TecnetDati
IT Professional Services

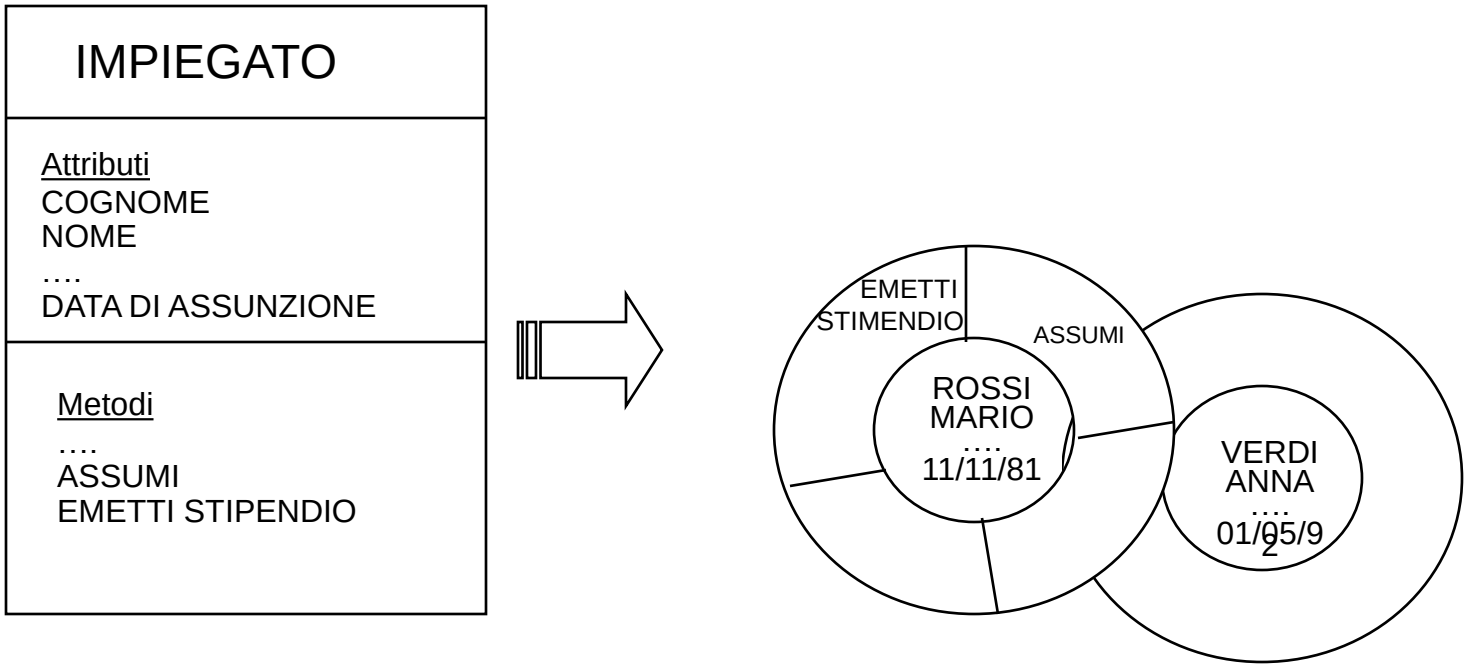
Il paradigma Object Oriented

Iolanda Salinari

gli oggetti

- un oggetto è un elemento o concetto del mondo reale che può essere identificato in modo univoco: un cliente, un articolo, un impiegato …
- ogni oggetto è costituito dall'insieme dei suoi dati caratteristici e delle funzioni che operano su di essi per fornire determinati servizi al resto del sistema
- ciascun oggetto è descritto attraverso:
  - **attributi**, che ne definiscono le caratteristiche
  - **metodi** (funzioni), che ne descrivono i comportamenti

gli oggetti e le classi



- Rossi Mario e Verdi Anna sono oggetti (**istanze**) della classe IMPIEGATO

gli oggetti e le classi

- tutti gli oggetti che condividono le stesse proprietà (attributi) e gli stessi comportamenti (metodi) possono essere raccolti in **classi**
- i metodi sono comuni per tutti gli oggetti della classe
- anche il formato dei dati è comune per tutti gli oggetti di una classe, mentre il contenuto può differire per ogni oggetto
- *la classe è responsabile di creare nuovi oggetti* (nuove istanze), tutti uguali, ma dotati ciascuno di una propria individualità

Incapsulazione

- *metodi e dati sono incapsulati nell'oggetto*; dall'esterno né metodi né dati sono direttamente accessibili
- i dati (attributi) di un oggetto possono essere manipolati solo attivando i suoi metodi
- un metodo è attivato dall'oggetto a fronte di una specifica richiesta di servizio (**messaggio**)
- solo l'interfaccia pubblica dell'oggetto, costituita dai messaggi che ad esso si possono inviare (servizi che l'oggetto può offrire), è nota all'esterno
- si comunica con un oggetto inviandogli un messaggio in cui si specifica che cosa si vuole da esso

effetti dell'incapsulazione

Ogni classe di oggetti è definita in due modi, uno esterno
(pubblico) ed uno interno (privato)

Livello esterno o interfaccia pubblica

- specifica i messaggi (e gli argomenti ad essi associati) che possono essere inviati agli oggetti della classe

Livello interno o privato

- definisce le proprietà dell'oggetto (i dati) e le modalità di trattamento di questi dati (i metodi)

effetti dell'incapsulazione

- l'interfaccia pubblica specifica i servizi che l'oggetto può fornire
- rappresenta ciò che l'oggetto "sa fare" nello scenario applicativo, le sue ***responsabilità***

L'incapsulazione soddisfa le seguenti esigenze:

- la necessità di distinguere nettamente tra specifica e implementazione di un servizio (operazione)
- il bisogno di modularità

information hiding

- gli oggetti ***nascondono*** la loro struttura all'ambiente circostante
- per usare un oggetto non è necessario conoscere la sua struttura interna, è sufficiente conoscere le operazioni che esso offre

Vantaggi dell' infomation hiding:

- semplicità d'uso
- effetti benefici sulla manutenzione del software: è possibile apportare variazioni all'oggetto in modo trasparente per le applicazioni che lo usano

i messaggi

- i messaggi sono i meccanismi di comunicazione tra gli oggetti
- il messaggio è lo stimolo a svolgere un'azione
- la ricezione di un messaggio da parte di un oggetto causa in questo l'esecuzione di un'operazione

struttura del messaggio

id oggetto nome-metodo (arg1, arg2, …)

Mioconto preleva (100 lire)

i metodi

UML distingue tra operazione e metodo:

- l'**operazione** è la specifica di un servizio offerto dalla classe
- il **metodo** ne è l'implementazione

Fanno parte della specifica il messaggio e gli argomenti ad
esso associati

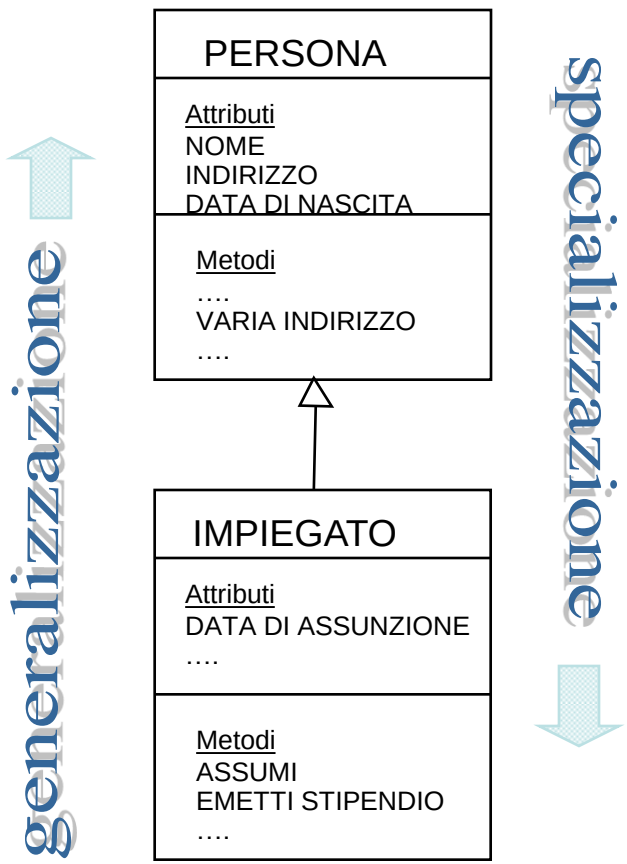
Il metodo consiste di un blocco di istruzioni in grado di:

- fornire i dati dell'oggetto
- variare i dati dell'oggetto
- eseguire altre operazioni relative all'oggetto

i metodi

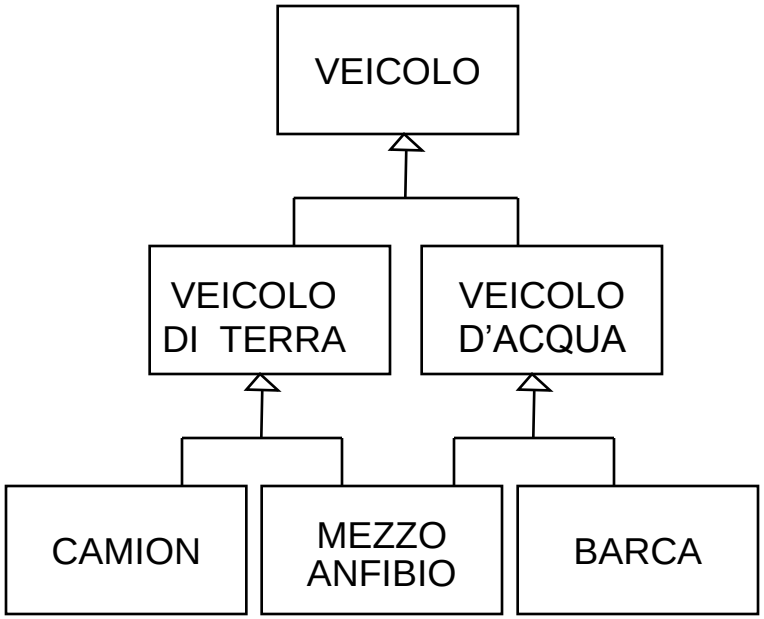
- quando un messaggio è inviato ad un oggetto, esso fa ciò che deve con i suoi dati interni e restituisce una risposta
- è responsabilità dell'oggetto ricevente conoscere e scegliere le funzioni opportune da applicare ai dati
- l'orientamento a oggetti sposta la ***responsabilità*** sui dati e le funzioni dal consumatore (client) al fornitore (server)

ereditarietà (specializzazione)



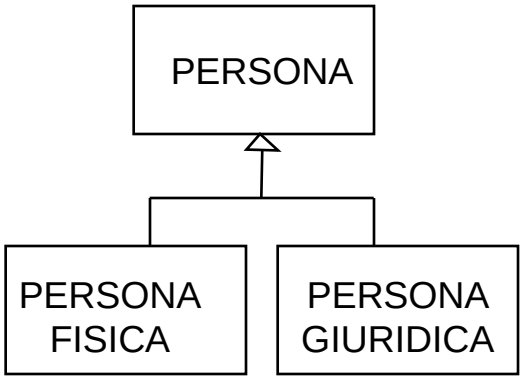
- permette ad una classe detta **sottoclasse** di essere definita a partire da un'altra classe detta **superclasse**
- tra le classi Persona e Impiegato esiste una relazione IS-A o **gerarchia di specializzazione-generalizzazione**
- la sottoclasse eredita gli attributi, le operazioni (e quindi anche i messaggi) della sua superclasse
- ogni sottoclasse può arricchire l'eredità ricevuta con un corredo proprio di attributi e operazioni
- l'ereditarietà promuove il ***riuso***

ereditarietà multipla



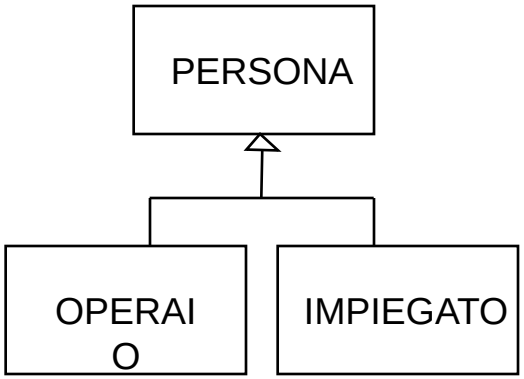
- quando una classe eredita da più superlassi si parla di ***ereditarietà multipla***
- nell'esempio *MEZZO ANFIBIO* eredita sia da *VEICOLO DI TERRA* che da *VEICOLO D'ACQUA*
- non tutti i linguaggi di programmazione OO prevedono questa possibilità
- ma… come vengono risolti i conflitti quando nelle superclassi esiste un attributo (o un'operazione) con lo stesso nome, ma significati diversi?... Quale sarà ereditato nella sottoclasse?

specializzazione esaustiva



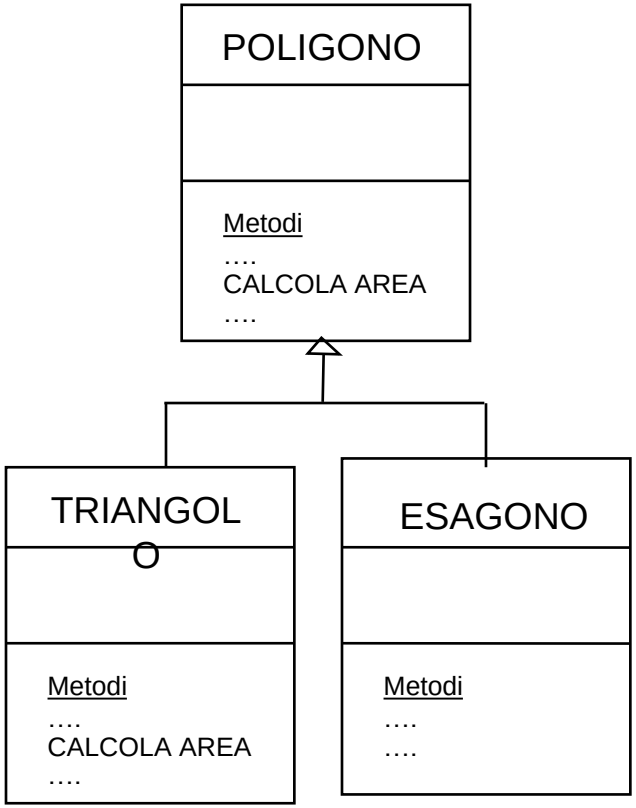
- nell'esempio la specializzazione è ***esaustiva*** o ***totale***
- la classe *PERSONA* è una **classe astratta**, non istanziabile
- è una classe "vuota": non esistono oggetti *PERSONA*, ma solo oggetti *PERSONA FISICA* o *PERSONA GIURIDICA*
- le classi astratte servono a raccogliere attributi e operazioni comuni a più sottoclassi
- le classi *PERSONA FISICA* o *PERSONA GIURIDICA* sono **classi concrete**

specializzazione non esaustiva



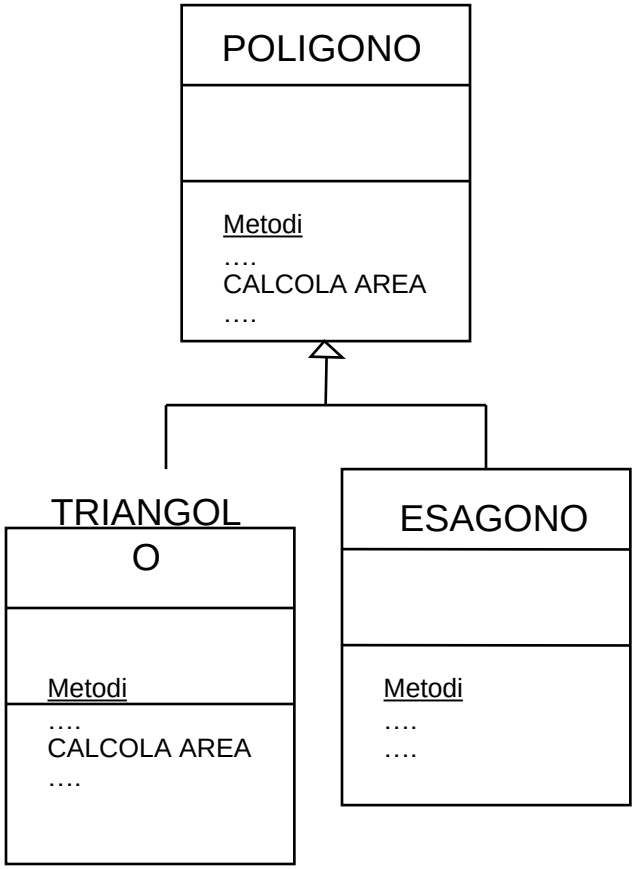
- nell'esempio la specializzazione è ***non esaustiva*** o ***parziale***
- la classe *PERSONA* è una **classe concreta**, istanziabile
- esistono oggetti *PERSONA*, che non sono né *OPERAIO* né *IMPIEGATO*

overriding



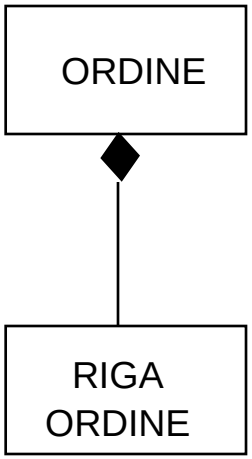
- consente di ***ridefinire*** qualche comportamento (operazione) o qualche informazione ereditati
- nell'esempio l'operazione *calcola area* è ridefinita nella sola sottoclasse TRIANGOLO
- nella superclasse POLIGONO l'operazione è implementata con il seguente algoritmo:
  *perimetro * apotema / 2*
- nella sottoclasse TRIANGOLO l'operazione è implementata con un altro algoritmo:
  *base * altezza / 2*

polimorfismo



- *significa letteralmente: capacità di assumere forme diverse*
- TRIANGOLO e ESAGONO appartengono alla stessa superclasse, entrambe le sottoclassi sanno rispondere al messaggio *calcola area*, ma lo fanno con un comportamento diverso (il metodo è diverso!)
- ***polimorfismo*** sta a indicare la capacità di un oggetto di attivare metodi diversi, a seconda della propria classe, alla ricezione del medesimo messaggio

aggregazione



- è il meccanismo per la costruzione di oggetti **complessi** aggregando oggetti **componenti** più semplici
- esprime la relazione ***is-part-of*** tra una *Parte* e il *Tutto*
- un oggetto aggregato
  - si presenta come un "Tutt'uno"
  - può essere usato senza doverne "conoscere" la composizione interna (principio di incapsulazione)

classe e tipo

una **classe**

- rappresenta l'insieme degli oggetti che condividono la stessa struttura interna (dati e metodi)
- è il modello di fabbricazione per più oggetti
- è un meccanismo orientato al ***riuso***

un **tipo**

- rappresenta la specifica dell'interfaccia di un insieme di oggetti, descrive come questi oggetti possono essere usati
- è un meccanismo orientato all'***usabilità***

Classe e tipo sono due modi diversi di guardare agli oggetti: l'oggetto è

- un **tipo** per l'utilizzatore
- una **classe** per il progettista

benefici dell'OO

Riusabilità

- gli oggetti delle classi possono essere riutilizzati in contesti diversi, attivandoli con gli opportuni ***messaggi***
- se l'oggetto non presenta tutte le funzioni richieste, si può specializzare sfruttando il meccanismo dell'***ereditarietà***

Robustezza del SW e manutenzione più semplice

- è nota la responsabilità di ciascun oggetto nello scenario applicativo: è sempre possibile identificare la componente che ha comportamenti anomali
- le modifiche al software sono localizzate: normalmente si deve modificare un metodo alla volta
- è possibile apportare variazioni all'oggetto in modo trasparente per le applicazioni che lo usano (quando le modifiche non hanno impatto sull'interfaccia pubblica)

benefici dell'OO

Integrità dei dati

- i dati possono essere manipolati solo da specifici metodi, che devono garantire il rispetto delle "regole di business"

Controllo della complessità

- ereditarietà e aggregazione sono criteri di astrazione che consentono di scomporre il sistema in componenti elementari, facilitando la comprensione del problema e il dominio della complessità

Interoperabilità

- il meccanismo dei messaggi permette la comunicazione e la collaborazione di componenti realizzate su piattaforme diverse
- ….

TecnetDati
IT Professional Services

Grazie per l'attenzione!

Tecnet Dati s.r.l.
C.so Svizzera 185 –
10149 - Torino (TO), Italia
Tel.: +39 011 7718090 Fax.: +39 011 7718092
P.I. 05793500017 C.F. 09205650154
www.tecnetdati.com