oggetti, classi e notazione UML

object orientation

- il paradigma object oriented è usato come linguaggio per esprimere modelli
  - domain models
  - design models
  - implemenetation models
  - ecc.
- tutto ciò che vedremo è legato alla modellazione
- applicabile alla programmazione
  - come caso particolare
  - non legato ad un linguaggio di programmazione specifico

Unified Modelling Language (UML)

- è un linguaggio visuale per esprimere modelli orientati agli oggetti
- UML è uno standard dell'Object Management Group (OMG)
  - www.omg.org
- approccio flessibile
  - notazione vasta e complicata
  - niente è obbligatorio
    - livello di dettaglio è adattabile alle varie esigenze
  - prevede meccanismi di estensione (*UML profiles*)

Unified Modelling Language (UML)

- dalla versione 1.5 dello standard:

  "The primary design goals of the UML are as follows:
  - Provide users with a ready-to-use, expressive visual modeling language to develop and exchange meaningful models.
  - Furnish extensibility and specialization mechanisms to extend the core concepts.
  - Support specifications that are independent of particular programming languages and development processes.
  - Provide a formal basis for understanding the modeling language.
  - Encourage the growth of the object tools market.
  - Support higher-level development concepts such as components, collaborations, frameworks and patterns.
  - Integrate best practices."

concetti fondamentali

- oggetto
- classe
- interfaccia
- (tipo)
- interazione tra oggetti

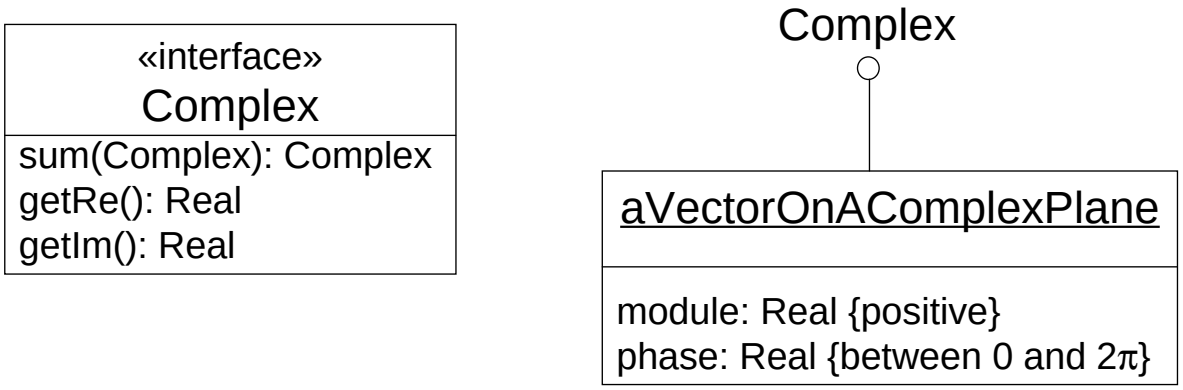
- UML prevede costrutti per esprimere dettagli di oggetti, classi, interfacce e interazioni tra oggetti

oggetti

- stato interno
  - nei casi semplici è un insieme di attributi (o proprietà o campi) che possono assumere dei valori in un certo insieme
  - può avere una struttura complessa per oggetti compositi
- interfaccia dell'oggetto
  - metodi o operazioni
  - sintassi
    - parametri e valori di ritorno
    - in programmazione si chiama prototipo
  - semantica
    - modi in cui lo stato si modifica quando l'oggetto è sollecitato tramite la sua interfaccia
    - può coinvolgere altri oggetti

stato esterno

- è il modello di stato che l'utente conoscere tramite l'interfaccia
- ad ogni stato interno corrisponde uno stato esterno
  - lo stesso stato esterno può essere rappresentato da più stati interni
- può essere
  - del tutto diverso dallo stato interno
  - esattamente uguale allo stato interno
  - meno espressivo dello stato interno (es. cache, lazy evalution, ecc.)
- un esempio



oggetti: identità

- due oggetti possono avere lo stesso stato ma sono comunque considerati distinti

5 **5** due *valori* uguali

due *oggetti* distinti

interfacce

- una *interfaccia* è una descrizione di una classe di oggetti software conformi a “certe modalità di utilizzo”

- una interfaccia si specifica mediante la descrizione
  - dello stato esterno
  - dei metodi dell’interfaccia (aspetto sintattico)
  - di ciò che tali metodi fanno (aspetto semantico) in termini di
    - modifiche dello stato esterno
    - valori di ritorno
    - interazioni con altri oggetti
    - creazione/distruzione di altri oggetti

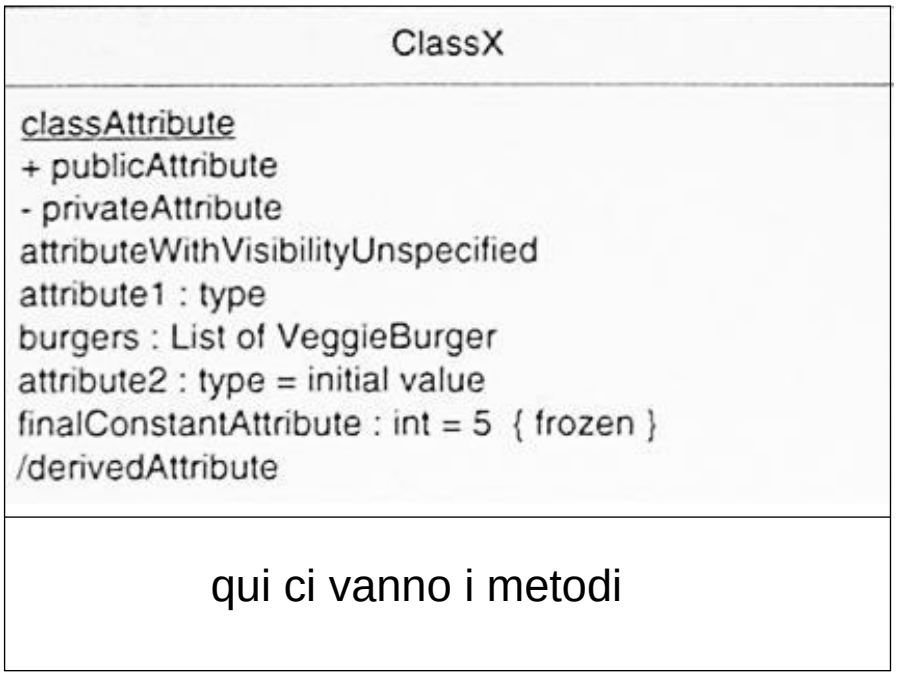
classi

- una *classe* è una descrizione di una classe di oggetti software che hanno
  - la stessa interfaccia
  - la stessa struttura dello stato interno e implementazione dei metodi
- una classe si specifica mediante la descrizione
  - dello stato interno (di tutti gli oggetti della classe)
  - dei metodi dell’interfaccia (pubblici) e di qualsiasi altra procedura “interna” di supporto (metodi privati)
  - di ciò che i metodi fanno in termini di
    - modifiche dello stato interno
    - valori di ritorno
    - interazioni con altri oggetti
    - creazione/distruzione di altri oggetti

tipi

- ogni volta che incontra la parola tipo leggi:
  classe o interfaccia

notazioni UML per e classi: gli attributi

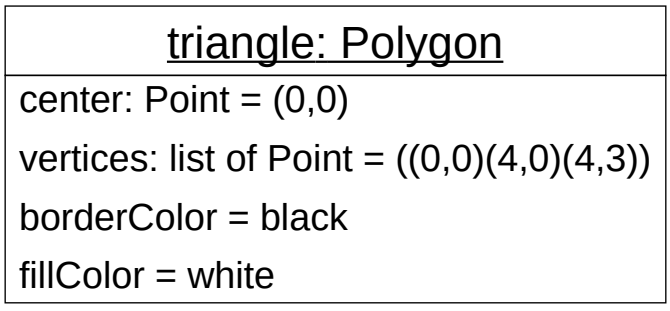


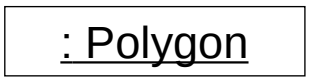
notazioni UML per le classi: i metodi

alcuni stereotypes standard

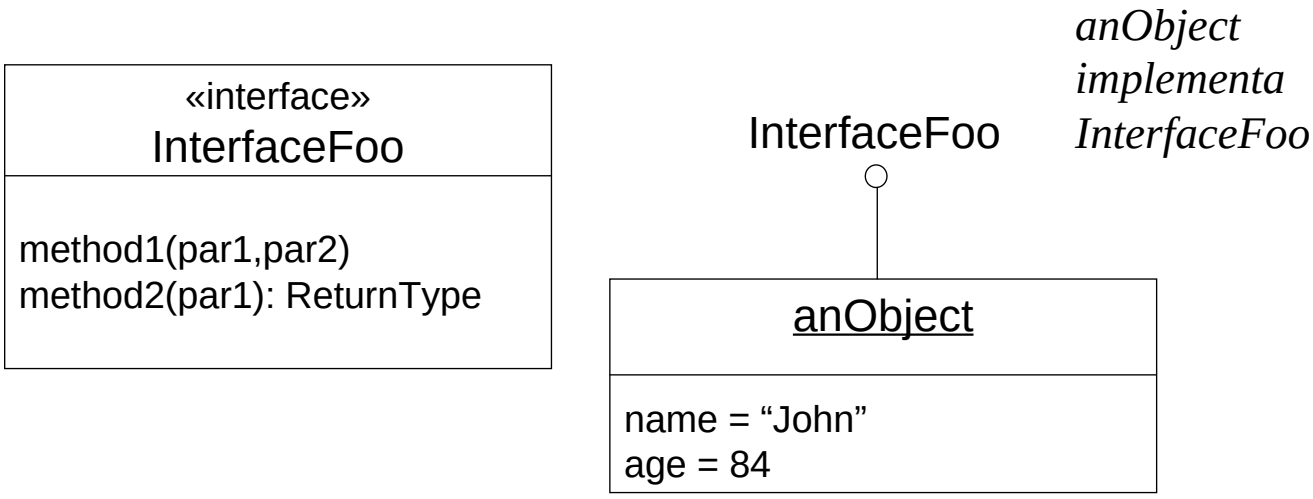
«interface»
«singleton»
«constructor»

notazione UML per gli oggetti





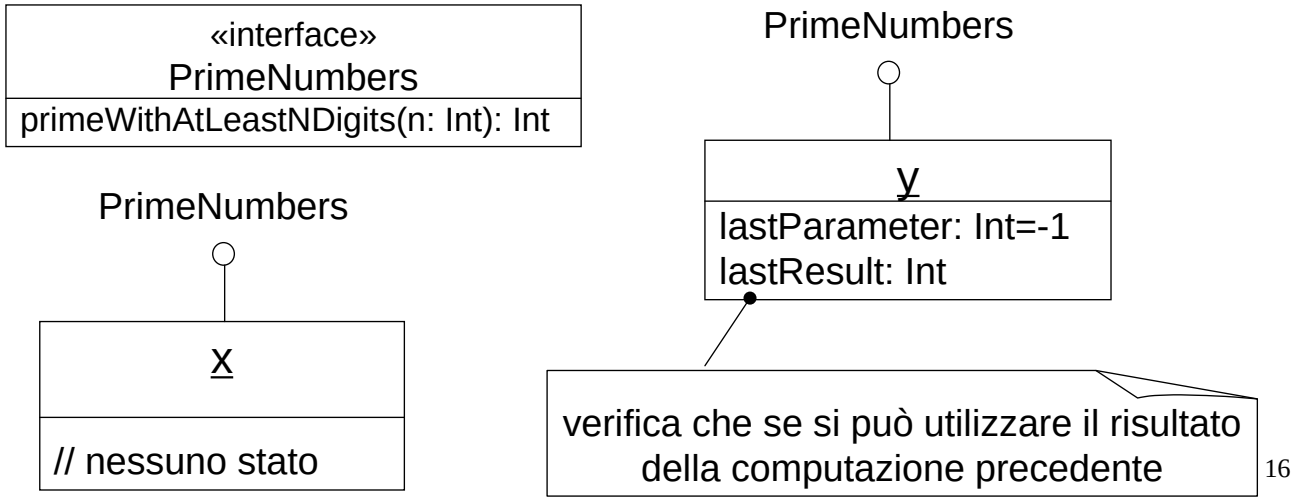
notazione UML per le interfacce



- una interfaccia non è istanziabile direttamente
- le istanze di una interfaccia sono le istanze di classi che implementano l’interfaccia

esercizio

- dare un esempio di interfaccia
- dare due oggetti diversi che la realizzano
  - uno con stato interno ed esterno uguali
  - l’altro il cui stato interno contiene più informazione dello stato esterno (a che serve questa informazione in più?)



interazioni tra oggetti

- messaggi tra oggetti = chiamata di metodi
- due ruoli
  - chi invia il messaggio (oggetto che chiama il metodo)
    - client object
  - chi riceve il messaggio (oggetto chiamato)
    - server object
- effetti dei messaggi
  - input: parametri attuali
  - output: valori di ritorno
  - effetti collaterali
    - su stato del destinatario
    - su altri oggetti

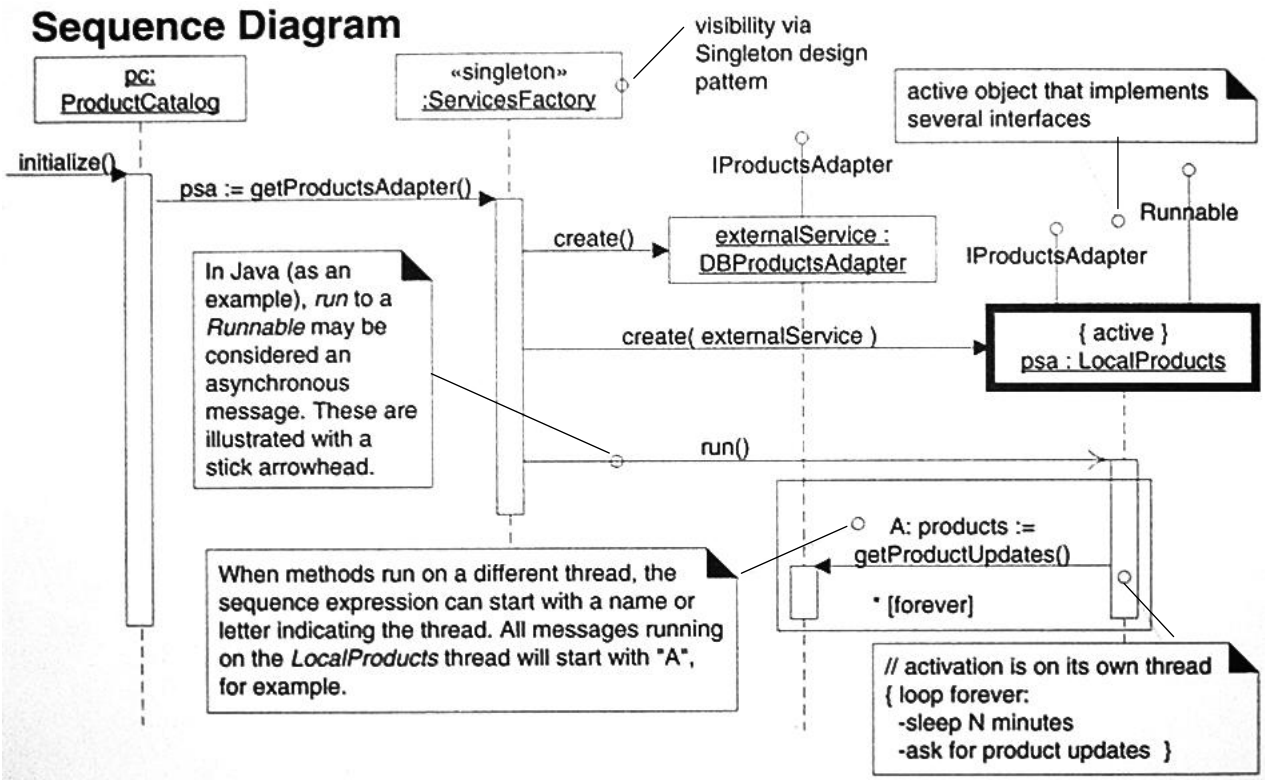
messaggi: sincroni e asincroni

- un messaggio è sincrono se chi invia si ferma attendendo la risposta
  - questo è il default in tutti i linguaggi di programmazione
  - è il caso di gran lunga più importante
- un messaggio è asincrono se chi invia il messaggio non attende la risposta
  - il supporto a questo concetto richiede un supporto per il concetto di thread
  - estremamente dipendente dal linguaggio di programmazione
  - per semplicità ignoriamo i messaggi asincroni

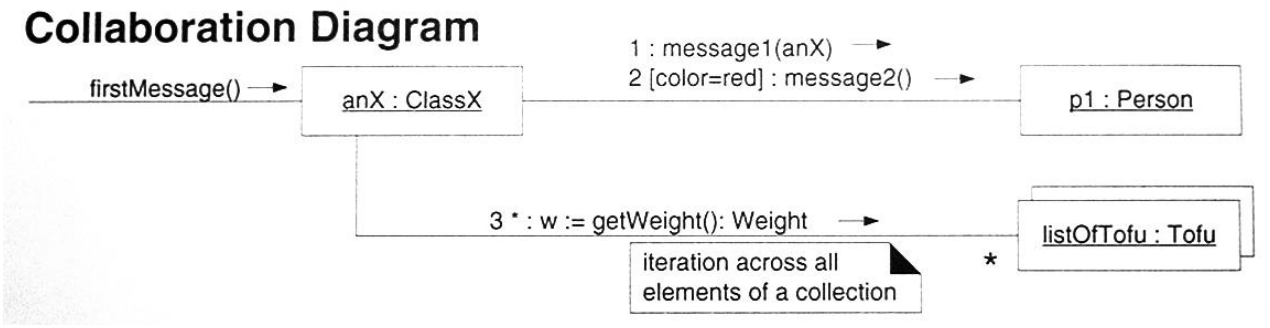
notazione UML per l'interazione

- interaction diagrams
  - sequence diagram: evidenzia l'evoluzione nel tempo dei messaggi e l'attività dei singoli oggetti
    - utili da affiancare a documenti use case
  - collaboration diagram: evidenzia le relazioni che scaturiscono dalle interazioni tra gli oggetti
  - stesso potere espressivo ma differenti trade-off di espressività
    - i sequence diagram sono più semplici da capire rispetto ai collaboration diagram ma occupano più spazio e quindi sono adatti a scenari semplici

sequence diagrams



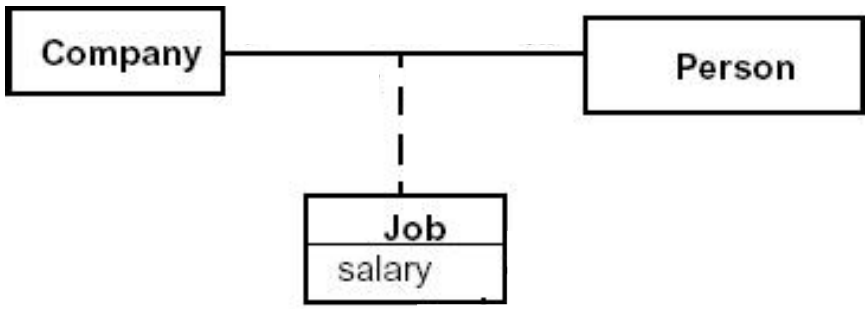
collaboration diagrams



esercizio

- considera gli oggetti:
  - cliente, distributore, magazzino, corriere
- un cliente contatta il distributore per ordinare della merce
- disegna un possibile sequence diagram e collaboration diagram

realzioni tra classi: associazione



- lo standard prevede anche associazioni più che binarie
  - es. ternarie

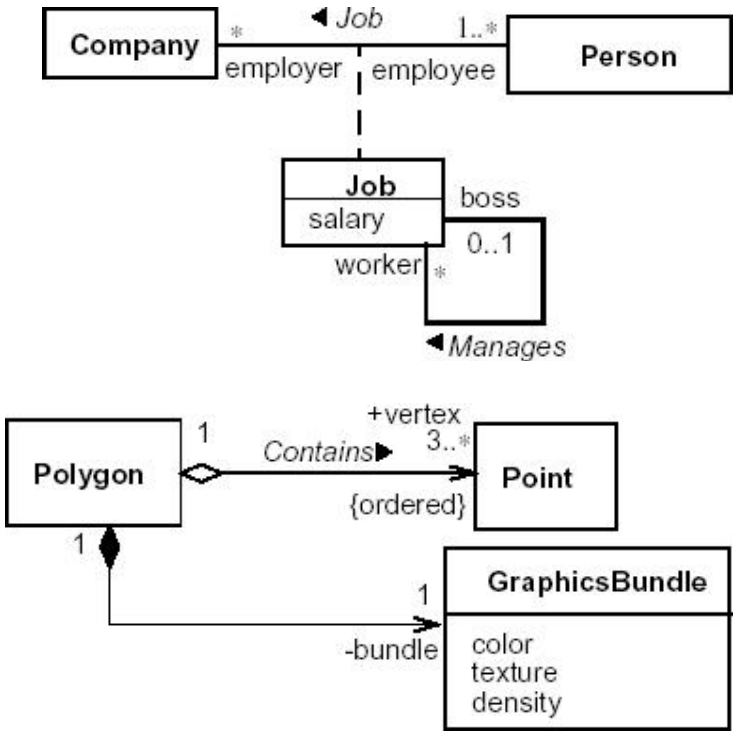
realzioni tra istanze: links

- un'istanza di una associazione è detta *link*
- un link è una tupla di istanze
  - per le associazioni binarie è una coppia

- negli interaction diagram i messaggi passano sui link

adornments

- molteplicità
- nomi di associazione
- ruoli e loro visibilità
- ordinamento
- navigabilità
- aggregation
  - composizione
  - sherable aggr.



aggregazione

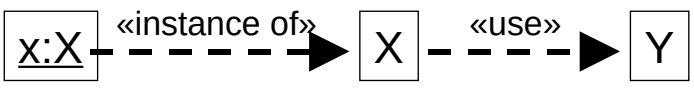
- aggreagzione
  - composition
    - gli oggetti parti sono contenute in al più un oggetto composto
    - la molteplicità sul composto dice se una parte può avere vita propria o no
    - propagation semantic
      - se un composto viene distrutto anche le sue parti vengono distrutte
      - se un composto viene copiato anche le sue parti vengono copiate
    - le istanze formano un insieme di alberi
  - shareble aggregation
    - semantica non ben definita in UML, da specificare nel modello
      - propagation?
    - le istanze formano un grafo aciclico
  - entrambe sono relazioni asimmetriche e transitive

ubiquità della composizione

- è utile pensare a qualsiasi oggetto come contenuto in un altro oggetto
  - tutti gli oggetti potrebbero così essere organizzati in un grande albero di relazioni di composizioni
  - ciascun oggetto ha un solo padre (l'oggetto contenitore)
- esercizio: individua il possibile contenitore (la risposta non è univoca) di
    - un file
    - una struttura dati (es. l'istanza di una lista)
    - un processo
    - un programma
    - un driver
    - un capitolo
    - una parola
    - una nota a piè di pagina
  - nota che le risposte non sono univoche
    - da che dipendono?

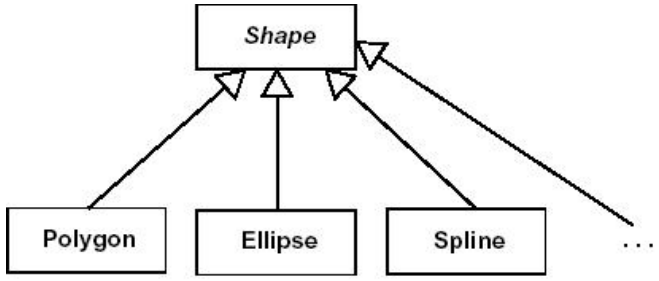
dipendenza semantica

- A dipende da B se esiste un cambiamento di B che modifica o invalida il comportamento di A
- A e B possono essere
  - classi
  - oggetti
  - package
  - use case
  - ecc,
  - A e B non necessariamente sono entrambi classi, oggetti, ecc.
- esempio
    - use
    - include
    - instance of (l'istanza dipende dalla classe)
- notazione:

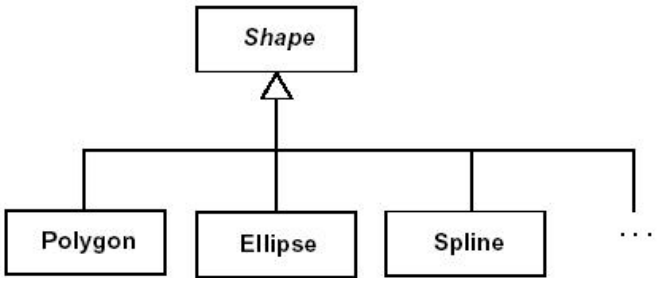


generalizzazione

- permette di definire dettagli del modello a vari livelli di astrazione



Separate Target Style



Shared Target Style

esercizio

- considera le classi:
  - cliente, distributore, magazzino, corriere
- disegna un possibile class diagram per tali classi
  - non è necessario specificare attributi e operazioni
  - poni attenzione invece alle associazioni e alle molteplicità