LEZIONE 3 - OBJECT DIAGRAM

Catia Trubiani

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica (DISIM) - Università degli Studi dell'Aquila

catia.trubiani@univaq.it

Alcune info sulle prossime lezioni

Mercoledì 12 Marzo non c'è lezione dalle 11:00 alle 13:00, ci sarà il parziale dalle 14:00 alle 16:00. Il parziale riguarderà class, package, e object diagrams.

Lunedì 31 Marzo (16:00 – 18:00) ci sarà lezione invece di venerdì 4 Aprile (11:00 – 13:00)

Lunedì 7 Aprile (16:00 – 18:00) ci sarà lezione invece di mercoledì 9 Aprile (14:00 – 16:00)

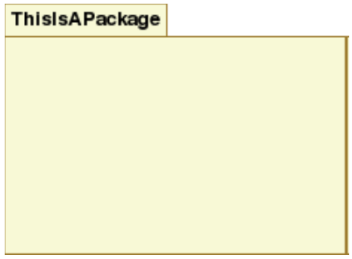
Lunedì 12 Maggio (16:00 – 18:00) ci sarà lezione invece di venerdì 16 Maggio (11:00 – 13:00)

Riepilogo della lezione precedente

Cos'è un **Package**? A cosa serve?

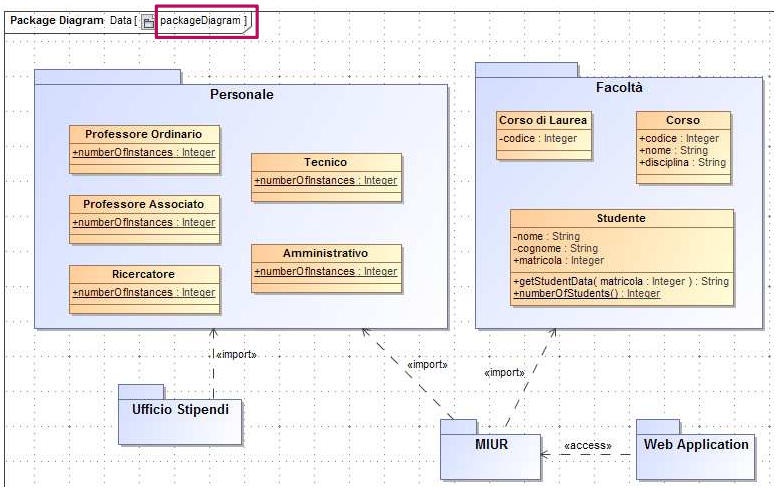
I package sono costrutti che permettono di raggruppare/organizzare entità da modellare

Rendono i diagrammi più semplici facilitandone la comprensione



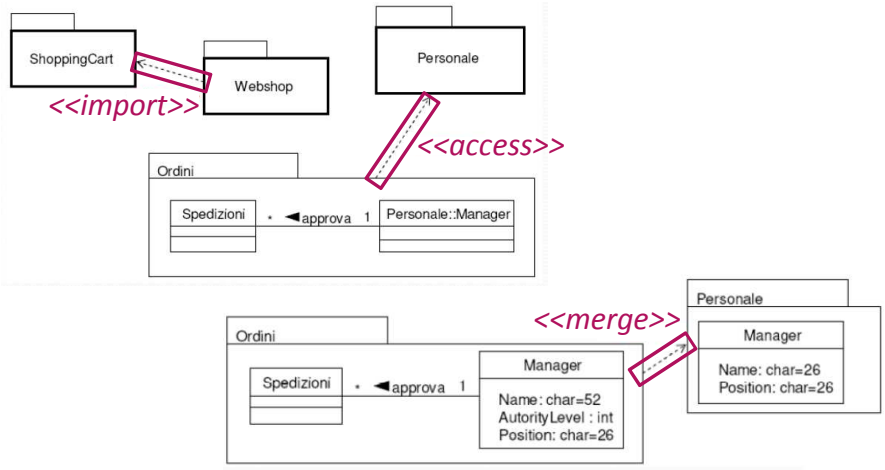
Riepilogo della lezione precedente

Cos'è un **Package Diagram**? A cosa serve?



Concetti principali

Quali sono le relazioni tra i packages? Qual'è il loro significato?



Object Diagram

L'Object Diagram è una "variante" del Class Diagram ed utilizza una notazione quasi identica

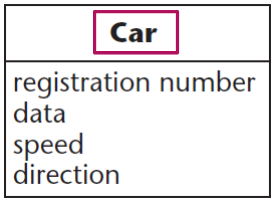
La differenza principale tra Object e Class Diagram è che il primo mette in relazione ISTANZE mentre il secondo mette in relazione TIPI di dato

Gli oggetti sono identificati con il loro «nome» (e non il tipo) e le relazioni sono relazioni di istanza

Qual'è la differenza tra classe e istanza??

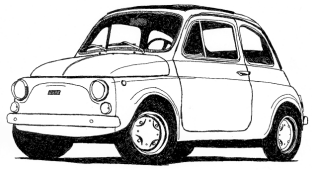
Il *nome* della CLASSE si riferisce al tipo di dato, il nome delle ISTANZE si riferisce agli oggetti reali

nome = 'Car'



la classe rappresenta un'entità astratta

nome = 'Cinquecento'



nome = 'Ferrari'

l'istanza rappresenta un'entità concreta

Object Diagram

Un oggetto si relaziona ad una classe allo stesso modo in cui un dato si relaziona ad un tipo di dato (nei linguaggi di programmazione)

Una classe può essere vista come un tipo di variabili e un oggetto (istanza) come una delle variabili di quel tipo. Creando una classe si definisce un nuovo tipo di dati, creando un oggetto si definisce una nuova variabile (di una tipologia già definita)

Object Diagram

L'Object Diagram è quindi "un'istanziazione" di un Class Diagram e mostra il sistema in esecuzione

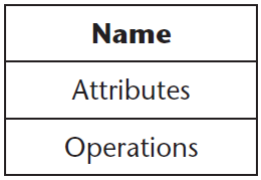
L'Object Diagram può essere utilizzato per semplificare un Class Diagram complesso mostrando le istanze effettive e le loro relazioni. Gli oggetti possono essere utilizzati anche nei diagrammi dinamici e mostrano la collaborazione tra un insieme di oggetti.

Classe vs Oggetto

Una **classe** modella una famiglia di entità del dominio di applicazione

- le proprietà (attributi)
- il comportamento (operazioni)

Raggruppa un insieme coeso di entità



Un **oggetto** si relaziona ad una classe allo stesso modo in cui un dato si relaziona ad un tipo (nei linguaggi di programmazione)

Object Oriented (O.O.)

La programmazione ad oggetti prevede di raggruppare in una zona circoscritta del codice sorgente (chiamata **classe**) la dichiarazione delle strutture dati e delle procedure che operano su di esse.

Le classi, quindi, costituiscono dei modelli astratti, che a tempo di esecuzione vengono invocate per istanziare o creare **oggetti** software relativi alla classe invocata.

Questi ultimi sono dotati di attributi (dati) e metodi (procedure) secondo quanto definito/dichiarato nelle rispettive classi.

Object Oriented (O.O.)

I principali concetti fondamentali del mondo O.O. sono:

- classe
- oggetto

- operazione
- metodo
- messaggio

- incapsulamento
- ereditarietà
- polimorfismo

Oggetto

Rappresenta un elemento del “**mondo reale**” che si sta modellando

Ha associato un tipo (definito dalla classe)

Può modellare un entità fisica, concettuale o software

È una manifestazione **concreta** di un’astrazione

Ha un’identità ben definita ed incapsula uno stato e un comportamento

Oggetto

Un **oggetto** è una istanza di una classe. È dotato di tutti gli attributi e i metodi definiti dalla classe, ed agisce come un fornitore di "messaggi" (i metodi) che possono essere attivati su richiesta da entità esterne (procedure o altri oggetti).

Inviare un messaggio ad un oggetto vuol dire invocare un metodo su quell'oggetto.

Classi ed oggetti in Java

```
public class Conto {
private static int numContiCreati;
private double saldo;
…
public Conto(double saldoIniziale, double tassoIniziale) {
        saldo = saldoIniziale;
        tasso = tasso;
        numContiCreati++;
}
…
public void deposita (double importo){
        if (importo < 0)
                System.out.println("errore: valore < 0 ");
        else
                saldo = saldo + importo;
        }
...
}
```

```
public class ProvaConto{
        public static void main(String[] args) {
                Conto contoPaperone = new Conto(100, 6.5);
                contoPaperone.deposita(240.61);
        }
}
```

oggetto *classe*

invocare un metodo su un oggetto

Cosa vuol dire istanziare un oggetto?!

Istanziare un oggetto vuol dire allocare memoria ed eventualmente inizializzarla secondo le specifiche definite dalla classe

Un oggetto viene identificato da una certa zona di memoria nella quale sono memorizzati gli attributi. Il valore di questi ultimi determina lo stato interno dell'oggetto.

Operazione, metodo, messaggio

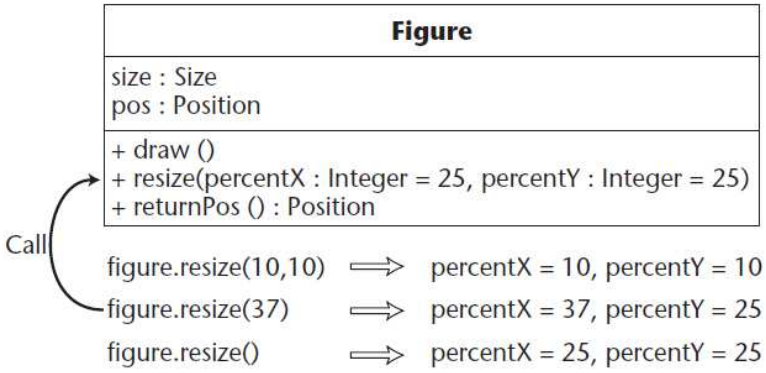
Operazione: specifica di una “interfaccia” (prototipo) che un oggetto mette a disposizione di altri oggetti

Metodo: implementazione vera e propria dell'operazione di un oggetto

Messaggio: è la richiesta di un metodo appartenente ad un oggetto A verso un metodo di un oggetto B

Definizione di parametri nelle operazioni

Cosa vuol dire invocare un metodo?! Come vengono usati i parametri di input dall’*esterno*?!



Incapsulamento

L'*incapsulamento* è la proprietà per cui i dati che definiscono lo stato interno di un oggetto sono accessibili ai metodi dell'oggetto stesso, mentre non sono visibili dall'esterno.

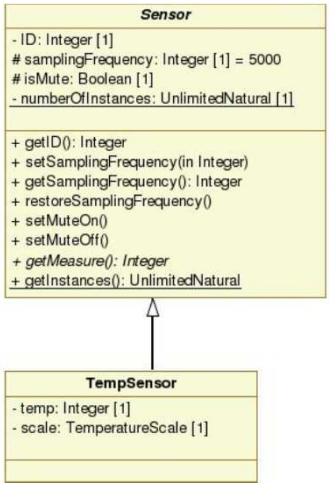
Per alterare lo stato interno dell'oggetto, è necessario invocarne i metodi, ed è questo lo scopo principale dell'incapsulamento.

L'incapsulamento permette quindi di vedere l'oggetto come un black-box di cui, attraverso l'interfaccia, è noto cosa fa ma non come lo fa.

Ereditarietà

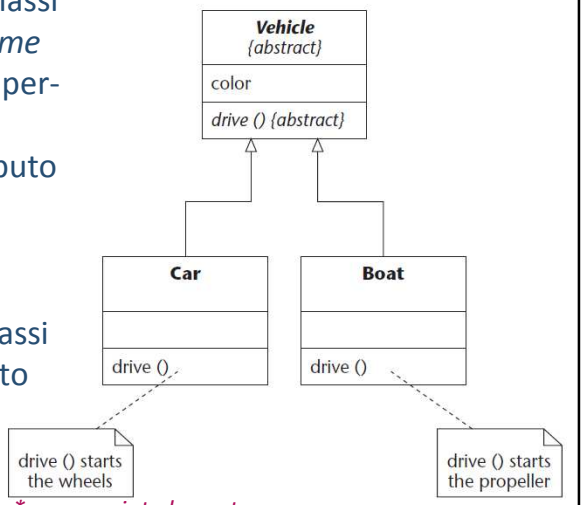
Il meccanismo dell'ereditarietà è utilizzato per permette di derivare nuove classi a partire da quelle già definite realizzando una gerarchia di classi

Una classe derivata attraverso l'ereditarietà (sotto-classe o classe figlia), mantiene i metodi e gli attributi delle classi da cui deriva (super-classi o classi padre)



Polimorfismo

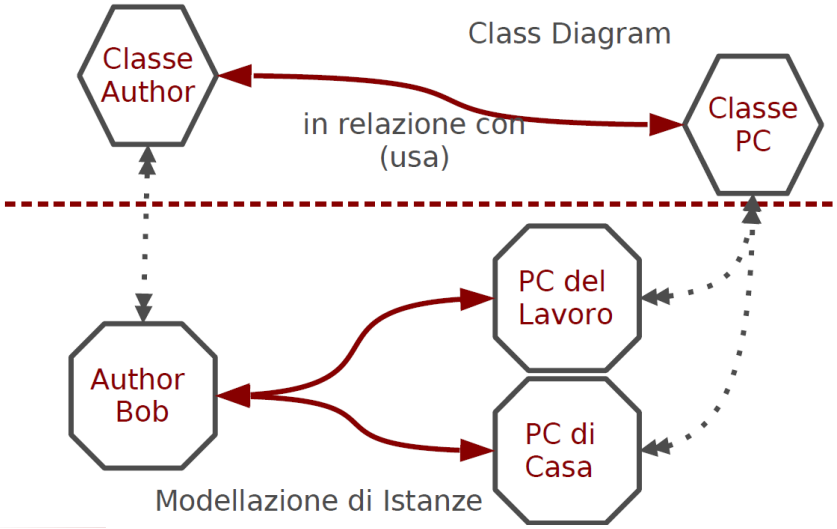
Le istanze delle sotto-classi possono assumere '*forme diverse*' rispetto alla super-classe. Le classi 'Car' e 'Boat' ereditano l'attributo 'color' e l'operazione 'drive'. Quest'ultima operazione viene però ridefinita nelle sotto-classi quindi il comportamento sarà diverso in base all'oggetto che invoca quel metodo.



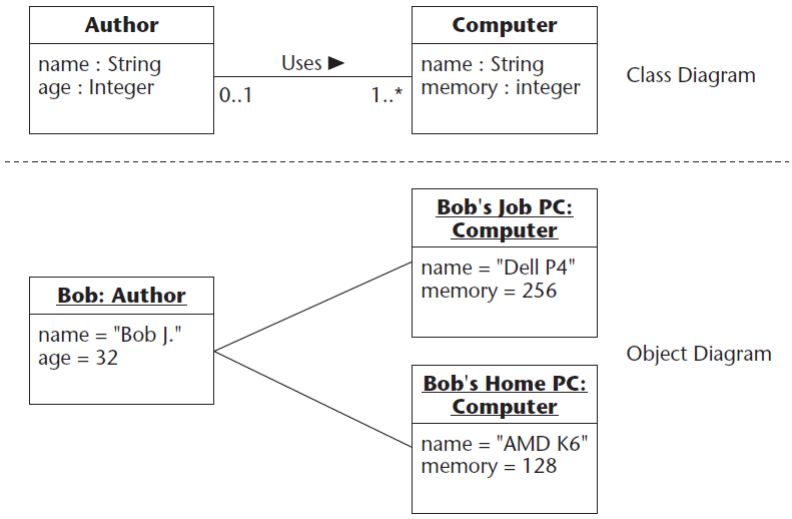
**sono avviate le ruote della macchina*

**è avviato il propulsore della barca*

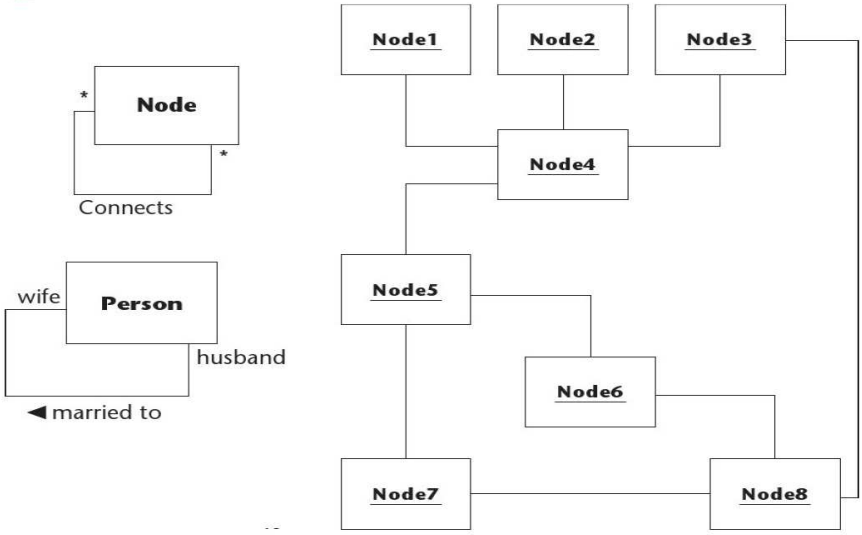
Classi vs Oggetti



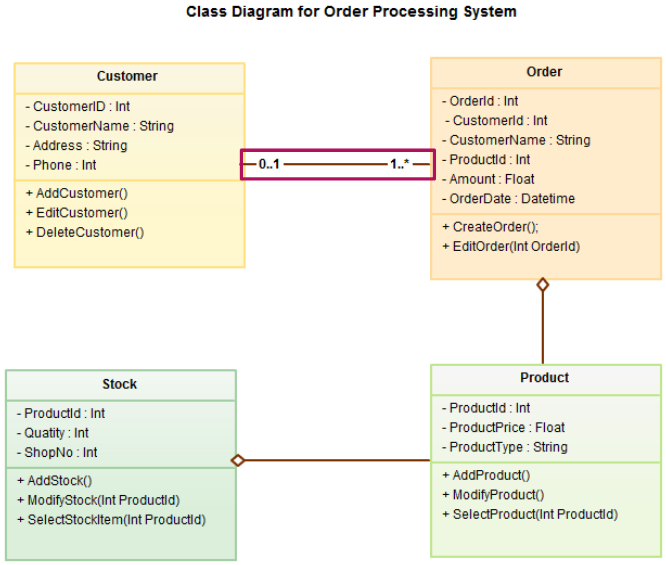
Class Diagram vs Object Diagram



Relazione di associazione tra oggetti

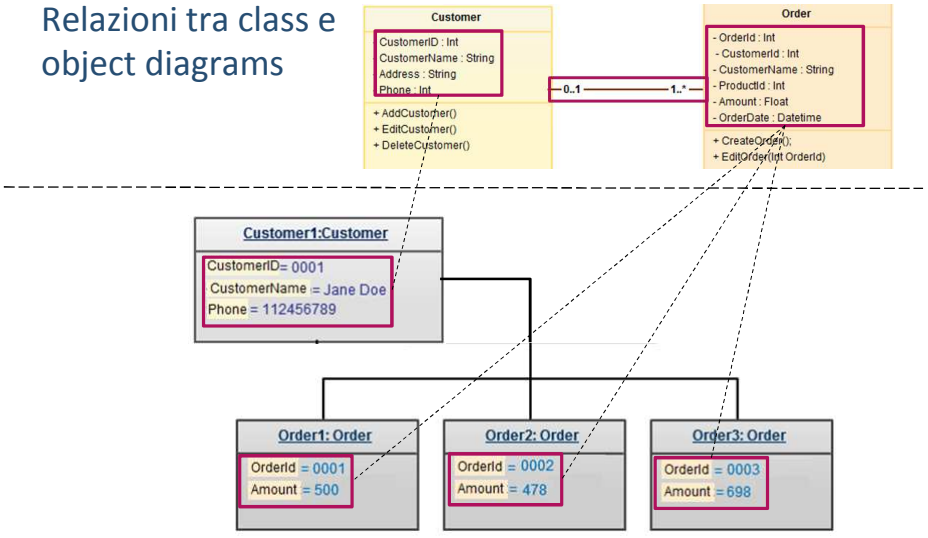


Un esempio di Class Diagram…



…e un possibile Object Diagram

Relazioni tra class e object diagrams



Object Diagram… proviamoci!

Si vogliono modellare:

Tre studenti di cui si conoscono nome, cognome, numero di matricola

studente1- nome: Zlatan, cognome: Ibrahimovic, matricola: 123456

studente2- nome: Michael, cognome: Schumacher, matricola: 123457

studente3- nome: Lilli, cognome: Gruber, matricola: 123458

Due corsi di laurea di cui si conoscono il codice, il nome

corsoDiLaurea1- cod: 12, nome: Scienze motorie

corsoDiLaurea2- cod: 13, nome: Scienze della comunicazione

Tre corsi di cui si conoscono il codice, il nome, e la disciplina a cui appartengono

corso1- cod: 121, nome: Calcio, disciplina: attività motoria

corso2- cod: 122, nome: Formula1, disciplina: attività automobilistica

corso3- cod: 131, nome: Storia Contemporanea, disciplina: attualità

Due professori di cui si conoscono nome, cognome, ruolo, e stipendio

prof1- nome: Fabio, cognome: Capello, prof. associato, stipendio: 2000

prof2- nome: Giorgio, cognome: Napolitano, prof. ordinario, stipendio: 2500

Object Diagram… proviamoci!

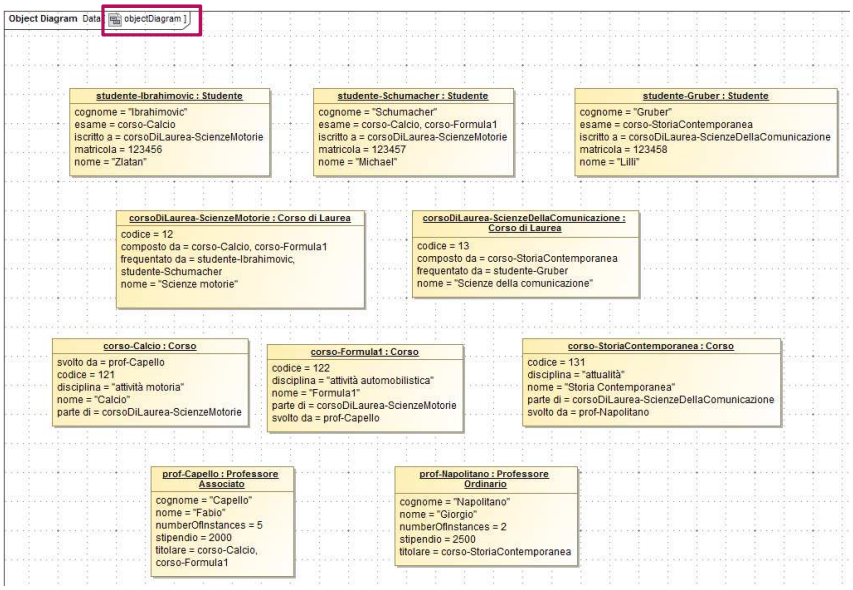
Altre informazioni:

Ibrahimovic e Schumacher sono iscritti al corso di laurea di scienze motorie, Gruber è iscritta al corso di scienze della comunicazione

Ibrahimovic ha sostenuto l'esame di calcio, Schumacher ha sostenuto gli esami di calcio e formula1, Gruber ha sostenuto l'esame di storia contemporanea

Fabio Capello è titolare del corso di calcio e formula1, Giorgio Napolitano è titolare del corso di storia contemporanea

Object Diagram



Questions?

catia.trubiani@univaq.it