



Programmazione III

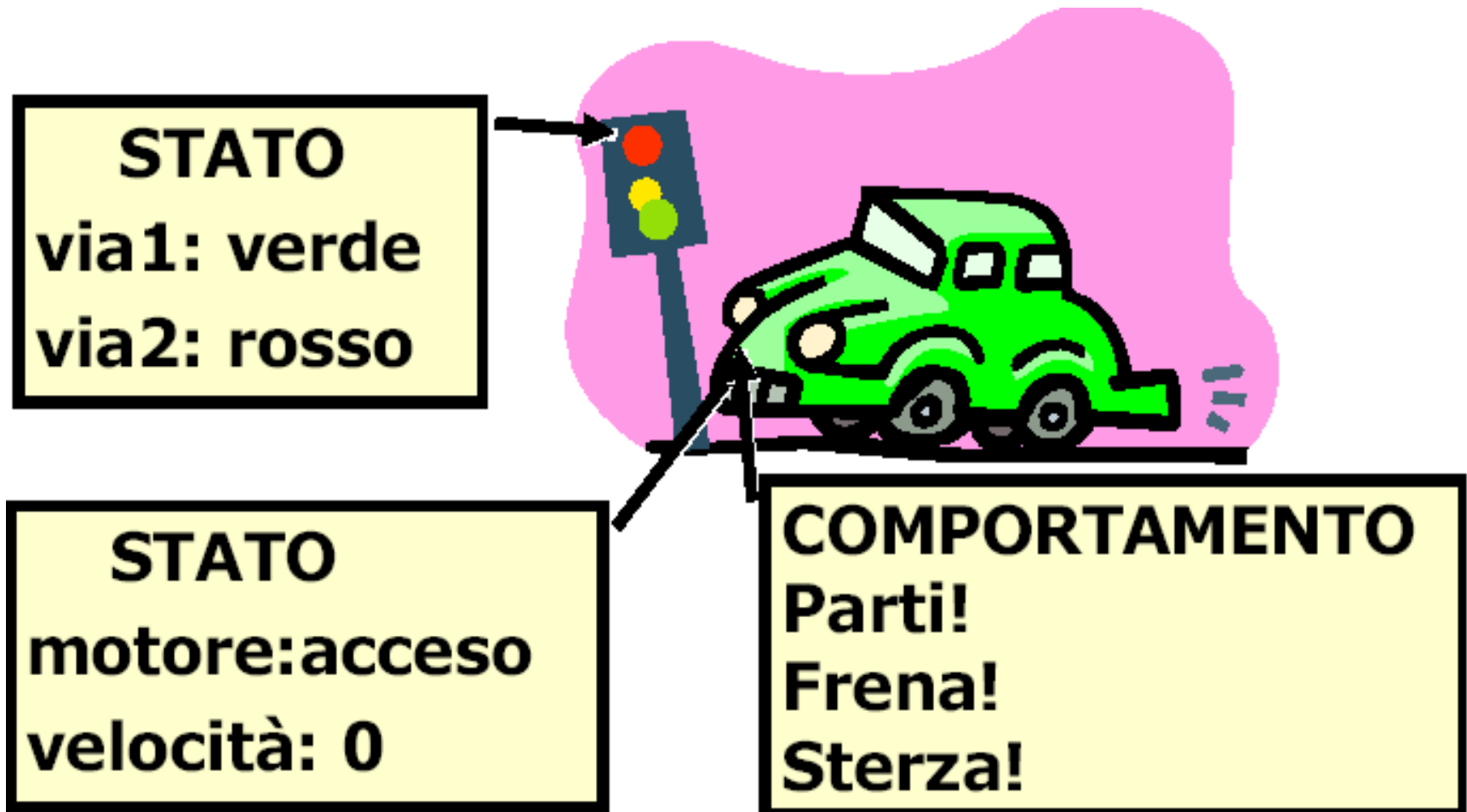
Prof.ssa Liliana Ardissono
Dipartimento di Informatica
Università di Torino

Introduzione alla progettazione a oggetti



Questa presentazione è distribuita sotto licenza Creative Commons CC BY ND

Modellare la realtà - I



Modellare la realtà - II



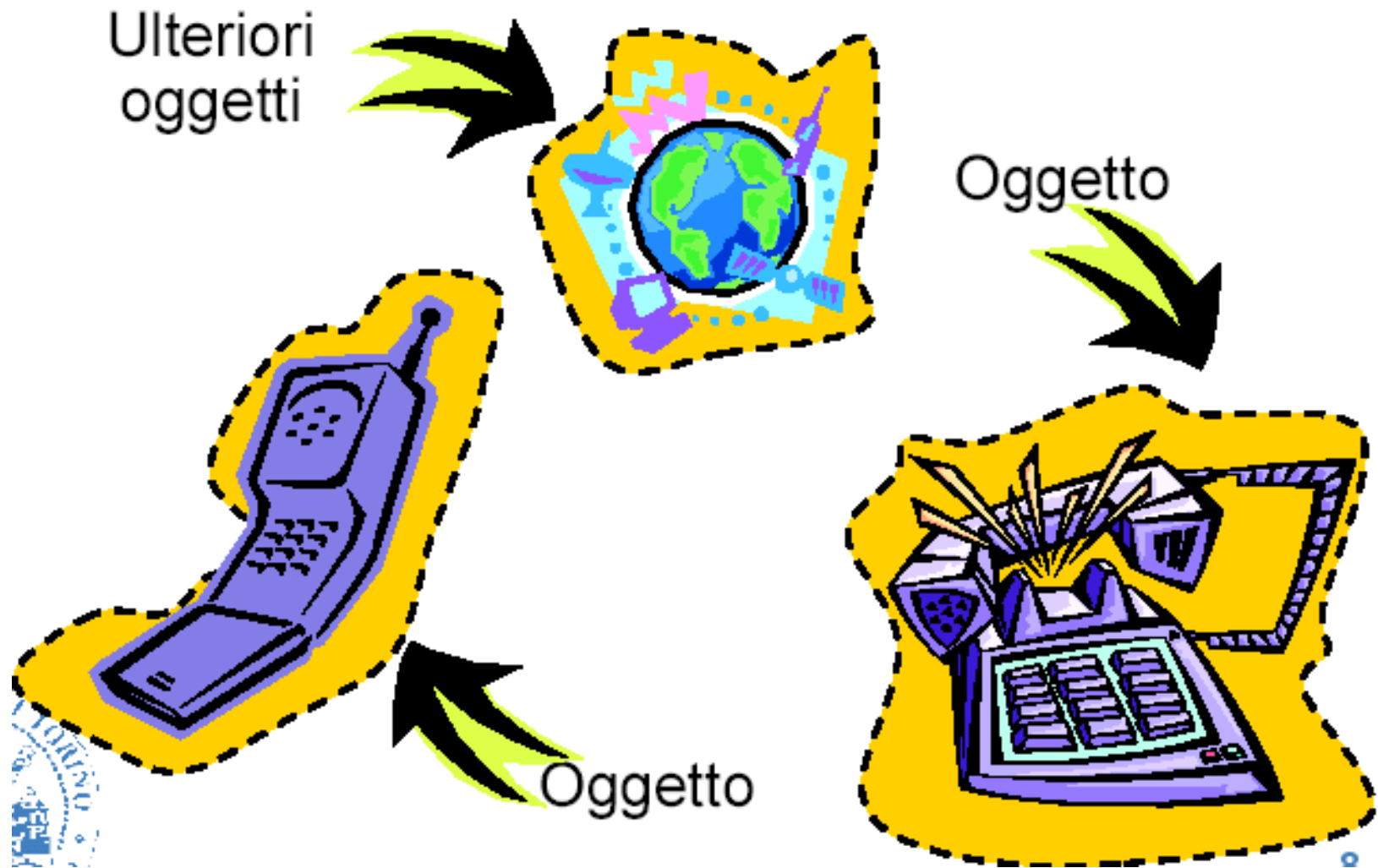
- **Stato**

- L'insieme dei parametri caratteristici che contraddistinguono un oggetto in un dato istante
- Modellato come insieme di **attributi**

- **Comportamento**

- Descrive come si modifica lo stato a fronte degli stimoli provenienti dal mondo esterno
- Modellato come insieme di **metodi**

Approccio nell'osservare il mondo



Oggetti e realtà



- Il mondo fisico è costituito da un insieme di oggetti variamente strutturati che interagiscono tra loro
- Ciascuno è dotato di:
 - Una propria **identità** (è riconoscibile)
 - Uno **stato** (ricorda la storia passata)
 - Un **comportamento** (reagisce a stimoli esterni in un modo prevedibile)
- Si può estendere la metafora al software
 - Ogni entità logica che deve essere manipolata può essere immaginata come un “oggetto”

Stato



- Ogni oggetto ha uno **stato**:
 - L'insieme dei parametri caratteristici che contraddistinguono un oggetto in un dato istante
- Composto da un **gruppo di “attributi”**
 - Ogni attributo modella un particolare aspetto dello stato
 - Può essere un valore elementare o un altro oggetto
- Implementato **mediante un blocco di memoria**
 - Contiene i valori degli attributi
- Principio fondamentale: **incapsulamento**
 - Lo stato “appartiene” all'oggetto
 - Un utente esterno non può manipolare direttamente lo stato di un oggetto

Comportamento



- Gli oggetti interagiscono a seguito di “richieste esterne” (invocazioni di metodi)
 - Dotate di eventuali parametri che ne specificano i dettagli
- Ogni oggetto sa reagire ad un ben determinato insieme di invocazioni di metodi
 - Costituiscono la sua interfaccia
- Ad ogni richiesta è associato un comportamento
 - Modifica dello stato
 - Invio di richieste verso altri oggetti
 - Comunicazione di informazioni (risultato)
- Implementato attraverso un blocco di codice (metodo)
- Principio fondamentale: delega - chi effettua la richiesta non vuole conoscere i dettagli di come la richiesta sia evasa

La programmazione orientata agli oggetti

(secondo Alan Kay – Smalltalk)

- Ogni cosa è un **oggetto**
- Un programma è un insieme di *oggetti* che si “dicono l’un l’altro” che cosa fare invocando i metodi reciprocamente offerti
- Ogni *oggetto* può contenere riferimenti ad altri *oggetti*
- Ogni *oggetto* ha un tipo (**classe**), cioè ogni *oggetto* ha proprietà strutturate in campi definiti dalla *classe*
- Tutti gli *oggetti* di un determinato tipo possono rispondere alle **stesse invocazioni di metodi**

Tipo di dato astratto



- Definisce le operazioni fondamentali sui dati, ma non ne specifica l'implementazione

Per es. una *lista* (astratta) è una sequenza ordinata di dati

- le operazioni possibili sono:
 - lettura sequenziale
 - inserimento/rimozione di un elemento in posizione *i*-esima

Una **struttura-dati** è vista come l'insieme di operazioni (*servizi*) offerti



Il ruolo dell'astrazione

- Astrazioni procedurali
- Astrazioni dei dati

Obiettivo: trattare cose complesse come primitive e nascondere i dettagli

Domande:

- Quanto è facile suddividere il sistema in moduli di astrazioni?
- Quanto è facile estendere il sistema?

La programmazione orientata agli oggetti

Object-oriented design = progettazione (e sviluppo) orientato agli oggetti = costruzione di sistemi **software visti come collezioni strutturate di (implementazioni di) strutture-dati astratte** [B. Meyer, "Object-oriented software construction ", Prentice Hall, 1988, cap.4.8]

Programmazione: procedurale vs OO

- Programmazione procedurale
 - Organizzare il sistema intorno alle procedure che operano sui dati
- Programmazione ad oggetti
 - Organizzare il sistema intorno ad oggetti che interagiscono mediante invocazione di metodi
 - Un oggetto *incapsula* dati e operazioni

La programmazione procedurale

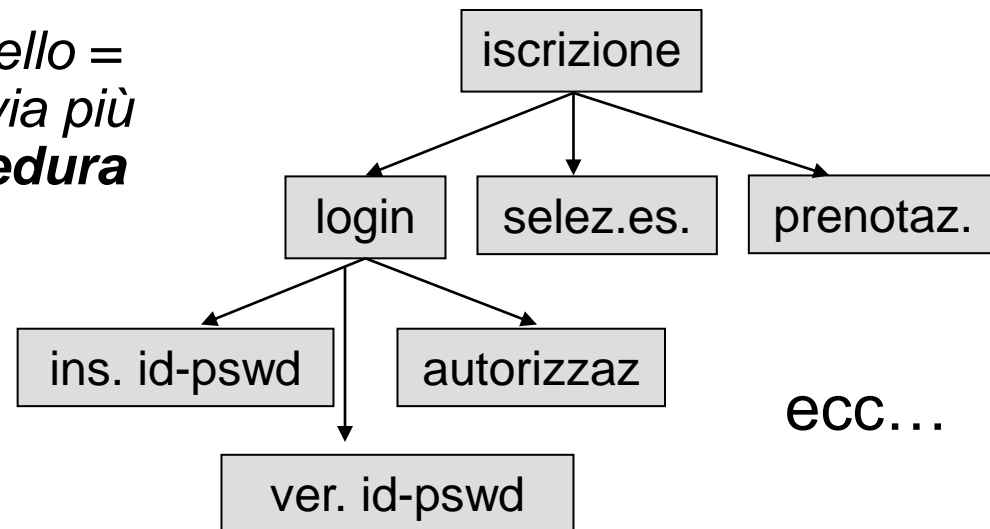


Metodo classico di software design = *top-down functional (structured) design* = scomposizione gerarchica funzionale (algoritmica)



paradigma procedurale: algoritmo = procedura = **sequenza di passi per raggiungere il risultato**

*Per es. iscrizione ad un appello = scomposizione in passi via via più semplici, elementari = **procedura** per iscriversi ad un appello*



PROGRAMMI = ALGORITMI + STRUTTURE-DATI

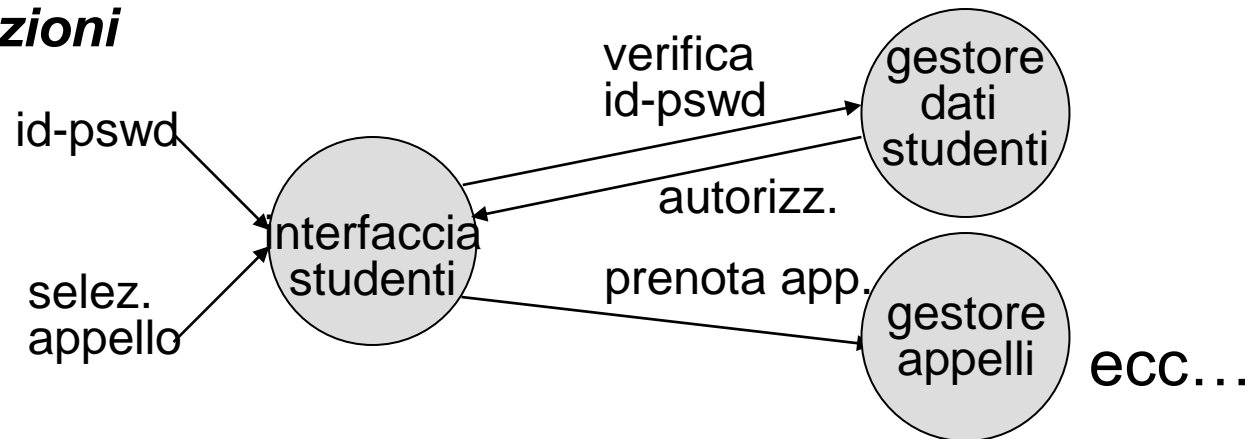
La programmazione orientata agli oggetti

Metodo alternativo = **object-oriented design** = si parte dagli oggetti, non dalle funzionalità!



paradigma ad oggetti: oggetti che interagiscono tra loro mediante invocazione di metodi = collaborazione per raggiungere il risultato

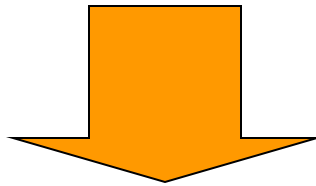
Per es. iscrizione ad un appello = **entità** coinvolte nell'attività e loro **relazioni**



**PROGRAMMI = OGGETTI (DATI + ALGORITMI) +
COLLABORAZIONE (INTERFACCE)**

Sviluppare un programma ad oggetti

- Un oggetto è un *fornitore di servizi*
- Un programma fornisce un servizio agli utenti e lo realizza utilizzando servizi di altri oggetti



- Obiettivo: produrre (o trovare librerie di oggetti già esistenti) l'insieme di oggetti che forniscono i servizi ideali per risolvere il problema

Progettare a oggetti



- Si parte dal testo delle specifiche
 - Si individuano i nomi e i verbi
- Tra i **nomi**, si individuano le possibili classi di oggetti
 - Con i relativi attributi
- Tra i **verbi**, si individuano metodi e relazioni
 - Di solito, i verbi di azione si modellano come metodi (“X apre Y”), quelli di stato come relazioni (“A si trova presso B”)
 - L’interazione tra due oggetti sottende l’esistenza di una relazione tra gli stessi

La programmazione orientata agli oggetti

Che cos'è un "oggetto"?

Passo 1: Distinguere classi e istanze

Passo 2: Distinguere interfaccia e implementazione

Passo 1: Distinguere classi e istanze

- **Un'istanza (oggetto)** è un'entità concreta, che esiste nel tempo (viene costruita e poi distrutta) e nello spazio (occupa memoria)
- **Una classe** è un'astrazione che rappresenta le **proprietà comuni** (struttura e comportamento) ad un insieme di oggetti concreti (istanze)

La programmazione orientata agli oggetti

Esempio: supponiamo di gestire una biblioteca, che contiene moltissimi libri

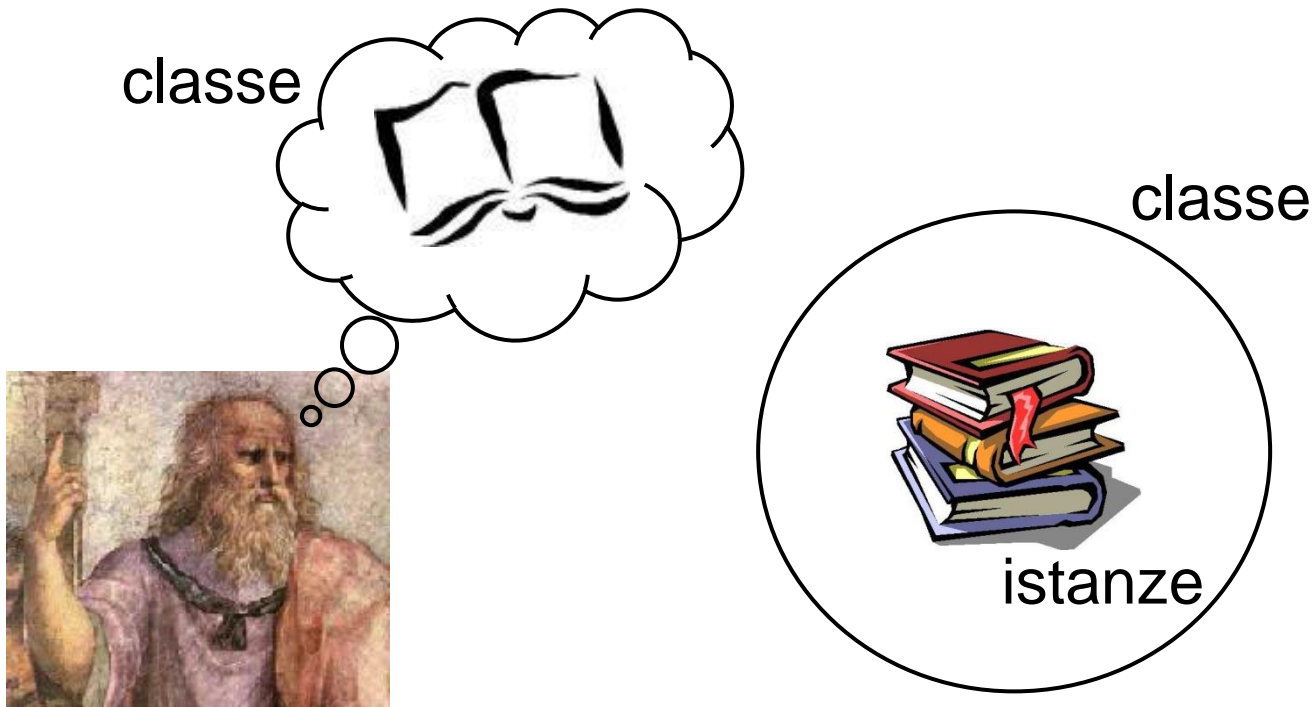
Una **classe** è...

- L'insieme di tutti i libri, la classe dei **libri**
- La proprietà *Libro*(*x*), che definisce l'appartenenza all'insieme dei libri, ed è vera per tutti i libri (gli oggetti *x* che sono libri)
- L' "idea platonica" di libro, il prototipo ideale di libro, che esiste solo nel mondo delle idee; tutti i libri della nostra biblioteca "partecipano" dell'idea di libro, da momento che sono libri!
- Il concetto mentale di libro, che esiste solo nella nostra testa e di cui i libri del mondo sono degli esempi concreti
- ...

La programmazione orientata agli oggetti

Un'**istanza (oggetto)** è...

- Un singolo libro concreto (che può essere preso in prestito, restituito, distrutto, fotocopiato, ecc...)



La programmazione orientata agli oggetti

⇒ Una **classe** può essere vista come la definizione di un **tipo di dato astratto**

Per esempio, supponiamo che la biblioteca riceva un nuovo libro. Per prima cosa il bibliotecario deve classificarlo come libro (e non rivista, CD Rom, o altro), dichiarando quindi che l'oggetto appena arrivato è un libro.

Questo equivale a dichiarare che **il nuovo oggetto è di tipo "libro"** (cioè che alla domanda "cos'è questo?" rispondiamo "è un libro!")

Un tipo è un modello (un *template*) che definisce il comportamento e la struttura di un'insieme di istanze (oggetti).

Per esempio, il tipo "libro" definisce le operazioni che possono essere fatte sui libri (istanze): prestito, restituzione, ecc...

La programmazione orientata agli oggetti

⇒ Un'**istanza** è un oggetto concreto (di un certo tipo, cioè appartenente ad una certa classe), caratterizzato da:

- **un'identità**: possibilità di identificare univocamente l'oggetto
- **uno stato**: l'insieme dei valori dei suoi attributi, in un certo tempo t
- **un comportamento**: l'insieme delle operazioni (funzionalità) offerte dall'oggetto, cioè le cose che l'oggetto è in grado di fare

La programmazione orientata agli oggetti



Torniamo al nostro esempio:

supponiamo di gestire una biblioteca, che contiene molti libri; nella biblioteca c'è un bibliotecario che classifica i nuovi libri, assegna i prestiti, ecc. e ci sono degli utenti che prendono in prestito i libri della biblioteca

Quali sono gli **oggetti** coinvolti nello scenario?

In particolare, quali sono le **classi** e quali le **istanze**?

Abbiamo bisogno dei seguenti **concetti (classi, tipi)**:

- la Biblioteca →
- il Bibliotecario → 
- il Libro → 
- l'Utente →



La programmazione orientata agli oggetti

Per ogni **concetto (classe, tipo)** di quali **proprietà (attributi, caratteristiche)** abbiamo bisogno per descriverlo in modo adeguato?



- per la Biblioteca:

- nome
- indirizzo
- orario apertura



- per il Bibliotecario:

- nome
- turno



- per il Libro:

- autore
- titolo
- editore
- collocazione



- per l'Utente

- nome
- cognome
- telefono

La programmazione orientata agli oggetti

Per ogni concetto (classe, tipo) di quante **istanze** (oggetti concreti) abbiamo bisogno?

- **una sola biblioteca** (un'istanza della classe Biblioteca), per la quale, per es:
 - nome = Biblioteca A. Gramsci
 - indirizzo = via Tizio 32, Roma
 - orario apertura = lun-sab 9:00-19:00
- **2 bibliotecari** (istanze della classe Bibliotecario), uno per il turno del mattino e uno per il turno del pomeriggio, per i quali, per es:



bibliotecario 1:

- nome = Paolo
- turno = mattino

bibliotecario 2:

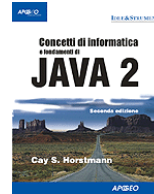
- nome = Luca
- turno = pomeriggio

La programmazione orientata agli oggetti

- **un grande numero di libri** (istanze della classe Libro), per i quali, per es:

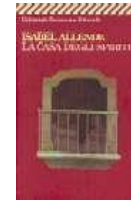
libro 1:

- o autore = C.S. Horstmann
- o titolo = Java 2
- o editore = Apogeo
- o collocazione = S21/L303



libro 2:

- o autore = I. Allende
 - o titolo = La casa degli spiriti
 - o editore = Feltrinelli
 - o collocazione = S13/L44
- ecc...



- **un certo numero di utenti** (istanze della classe Utente), per i quali, per es:

utente 1:

- o nome = Maria
 - o cognome = Bianchi
 - o telefono = 011 1234567
- ecc...

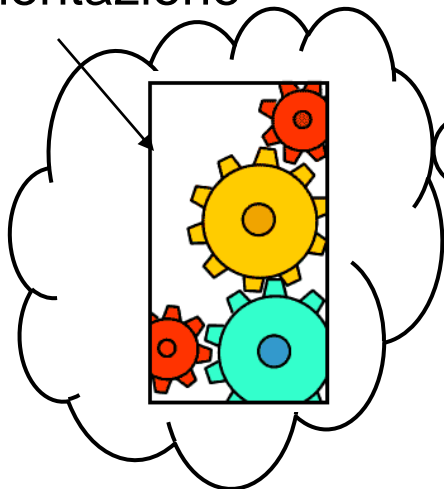
La programmazione orientata agli oggetti

Passo 2: Distinguere interfaccia e implementazione

Quando definisco una **classe (tipo)** ne definisco:

- l'**interfaccia** = la “vista esterna” = l’insieme di operazioni che le sue istanze potranno fare
- l'**implementazione** = la “vista interna” = la definizione dei meccanismi che realizzano le operazioni definite nell’interfaccia

implementazione



interfaccia





- 

restituzione(libro)

prenotazione(libro)

- questi servizi (operazioni) sono **accessibili al pubblico**

= interfaccia

La programmazione orientata agli oggetti

Come li **implementa** (realizza)?

prestito(libro) →

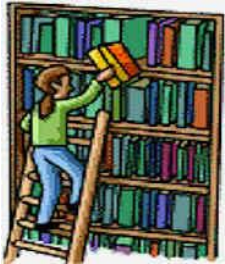
procedura, per trovare il libro,

prenderlo, darlo all'utente,

registrare il

prestito sulla scheda, ...

**non sono visibili
al pubblico**



= implementazione

La programmazione orientata agli oggetti

L'**interfaccia** definisce dunque il **comportamento** di un oggetto: nell'esempio, i servizi, cioè le operazioni di *prestito(libro)*, *restituzione(libro)*, *prenotazione(libro)* definiscono il comportamento dei bibliotecari (cioè di tutte le istanze della classe Bibliotecario: Paolo e Luca nell'esempio)

Come avviene l'**interazione** con un oggetto?

- avviene con un'**istanza** (con Paolo o Luca) e non con la classe (non si interagisce con il concetto di Bibliotecario!)
- **Invocando un metodo su un'istanza**, con il quale gli si chiede il servizio desiderato; nell'esempio di deve parlare o scrivere a Paolo o Luca per avere un libro in prestito, o per restituirlo, o prenotarlo

La programmazione orientata agli oggetti



ogni biblioteca può
1 avere tanti (n) bibliotecari **n**

ogni bibliotecario può appartenere
ad una sola biblioteca



1

m

ogni
biblioteca
può avere
tanti (n)
libri

ogni
libro può
appartenere
ad una sola
biblioteca

n



ogni biblioteca
può
avere tanti (n)
utenti
ogni
utente può
essere iscritto
a tante (m)
biblioteche

n



La programmazione orientata agli oggetti

I principi fondamentali dell'*object-oriented*:

- **Astrazione**



Un'astrazione rappresenta **le caratteristiche essenziali e distintive di un oggetto**, dal punto di vista di chi lo guarda
[Grady Booch, *Object Oriented Design*, Benjamin/Cummings, 1991 p. 39]

La programmazione orientata agli oggetti

- ***Incapsulamento*** (*information hiding*)



L'incapsulamento (o *information hiding*) è il principio secondo cui la struttura interna, il funzionamento interno, di un oggetto **non deve essere visibile** dall'esterno

La programmazione orientata agli oggetti

⇒ ogni oggetto è costituito da 2 parti:

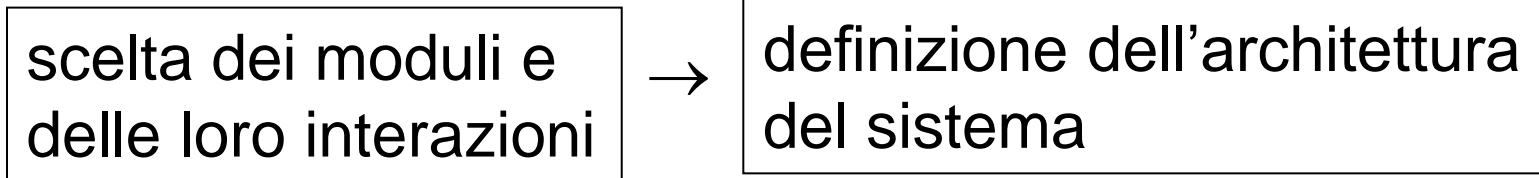
- l'*interfaccia* (vista “esterna”) → visibile
- l'*implementazione* (vista “interna”) → nascosta

L'incapsulamento (o *information hiding*) è il processo che **nasconde** quei dettagli, relativi al funzionamento di un oggetto, che non costituiscono le sue caratteristiche essenziali e distintive [BOOCH, p. 46]

La programmazione orientata agli oggetti

- **Modularità**

La modularità consiste nella suddivisione di un sistema in una serie di **componenti indipendenti**, che interagiscono tra loro per ottenere il risultato desiderato



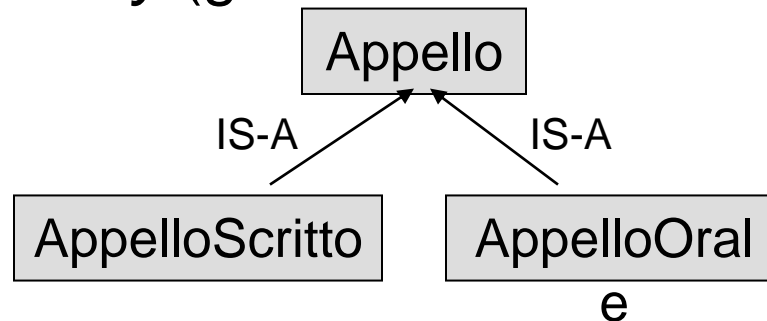
La programmazione orientata agli oggetti

Struttura gerarchica

In un sistema complesso, le due principali **gerarchie** sono:

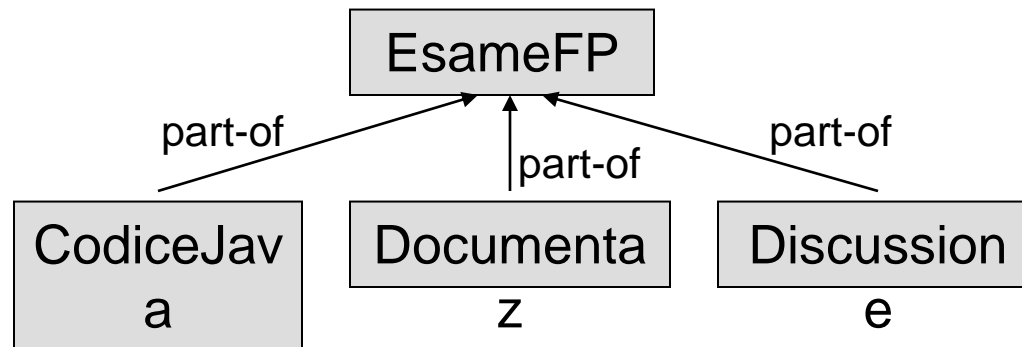
- *kind-of hierarchy* (gerarchia di **classi** e **sotto-classi**)

Per es.



- *part-of hierarchy* (gerarchia di **parti**)

Per es.



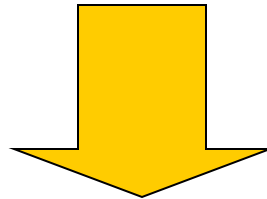
Vantaggi dell' dell'approccio *object-oriented*



- Riuso
- Maggiore leggibilità
- Dimensioni ridotte
- Estensione e modifica più semplici
- Compatibilità
- Portabilità
- Manutenzione del software semplificata
- Migliore gestione del team di lavoro



- Approccio procedurale:
 - occorre conoscere tutto il software
- Approccio ad oggetti:
 - Occorre conoscere l'interfaccia delle classi ma non l'implementazione



- Molto più conveniente in termini di costi



Ringraziamenti

Grazie alla Prof.ssa Annamaria Goy del
Dipartimento di Informatica dell'Università di
Torino per aver redatto la prima versione di queste
slides.