Università degli Studi di Palermo

Corso di Laurea
Ingegneria Gestionale

Lezione 8

La Programmazione Basata su Oggetti

Ing. Massimo Cossentino

Sistemi Informativi Aziendali

a.a. 2008/2009

Concetti fondamentali : gli oggetti

- Ovunque guardiamo vediamo oggetti: persone, animali, piante, automobili, computer, etc.
- Le persone pensano in termini di oggetti (o di categorie di oggetti)
- I programmi per computer (ed i programmi Java) possono essere visti come oggetti, costituiti a loro volta da altri oggetti software interconnessi.
- Gli oggetti (nel mondo reale) si dividono in due categorie: animati ed inanimati.

Concetti fondamentali : gli oggetti

- Gli oggetti animati sono vivi, si muovono, quello inanimati sembrano non far nulla -> tutti hanno qualcosa in comune.
- Possiedono *attributi* (come dimensione, forma, peso, colore) e tutti mostrano un *comportamento* (es. la palla rimbalza, il bambino piange, la radio suona…).
- Al livello software è quindi importante studiare i tipi di attributi e i comportamenti che gli oggetti software mostrano
- Gli uomini imparano a conoscere gli oggetti studiando/ osservando i loro attributi ed il lo comportamento.

Cosa è un oggetto?

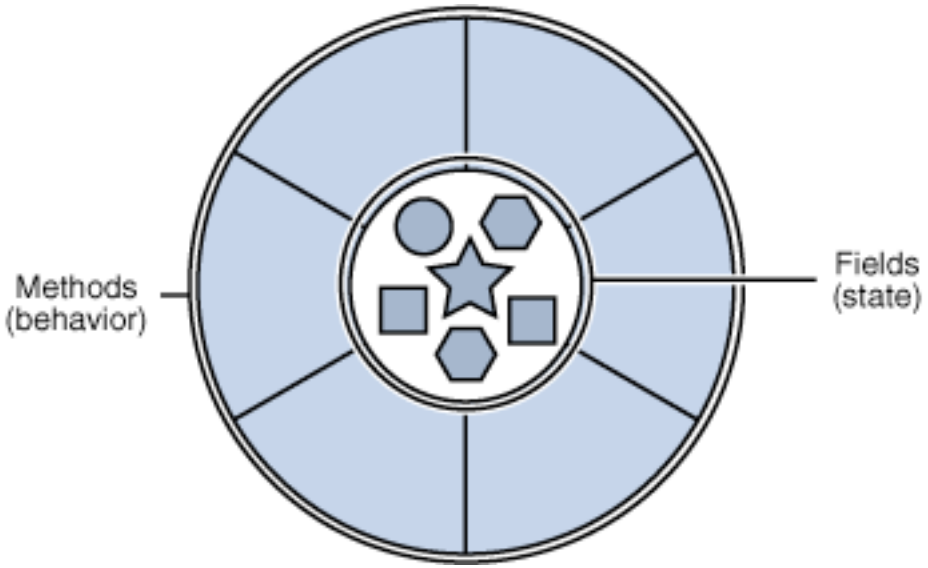
- Gli oggetti del mondo reale condividono due caratteristiche, essi hanno:
  - uno stato
  - Un comportamento
- Un cane ad esempio:
  - Lo stato è identificato dai valori dati a: nome, colore, razza, …
  - Il comportamento potrebbe essere uno dei seguenti: abbaia, scodinzola, odora, …
- Per una bicicletta:
  - Stato, valori dati a: marcia ingranata, cadenza della pedalata, velocità attuale
  - Comportamento: cambia marcia, cambia cadenza di pedalata, frena

Alcune di queste slide sono ispirate dal tutorial sun: http://java.sun.com/docs/books/tutorial/java/index.html

Cosa è un oggetto?

In altre parole in un ciascun oggetto che ci circonda possiamo trovare:

- Degli *attributi* a cui assegnare dei valori (che determinano lo stato)
- Dei comportamenti, descritti da *metodi*.



Cosa è un oggetto?

Gli oggetti software sono concettualmente simili agli oggetti del mondo reale: anch'essi hanno uno stato e un comportamento.

Un oggetto memorizza il suo stato in variabili dette anche attributi (*attribute*) o campi (field)

Il comportamento viene esposto tramite funzioni o metodi (*method*).

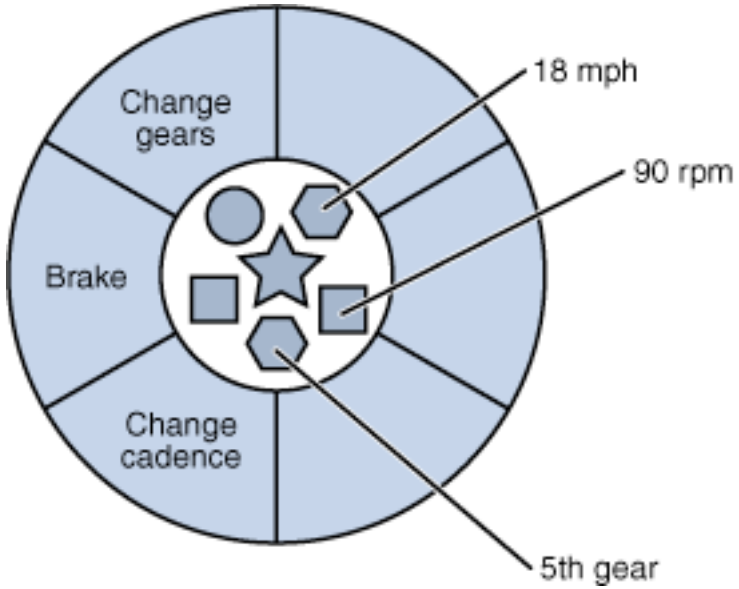
I metodi agiscono sullo stato dell'oggetto a cui appartengono e sono il meccanismo di comunicazione principale tra oggetti (invocazione di metodi)

Cosa è un oggetto?

- Un buono stile di programmazione prescrive l'utilizzo dell'incapsulamento dei dati (*data encapsulation*)
  - Lo stato dell'oggetto è interno (o privato) e può essere manipolato solo tramite un metodo
- Esempio: automobile
  - Attributi (che descrivono lo stato):motore_acceso, velocità, luci_accese
  - Metodi: accendi, accelera, frena, accendiLuci

Cosa è un oggetto?

- Esempio: bicicletta



Cosa è un oggetto?

Perché non mettere tutto il codice in un unico contenitore (oggetto, funzione, unità, modulo, …)?

- Modularità: il codice di un oggetto può essere scritto e manutenuto indipendentemente dal resto del codice
- Information-hiding: limitando l'interazione con i soli metodi di un oggetto i dettagli della sua struttura interna sono nascosti al mondo esterno
- Riuso: se un oggetto esiste già, può essere riusato in altre parti del programma o in un altro programma.
- Componibilità e debug: se un oggetto crea problemi può essere rimosso dalla applicazione e al suo posto se ne può mettere un altro che svolge le stesse funzioni (anche nel mondo reale funziona così: se il motore di un auto si rompe, si cambia il solo motore non l'intera auto)

Cosa è una classe?

- Nel mondo reale si trovano spesso parecchi oggetti dello stesso tipo.
- Per esempio esistono molte biciclette, tutte dello produttore e modello.
  - Ogni bicicletta contiene gli stessi componenti e nasce dallo stesso progetto.
- In termini object-oriented si dice che una certa bicicletta (la mia ad esempio) è una instanza della classe (*instance of class*) di oggetti di nome Bicicletta.
- La classe Bicicletta è il progetto da cui la mia bicicletta è stata creata.

Ecco un esempio di codice java (che verrà studiato in seguito):

```
class Bicycle {
        int cadenza = 0;
        int velocita = 0;
        int marcia = 1;
        void cambiaCadenza (int nuovoValore) {
             cadenza = nuovoValore;
        }
        void cambiaMarcia(int nuovoValore) {
             marcia = nuovoValore;
        }
        void accelera (int incremento) {
             velocita = velocita + incremento;
        }
        void applicaFreni(int decremento) {
             velocita = velocita - decremento;
        }
        void stampaStato() {
             System.out.println("cadenza:"+cadenza+" velocita:"+velocita+"
               marcia:"+marcia);
        }
}
```

Concetti fondamentali : gli oggetti

- Oggetti diversi possono avere attributi simili e mostrare comportamenti simili.
- La progettazione orientata agli oggetti modella il software in termini simili a quelli che gli uomini usano per descrivere gli oggetti del mondo reale.
  - Viene sfruttato il concetto di classe -> gli oggetti di una stessa classe hanno le stesse caratteristiche.
  - Il concetto di ereditarietà (anche multipla) con cui è possibile derivare nuove classi "assorbendo" le caratteristiche di classi già esistenti ed aggiungendone di nuove
    - Es: un oggetto della classe 'spider' ha le caratteristiche della classe più generale 'automobile' ma in più ha il tettuccio apribile

Concetti fondamentali : gli oggetti

- La progettazione orientata agli oggetti modella i componenti software proprio come vengono descritti gli oggetti del mondo reale, usando i loro attributi ed il loro comportamento.
- Gli oggetti nel mondo reale interagiscono tra di loro comunicando -> La OOP modella anche la comunicazione tra gli oggetti tramite messaggi.
- La OOP incapsula gli attributi e le funzionalità degli oggetti -> gli oggetti hanno la proprietà di nascondere le informazioni:
  - sebbene essi possano comunicare tra loro attraverso specifiche interfacce, non possono sapere come altri oggetti sono implementati.
  - I dettagli dell'implementazione sono nascosti all'interno degli oggetti stessi.

Concetti fondamentali : gli oggetti

- Nascondere le informazioni (Information Hiding) è cruciale nell'ambito della progettazione software.

- Nella programmazione procedurale (linguaggi tipo C, Pascal) la programmazione tende ad essere orientata all'azione-> l'unità di programmazione è la funzione , gruppi di azioni che svolgono qualche compito comune vengono trasformate in funzioni e queste ultime e loro volta raggruppate a formare programmi.
- In Java l'unità di programmazione è la classe da cui gli oggetti vengono istanziati (creati); le classi Java contengono metodi (che implementano delle funzionalità) e campi (che implementano attributi).

Concetti fondamentali : gli oggetti

- Creare un programma in Java significa creare delle classi, ogni classe contiene campi ed un insieme di metodi per manipolare questi campi per fornire determinati servizi ai clienti.
- Le classi vengono usate come mattoni per costruire nuove classi.
- Le classi hanno anche relazioni con altre classi chiamate associazioni.
- Un software ad oggetti viene scritto come aggregati di classi, ogni 'aggregato' può essere riusato per creare altri programmi.
- Parola d'ordine RIUSO.

Un editor Java: BlueJ

- Introduzione all'uso di BlueJ
  - Interfaccia del programma
  - Compilazione
  - Esecuzione dei programmi
  - Un esempio: la classe Bicycle già vista prima
    - Uso del comando new per instanziare un oggetto
    - Comando Inspect per verificare lo stato dell'oggetto
    - Invocazione dei metodi della classe per alterarne lo stato

Università degli Studi di Palermo

I primi comandi Java

Un semplice Programma in Java

```
// programma che scrive una riga di testo
public class Welcome {

  public static void main(String args[])
  {
    System.out.println(”Welcome to java programming");
  } //fine del metodo main
} //fine della classe Welcome
```

Un semplice Programma in Java

- I commenti:

  // commento a riga singola

  /* commento
  a riga multipla, può essere distribuito su più righe*/

  /** commento Javadoc */
  javadoc è una utility che legge i commenti in Javadoc e li utilizza per preparare automaticamente la documentazione di un programma in HTML.

  Dimenticare di chiudere i commenti può generare errori di sintassi: violazioni alle regole del linguaggio. (vengono rilevati in fase di compilazione)

Un semplice Programma in Java

```
// programma che scrive una riga di testo
public class Welcome {

  public static void main(String args[])
  {
    System.out.println(”Welcome to java programming");
  } //fine del metodo main
} //fine della classe Welcome
```

Un semplice Programma in Java

public class Welcome {

- Rappresenta l'inizio di una dichiarazione di classe per la classe Welcome.
- Ogni programma Java consiste di almeno una dichiarazione di classe definita dal programmatore.
- La parola chiave class introduce una dichiarazione di classe ed è immediatamente seguita dal nome della classe.
- Per convenzione i nomi delle classi iniziano per lettera maiuscola (Java è sensibile alle maiuscole e minuscole)

Un semplice Programma in Java

```
public static void main(String args[])
 {
```

- è il punto di partenza di ogni applicazione Java.
- il metodo main fa partire l'esecuzione di un'applicazione Java.
- Le parentesi dopo main indicano che main è blocco di codice del programma detto metodo.
- Le dichiarazioni di classe Java contengono uno o più metodi, in un programma uno di questi metodi deve essere chiamato main e deve essere definito come mostrato (conseguenza la JVM non è in grado di eseguire l'applicazione)
- I metodi sono in grado di eseguire delle attività e ritornare del informazioni -> la parola void indica che questo metodo non ritorna nessuna informazione

Un semplice Programma in Java

System.out.println(”Welcome to java programming");

- Comunica al computer che deve eseguire un’azione , ovvero visualizzare una stringa di caratteri contenuti tra doppie virgolette.
- Vengono chiamate stringhe di caratteri o messaggi di testo e gli spazi non vengono ignorati.
- System.out è un oggetto speciale chiamato output standard che permette alle applicazioni Java di visualizzare sequenze di caratteri nella finestra di comando.
- Il metodo System.out.println visualizza una riga di testo nella finestra di comando ed il testo tra virgolette è l’argomento del metodo.

Un semplice Programma in Java

```
// programma che scrive una riga di testo
public class Welcome {

  public static void main(String args[])
  {
    System.out.println(”Welcome to java programming");
  } //fine del metodo main
} //fine della classe Welcome
```

Errori Tipici

- Dimenticare di chiudere una parentesi -> errore di sintassi.
- Dimenticare un punto e virgola –> errore di sintassi.
- Quando il compilatore segnala errori di sintassi, l'errore potrebbe non trovarsi nella riga indicata ma in quelle sopra.
- *bad command or filename* oppure *javac:command not found* oppure *javac is not recognized…* -> non avete installato correttamente la JVM oppure non avete settato correttamente il Path.
- *public ClassName must be defined in a file called ClassName* -> non c'è corrispndenza tra il nome della classe che volete compilare e quello del file.
- *Exception in thread main java.lang.NoClassDefFoundError* -> la variabile di ambiente potrebbe non essere stata settata correttamente

Tipi e Variabili

- Java (come altri programmi) non manipola solo stringhe e numeri ma anche dati più complessi (es.conti bancari, informazioni impiegati etc.)
- Java è perfettamente progettato per gestire questi tipi di dati (o oggetti) complessi.
- In Java si definiscono le classi per definire il comportamento di questi oggetti.

Tipi e Variabili

- In Java ogni valore è di un determinato tipo.
- Il tipo indica cosa si può fare con i valori (p.es. con oggetti di tipo System.out si può invocare il metodo println oppure con due numeri di tipo intero di può fare la somma).
- Spesso si vogliono memorizzare i valori per utilizzarli in seguito -> la variabile è una porzione della memoria del computer dotata di un tipo, un nome ed un contenuto.

Tipi e Variabili

- Sintassi Java per la **definizione di Variabile**:

  nomeTipo nomeVaribile = valore;

  nomeTipo nomeVariabile;

  Esempio:

  String greeting = “Welcome in Java”;

  Obiettivo:

  definirn una nuova variabile di tipo nomeTipo e fornire eventualmente un valore iniziale

Tipi e Variabili

- Le variabili possono essere utilizzate al posto degli oggetti che memorizzano.

  per esempio:

  ```
  String greeting = “Hello, World”;
  PrintStream printer = System.out;

  printer.println(greeting) ;
  ```

  al posto di

  ```
  System.out.println(“Hello, World”);
  ```

Tipi e Variabili

- Quando si dichiara una variabile si deve decidere di che tipo è e che nome dargli.
- Il tipo dipende dall'utilizzo che se ne deve fare.
- Il nome di una variabile di un metodo o di una classe si chiama identificatore.
- Gli identificatori di variabili,metodi e classi sono composti di lettere, cifre, caratteri dollaro e segni di sottolineatura ma non possono cominciare con una cifra.
- Non si possono usare altri simboli come per esempio ! ? %.
- gli spazi non sono ammessi all'interno degli identificatori.
- Le parole riservate non si possono usare.
- anche gli identificatori sono sensibili alle maiuscole e minuscole.

Tipi e Variabili

- Per convenzione i nomi delle variabili dovrebbero iniziare con lettera minuscola.
- Si possono usare alcune lettere maiuscole all'interno, p.es.:

```
int luckyNumber = 13;
```

- Per modificare il valore di una variabile si usa l'operatore si assegnazione (=).

```
luckyNumber = 12;
```

- L'assegnazione provoca la sostituzione (in memoria) del valore originale della variabile.

Tipi e Variabili

- Sintassi di assegnazione:

  nomeVariabile = valore;

esempio:

  luckyNumber =13;

obiettivo:

  assegnare un valore ad una variabile definita in precedenza.

Tipi e Variabili

- L'operatore (=) indica un'azione atta a sostituire il nome memorizzato in una variabile -> non è sinonimo di uguaglianza.
- E' errato usare una variabile a cui non è mai stato assegnato un valore, es:

```
int luckyNumber;
System.out.println(luckyNumber);
```

Oggetti, classi e metodi

- Gli oggetti sono entità di un programma che si possono manipolare invocando metodi.
- Un metodo è una sequenza di istruzioni che accede ai dati di un oggetto.
- Una classe specifica i metodi che possono essere applicati ai suoi dati.
- Esempio:
  - il metodo System.out appartiene alla classe PrintStream che ha due metodi println e print.
  - l'oggetto "Hello, World" appartiene alla classe String che mette a disposizione diversi metodi da applicare agli oggetti di tipo String .
  - il metodo length conta il numero di caratteri

```
String greeting ="Hello, World";
int n = greeting.length();
```

Oggetti, classi e metodi

- I metodi di una classe costituiscono la sua interfaccia pubblica.
- L'interfaccia pubblica di una classe specifica cosa si può fare con i suoi oggetti mentre l'implementazione nascosta specifica come si svolgono le azioni -> implementazione o realizzazione privata descrive i dati interni agli oggetti e le istruzioni per le esecuzioni dei propri metodi.
- La realizzazione privata non è nota ai programmatori che non conoscono "l'interno" di un oggetto.

Oggetti, classi e metodi

- Alcuni metodi hanno bisogno di valori di ingresso che specificano il tipo di elaborazione che deve essere svolta.

```
System.out.println("Hello, World");
System.out.println(greeting);
```

- I dati forniti in ingresso ad un metodo si chiamano parametri.

- Ci sono parametri espliciti e parametri impliciti
  - il parametro implicito è l'oggetto con cui viene invocato il metodo stesso.
- Alcuni metodi richiedo un parametro esplicito altri no, inoltre alcuni restituiscono un dato in uscita altri no.
  - il valore restituito da un metodo è il risultato che il metodo ha calcolato perché venga utilizzato nel codice che ha invocato il metodo stesso.

Es. : `int n = greeting.length();`

Oggetti, classi e metodi

- Il valore restituito da un metodo può anche essere utilizzato direttamente come parametro di un altro metodo.

  Es. :

  ```
  System.out.println(greeting.length());
  ```

- Non tutti i metodi restituiscono un valore, per es. il metodo printl() interagisce con il sistema operativo per la stampa a video della stringa immessa.

Università degli Studi di Palermo

Analisi e progettazione OO

Analisi e progettazione OO

- Quando i programmi (o analogamente i problemi che essi risolvono) sono molto grandi e complessi non basta sedersi davanti ad un computer e programmare.
- Per creare una buona soluzione si deve seguire un procedimento dettagliato per ottenere una analisi dei requisiti del progetto (i.e. stabilire cosa il sistema deve fare) e per sviluppare un progetto che soddisfi tali requisiti (i.e. come costruire il sistema).
- Un procedimento che comprende analizzare e progettare il sistema da un punto di vista orientato agli oggetti si chiama *Procedimento di analisi e progettazione orientato agli oggetti.*
- Ogni gruppo dovrebbe stabilire un procedimento comune ed un modo uniforme (linguaggio) di comunicare i risultati del procedimento all'interno del gruppo.
- Esistono molti procedimenti di analisi e progettazione orientati agli oggetti -> un linguaggio grafico molto utilizzato: UML

Analisi e progettazione OO

- UML: Unified Modelling Language, sviluppato nelle prima metà degli anni '90 è attualmente lo schema di rappresentazione grafica per modellare i sistemi orientati agli oggetti ed ha unificato diversi tipi di notazioni comunemente usate.

Un modello è una semplificazione della realtà

I ***modelli***

- ci aiutano a "visualizzare" un sistema *come è* o *come vorremmo che fosse*
- ci permettono di specificare la struttura o il comportamento di un sistema
- ci forniscono un "template" che ci guida nella costruzione di un sistema
- documentano le decisioni che abbiamo preso

Analisi e progettazione OO

- Divide et impera: tramite i modelli ci focalizziamo su un solo aspetto alla volta
- ogni modello può essere espresso a differenti livelli di precisione
- per un sistema non banale:
  - non un solo modello
  - ma un piccolo insieme di modelli, che possono essere costruiti e studiati separatamente, ma che sono strettamente interrelati
- UML è un linguaggio (e notazione) universale, per la creazione di modelli software

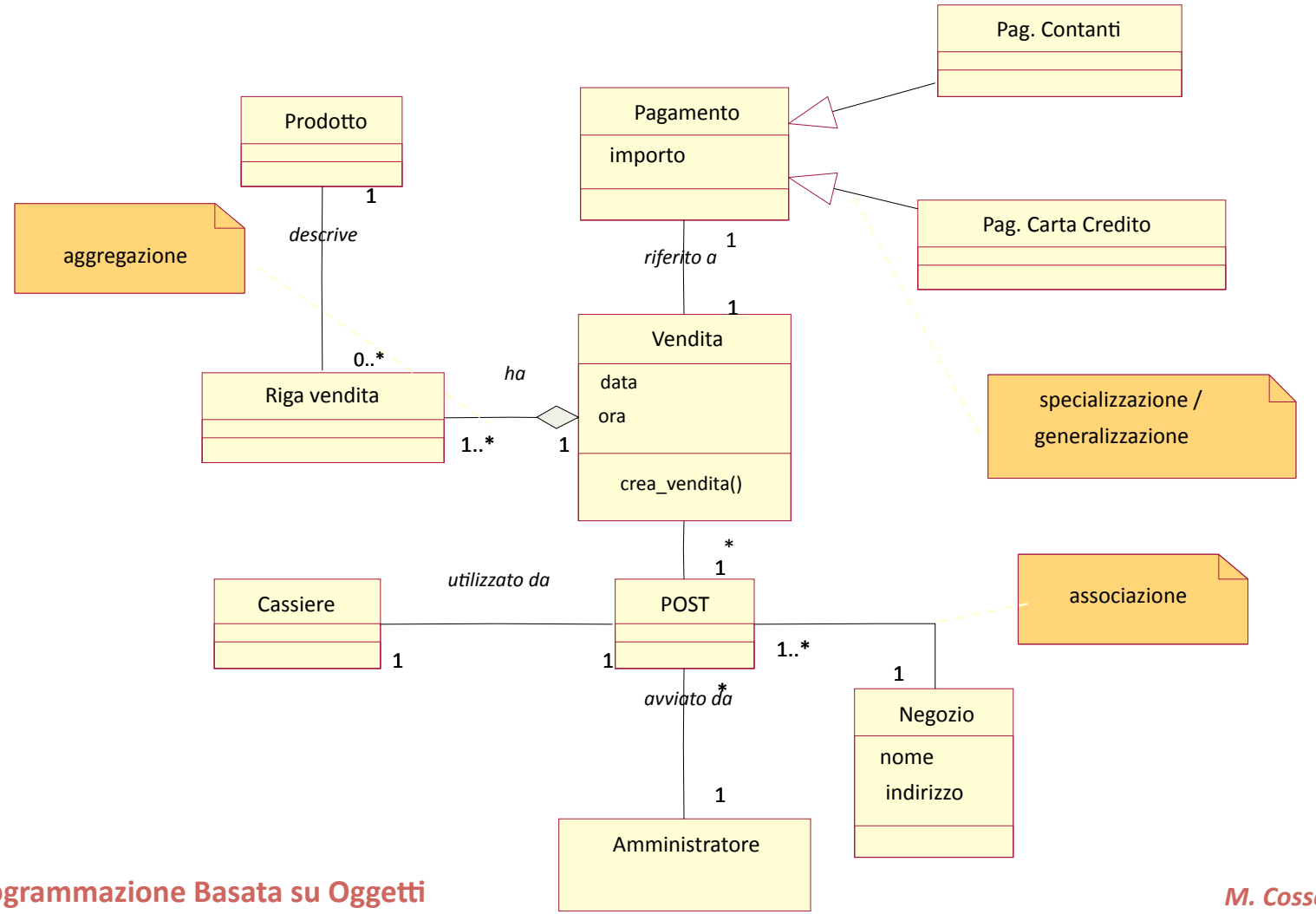
Analisi e progettazione OO

- UML prevede una serie di modelli diagrammatici
  - alcuni elementi (ad es. la "classe") hanno una icona che li rappresenta graficamente
- **livello "logico":**
- dei casi d'uso - Use Case Diagram
- delle classi - Class Diagram
- di sequenza - Sequence Diagram
- di collaborazione - Collaboration Diagram
- di transizione di stato - Statechart Diagram
- delle attività - Activity Diagram
- **livello "fisico":**
- dei componenti - Component Diagram
- di distribuzione dei componenti - Deployment Diagram

Analisi e progettazione OO

- è il caposaldo dell'object oriented
- rappresenta le classi di oggetti del sistema con i loro attributi e operazioni
- mostra le relazioni tra le classi (associazioni, aggregazioni e gerarchie di specializzazione/ generalizzazione)
- può essere utilizzato a diversi livelli di dettaglio (in analisi e in disegno)

Analisi e progettazione OO



Dal problema al codice

- Il processo di sviluppo di un software è costituito dalle fasi di analisi, progettazione e realizzazione.
- Nella fase di analisi si vuole ottenere una completa descrizione di ciò che il prodotto (software) dovrebbe fare.
- Nella fase di progettazione si identificano le classi, le loro responsabilità e le relazioni reciproce.
- Nella fase di realizzazione viene prodotto il programma che viene poi installato e collaudato.

Dal problema al codice

- La fase di analisi trasforma una vaga comprensione del problema in una descrizione precisa dei compiti che devono essere portati a termini dal software.
- Il risultato è una descrizione testuale accurata detta specifica funzionale con le seguenti caratteristiche:
  - definisce completamente i compiti che devono essere eseguiti
  - è priva di contraddizioni interne
  - è comprensibile sia per esperti del problema che per sviluppatori di software
  - può essere revisionata da più persone
  - può essere messa a confronto con la realtà dei fatti

Dal problema al codice

- La fase di progettazione ha come risultato l'organizzazione dei singoli compiti di programmazione in una serie di classi correlate.
- Obiettivi principali:
  - identificare le classi
  - identificare le responsabilità di tali classi
  - identificare le relazioni tra le classi
- Risultati:
  - descrizione testuale delle classi e delle loro principali responsabilità
  - diagrammi di relazioni tra classi
  - diagrammi dei principali scenari di utilizzo
  - diagrammi di stato degli oggetti il cui comportamento sia dipendente da uno stato.

Dal problema al codice

- Identificare le classi:
  - dalla descrizione del problema cercare i sostantivi
- per identificare le responsabilità si prendono in considerazione i verbi nella descrizione del problema.
- una responsabilità deve appartenere ad una sola classe.
- Le relazioni tra le classi:
  - Dipendenza (“usa”)
  - Aggregazione (“ha”)
  - Ereditarietà (“è”)