oggetto di valutazione.
Pre-Condizioni:	il questionario appartiene alla popolazione dei questionari progettati; il questionario non è stato "chiuso"; lo studente non ha già compilato il questionario

Responsabilità	• Memorizzare le informazioni del compilatore
	  ○ nome
	  ○ cognome
	• Memorizzare giorno, ora ed anno di compilazione
	• Memorizzare le informazioni della compilazione
	  ○ Nome questionario
	  ○ Risposte
	• Verificare la compilazione
	  ○ Studente
	  ○ Compilazione (aperta, chiusa)

Post-Condizioni per Successo:	I dati memorizzati saranno oggetto d'elaborazione statistica
Post-Condizioni per Fallimento:	Nessuna modifica è apportata all'insieme delle compilazioni memorizzate
Attore Primario:	Addetto all'inserimento
Evento innescante:	riportare una compilazione in formato cartaceo in formato elettronico

Scenari di base ed alternativi: Esempio

Scenario di base
1. L'addetto all'inserimento inserisce le informazioni relative all'intestazione del questionario 1.1 nome studente 1.2 cognome studente 1.3 numero di matricola 1.4 giorno di sottomissione del questionario
2. L'addetto all'inserimento inserisce le risposte
3. L'addetto all'inserimento avvia la registrazione del questionario compilato
4. Il sistema registra la compilazione
Scenari alternativi
5. L'utente inserisce risposte non riconosciute come valide 5.1 il sistema visualizza un messaggio che informa l'utente di quali risposte non risultano valide
6. I dati dello studente sono errati

Casi d'Uso: Errori Comuni

Errori Comuni

- usare i casi d'uso per descrivere le funzionalità interne del sistema anziché le modalità di utilizzo del sistema da parte degli attori
- frammentare i casi d'uso in modo eccessivo
- assegnare un'eccessiva importanza ai diagrammi
- fornire delle descrizioni improprie
- separare i comportamenti sulla base del tipo di architettura del sistema

CLASSI e OGGETTI

Oggetto …

⇨ Definizione generale

- Un oggetto rappresenta una qualsiasi cosa, reale o astratta, che abbia un concetto associato

⇨ Definizione più specifica

- "Un oggetto rappresenta un articolo (item), un'unità o entità individuale, identificabile, astratta o reale che sia, con un ruolo ben definito nel dominio del problema e un confine altrettanto ben definito" (Booch)

… Oggetto: Generalità …

- Gli Oggetti interagiscono tra di loro richiedendo, reciprocamente, servizi o inviando messaggi
  - in risposta ad una richiesta, un oggetto può invocare un'operazione che ne può cambiare il proprio stato
- "Un oggetto possiede stato, comportamento e identità; la struttura e il comportamento di oggetti simili sono definiti nella loro classe comune; i termini di istanza (di classe) e oggetto sono interscambiabili" (Grady Booch)

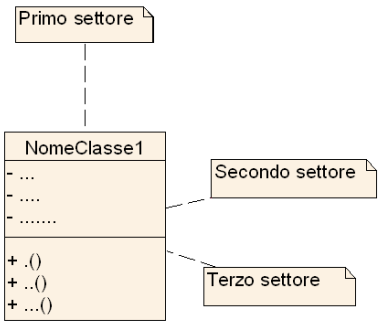
… Oggetto: Generalità

- Ogni oggetto è costituito da:
  - Stato
    - Lo Stato di un oggetto è un concetto dinamico e, in un preciso istante di tempo, è dato dal valore di tutti i suoi attributi e dalle relazioni instaurate con altri oggetti
  - Comportamento
    - costituito dalle operazioni visibili all'esterno, "il comportamento stabilisce come un aggetto agisce e reagisce, in termini del cambiamento del proprio stato e del transito dei messaggi" (Grady Booch)
  - Identità
    - è la caratteristica che distingue un oggetto da qualunque altro; ogni oggetto ha una propria identità

Classe

- Definizione
  - una classe rappresenta un'astrazione di oggetti simili, i.e. oggetti che possiedono la stessa struttura (attributi) e lo stesso comportamento (operazioni)
- Definizione formale
  - una classe equivale ad un tipo di dato astratto (incapsulamento di dati e operazioni sui dati) dove un oggetto è un elemento del tipo

Classe: Notazione

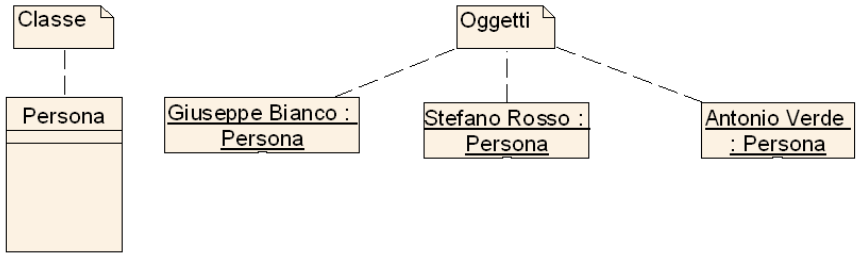


Una classe è rappresentata da un rettangolo con tre settori (compartment) separati da una linea continua.
Il primo settore, in alto, visualizza il nome della classe ed eventualmente lo stereotipo, il secondo, nel mezzo, una lista di attributi (con l'opzione di indicare il tipo ed i valori), il terzo, in fondo, una lista di operazioni (per ognuna delle quali è opzionale indicare i parametri in ingresso e i tipi di dati restituiti).
Il primo settore deve essere sempre presente, inoltre è possibile aggiungere nuovi settori

Oggetto: Notazione

- Per rappresentare gli oggetti si utilizza lo stesso simbolo usato per le classi
  - il nome della classe è sottolineato ed è preceduto da ":" per indicare che si tratta di un oggetto
  - è opzionale indicare il nome dell'oggetto

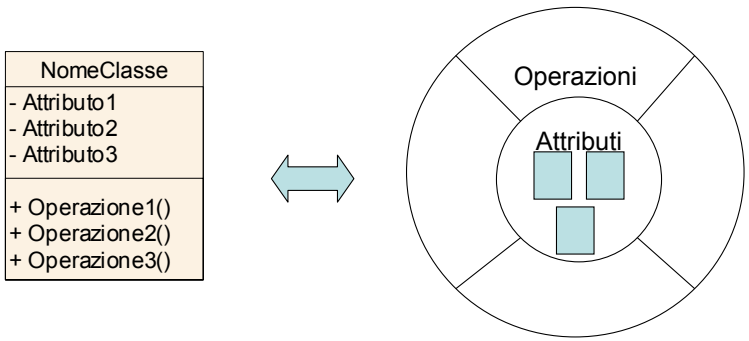
Classi e Oggetti



Un diagramma degli Oggetti è un'istanza del diagramma delle Classi; mostra, infatti, un'istantanea del diagramma delle Classi in un dato istante di tempo.
L'uso del diagramma degli Oggetti è abbastanza limitato.
Gli Oggetti sono importanti perchè usati nei diagrammi di sequenza, collaborazione e degli stati.

Concetto chiave: Incapsulamento

- Ogni attributo può essere manipolato solo dalle operazioni della classe cui appartiene



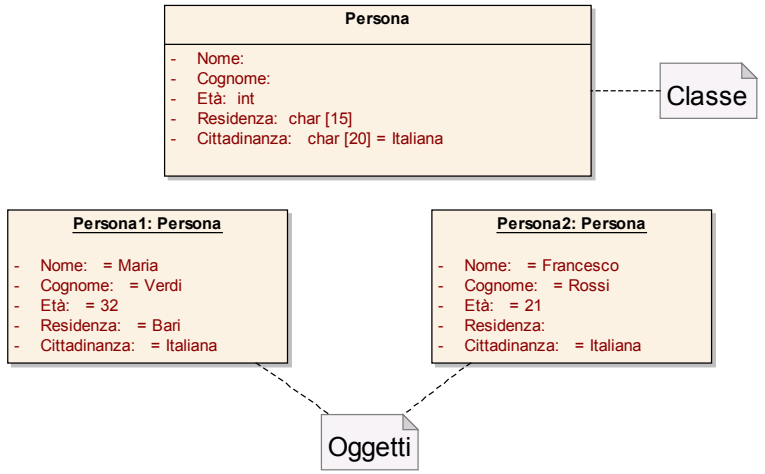
Attributi ...

- Definizione
  - è una proprietà statica di un oggetto
- Esempi
  - nome, età, peso sono attributi della classe Persona
  - colore, peso, anno-modello sono attributi della classe Auto
  - gli attributi possono essere semplici o composti

… Attributi …

- Proprietà
  - i nomi degli attributi devono essere unici all'interno di una classe
  - i valori degli attributi definiscono lo stato ma non l'identità di un oggetto
  - Per ogni attributo si possono specificare
    - *visibilità* + *nome* + [: *espressione-di-tipo*] + [*molteplicità*] + [= *valore-iniziale*]
    - se specificato *valore-iniziale*, l'attributo assumerà tale valore ad ogni istanza della classe

… Attributi: Esempio



Operazioni …

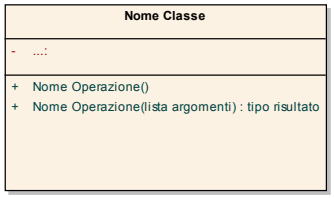
- Definizione
  - è un'azione che un oggetto esegue su un altro oggetto e che determina una reazione
- Proprietà
  - le operazioni manipolano gli attributi dell'oggetto
  - l'implementazione di un'operazione prende il nome di metodo
  - il metodo specifica l'algoritmo o la procedura associata ad un'operazione

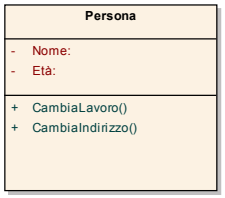
… Operazioni …

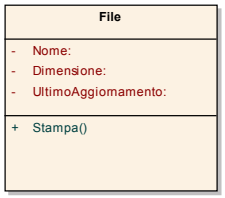
- Tipi di operatore
  - selettore (query)
    - accedono allo stato dell'oggetto senza alterarlo (es. "lunghezza" della classe coda)
  - Modificatore
    - alterano lo stato di un oggetto (es. "appendi" della classe coda)
  - Costruttore (*operatore di base)
    - crea un nuovo oggetto e/o inizializza il suo stato
  - Distruttore (*operatore di base)
    - distrugge un oggetto e/o libera il suo stato

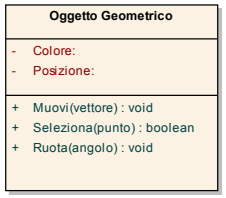
*realizzato con modalità diverse a seconda del linguaggio di programmazione

… Operazioni: Esempio









Visibilità degli Elementi di una Classe …

⇨ Definizione
- La visibilità di un elemento di una classe indica quali classi, oltre necessariamente a quella di appartenenza, possono accedere e usare l'elemento

⇨ Tipi di visibilità
- Pubblica
  - ogni classe esterna che abbia visibilità della classe di appartenenza dell'elemento considerato [l'elemento è preceduto dal simbolo +]
- Privata
  - nessuna oltre alla classe di appartenenza [l'elemento è preceduto da -]

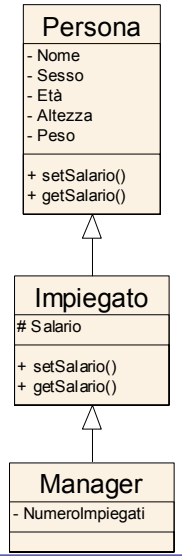
… Visibilità degli Elementi di una Classe …

- Protetta
  - qualsiasi classe specializzata dalla classe di appartenenza [l'elemento è preceduto da #]
- Package
  - tutte le classi che appartengono al package in cui è definita la classe di appartenenza [l'elemento è preceduto da ~]

… Visibilità degli Elementi di una Classe

⇨ Proprietà
- Se la visibilità non viene specificata non si opera alcuna assunzione
- Tipi addizionali di visibilità potrebbero essere definiti (in realtà tutte le forme di visibilità non pubblica sono language-dependent).

Esempio di Visibilità



Classi Essenziali

Classi Essenziali …

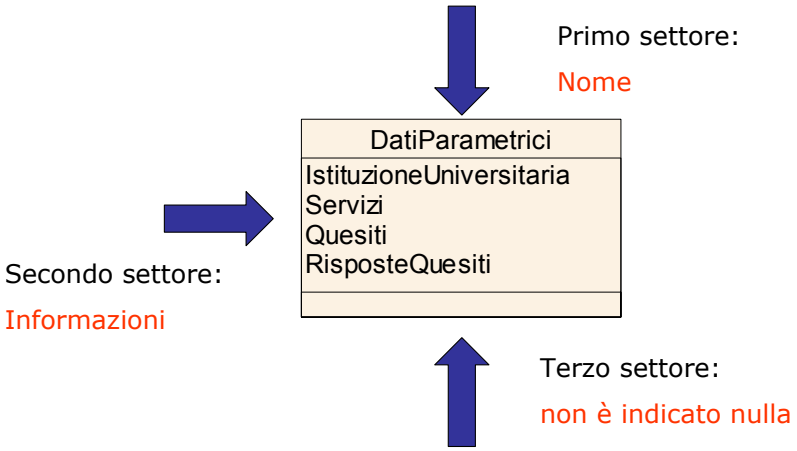
- Definizione di Classe Essenziale*
  - rappresenta una concettualizzazione di oggetti simili, i.e. oggetti del dominio applicativo che possiedono le stesse capacità (responsabilità) e le stesse informazioni che utilizzano e/o generano le capacità

*estensione dei concetti base dell'UML

Classi Essenziali: Notazione …

- analoga, dal punto di vista grafico, a quelle delle classi prevede però, sempre, la presenza di due soli settori
  - nome (della classe)
  - informazioni che utilizzano e/o generano le capacità della classe
- le capacità sono indicate mediante una descrizione testuale

Classi Essenziali: Esempio …



… Classi Essenziali: Esempio

- Descrizione testuale di DatiParametrici

  è responsabile della definizione e della descrizione dei Dati Parametrici necessari per la progettazione dei questionari. In particolare è responsabile
  - della descrizione dell'Università e delle sue istituzioni
  - dei servizi oggetto di valutazione del questionario, della loro codifica, del loro inventario
  - dei risultati di qualità desiderati ed accettabili
  - della definizione dei quesiti e per ognuno di questi, dove previsto, dell'insieme delle risposte

    * dati necessari per la progettazione dei questionari

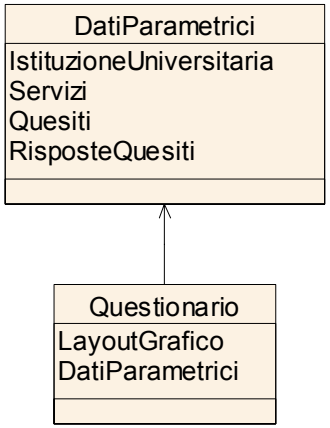
Diagramma delle Classi essenziali

- Definizione
  - Grafico che rappresenta le classi essenziali e le relazioni esistenti tra queste
- Tipi di relazione
  - è ammessa solo la relazione di associazione unidirezionale

Associazione unidirezionale …

- Definizione
  - indica la presenza di uno scambio di informazioni tra due classi, una richiedente e l'altra fornitrice
- La direzione della relazione è rappresentata da una linea con una freccia che punta alla classe fornitrice

… Associazione unidirezionale

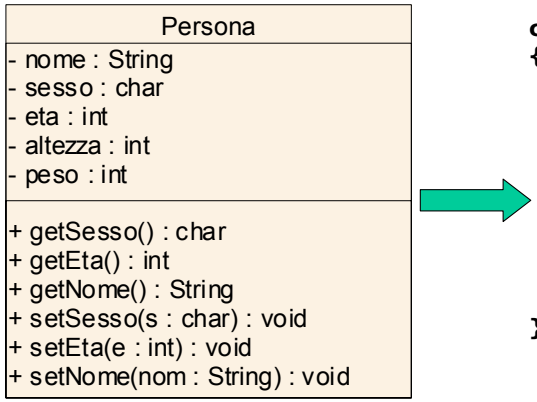


Implementazione fisica di una classe

Implementazione fisica di una classe …

- Esiste una corrispondenza tra la rappresentazione UML di una classe e la sua implementazione in un linguaggio Object Oriented

Implementazione fisica: Esempio in JAVA

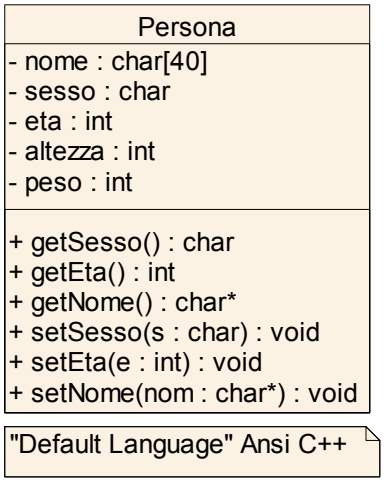



```
class Person
{
  public char     getSesso();
  public int      getEta();
  public String   getNome();
  public void     setSesso(char ses);
  public void     setEta(int a);
  public void     setNome(char nom);
  private char    sesso ;
  private int     eta, altezza, peso;
  private String  nome;
};
```

Implementazione in JAVA

Nel tool utilizzato, Rational Rose Enterprise Edition, JAVA è stato impostato come "Default Language"

Implementazione fisica: Esempio in C++




```
Class Person
{
  public:     //member functions
   char       getSesso();
   int        getEta();
   char*      getNome();
   void       setSesso(char s);
   void       setEta(int a);
   void       setNome(char *);
  private:    //data members
   char       Sesso ;
   int        eta, altezza, peso;
   char       nome[40];
};
```

Implementazione in C++

Diagramma delle Classi

Diagramma delle classi

- Grafico che fornisce una vista strutturale (statica) del sistema in termini di
  - classi
  - attributi
  - operazioni
  - relazioni tra classi
- Osservazione
  - non fornisce alcuna informazione temporale

Relazioni tra classi

- le principali relazioni tra classi sono
  - Associazione
    - Aggregazione
    - Composizione
  - Generalizzazione
  - Dipendenza

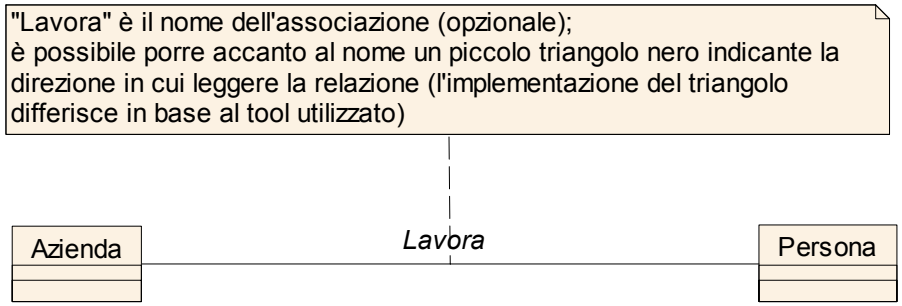
Relazione di Associazione

- Definizione
  - connessione semantica tra due o più classi che è tradotta in una connessione tra oggetti istanze delle classi
- Osservazione
  - è sempre bidirezionale

Tipi di Relazione di Associazione

- Le associazioni possono essere
  - binarie (coinvolgono due classi)
  - n-arie (coinvolgono tre o più classi, sono poco usate)
- Osservazione
  - una associazione binaria è un caso speciale di relazione n-aria con una propria notazione

Associazione binaria



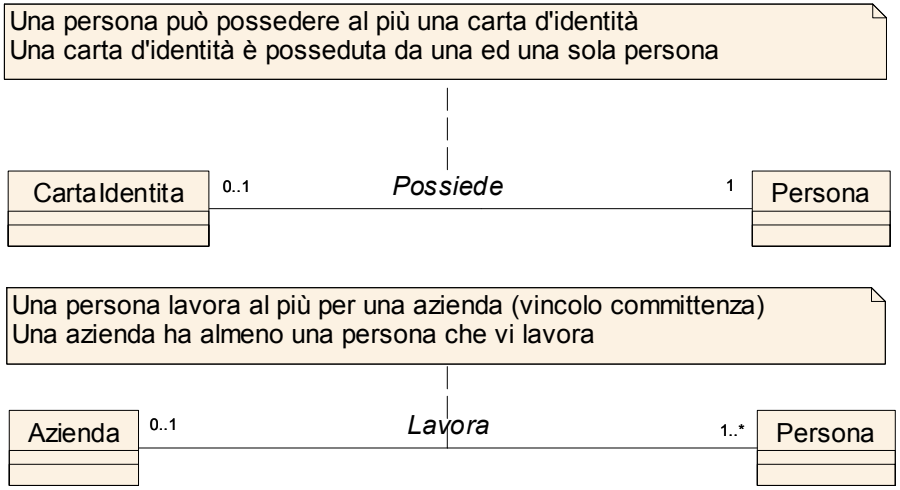
Molteplicità nelle Associazioni ...

- specifica quante istanze di una classe possono essere associate con una singola istanza di un'altra classe
- Osservazione
  - se la molteplicità non è indicata non si opera alcuna assunzione

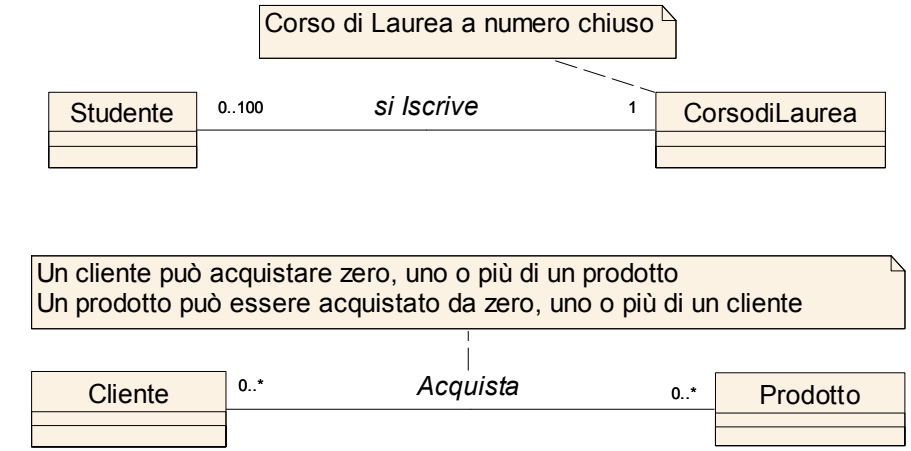
… Molteplicità nelle Associazioni

- Notazione
  - stringa di testo che comprende sequenze di intervalli di interi separate da una virgola
  - un intervallo, potenzialmente illimitato, è nel formato
    - **limite inferiore..limite superiore**
- Esempi di molteplicità
  - 0..1 (da zero ad una istanza)
  - 0..* (zero o più istanze)
  - 1..* (una o più istanze)
  - 1..6 (da una a sei istanze)
  - 1..3, 7..10, 15, 19..*

Molteplicità nelle Associazioni: Esempi …



… Molteplicità nelle Associazioni: Esempi



Ruoli nelle Associazioni …

- I ruoli forniscono una modalità per attraversare relazioni da una classe ad un'altra
  - sono posti alla fine della relazione con l'altra classe
  - possono essere usati in alternativa ai nomi delle associazioni
  - sono spesso usati per relazioni tra oggetti della stessa classe

Ruoli nelle Associazioni: Esempio …

- indica il ruolo che la classe riveste nell'associazione
  - Osservazione
    - il nome di un ruolo è espresso da un sostantivo mentre il nome di una associazione da un verbo

… Ruoli nelle Associazioni: Esempio

Implementazione associazione …

- Generalità
  - non esiste un mapping diretto su un costrutto di un linguaggio di programmazione
  - può essere implementata introducendo degli attributi ad hoc che contengono dei riferimenti alle istanze delle classi associate
  - il riferimento a più istanze di una classe può essere modellato con liste, array, etc.

… Implementazione associazione …

- Per realizzare l'associazione tra gli oggetti occorre che entrambe le classi siano dotate di una operazione la cui semantica sia "crea un riferimento all'altra classe"; tale operazione riceve il riferimento della classe con la quale creare l'associazione. all'interno di un'altra classe si potrà dunque:
  - creare entrambi gli oggetti
  - associare un oggetto all'altro passando prima all'uno e poi all'altro il riferimento dell'altro oggetto tramite l'operazione "crea un riferimento all'altra classe"

… Implementazione associazione

- Osservazione
  - l'associazione si realizza solo a run-time
- Direzionalità dell'Associazione
  - in fase di implementazione si può decidere di codificarla solo in una delle due direzioni, in questo modo però si può percorrere l'associazione solo nella direzione implementata

Implementazione uno a uno …

```
class A {
    private B associazioneX; //nome associazione o nome ruolo
    public   associaPartner(B partner);
};
class B {
    private A associazioneX; //nome associazione o nome ruolo
    public   associaPartner(A partner);
};
```


Implementazione uno a uno (JAVA)

```
class CartaIdentita {
    private     Persona possessore;
    private     String numero;
    public void  associaPersona(Persona partner);
    CartaIdentita (String numcarta);
};
class Persona {
    private     CartaIdentita documento;
    private String nome;
    public void  associaCartaIdentita(CartaIdentita partner);
    Persona (String nomPersona);
};
```


Uso dell'associazione uno a uno (JAVA)

```
{
    //istanziazione delle variabili
    CartaIdentita cartiden = new
      CartaIdentita("AC1234");
    Persona pers = new Persona("Pasquale");
    //creazione dell'associazione tra i due oggetti
    cartiden.associaPersona(pers);
    pers.associaAzienda(cartiden);
    // … segue
};
```

Implementazione uno a uno (C++)

```
class CartaIdentita {
    private:
        Persona* possessore;
        char numero[20];
    public:
        CartaIdentita (char* numcarta);
        void associaPersona(Persona* partner);
};
class Persona {
    private:
        CartaIdentita* documento;
    public:
        void associaCartaIdentita(CartaIdentita* partner);
};
```


Uso dell'associazione uno a uno (C++)

```
{
   //istanziazione delle variabili
   CartaIdentita cartiden = new CartaIdentita("AC1234");
   Persona pers = new Persona("Pasquale");
   //creazione dell'associazione tra i due oggetti.
   cartiden.associaPersona(&pers);
   pers.associaCartaIdentita(&cartiden);
   // … segue
};
```


Implementazione uno a molti …

- una associazione uno a molti tra una classe A ed una classe B descrive una connessione semantica per cui a ciascun oggetto di A possono corrispondere zero, uno o più di un oggetto di B mentre ad un oggetto di B corrisponde al più un oggetto di A
  - la classe che partecipa con molteplicità 1 nell'associazione rappresenta la relazione attraverso una variabile di tipo lista di riferimenti alla classe associata
  - se è noto il numero massimo di oggetti associati si usa una lista statica, in caso contrario una lista dinamica

… Implementazione uno a molti

```
Class A {
      private Lista_a_B; //lista di oggetti di B
      public   aggOggetto (B newobj);
};
Class B {
      private  A associazioneX; //nome associazione o
      nome ruolo
      public   associaPartner(A partner);
};
```

Implementazione uno a molti (JAVA)

```
class CorsodiLaurea {
    private ArrayList studenti;
    private String denominazione;
    public void aggStudente (Studente newstud);
    CorsodiLaurea (String denomcorso);
};
class Studente {
    private CorsodiLaurea iscrizione;
    private String nome;
    public void associaCdl(Laurea partner);
    Studente(String nome, String numeromat);
};
```


Uso dell'associazione uno a molti (JAVA)

```
{
    //istanziazione delle variabili
    CorsodiLaurea cdl = new CorsodiLaurea(ICD);
    Studente stud1 = new Studente("Pasquale", "123456");
    Studente stud2 = new Studente("Nicola", "123457");
    // …
    //creazione dell'associazione tra i due oggetti.
    stud1.associaCdl(cdl);
    stud2.associaCdl(cdl);
    cdl.aggStudente(stud1);
    cdl.aggStudente(stud2);
    // … segue
};
```


Implementazione uno a molti (C++)

```
Class CorsodiLaurea {
        private:
            Studente *studenti[];
            studiscritti int; /* a run-time indicherà il numero di
            studenti iscritti */
        public:
            CorsodiLaurea (int numStud);
            void aggStudente (Studente* newstud);
};
Class Studente {
        private:
            CorsodiLaurea iscrizioneLaurea;
        public:
            void associaCdl(Laurea* partner);
};
```

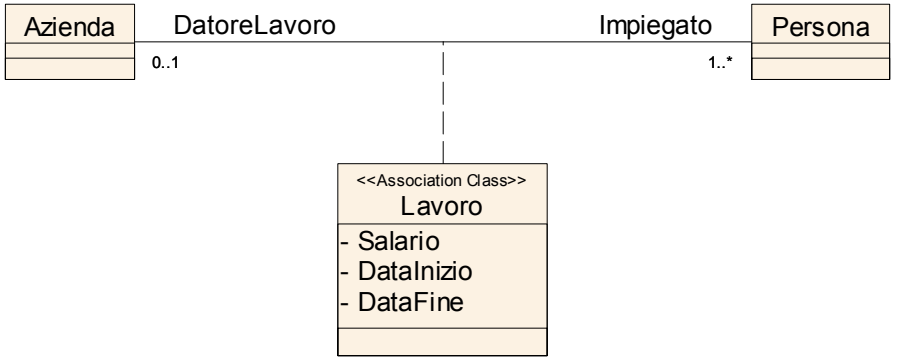

Uso dell'associazione uno a molti (C++)

```
{
    //istanziazione delle variabili
    CorsodiLaurea cdl = new CorsodiLaurea(ICD);
    Studente stud1 = new Studente("Pasquale", "123456");
    Studente stud2 = new Studente("Nicola", "123457");
    //creazione dell'associazione tra i due oggetti.
    stud1.associaCdl(&cdl);
    stud2.associaCdl(&cdl);
    cdl.aggStudente(&stud1);
    cdl.aggStudente(&stud2);
    // … segue
};
```

Classe Associativa …

- A volte le proprietà di un'associazione sono proprie della relazione e non delle classi coinvolte, pertanto può essere opportuno definire una classe associativa
- Proprietà
  - è utilizzata per modellare proprietà delle associazioni
  - ogni attributo di una classe associativa contiene un valore per ogni istanza di associazione
  - è utilizzata, in genere, nelle associazioni molti a molti

… Classe Associativa: Esempio



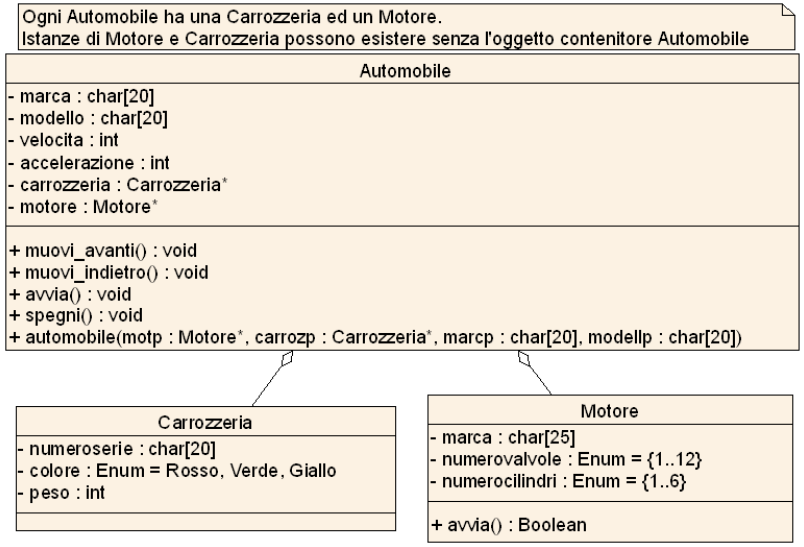
Relazione di Aggregazione

- è una forma speciale di associazione, molto diffusa, che definisce una relazione tra un "tutto" ed un insieme di parti di cui il "tutto" è costituito (*whole-part relationship*)
- Osservazione
  - il "tutto" corrisponde ad una classe contenitore mentre ogni parte corrisponde ad una classe contenuta

Proprietà

- Siano A,B,C tre classi, valgono le seguenti proprietà
  - transitiva: se A è parte di B e B è parte di C allora A è parte di C
  - antisimmetrica: se A è parte di B allora B non è parte di A
- a *run-time* un oggetto contenuto sopravvive all'oggetto contenitore

Esempio



Implementazione

- Per realizzare l'aggregazione è necessario
  - definire dei riferimenti alle istanze aggregate nella parte privata dalla classe aggregante
    - il numero di riferimenti dipende dalla molteplicità
- oppure
  - aggiungere una opportuna struttura attraverso la quale mantenere le istanze aggregate
- ed infine
  - definire delle operazioni per esportare i riferimenti delle istanze aggregate

Implementazione dell'Esempio (C++) …

```
//file Automobile.h; inoltre sono da includere gli
   header file "Motore.h" e "Carrozzeria.h"
class Automobile {
    public:
        //il costruttore riceve un puntatore per ognuno degli
          oggetti aggregati; una parte del codice è omessa
        Automobile(… , Motore* motp, Carrozzeria* carrozp);
        // seguono il distruttore e le altre operazioni
    private:
        Carrozzeria* carrozzeria; //traduzione aggregazione
        Motore* motore; //traduzione aggregazione
        // altri attributi da definire
};
```


… Implementazione dell'Esempio (C++)…

```
#include "Automobile.h"
main () { //usa classe Automobile
    //definisce e inizializza oggetto di tipo Motore
    Motore m("AZ123","Ferrari", 6,12,1000);
    //definisce e inizializza oggetto di tipo Carrozzeria
    Carrozzeria c("12345ASA", "RossoFerrari",1500);
    //definisce e inizializza puntatore a motore
    Motore* puntatoreMotore=&m;
    //definisce e inizializza puntatore a carrozzeria
    Carrozzeria* puntatoreCarrozzeria=&c;
    //definisce e inizializza oggetto Automobile
    Automobile
      auto1(puntatoreCarrozzeria,puntatoreMotore);
} //al termine sono separatamente distrutti gli oggetti m,c e auto1
```

Relazione di Composizione

- è un caso particolare di aggregazione in cui gli oggetti contenuti (componenti) non hanno vita propria ma esistono in quanto parte della classe contenente (composta)
- Proprietà
  - la classe contenente è responsabile della creazione e della distruzione degli oggetti contenuti
  - le classi contenute sono di uso esclusivo della classe componente
  - la molteplicità dal lato della classe composta deve essere al più uguale ad uno, mentre può essere qualsiasi per le classi componenti

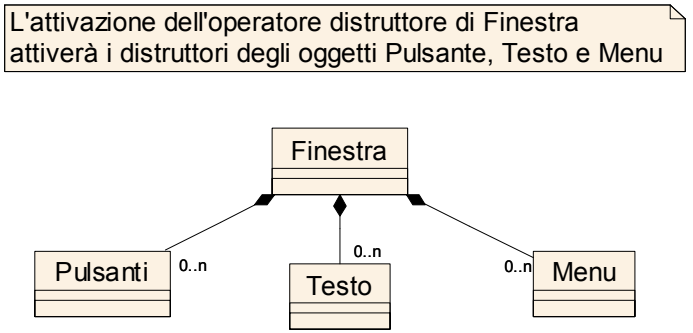
Implementazione

- Per tradurre tale relazione si deve:
  - aggiungere una variabile membro alla classe composta del tipo della classe componente;
  - implementare il costruttore della classe componente in modo da richiamare il costruttore della classe composta
  - il distruttore, o un suo sostitutivo, della classe composta provvederà a distruggere anche gli oggetti componenti
- Osservazione
  - in JAVA non esiste il distruttore

Implementazione

```
class Contenitore {
    private:
        Contenuto oggContenuto;
    public:
        //costruttore: costruisce oggetto x di tipo X,
        Contenitore(int i): oggContenuto(i){}; //richiama costruttore
          di Contenuto
        //se esiste è richiamato il distruttore
        ~C3(); //richiama implicitamente il distruttore di X
        //operazione che lavora sull'oggetto contenuto
        void fc() {
          x.fx(); //opera su oggetto x; fx è un'operazione di x
        }
}
```


Esempio



Implementazione (C++)

```
#include "Testo.h"
class Finestra {
    private:
        //struttura dati
        Testo oggTesto;
        //…
    public:
        Finestra(int i): testo(char[100] testp){};
        ~Finestra();
        void visualizzafinestra() {
          oggTesto.formattatesto()
        }
}
```


Implementazione (JAVA)

```
class Finestra {
    private Testo oggTesto;
    public Finestra(int i) {
      // il garbage collector "sa" rilasciare la memoria allocata
      con la new
      oggTesto = new Testo; //
      //…
    }
    public void visualizzafinestra() {
          oggTesto.formattaTesto();
        }
}//il garbage collector al rilascio della memoria di un oggetto
   Finestra rilascerà anche la memoria degli oggetti componenti
```


Relazione di Generalizzazione …

⇨ Definizione
- relazione tassonomica tra una classe più generale ed una o più classi specifiche

⇨ Proprietà
- le classi specifiche sono totalmente consistenti con la classe più generale
- la classe generale è detta superclasse
- ogni classe specializzata è detta sottoclasse
- la relazione può essere letta come
  - "è un tipo di" (verso di generalizzazione)
  - "può essere un" (verso di specializzazione)

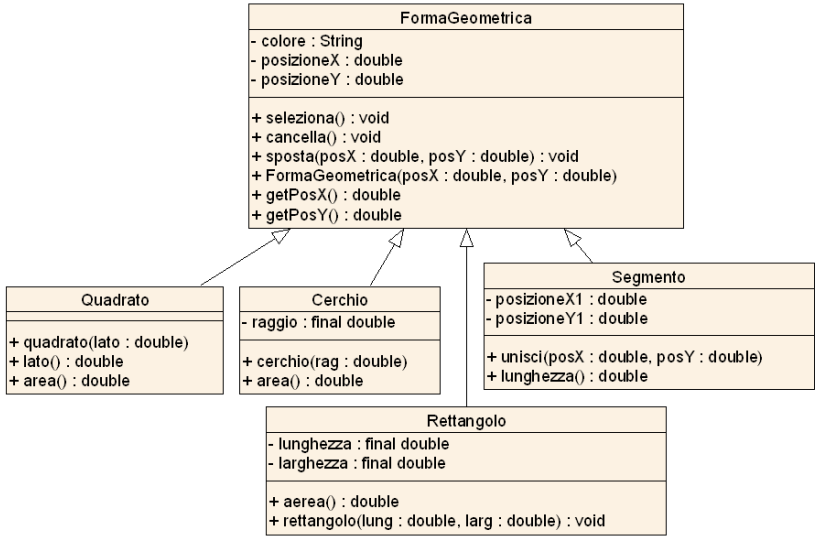
… Relazione di Generalizzazione

- le classi specializzate incorporano (ereditano) la struttura ed il comportamento delle classi più generali
- l'ereditarietà è il meccanismo di condivisione delle proprietà degli oggetti in una gerarchia di generalizzazione
  - l'ereditarietà gode della proprietà transitiva
- è supportata dai costrutti dei linguaggi di programmazione object-oriented

Differenze tra superclasse e sottoclasse

- la sottoclasse può
  - contenere nuovi attributi e operazioni non inclusi nella superclasse
  - ridefinire i metodi delle operazioni della superclasse (*overriding*)

Ereditarietà: Esempio …



… Ereditarietà: Esempio (JAVA) …

```
class FormaGeometrica {
    private String colore;
    private double posizioneX;
    private double posizioneY;
    public void seleziona();
    public void cancella();
    public void sposta(double posX, double posY);
};
```


… Ereditarietà: Esempio (JAVA)

```
class Cerchio extends FormaGeometrica {
    final double raggio;
    Cerchio(double rag) { this.raggio = rag;}
    double area() { return Math.PI * raggio*raggio;}
};
class Quadrato extends FormaGeometrica {
    Quadrato(double lato) {super(lato, lato);}
    double lato() {return lunghezza; }
};
class Segmento extends FormaGeometrica {
    public unisci (double posX, double posY);
};
```

… Ereditarietà: Esempio (JAVA)

```
class Rettangolo extends FormaGeometrica
  {
   final double lunghezza;
   final double larghezza;
   Rettangolo(double lung, double larg) {
   this.lunghezza = lung;
   this.larghezza = larg;}
   double area() { return lunghezza * larghezza; }
}
```


Uso dell'ereditarietà (JAVA)

```
public class FigGeometricaTest {
   public static void main(String args[]) {
      FiguraGeometrica rettangolo = new Rettangolo(1.0, 1.0,
      2.0, 3.0);
      FiguraGeometrica cerchio = new Cerchio(2.0, 2.0, 2.0);
      FiguraGeometrica segmento = new Segmento (1.5,
      1.5);
      System.out.println("Area Rettangolo: " +
      rettangolo.area())
      System.out.println("Area Cerchio: " + cerchio.area() +
      "Centro:");
      segmento.unisci (2.0, 2.0);
      System.out.println("Lunghezza segmento: " +
      segmento.lunghezza());}
}
```


Classe Astratta …

⇨ Definizione
- una classe si dice astratta (*abstract)* se ha almeno un'operazione il cui metodo non è definito; anche l'operazione in questione si dice astratta

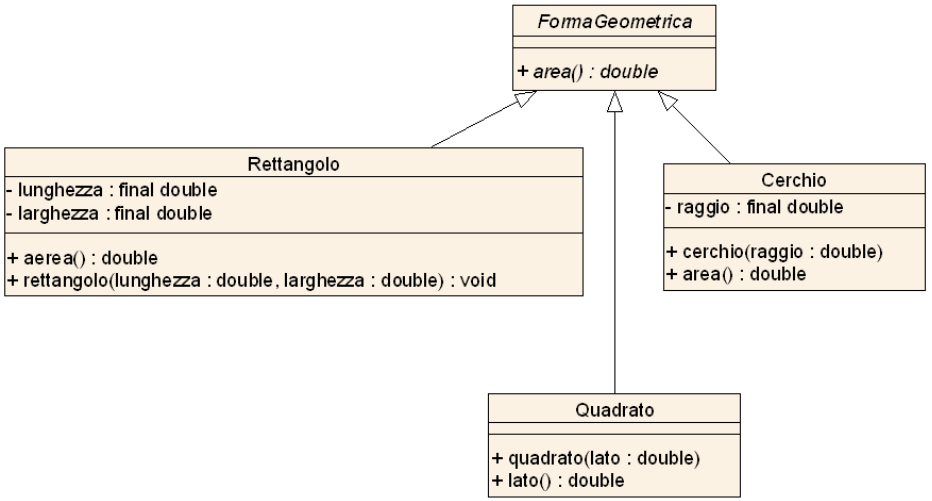
⇨ Proprietà
- non è direttamente implementabile
- deve avere delle sottoclassi
- ogni sottoclasse di una classe astratta eredita le operazioni astratte e ne realizzare l'implementazione (metodo), altrimenti essa stessa diventa una classe astratta

… Classe Astratta

è usata per
- mettere a fattor comune un'astrazione di un certo tipo
- favorire il riuso

⇨ Osservazione
- non tutti i linguaggi di programmazione possiedono un costrutto per la definizione di una classe astratta
- in C++ una classe che contiene almeno una operazione virtuale pura si definisce astratta

Classe Astratta: Esempio …



… Classe Astratta (JAVA) …

```
abstract class FiguraGeometrica {
    abstract double area();
}
class Cerchio extends FiguraGeometrica {
    final double raggio;
    Cerchio(double raggio) { this.raggio = raggio;}
    double area() { return Math.PI *
      raggio*raggio;}
} //esempio tratto da EFFECTIVE JAVA, Programming Language
  Guide, by Joshua Bloch,Foreword by Guy Steele, pag.101
```


… Classe Astratta (JAVA)

```
class Rettangolo extends Shape {
    final double lunghezza;
    final double larghezza;
    Rettangolo(double lunghezza, double larghezza) {
    this. lunghezza = lunghezza;
    this. larghezza = larghezza;}
    double area() { return lunghezza * larghezza; }
}
class Quadrato extends Rectangle {
    Quadrato(double lato) {super(lato, lato);}
    double lato() {return lunghezza; }
} //esempio tratto da EFFECTIVE JAVA, Programming Language Guide, by
  Joshua Bloch,Foreword by Guy Steele, pag.101
```


Polimorfismo

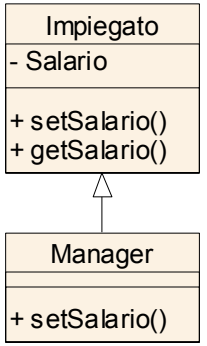
- se due o più sottoclassi ereditano un'operazione (anche astratta) dalla stessa superclasse, la implementano o ne modificano l'implementazione ereditata, assegnano a tale operazione la proprietà del polimorfismo (*polys*=molto + *morphè*=forma)

- Proprietà
  - un'operazione polimorfica ha un comportamento diverso per ogni classe in cui ne è (ri)definito il metodo

Tipo di Polimorfismo: Overriding

- Overriding
  - ridefinizione in una classe specializzata dei metodi di una o più operazioni della superclasse

Esempio di Overriding



Tipo di Polimorfismo: Overloading

- Overloading
  - presenza in una stessa classe di operazioni accomunate dallo stesso nome ma con differenti metodi e aree di passaggio parametri (*firme*)
- Osservazione
  - un operatore tipicamente *overloaded* è il costruttore
  - il compilatore "sa" quale operazione richiamare basandosi sulla *firma*

Esempio di Overloading built-in (JAVA)

- l'operatore + è overloaded in JAVA; è infatti usato per la
- somma di interi
  - int i, j;
  - i = 3; j = 5;
  - i = i + j; // i conterrà il valore 8
- concatenazione di stringhe
  - String strtesta = new("Linguaggio");
  - String strcoda = new("JAVA");
  - strtesta +=strcoda; // strtesta conterrà la stringa "LinguaggioJAVA"

Utilità dell'Overloading (C++)

- In C esistono tre funzioni che calcolano il valore assoluto di un numero
  - abs() // valore assoluto di un int
  - labs() // valore assoluto di un long
  - fabs() // valore assoluto di un double
- In C++ esiste solo
  - abs() // valore assoluto di un int, long, double
- Vantaggio
  - è il compilatore che "sceglie" il metodo specifico permettendo al programmatore di ricordare solo l'azione generale

Esempio di Overloading (JAVA)

```
class Persona {
    String nome, cognome;
    Persona ();
    Persona (String nome, String cognome);
    //…
};
// uso dei due costruttori
Persona pers = new Persona();
Persona pers1 = new Persona("Pasquale",
  "Ardimento");
```


Overloading: Differenze tra JAVA e C++

- in Java é possibile definire più versioni di una operazione o di un costruttore definendo *firme* differenti mentre in C++ è possibile modificare anche il tipo di dato restituito con la condizione però che la *firma* sia unica
- in Java, al contrario del C++, non é consentito l'overloading degli operatori matematici, relazionali, logici, etc

Interfaccia di una Classe …

- Interfaccia
  - elenco delle operazioni di una classe visibili all'esterno
  - definisce un protocollo di comportamento che può essere implementato da qualunque classe in qualunque punto della gerarchia
- Caratteristiche
  - tutti gli oggetti possiedono un'interfaccia
  - ogni interfaccia, spesso, specifica solo una parte del comportamento di una classe
  - una interfaccia non ha implementazione, stati o associazioni

… Interfaccia di una Classe

- ogni oggetto è in grado di soddisfare solo tipi ben definiti di richieste, specificati dalla propria interfaccia
- è formalmente equivalente ad una classe astratta e la si può considerare come una classe astratta
- può essere rappresentata con lo stesso simbolo di una classe senza i *compartimenti* degli attributi e delle operazioni oppure mediante un piccolo cerchio al di sotto del quale è posto il nome
- non tutti i linguaggi presentano costrutti per realizzare le classi astratte, e.g. C++ non le supporta

Differenze tra Classe Astratta ed Interfaccia di una classe

- ⇨ al contrario di una classe astratta una interfaccia non può implementare alcuna operazione
- ⇨ una classe può implementare molte interfacce ma può avere soltanto una superclasse
- ⇨ una interfaccia non è parte della gerarchia di una classe. Classi non in relazione tra loro possono implementare la stessa interfaccia

Esempio di Interfaccia di una Classe

```
public interface List extends Collection {
    //manipolazione degli elementi basata sulla loro posizione numerica nella
      lista
    Object get(int index);
    Object set(int index, Object element);
    void add(int index, Object element);
    Object remove(int index);
    abstract boolean addAll(int index, Collection c);
    // Ricerca di uno specifico elemento e restituzione della sua posizione
    int indexOf(Object o);
    int lastIndexOf(Object o);
    //…
} // in JAVA esistono due implementazioni: l'ArrayList e la LinkedList
   (dal Tutorial "The Really Big Index")
```


Relazione di Dipendenza

- ⇨ Definizione
  - indica una connessione "semantica" tra due classi in cui una delle due è dipendente dall'altra
- ⇨ Osservazione
  - una modifica nell'elemento indipendente ha effetti su quello dipendente
  - la dipendenza può essere di varia natura

Relazione di Dipendenza

- In UML sono citate diverse relazioni di dipendenza alcune delle quali sono
  - <<friend>> indica la possibilità di accesso al contenuto di un'altra classe indipendentemente dalla visibilità prevista dalla classe target (utilizzata in C++)
  - <<use>> indica che un elemento richiede la presenza di un altro per il suo corretto funzionamento; potrebbe essere stereotipata come
    - <<call>> una operazione di una classe chiama una operazione di un'altra classe di cui possiede un riferimento
    - <<instantiate>> una operazione di una classe crea istanze di un'altra classe

Relazione <<friend>>: Esempio …

```
class monete {
    enum unità {penny, euro, dollaro, sterlina};
    friend class totale;
};
class totale {
      monete:: unità denaro;
    public:
      void setm();
      void getm();
}tot;
```


… Relazione <<friend>>: Esempio

```
void amount::setm()
{
    denaro = monete::euro;
}
Main()
{
    tot.setm();
    cout<<tot.getm() //visualizza il numero 1
    Return 0;
}
```

- quando una classe è friend di un'altra ha accesso agli attributi definiti nell'altra classe ma non ne eredita né gli attributi né le operazioni
- nella pratica sono raramente impiegate

Diagrammi di Interazione

Diagramma di Sequenza
Diagramma di Collaborazione

Interazione

- una interazione specifica i dettagli della comunicazione che dovrebbe aver luogo per realizzare un particolare compito
- una comunicazione corrisponde ad un messaggio, che definisce i ruoli (delle istanze) di chi lo invia e di chi lo riceve, e all'azione che causerà la comunicazione

Diagramma di sequenza …

- Definizione
  - grafico che mostra un'interazione o un insieme di interazioni tra due o più oggetti mediante una sequenza temporale di azioni
- Osservazione
  - i diagrammi di sequenza appartengono alla Vista dei casi d'uso e modellano le singole istanze di ogni caso d'uso (scenari)

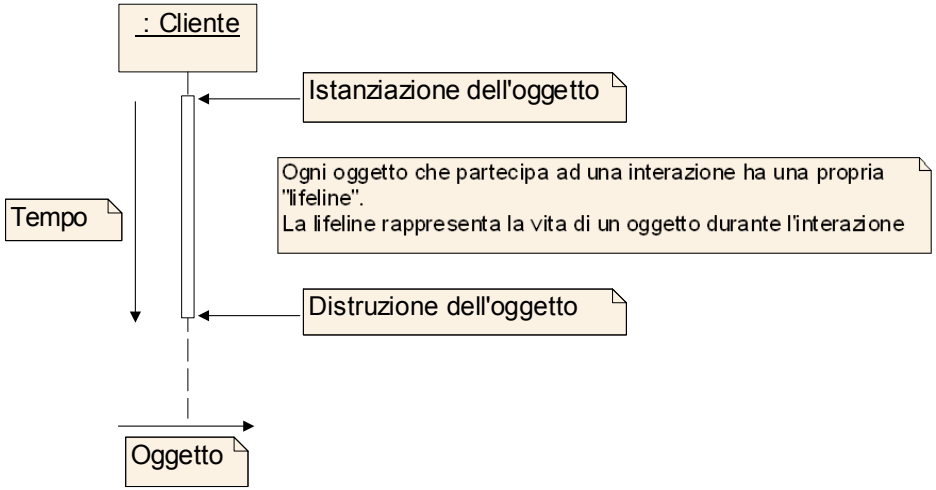
… Diagramma di sequenza

- ogni scenario di base è modellato da un singolo diagramma di sequenza
- ogni scenario alternativo può essere modellato in uno dei due seguenti modi
  - un singolo diagramma di sequenza
  - all'interno del diagramma di sequenza dello scenario base

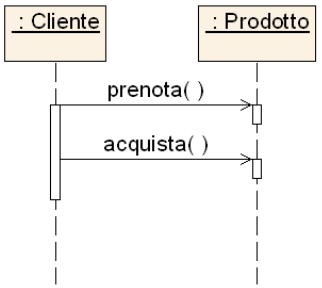
Elementi di un Diagramma di sequenza

- Tempo
- Oggetti
- *Lifeline* (periodo di vita) degli oggetti
  - costruzione dell'oggetto
  - distruzione dell'oggetto
- Messaggi
  - da un oggetto verso un altro oggetto
  - da un oggetto verso sé stesso (Auto-Delega)
- Valore di Ritorno (di un messaggio)
- Iterazione

Lifeline di un Oggetto

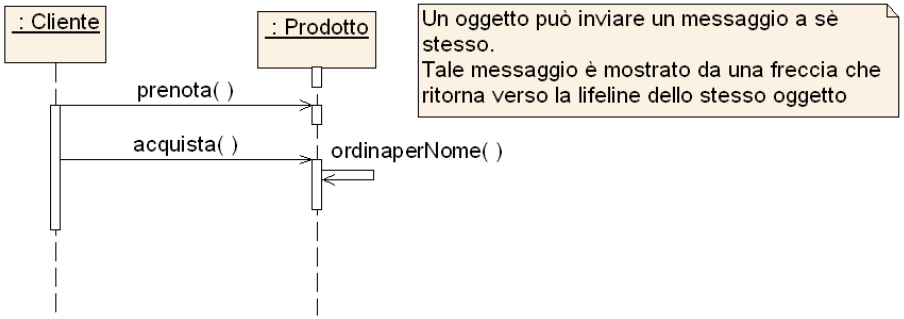


Messaggi



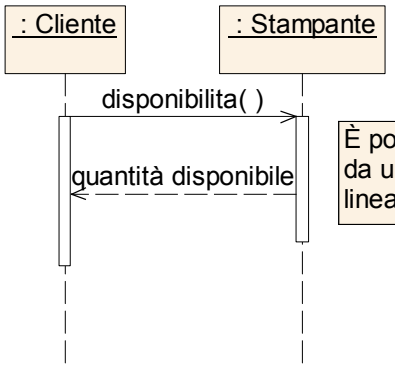
Un messaggio rappresenta una comunicazione tra oggetti.
I messaggi includono il nome del messaggio, la lista di argomenti (opzionale) ed informazioni di controllo (opzionale).
L'ordine con il quale si presentano i messaggi è mostrato dall'alto verso il basso

Auto-Delega



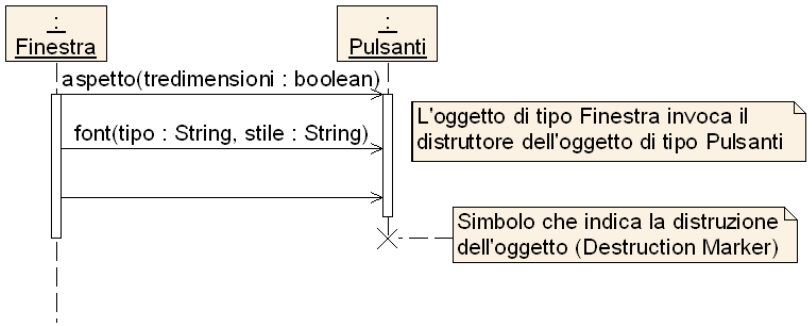
Un oggetto può inviare un messaggio a sè stesso.
Tale messaggio è mostrato da una freccia che ritorna verso la lifeline dello stesso oggetto

Valore di ritorno

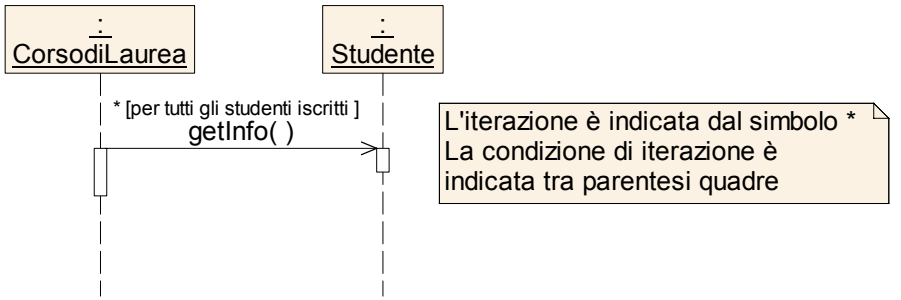


È possibile indicare il valore restituito da un messaggio (mostrato da una linea tratteggiata)

Distruzione di un oggetto



Iterazione



Esempio: Richiedere in prestito un libro

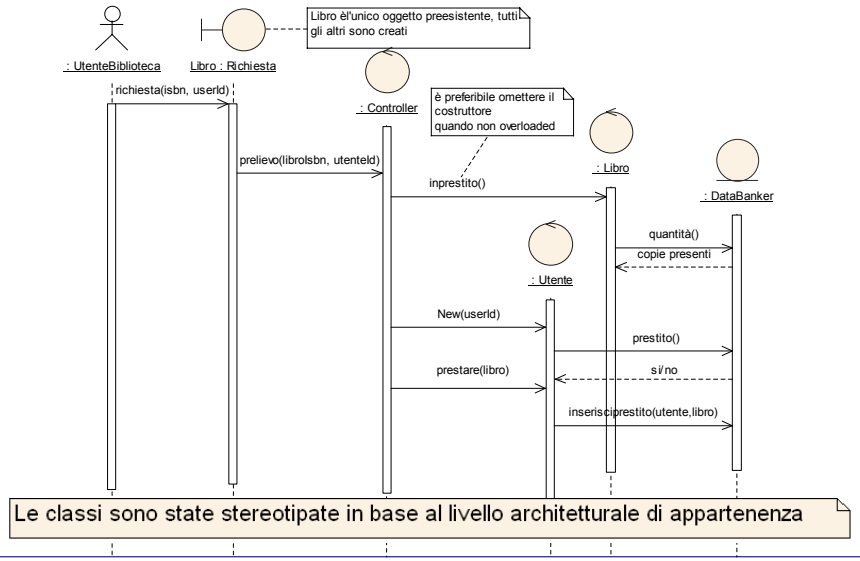


Diagramma di Collaborazione

Collaborazione

⇨ Definizione
- descrive come una operazione o un classificatore, ad esempio un caso d'uso, è realizzato da un insieme di classificatori e associazioni usate in uno specifico modo

⇨ Osservazione
- la descrizione di un caso d'uso è rivolta ai casi d'uso e alle associazioni in generale
- la descrizione di un'operazione include gli argomenti e le variabili locali dell'operazione, così come le associazioni ordinarie legate al classificatore che possiede l'operazione

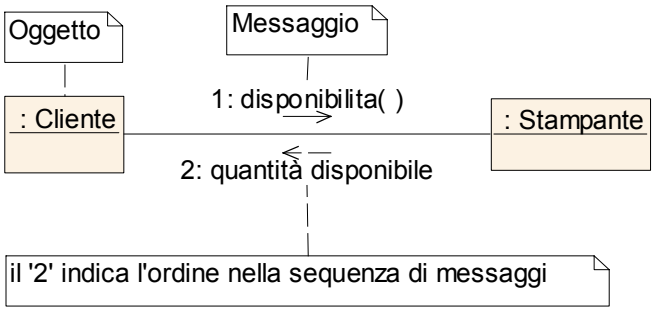
Diagramma di collaborazione

⇨ Definizione
- è un grafico che mostra o una collaborazione, che contiene un insieme di ruoli che devono essere rivestiti dalle istanze dei classificatori e dalle loro relazioni, o un istanza o un insieme di istanze di collaborazioni che contengono una collezione di istanze e relazioni

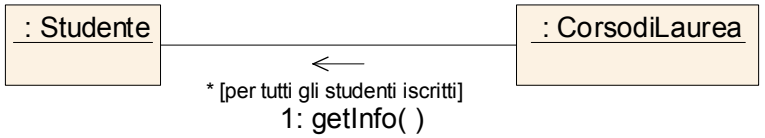
Elementi di un diagramma di collaborazione

- Oggetti
- Messaggi
  - da un oggetto verso un altro oggetto
  - da un oggetto verso sé stesso (Auto-Delega)
- Numeri di sequenza
- Iterazione

Messaggi

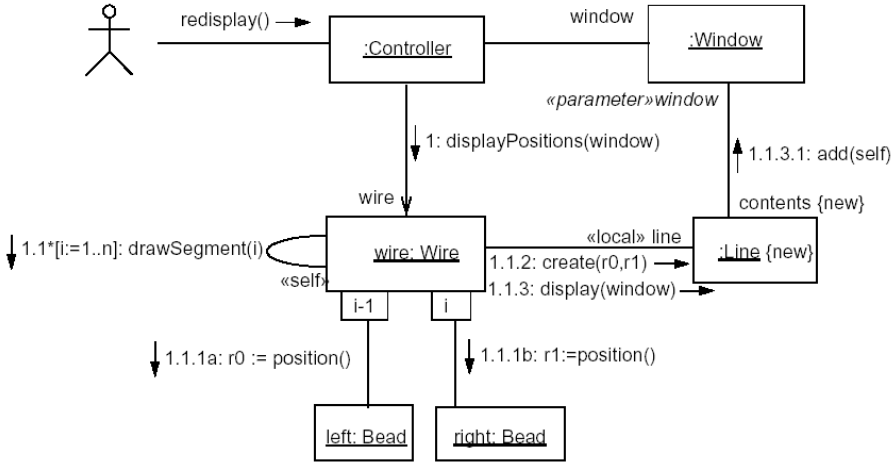


Iterazione



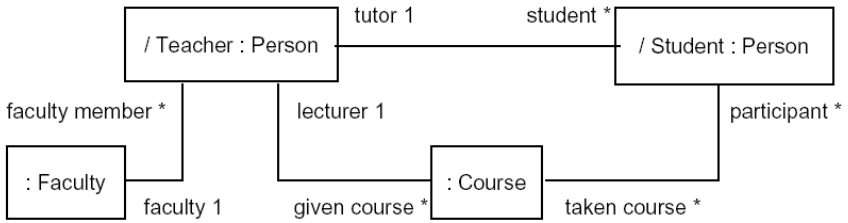
La sintassi per definire l'iterazione nei diagrammi di collaborazione è simile a quella usata nei diagrammi di sequenza

Esempio a livello di istanze



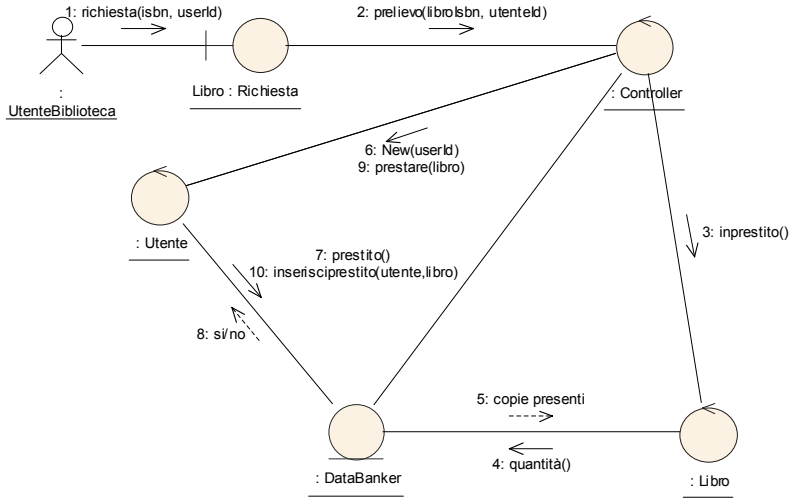
Tratto da UML versione 1.5 (pag.516)

Esempio a livello di specifica



Tratto da UML versione 1.5 (pag.516)

Esempio: Richiedere in prestito un libro



Sequenza vs Collaborazione

- i diagrammi di sequenza pongono più enfasi sulla sequenza temporale delle operazioni
  - è preferibile usarli per specifiche *real-time* e per scenari complessi
- i diagrammi di collaborazione indicano le relazioni tra oggetti

Diagramma degli Stati

Diagramma degli Stati …

- Definizione
  - è un grafico con nodi ed archi in cui i nodi rappresentano gli stati di una classe e gli archi, direzionali, rappresentano le transizioni di stato
- Semantica
  - mostra il comportamento di una classe mediante gli stati che può assumere e le sue reazioni (cambiamenti di stato) al verificarsi di cause esterne (eventi)

… Diagramma degli Stati

- Osservazioni
  - tipicamente usato per descrivere il comportamento di istanze di classi
  - ogni arco ha il nome dell'evento che genera la transizione che esso rappresenta
  - una transizione può essere controllata attraverso una condizione. In tal caso una condizione si può verificare se accade l'evento e se è vera la condizione che la controlla

Stato …

- Definizione
  - Condizione che un oggetto ha nel corso della propria vita o di una interazione durante la quale un oggetto soddisfa delle condizioni, effettua delle azioni o attende qualche evento
- Osservazione
  - concettualmente, un oggetto resta in uno stato per un intervallo di tempo. Comunque, la semantica permette di modellare stati attraverso dei flussi così come transizioni che non sono istantanee

… Stato

- lo stato influenza il comportamento
  - l'oggetto può reagire diversamente ad uno stesso evento esterno in base al proprio stato
- ad esempio
  - l'iscrizione di uno studente ad un corso di laurea a numero di chiuso dipende dallo stato delle iscrizioni (periodo, disponibilità di posti, numero di iscritti, etc..)
- inoltre uno stato perdura nel tempo
  - finché non si verifica un evento che ne causa il cambiamento (es. un prestito di un libro muta lo stato di libro da 'presente' in 'in prestito')

Il cambio di stato causato da un evento si chiama transizione

Elementi descrittivi di uno Stato

- Nome
  - identificatore dello stato [obbligatorio]
- Variabili di Stato
  - attributi che descrivono lo stato. Qualche volta questi sono temporanei (ad esempio un contatore) [opzionale]
- Sequenza di eventi che determina lo stato
- Condizioni che caratterizzano lo stato

Eventi …

- Definizione
  - un evento è qualcosa che accade in un istante; ad esempio il volo AZ321 parte da Bari
- Osservazione
  - l'istante in cui accade un evento è un attributo implicito di tutti gli eventi
  - un evento può essere solo un segnale di temporizzazione, altri eventi trasportano informazioni tra due processi

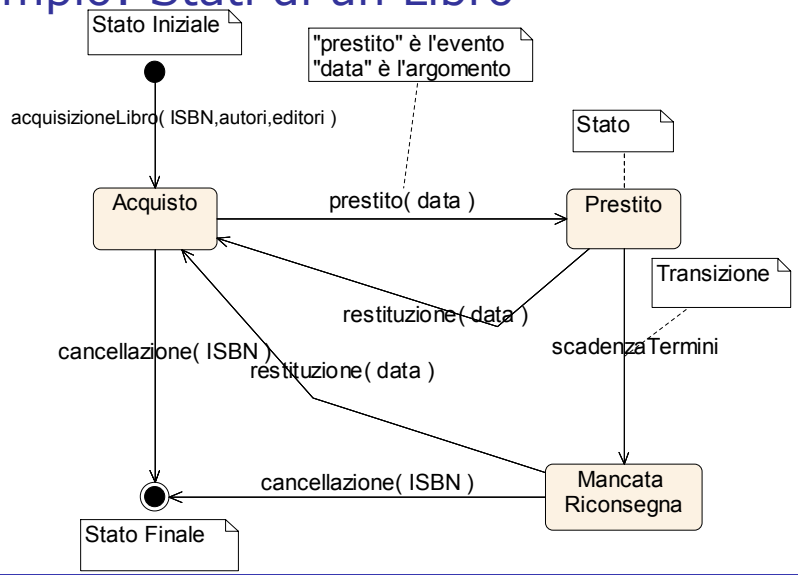
… Eventi

⇨gli eventi possono essere raggruppati in classi con struttura e comportamento comuni: partenze dei voli è una classe a cui appartengono tutti gli eventi di partenze

Identificazione degli stati

- per ciascun oggetto occorre identificare tutti gli stati in cui può trovarsi
- trascurare gli attributi ininfluenti
- alcuni attributi non modificano in modo qualitativo il comportamento di un oggetto, ma al più i valori degli eventi che l'oggetto produce.
- individuare le condizioni limite
- trovare tutti i confini e i limiti dello stato
- definire un corretto livello di astrazione di stati ed eventi, ad esempio
  - per un sito di e-commerce l'acquisto di un prodotto è un evento, ma per il software di controllo tale funzionalità corrisponde a decine di eventi distinti

Esempio: Stati di un Libro



Diagrammi Implementativi

Diagramma delle Componenti
Diagramma di Configurazione

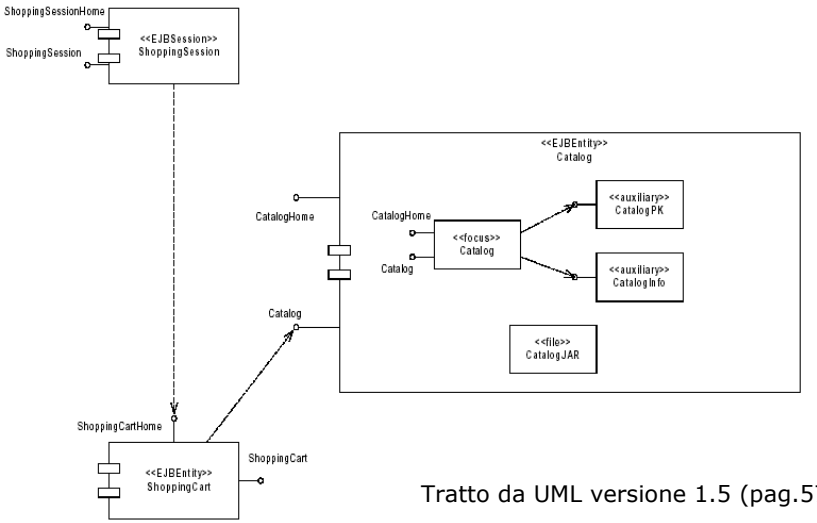
Diagramma delle Componenti

- Definizione
  - grafico che mostra le componenti e le relazioni, di dipendenza, tra queste
- Osservazione
  - non mostra le istanze delle componenti

Componente

- Definizione
  - una parte del sistema che è sostituibile, modulare, che incapsula l'implementazione ed espone un insieme di interfacce
- Esempi di componenti
  - file del codice sorgente e/o file eseguibili
  - DLL (Dynamic Link Library)
  - EJB (Enterprise JavaBeans)
  - …
- Osservazione
  - di solito diverse classi costituiscono una componente, esponendo solo una parte limitata del loro comportamento

Esempio



Tratto da UML versione 1.5 (pag.570)

Diagramma di configurazione …

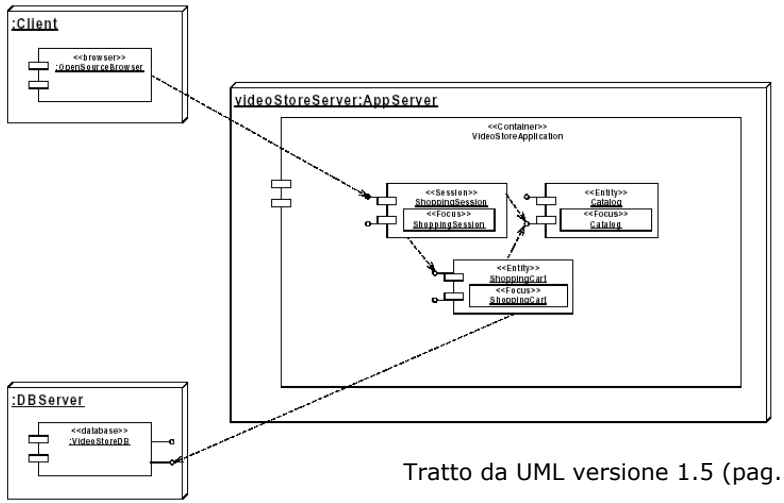
- Definizione
  - grafico che mostra la configurazione degli elementi attivi a *run-time* e le componenti software, i processi e gli oggetti che li costituiscono
- Notazione/Semantica
  - è costituito da nodi che corrispondono ad una risorsa computazionale e che possono contenere istanze di componenti e/o oggetti
  - e da archi che rappresentano una comunicazione e tipicamente indicano una relazione di utilizzo

… Diagramma di configurazione

⇨ Osservazione

- le componenti che non esistono come entità a run-time, perché compilate separatamente, non sono mostrate (dovrebbero comunque essere mostrate nel diagramma delle componenti)
- istanze di componenti software rappresentano le attivazioni a run-time delle unità di codice

Esempio



Tratto da UML versione 1.5 (pag.572)

Bibliografia …

- http://www.omg.org/uml/ (specifiche dell'OMG)
- Association for Computing Machinery, OOPSLA'87 Conference Proceedings, special issue of SIGPLAN Notices, Vol. 22, No. 12, December 1987
- I. Jacobson, "Object-Oriented Development In an Industrial Environment", OOPSLA'87 Conference Proceedings, special issue of SIGPLAN Notices, Vol. 22, No. 12, December 1987, pp. 183-191
- I. Jacobson, M. Christerson, P. Jonsson, and G. Övergaard, "Object-Oriented Software Engineering: A Use Case Driven Approach", Addison-Wesley, Reading, Massachusetts, 1992
- D. Coleman, P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes, and P. Jeremaes, "Object-Oriented Development: The Fusion Method", Prentice Hall, Englewood Cliffs, New Jersey, 1994
- http://www.analisi-disegno.com/

… Bibliografia …

- G. Booch, "Object-Oriented Analysis and Design - With Applications", Second Edition, Benjamin/Cummings, Menlo Park, California, 1994
- S. Cook and J. Daniels, "Designing Object Systems - Object Oriented Modelling With Syntropy", Prentice Hall, Englewood Cliffs, New Jersey, 1994
- G. Booch and J. Rumbaugh, "Unified Method: User Guide, Version 0.8", Rational Software Corporation, Santa Clara, California, 1995
- I. Jacobson, M. Ericsson, and A. Jacobson, "The Object Advantage: Business Process Reengineering With Object Technology", Addison-Wesley, Reading, Massachusetts, 1995
- L. Vetti Tagliati, "UML e Ingegneria del Software", scaricabile gratuitamente, previa registrazione a partire dall'indirizzo:http://www.mokabyte.it/umlbook/index.htm/

… Bibliografia

- R.Lee, W. M. Tepfenhart “UML and C++: A Practical guide to object-oriented development”, Prentice Hall, 1997
- G. Booch, I.Jacobson and J. Rumbaugh, “The Unified Modeling Language Reference Manual", Addison Wesley, 1999