

Sistemi Informativi:

La determinazione dei dati

Argomenti

- Principi dell'approccio object oriented
- Individuazione oggetti
- Individuazione delle classi
- Esempi

Approccio object oriented

crea un modello in base ai seguenti principi:

- astrazione
- incapsulamento
- modularita'
- gerarchia

Astrazione

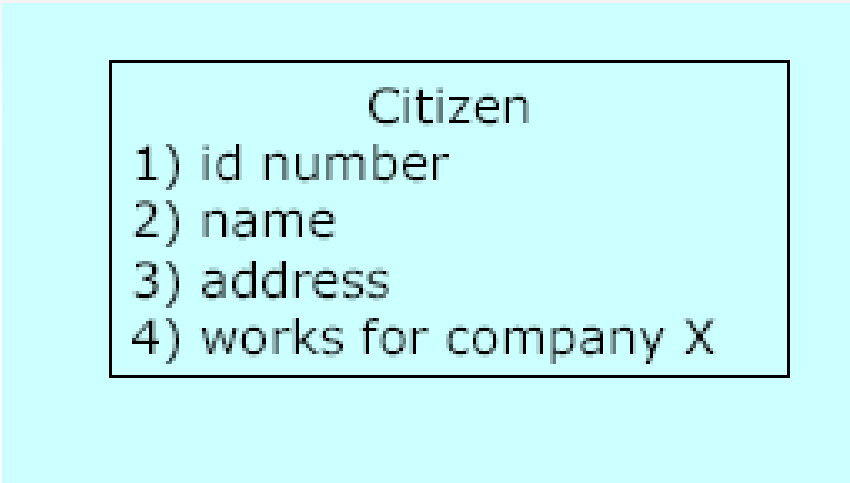
Il principio piu' generale di modellazione
un processo che ci consente di concentrarci sugli
aspetti piu' importanti, trascurando i dettagli
l'astrazione permette di gestire la complessita'
minimizzando gli aspetti che rendono una
identita' differente dalle altre

domanda

quale dei due modelli e' piu' astratto?



Citizen



Citizen

- 1) id number
- 2) name
- 3) address
- 4) works for company X

incapsulamento

separa gli utenti dalla realizzazione

gli utenti dipendono solo dall'interfaccia

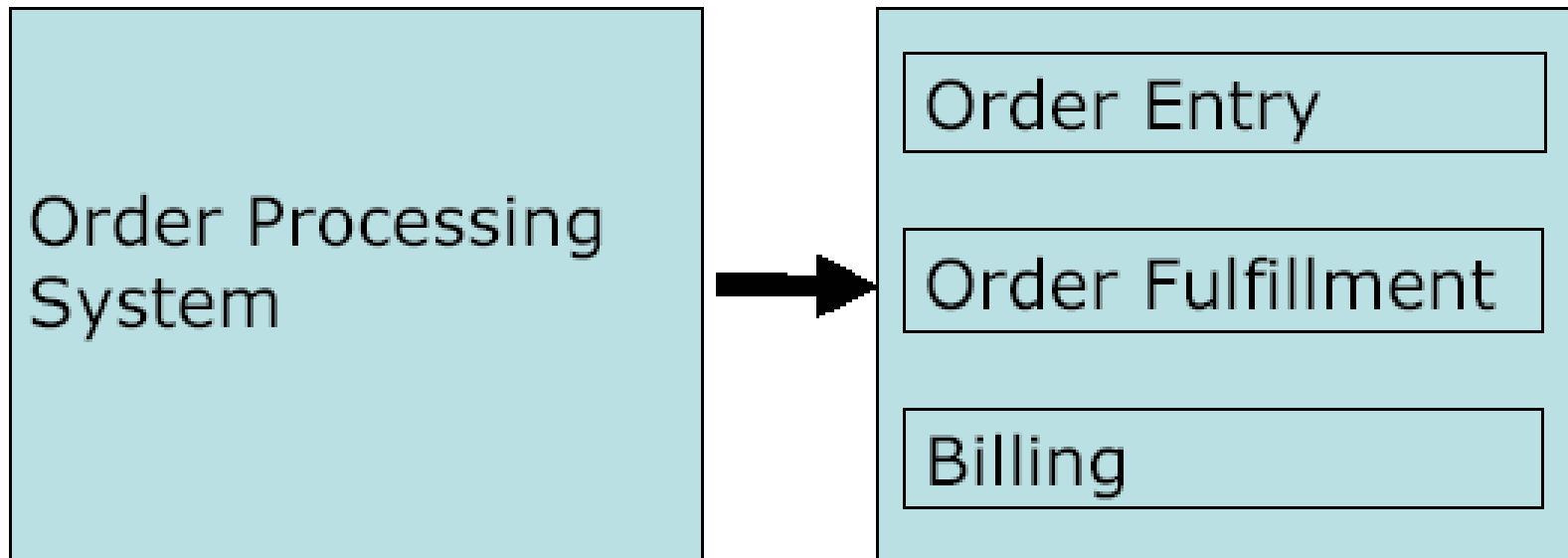
L'oggetto e' una "black box" di cui conosciamo solo il comportamento esterno



Courtesy Rational Software

Modularita'

scomporre un elemento in componenti piu' elementari, autosufficienti, sostituibili

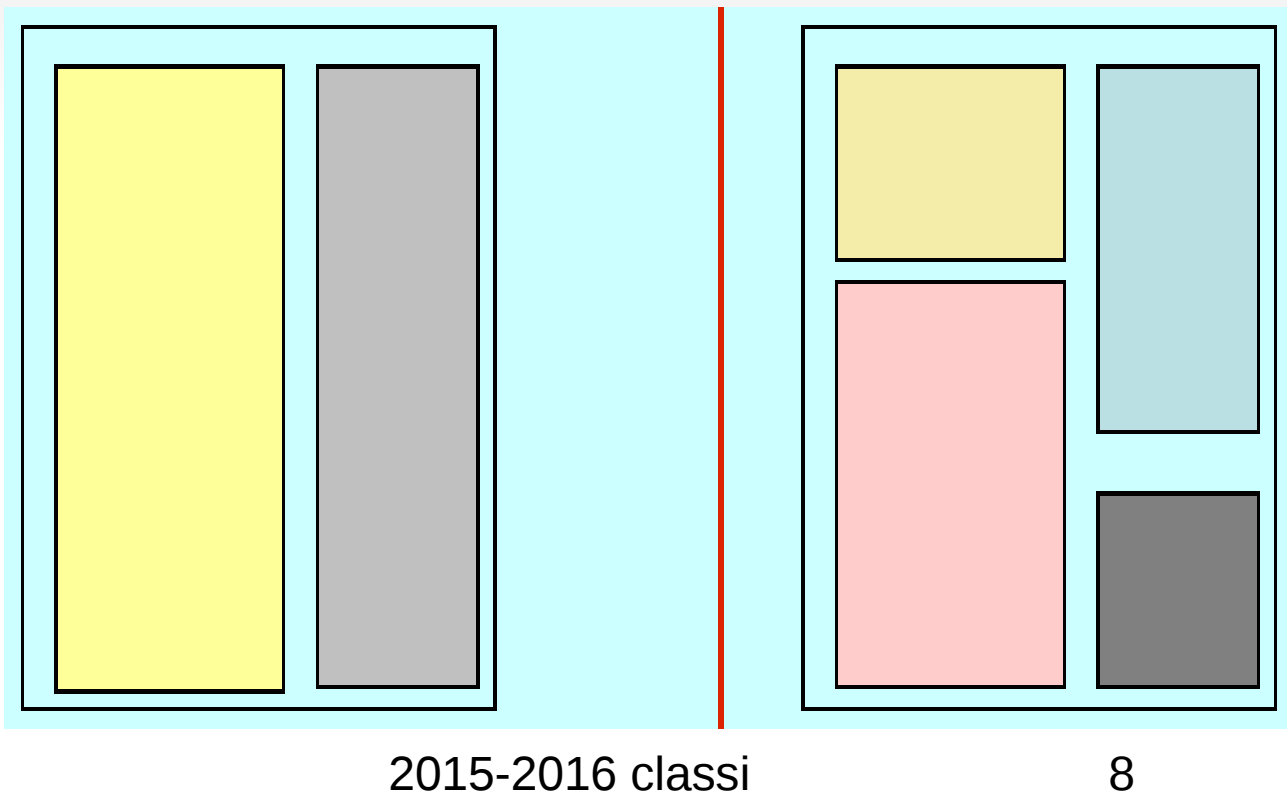


domanda

quale dei due sistemi e' piu' modulare?

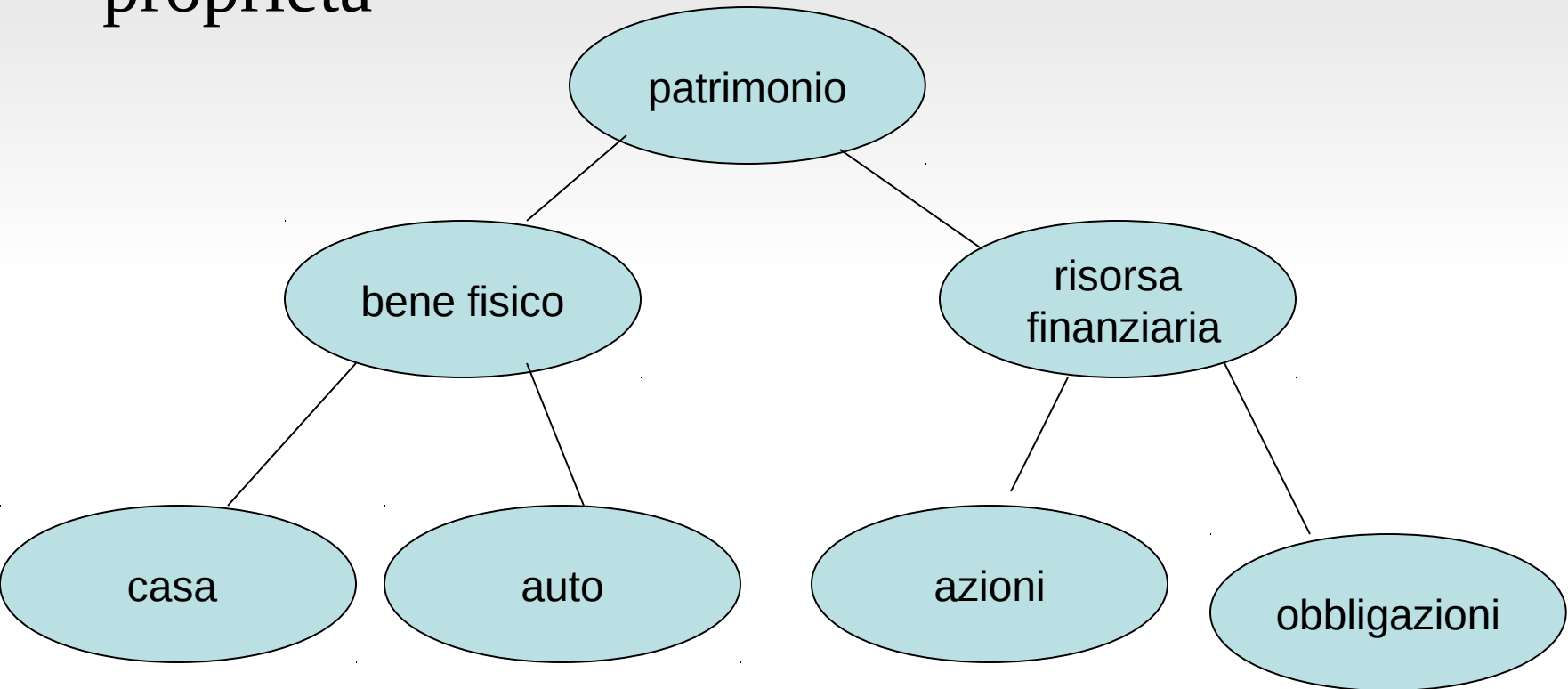
vantaggi o svantaggi?

come sono costruite attualmente le automobili?



gerarchia

generalizzare gli elementi in base alle loro proprietà'



domande

fate degli esempi di gerarchia

quali proprietà sono comuni e distinte?

modi di organizzare files in un file system?

una genealogia?

e' la stessa cosa di una gerarchia?

Oggetto e UML

Secondo Rumbaugh:

- “un entità discreta con un contorno ben definito che incapsula stato e comportamento, un’istanza di una classe

Proprietà’:

- identità’
- stato
- Comportamento

Gli oggetti si scambiano messaggi

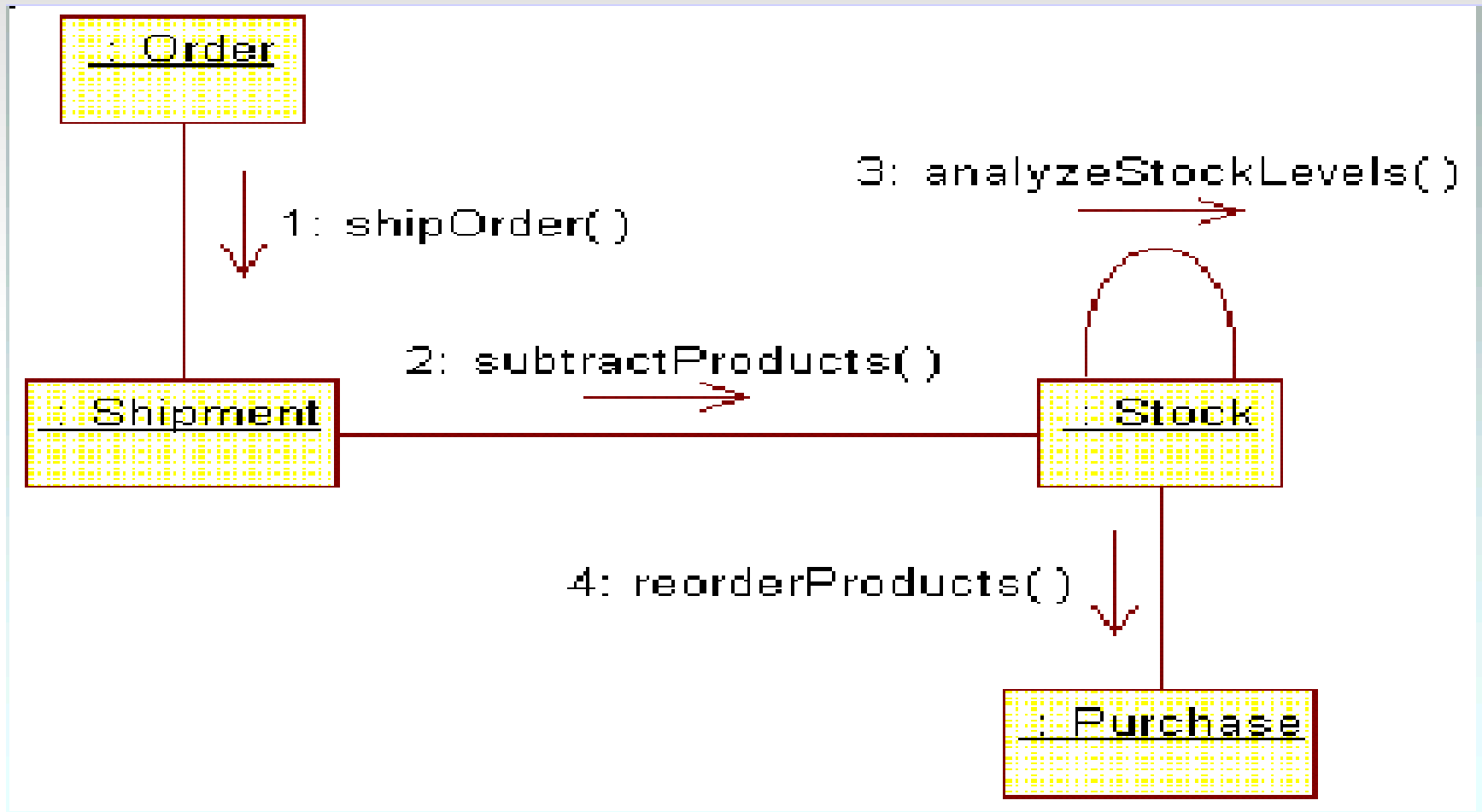
Oggetti e sistemi naturali

In UML un oggetto è sempre scritto **sottolineato**

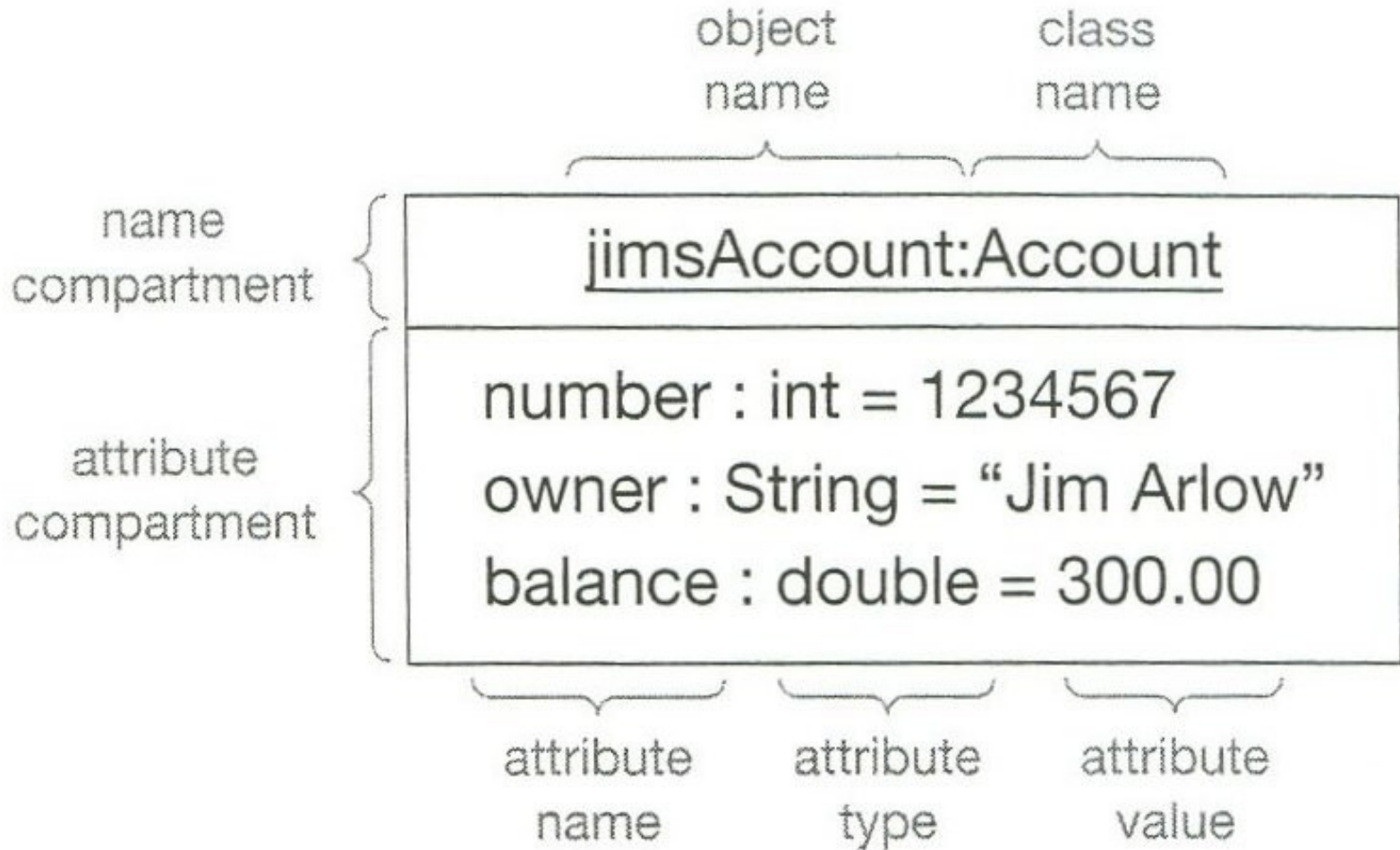
Notazione

- **oggetto: Classe**

Collaborazione tra oggetti tramite messaggi



Oggetto in UML (Arlow..)



Identita' di un oggetto

Identificatore di oggetto (OID)

Riferimento ad un oggetto

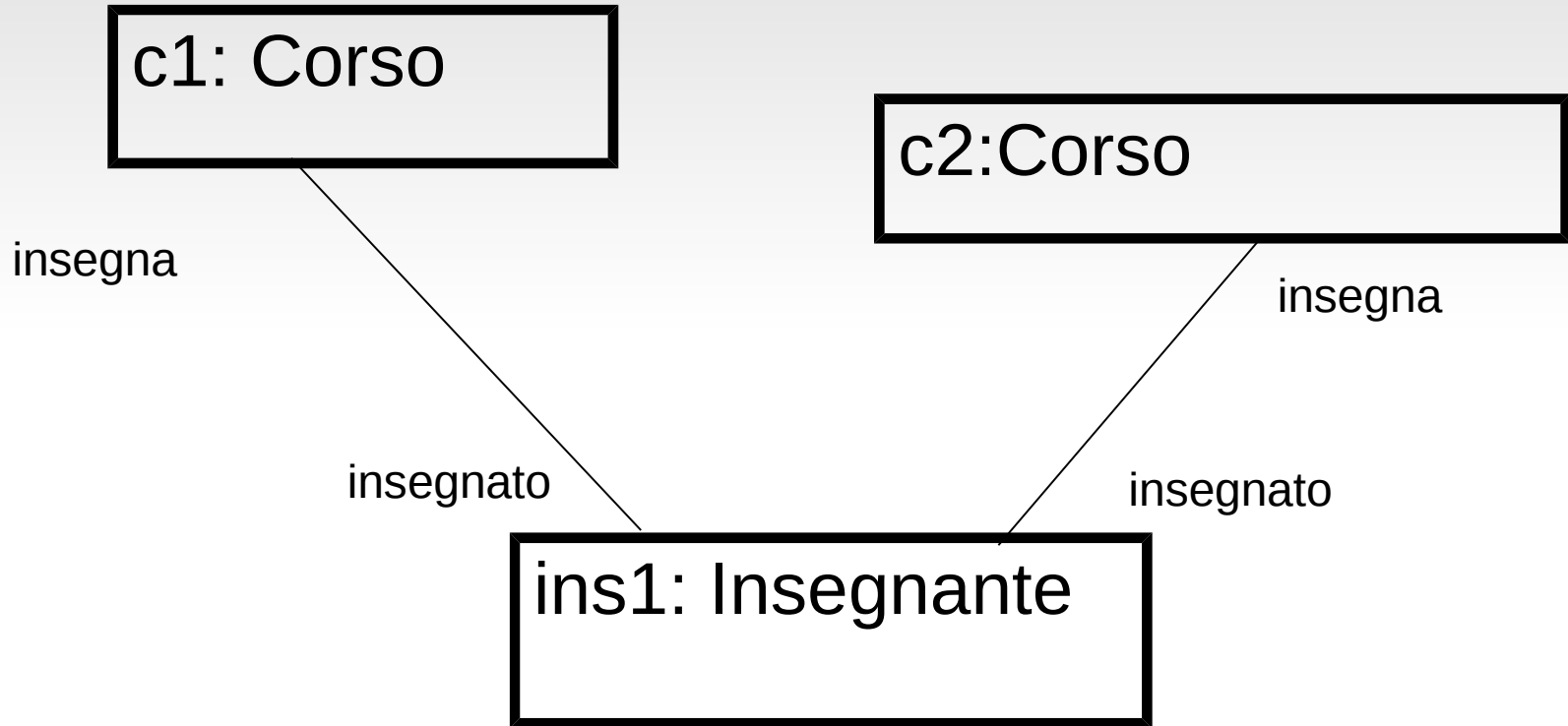
Durata di un oggetto

- Transiente
- Persistente

Comunicazione tra oggetti tramite:

- OID persistenti
- OID transienti

Relazioni tra oggetti in UML



Modellazione delle classi

Tecnica basilare per lo sviluppo di un sistema ad oggetti: un sistema e' costituito da un insieme di oggetti classificati e collaboranti

Oggetti con le stesse proprieta' **raggruppati** in una stessa classe

Il processo e' collaborativo ed incrementale

Strumenti CASE:

- Per facilitare la collaborazione
- Per automatizzare task ripetitivi e fornire uno standard per lo sviluppo del processo

Scoperta delle classi

Analisti differenti forniranno modelli differenti per la stessa applicazione

Tecniche di scoperta

- Sostantivi della frase
- Teoria della classificazione
- Guidato dai casi d'uso
- Classi-responsabilita'-collaboratori (CRC)
- Mixed
- Esperienza

Uso dei sostantivi

Sostantivi considerati come classi potenziali

3 tipi di classi:

- Irrilevanti
- Rilevanti
- Fuzzy (incerte, vaghe)

Si basa sulla correttezza e completezza del documento

Classi ?

Tecnica: usare un attributo/classe descrizione per descrivere qualcosa indipendentemente dalla sua esistenza nel sistema, es. Prodotto non disponibile a magazzino

Per evitare la perdita se cancellassimo tutte le istanze del prodotto

Per ridurre informazioni ripetute

Teoria della classificazione

Un esempio di classificazione:

- Concetti (Prenotazione)
- Eventi (Arrivo)
- Organizzazione (Dipartimento)
- Persone (Passeggero)
- Luoghi (Agenzia di viaggio)

Puo' servire come supporto per ragionare a meta-livello (classificazione delle classi trovate) piu' che scoprire nuove classi

Puo' aiutare a decidere come ripartire gli attributi tra le classi (associa Data ad un evento non ad un concetto)

Slegato dai requisiti (a meno che il sistema non debba implementare una tassonomia)

Approccio tramite casi d'uso

Bottom up

Simile a quello dei sostantivi

Si basa sulla completezza dei documenti relativi

Puo' essere sbilanciato verso la scoperta delle funzionalita'

Trovare gli oggetti nei casi d'uso

Per tutti I casi:

- Trovare I termini ambigui per utenti o sviluppatori per chiarire il flusso degli eventi (partendo dai termini dell'utente)
- Identificare le entita' reali che ill sistema deve gestire. Esempi: AgentePS, Responsabile, Risorsa
- Identificare le procedure reali che il sistema deve gestire : Esempio: PianoOperativoEmergenza
- Identificare sorgenti o destinazioni dei dati. Esempio : Stampante
- Identificare le interfacce. Esempio: StazioneDiPolizia
- Analisi testuale per trovare altri oggetti (Abott)
- Modellare il flusso di eventi con un diagramma di sequenza
- In generale verificare la correttezza dei quantificatori: qualunque, tutti, nessuno... ed avverbi: mai, sempre....

Come trovare le classi?

Conoscere il dominio del problema

Applicare intuito e conoscenze generali

Esaminare il flusso degli eventi e gli oggetti che partecipano

Applicare degli schemi a quel tipo di sistema (design patterns nella parte 2 del corso)

Tentare di stabilire una tassonomia

Analysis testuale del flusso eventi (Abbott Textual Analysis, 1983)

I sostantivi sono buoni candidati ad essere classi

Corrispondenza fra parti del discorso ed elementi (Abbott, 1983)

<i>Elemento</i>	<i>componente del modello</i>	<i>Esempio</i>
Nome proprio	oggetto	Jim Smith
Nome comune	classe	Giocattolo
essere	ereditarieta'	e' un (tipo-di)
avere	aggregazione	ha
verbo modale	vincolo	deve essere
aggettivo	attributo	basso
altri verbi	metodo	comprare

Euristiche per la modellazione

Stabilire riunioni per identificare gli oggetti

differenziare tra oggetti di confine del sistema(boundary), entita' e controllo

trovare associazioni e loro molteplicita'

– cardinalita' insolite spesso generano nuove classi

trovare le aggregazioni

trovare ereditarieta': cercare tassonomie, categorizzare

Euristiche per la modellazione

in generale conviene:

- analizzare a fondo ogni singolo caso d'uso come quasi a se' stante
- di fronte al dubbio se un particolare elemento sia attributo od oggetto, **preferire oggetto** (es indirizzo)
 - alla fine se e' veramente stupido e passivo modellarlo e' stato semplice
 - quasi sempre all'inizio tutto sembra semplice ma poi le cose si complicano... meglio prevenire
- dopo che si e' scomposto il caso in componenti elementari, si puo' passare a fattorizzare componenti comuni, ereditarieta' e quant'altro, per evitare duplicazioni

Class Responsibility Collaborators

Simulazione del funzionamento del sistema tramite attori impersonanti le classi

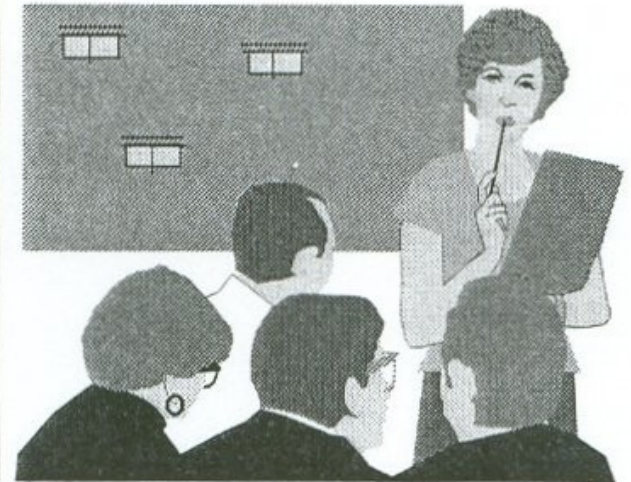
Identificazione delle classi basata sui messaggi che esse si devono trasmettere per partecipare ai casi d'uso

Utile a:

- Determinazione di proprietà della classe
- Verifica di classi scoperte con altri metodi

Class Responsibility Collaborators

Class name: BankAccount	
Responsibilities: Maintain balance	Collaborators: Bank



I post-it si attaccano alla lavagna e si disegnano delle frecce per le collaborazioni

Approccio misto

Usa tutte le tecniche precedenti

Ne' top-down ne' bottom-up

Un possibile modo di procedere:

- Classi importanti derivate da conoscenza del dominio
- Tecnica dei sostantivi e dei casi d'uso aggiungono classi
- Tecnica di CRC per la simulazione
- Teoria della classificazione come guida

Concludendo

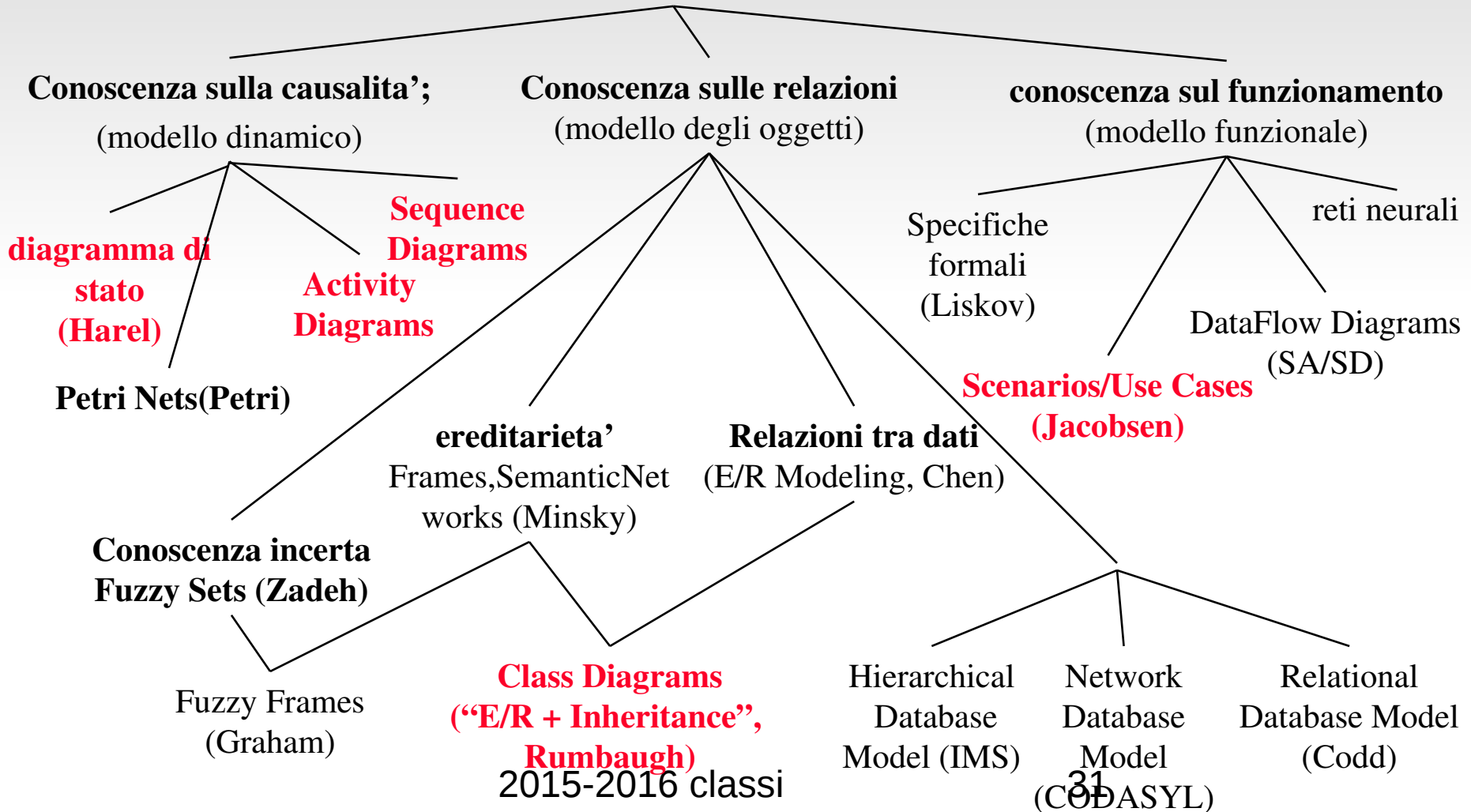
Non bisogna legarsi a nessuna tecnica particolare: e' l'esperienza che suggerisce come e quando utilizzarle

Come gia' detto, il processo e' incrementale e condotto parallelamente alla modellazione di altri aspetti del sistema che a loro volta forniscono ulteriori informazioni

Un cruciverba multidimensionale

Descrizione di sistemi complessi (Naturali, sociali, artificiali)

Epistemologia
Descrizione della conoscenza di un sistema



Classe

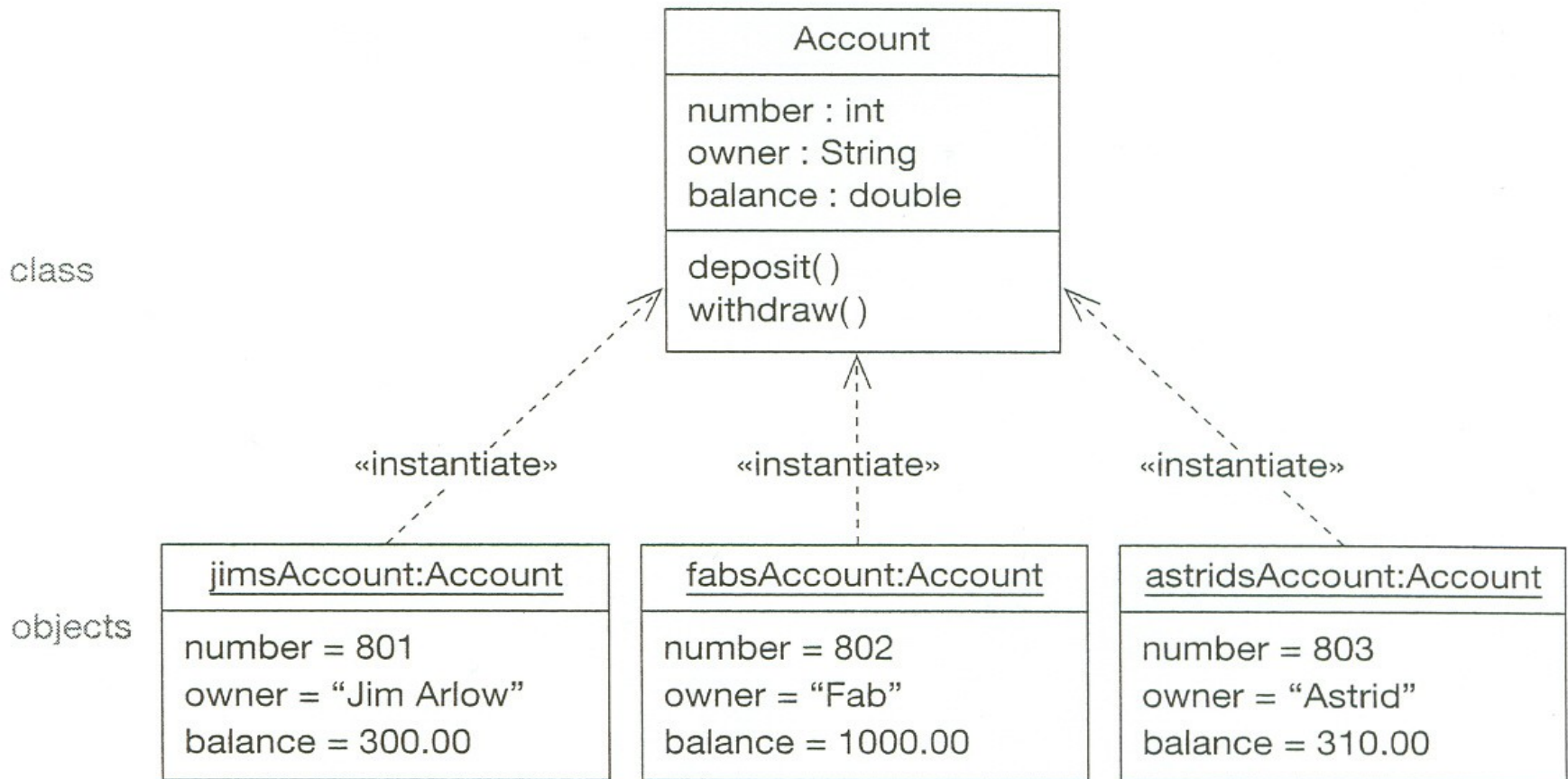
Secondo Rumbaugh:

- “il descrittore di un insieme di oggetti che condividono gli stessi attributi, operazioni, relazioni, comportamento”

ogni oggetto istanza di una sola classe

Gli oggetti non cambiano praticamente mai classe
scegliere la miglior classificazione e' fondamentale
la miglior classificazione per un problema puo'
variare a seconda dei requisiti

Relazione classe oggetto (Arlow..)



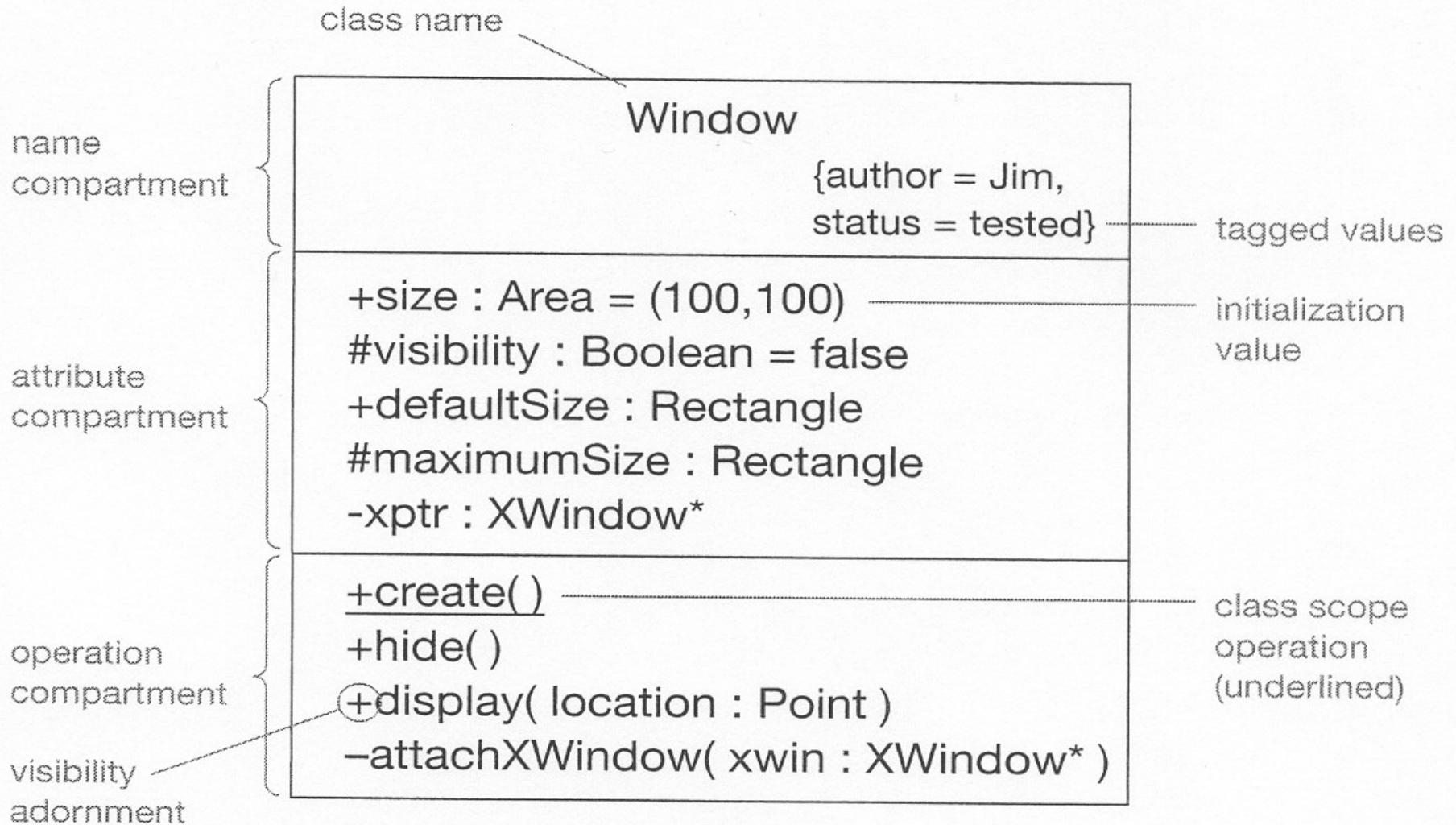
Classe

la relazione <<instantiated>> e' lo stereotipo di una relazione di dipendenza

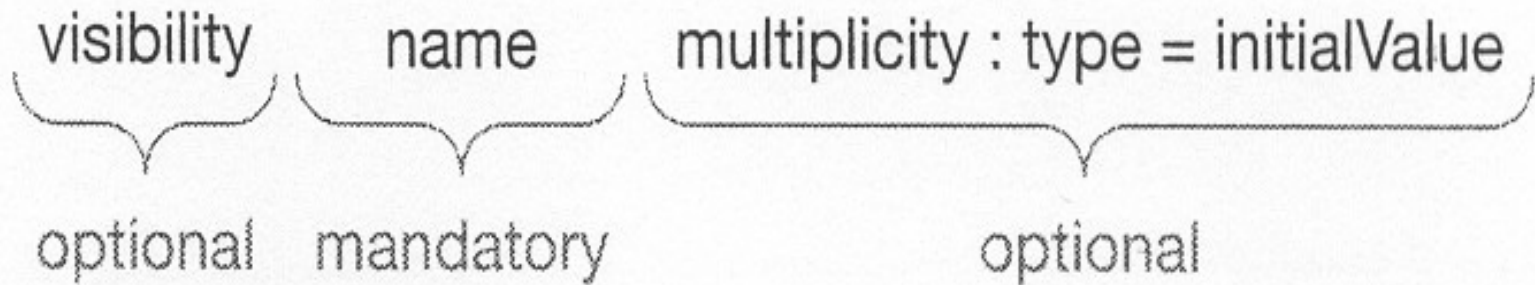
dipendenza: “una relazione tra due elementi in cui il cambiamento di uno dei due (fornitore) puo' influenzare o fornire informazioni all'altro (cliente)

Nota bene: nel modello **relazionale** solo dati predefiniti: niente **ricorsivita'** (parte 2 del corso)

Classe in UML (Arlow..)



Classe in UML: sezione attributi (Arlow..)



NB convenzionalmente la classe ha nome maiuscolo come gli attributi di classe, quelli di istanza minuscolo ma le parole successive iniziano con la maiuscola es. SaldoDisponibile, detta “Camel Notation”

attributi ed operazioni

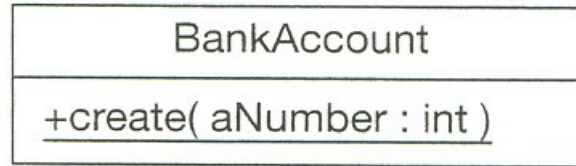
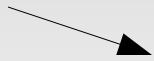
- non cercate di forzare gli attributi e le operazioni nelle classi che avete già trovato. create piuttosto nuove classi :
 - es Impiegato , Ditta, **mansione**
 - mansione puo' essere un attributo di Impiegato, oppure se Mansione ha altri attributi come **elencoCompiti**, meglio creare una nuova classe **Mansione**

Incapsulamento e visibilita' degli elementi

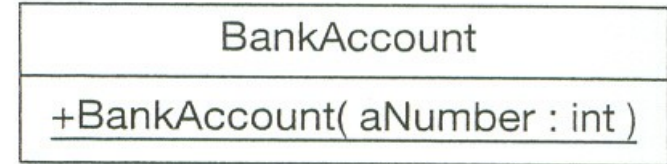
+	pubblico	Tutti gli elementi che possono accedere alla classe possono accedere all'elemento
-	privato	solo operazioni della stessa classe possono accedere all'elemento
#	protetto	operazioni della stessa classe o di classi derivate possono accedere all'elemento
~	package	tutti gli elementi del package o di package annidati puo' accedere

Classe in UML: lo "scope" (ambito) (Arlow..)

costruttori

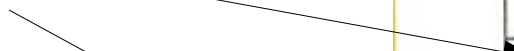


Generic constructor name



Java/C#/C++ standard

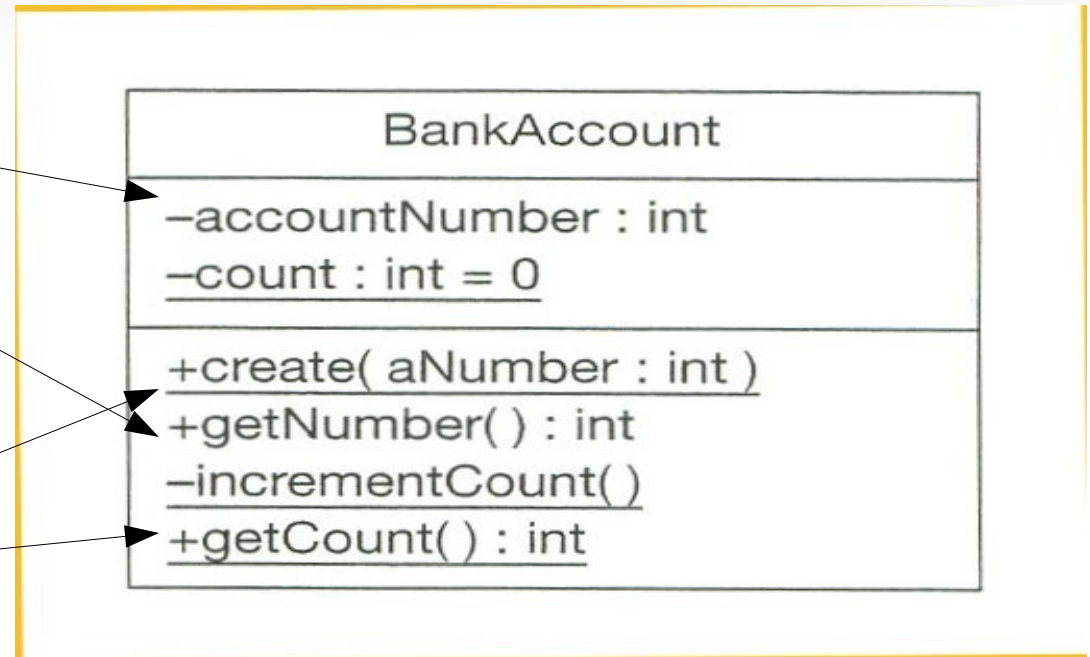
privati



class scope



pubblici



operazioni

Realizzano gli algoritmi

Realizzano lo scambio di messaggi tra oggetti

Possono avere parametri

Si eseguono alla ricezione di un messaggio: di solito hanno lo stesso nome del messaggio

Due tipi :

- Messaggio semplice (asincrono)
- Chiamata e risposta (sincrono)
 - nel sincrono il mittente e' bloccato in attesa della risposta

Operazione, Firma , Metodo

Operazione: Una funzione o trasformazione applicabile agli oggetti di una classe .tutti gli oggetti di una classe condividono le stesse operazioni (***fase di analisi***)

Firma: Numero e tipo dei parametri e del valore restituito. (***fase di progetto delle classi***)

Metodo: Implementazione di una operazione della classe (***fase implementativa***)

Operazione polimorfica: la stessa operazione si applica a diverse classi

Workorder
File_name: String Size_in_bytes: integer Last_update: date Stickers: array[max]
print() delete() open() close() write() read()

operazioni di istanza (oggetto)

possono avere parametri e valore di ritorno

possono modificare lo stato dell' oggetto

cambiando il valore dei suoi attributi

possono mandare messaggi ad altri oggetti (che a loro volta possono modificare il loro stato)

dopo l'invocazione di un messaggio, il flusso torna all'oggetto chiamante

operazioni di classe

hanno senso nell'ambito di classe e non si applicano alle istanze

– costruttori:

- creano una nuova istanza di oggetto della classe secondo criteri di inizializzazione specificati nell'implementazione (piu' tardi) es `Persona(Eta' 0)`

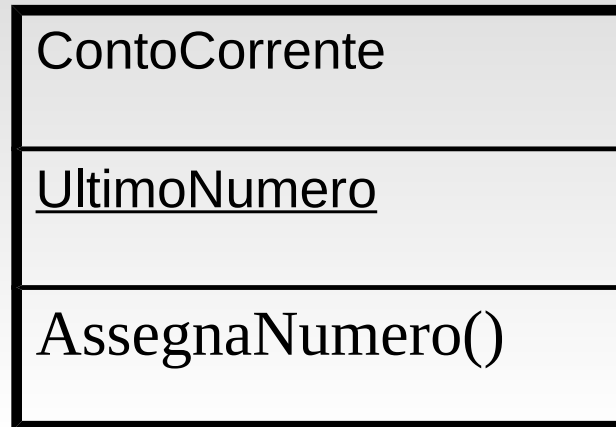
– distruttori

- cancellano un istanza e liberano le risorse destinate a suo uso esclusivo

– operazioni hanno senso per **l'insieme** delle istanze

- es calcola “eta' media della popolazione”, ipotizzando una classe “Persona”
- Altro es assegna il prossimo numero al nuovo conto corrente

operazioni di classe



Classi singleton (una istanza)

Sono oggetti di cui ha senso crearne uno solo

Siccome qualunque oggetto in un sistema ad oggetti appartiene ad una classe,

- Si crea la classe di cui creeremo una sola istanza

Es. ConsiglioAmministrazione

Criteri per l'individuazione di una classe persistente

Detta in genere Entita'

Deve descrivere un insieme di oggetti

- Le classi singole poco frequenti nelle entita'

Definita da un insieme di attributi

- Chiavi per l'identificazione
- Identificatore univoco - OID

Ha sempre le 4 operazioni CRUD (Create, Read, Update, Delete)

Classe nella fase di analisi

modella aspetti importanti del sistema, es.

Prodotto, Cliente

deve corrispondere in modo chiaro ad un concetto del mondo reale

solo gli attributi e le operazioni piu' importanti sono modellati: a volte saranno espansi in insiemi di attributi e composizioni di operazioni

Classe nella fase di analisi

mancano un gran numero di dettagli,

alcuni attributi ed i loro tipi es

`dataApertura`,

operazioni ed i loro parametri, es

`stampaUltimeOperazioni()`

visibilita' degli elementi

**limitare i dettagli finche'
non diventano necessari**

ContoCorrente
numero intestatario saldo
deposito() prelievo() calcoloInteresse()

Classe nella fase di analisi

- hanno un compito ristretto e chiaro
- alta coesione interna
 - I loro attributi sono strettamente collegati
- relativa indipendenza dalle altre
- un nome che ne rifletta lo scopo

Classe nella fase di analisi

es. CestinoAcquisti potrebbe avere:

- aggiungiProdotto()
- rimuoviProdotto()\
- mostraProdotti()

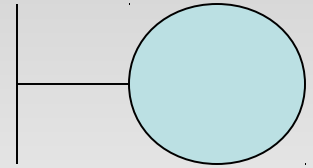
mentre

- accettaPagamento(), stampaFattura()..
- potrebbero appartenere ad una classe ServizioCassa

Tre tipi di classi

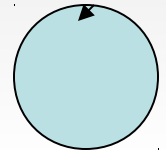
Confine <<boundary>>

- media l'interazione fra sistema ed ambiente (spesso interfacce)



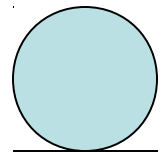
Controllo <<control>>

- Controlla l'esecuzione dei casi d'uso



Persistente <<entity>>

- modella i dati permanenti



classi <<boundary>>

- interfacce utente
- interfacce con altri sistemi
- con strumenti esterni tipo sensori

classi <<controllo>>

- nessuna regola precisa: cercare la soluzione piu' naturale
- nome finisce in
 - -Controller , -Manager
- gestiscono spesso diversi casi d'uso

classi <<entity>>

- Sono I dati permanenti del sistema
- entrano in diversi casi d'uso
- manipolate dalle classi controllo
- accettano e forniscono informazioni alle classi <<boundary>>
- spesso molto semplici → facilitano l'accesso ai dati per altri sistemi
- In DB relazionale identificati da una Primary Key (chiave primaria) PK (seconda parte corso)

classi ricorrenti

Magazzino

Prodotto

Fattura

...

– non serve reinventare la ruota: se si puo’
meglio soluzioni standard

domande

Spiegare quali affermazioni sono vere o false:

- **Utente** e' un esempio di **classe**
- **Mario** e' un esempio di **classe**
- **creaLavoro** e' un esempio di **classe**
- **codiceFiscale** e' un esempio di **operazione**
- **rinominaCartella** e' un esempio di **attributo**

Esempio: iscrizione universitaria

Requisito:

- Ogni diploma di laurea ha un insieme di corsi obbligatori e uno di facoltativi:

Classi rilevanti: **Diploma, Corso**

Classi vaghe (concetto non immediatamente rilevante): **corso obbligatorio, corso facoltativo**

- Perché obbligatorio e facoltativo non comporterebbero creazioni di classi distinte?

Esempio: universita'

I concetti di **Corso Obbligatorio** e **Corso Facoltativo** sono relativi ad uno specifico piano di studi

Possono quindi variare in funzione di esso ->

- Non devono essere modellati tramite classi
- Un oggetto non deve in generale cambiare classe, mettere le proprieta' variabili negli attributi

A volte le classi “Vaghe” sono difficili da modellare perche'

Non si sa come definire / dove mettere I loro attributi

Esempio: iscrizione universitaria

Altri requisiti:

- Ogni corso ha un livello ed un numero di crediti
- Un corso puo' appartenere a diversi diplomi
- Ogni diploma ha un numero minimo di crediti necessari al suo conseguimento
- Gli studenti possono combinare le offerte di corsi in programmi di studio per ottenere il diploma

Esempio: iscrizione universitaria

Classi rilevanti	Classi fuzzy
Corso	CorsoObbligatorio
Diploma	CorsoFacoltativo
Studente	ProgrammaDiStudio
OffertaDiCorso	

Esempio: affitto video

Requisiti:

- Le videocassette sono in formato Beta o VHS
- I VideoDischi sono in formato DVD
- Ogni film ha uno specifico periodo di affitto in giorni con relativa tariffa
- Il negozio deve sapere di ogni film quante copie ed in quale formato sono disponibili
- La condizione attuale di ogni singolo disco o nastro deve essere conosciuta

Esempio: affitto video

Classi rilevanti	Classi fuzzy
Film SupportoDiRegistrazione	CondizioniDiAffitto
VideoCassetta	
VideoDisco (o DVD)	
NastroBeta	
NastroVHS	

Esempio: contact management

Requisiti:

- Mantenere contatti con clienti attuali e potenziali
- Ottenere nuovi contratti
- Conservare nome, telefono, indirizzo delle organizzazioni e delle persone di contatto relative
- Organizzare attività degli impiegati riguardo alle persone di contatto
- Gli impiegati possono schedare compiti per se' o altri impiegati
- Un compito e' un insieme di eventi che occorrono per realizzare un determinato obiettivo del cliente
- Esempi di eventi sono: telefonate, visite, fax etc.

Esempio: contact management

Classi rilevanti	Classi fuzzy
Organizzazione	OrganizzazioneCorrente
Contatto	OrganizzazionePotenziale
Impiegato	IndirizzoPostale
Compito	
Evento	

Rilevanza delle classi

La distinzione tra classi rilevanti e secondarie e' in definitiva **poco** rilevante:

- Se una classe e' poco rilevante probabilmente sara' poco impegnativo modellarla
- Una fase di analisi accurata puo' far risparmiare molto in fase di progetto ed implementazione
- **Dopo** aver analizzato e compreso in dettaglio il problema si potranno in seguito fare eventuali semplificazioni ed eliminazioni (astrazioni)
- Preferire classi ad attributi puo' portare a soluzioni piu' generali e riutilizzabili

Specifica delle classi

Nel class diagram:

- Ogni classe ha un nome (magari anche un codice) singolo con iniziali maiuscole se composto (es CodiceFiscale)
- appropriato: evitare abbreviazioni (oggiogiorno)

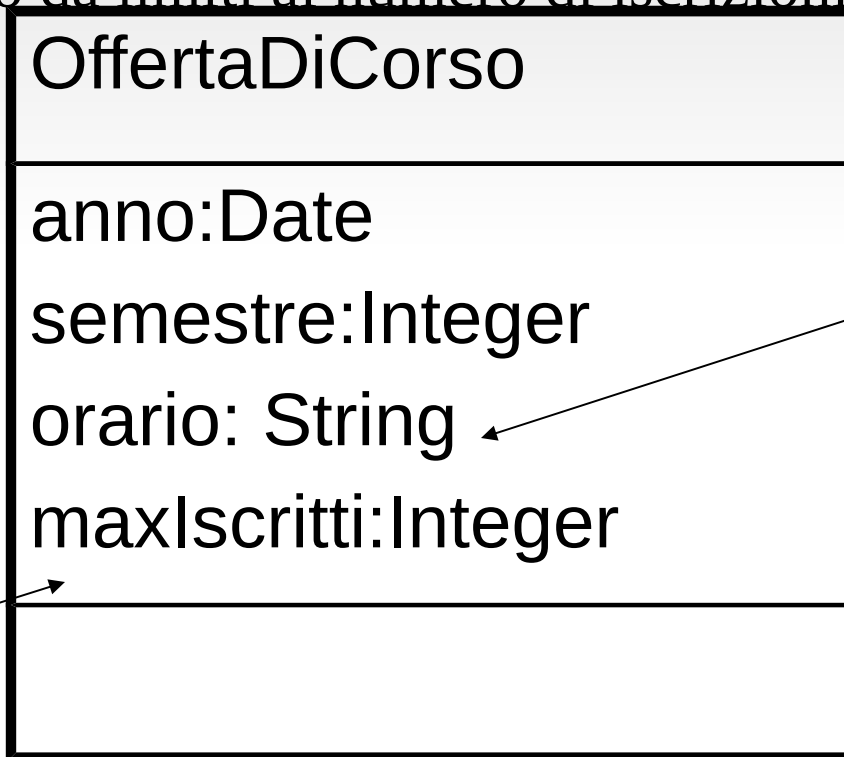
Definizione delle proprietà della classe

- Attributi (inizialmente quelli che individuano dati persistenti) – di solito senza lettera maiuscola
- Operazioni (anche in fase successiva)

Esempio: iscrizione universitaria

Altro requisito:

- La scelta dei corsi puo' essere limitata da conflitti di orario o da limiti al numero di iscrizioni

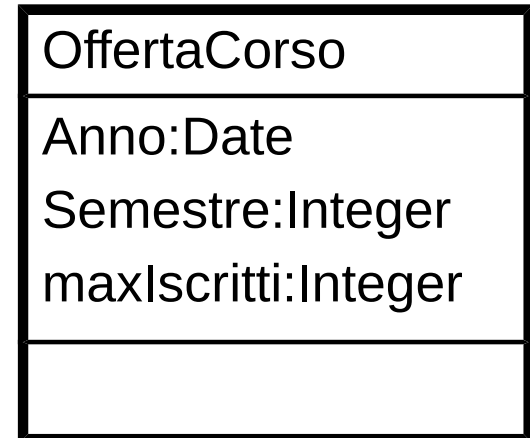
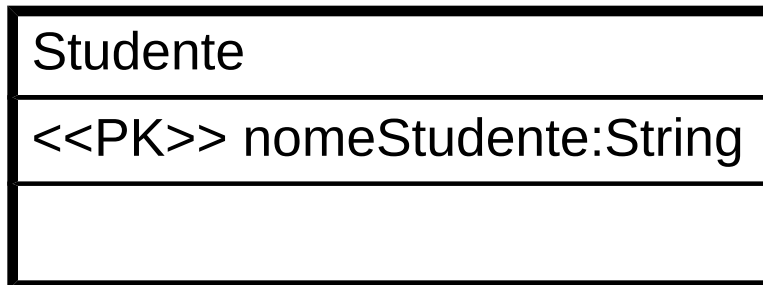
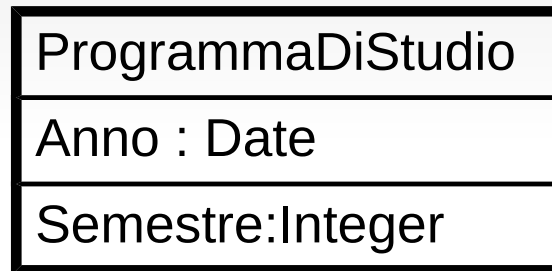
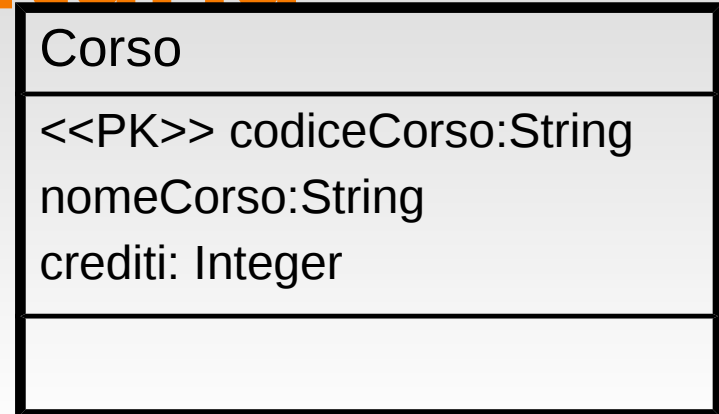
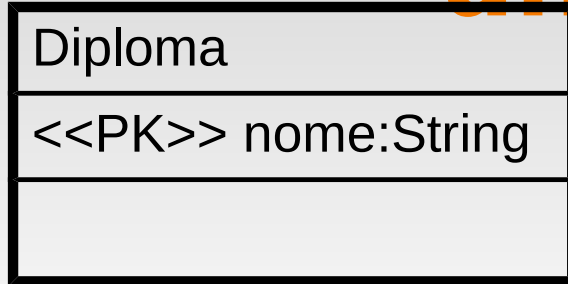


Esempio: iscrizione universitaria

Altro requisito:

- Il programma di studi e' inserito nel sistema on-line
- Il sistema controlla la validita' del programma e notifica eventuali incongruenze
- I problemi saranno risolti con l'aiuto di un tutor
- Il programma finale e' soggetto all' approvazione di un delegato accademico e inviato all'ufficio di registro

Esempio: iscrizione universitaria



Che fare con studenti omonimi? perche' gia'
definire PK?

Esempio: affitto video

Altro requisito:

- La tariffa differisce a seconda che si tratti di disco o cassetta, ma e' identica per i due tipi di cassetta

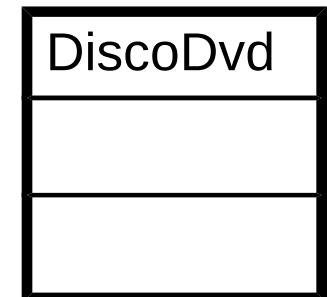
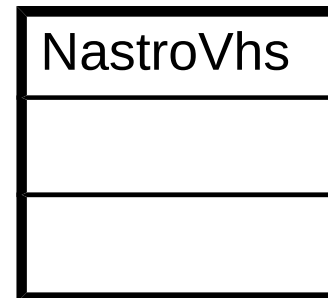
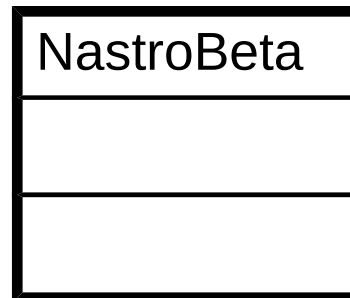
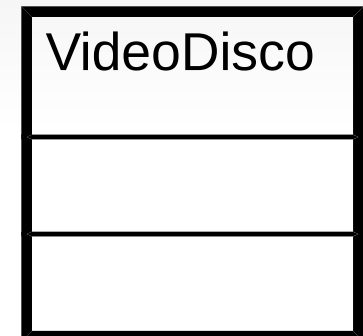
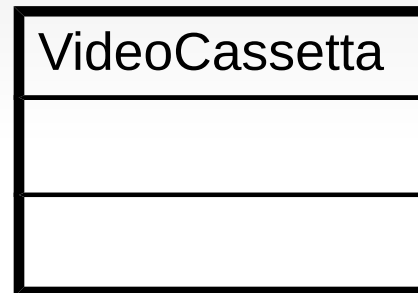
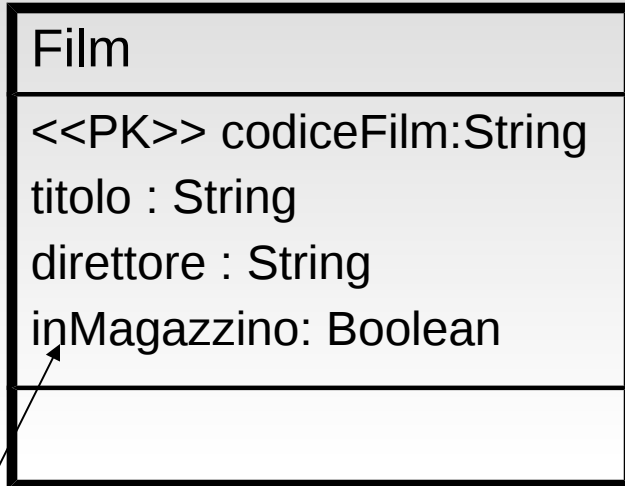
CondizioniDiAffitto
affittoInGiorni :Integer tariffa: Currency

Esempio: affitto video

Altri requisiti:

- Il sistema deve consentire l'uso di eventuali futuri diversi formati e supporti in aggiunta a quelli iniziali
- Gli impiegati di frequente per individuare un film usano un codice anziché il nome
- Lo stesso titolo di film può avere diverse edizioni realizzate da direttori differenti

Esempio: affitto video



Esempio: gestione contatti

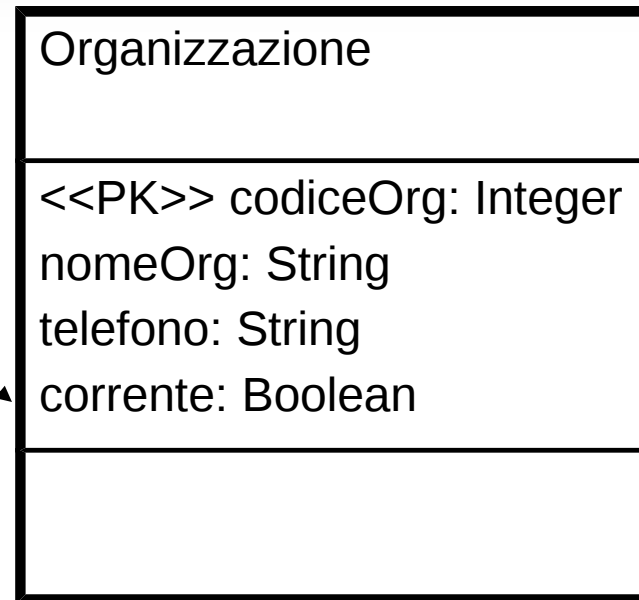
Altro requisito:

- Un cliente e' corrente se esiste un contratto di fornitura di prodotti o servizi; la gestione dei contratti e' fuori dall' ambito del sistema

Eliminiamo le classi fuzzy:

OrganizzazioneCorrente

OrganizzazionePotenziale

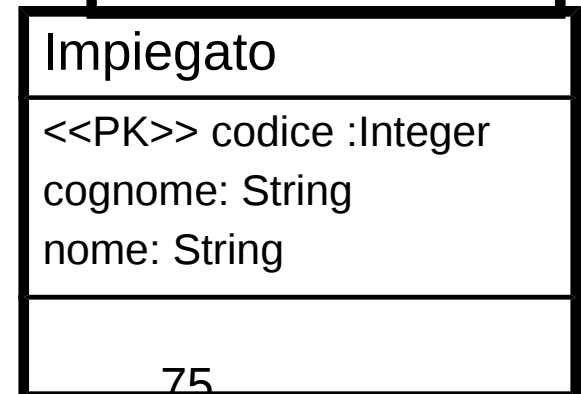
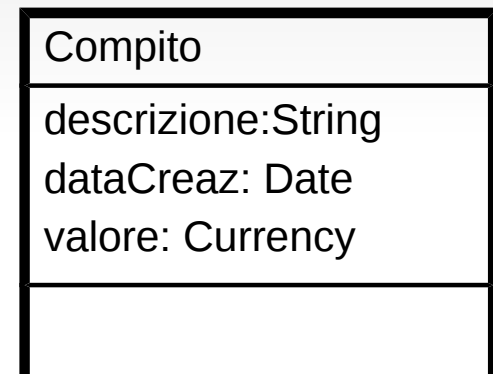
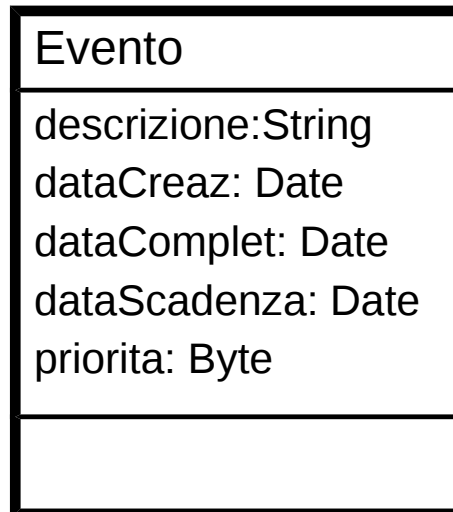
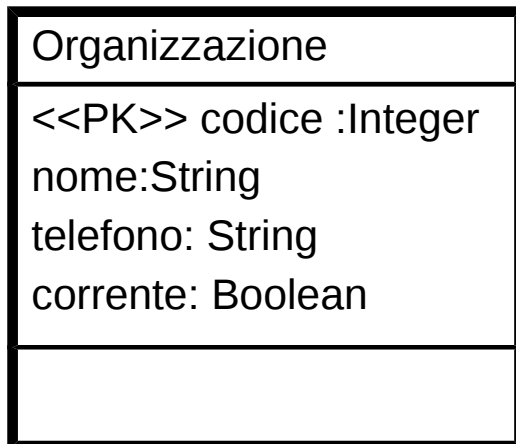
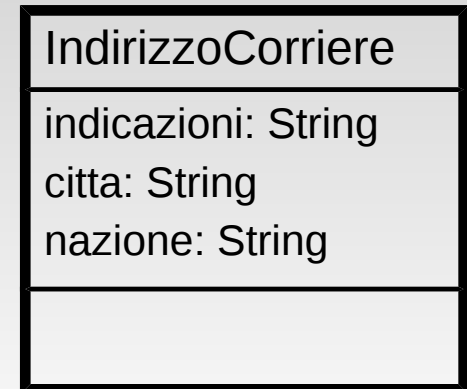
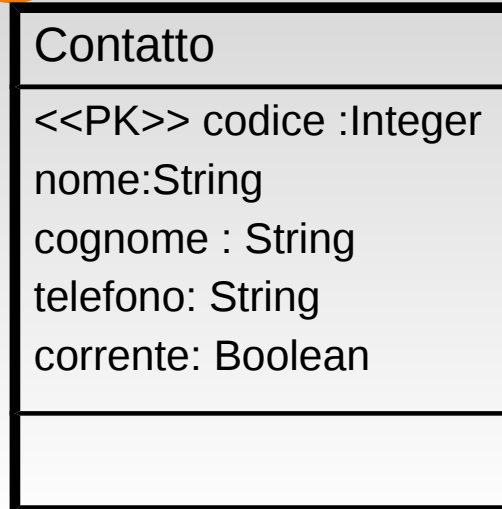
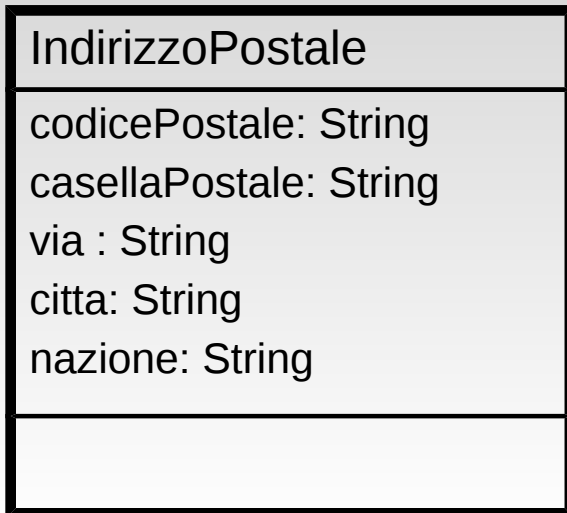


Esempio: gestione contatti

Altri requisiti:

- Fornire elenchi di contatti organizzati secondo indirizzi postali o corrieri di spedizione
- Registrare data e ora della creazione di un compito e del suo completamento
- Memorizzare il valore veniale di un compito
- Gli eventi associati ad un impiegato sono mostrati sotto forma di calendario giornaliero, con visualizzata la priorit  (alta media o bassa)
- Alcuni eventi non hanno una data di scadenza
- La data di scadenza di un evento puo' essere modificata ma non quella di creazione
- Il sistema memorizza gli identificativi degli impiegati che hanno creato un evento o un compito , che sono incaricati di eseguirlo, che lo hanno completato

Esempio: gestione contatti



Esempio: telemarketing

- Altri requisiti:
 - Ogni campagna ha un titolo utilizzato per riferimento
 - Ha un codice unico interno di riferimento
 - Ha durata prefissata
 - Al termine, i premi sono estratti ed i loro vincitori avvisati

Campagna
<<PK>> codice :String
titolo: String
dataInizio: Date
dataFine: Date
dataEstrazione: Date
bigliettiTotali : Integer
bigliettiVenduti : Integer

Premio
descrizione :String
valore : Currency
classifica: Integer

Esempio: telemarketing

Altri requisiti:

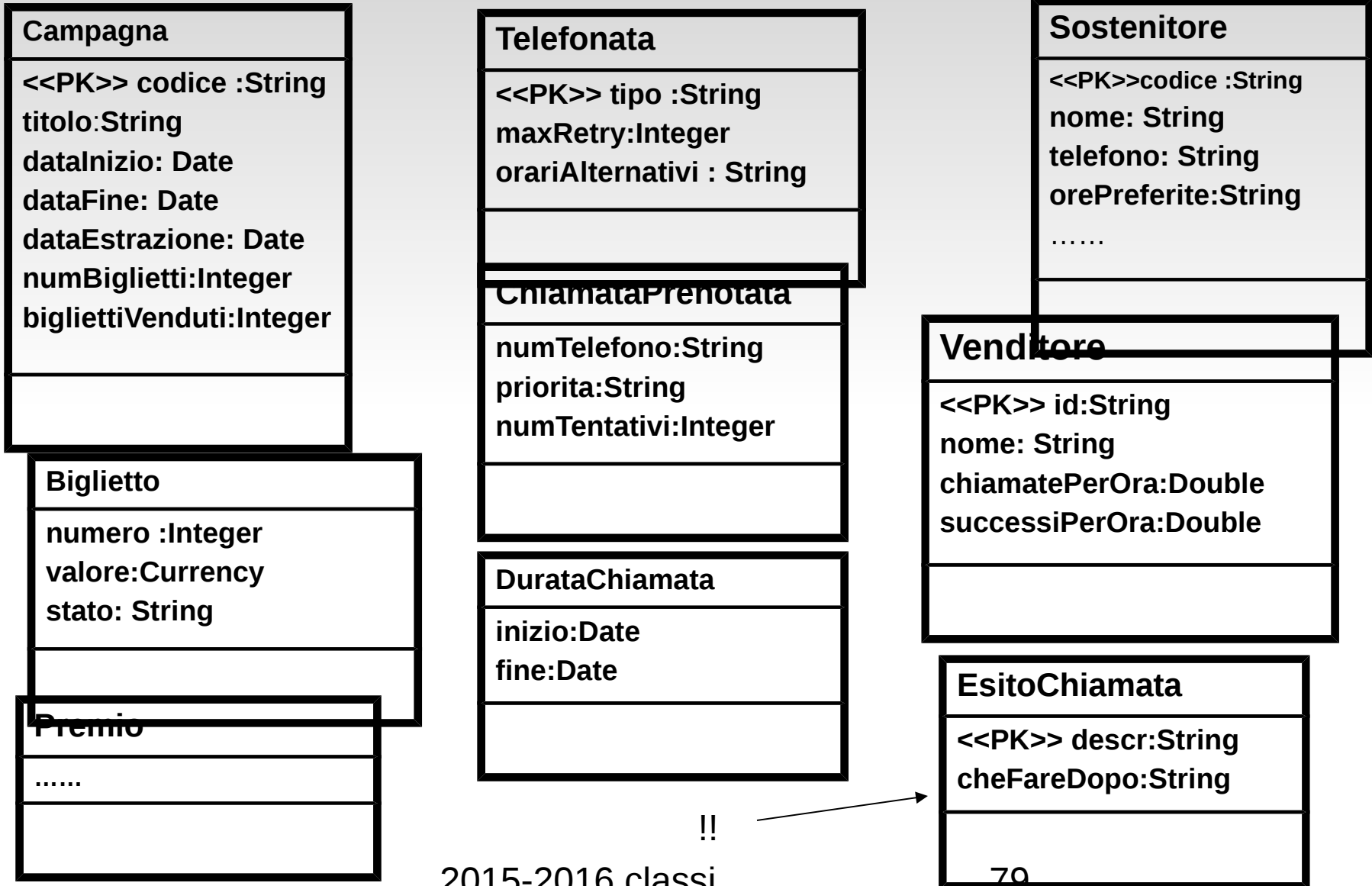
- I biglietti hanno un numero unico all'interno di ogni campagna
- Si conosce il numero di biglietti totali, prenotati, venduti, estratti
- Per conoscere le prestazioni dei venditori, viene registrato il numero di chiamate, la durata, l'esito della stessa
- Si conservano le seguenti informazioni relative ai sostenitori
 - Dettagli del contatto (indirizzo telefono ..)
 - Dettagli storici sull'adesione a campagne precedenti
 - Preferenze e vincoli del sostenitore come orari preferiti per telefonare , carta di credito etc.

Esempio: telemarketing

Altri requisiti:

- Le telefonate sono effettuate in base a prioritá'
- Le telefonate senza esito sono riefettuate:
 - variando l'orario di chiamata
 - vi e' un limite variabile al numero di tentativi a seconda del tipo di chiamata (normale, pagamento ..)
- L'esito della chiamata rientra nelle categorie successo (vendita biglietti) , fallimento, richiamare dopo, nessuna risposta, occupato, segreteria telefonica, fax, numero sbagliato, numero inattivo.

Esempio: telemarketing



ordine desiderabile di definizione di classe in questa fase

1. classi e **nomi** degli attributi/operazioni principali
2. valorizzare il tipo degli attributi
 - in caso di dubbio scegliere sempre la soluzione piu' generale:
 - preferire stringa agli altri tipi elementari
 - preferire oggetto a tipo elementare: il che puo' portare di nuovo al punto 1.

Attributo o classe? es

N.telefono

- Vogliamo memorizzare il numero di telefono
- 0113456789.

discutiamo le varie possibilita'. numero stringa oggetto

esercizio

Classi del sistema albergo