Laboratorio di Algoritmi e Strutture Dati

Nicola Di Mauro
– ndm@di.uniba.it –

Dipartimento di Informatica
Università degli Studi di Bari "Aldo Moro"

Introduzione OO

http://www.di.uniba.it/~ndm/corsi/lasd/

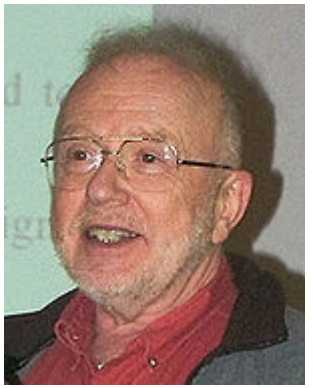
Object Oriented Programming

Tag cloud from wikipedia OO programming

Programmazione Object Oriented

- Nella programmazione imperativa è possibile accedere alle variabili globali da qualsiasi parte del programma
  - questa disciplina nei programmi molto grandi diventa ingestibile
    - i moduli non possono essere sviluppati indipendentemente
      David Parnas, "On the Criteria to Be Used Decomposing Systems Into Modules", Communications of the ACM, **1972**

      information hiding



  - Incapsulare ogni variabile globale in un modulo insieme ad un gruppo di operatori che possono accedere a tali variabili
  - Gli altri moduli possono accedere indirettamente alle variabili solo per mezzo di questi operatori
- La programmazione Object-Oriented è una disciplina che si basa su oggetti per imporre una struttura modulare sui programmi

Oggetti e classi

- Un oggetto è definito come un gruppo di procedure che condividono uno stato
  - Peter Wegner, Concepts and paradigms of object-oriented programming. OOPS Messenger 1(1): 7-87, 1990
- Definiremo un oggetto come
  - una collezione di dati (stato dell'oggetto), e
  - procedure in grado di alterare lo stato (comportamento)
- Oggetto
  - un robot il cui stato è rappresentato dalla sua posizione in una stanza, l'angolazione degli arti, ...
  - un oggetto robot deve avere un nome, che lo distingue dagli atri robot
- Classe
  - la collezione di tutti i robot è detta classe
  - Definiremo classe una collezione di oggetti che condividono gli stessi attributi, dove un attributo è il tipo di un dato o di un metodo che manipolano i dati.

Costruttore e distruttore

- Agli oggetti sono associati operazioni e valori
- Exempio
  - Coda è la classe di tutti gli oggetti coda
  - q è un oggetto della classe Coda
  - opearazioni su q: creaCoda, aggiungi(q,i), primo(q), rimuovi(q) e CodaVuota(q).
  - Lo stato di default di q dovrebbe essere quello di coda vuota. Altrimenti lo stato di q includerà tutti gli elementi aggiunti a q
  - Nei linguaggi orientati ad oggetti, creaCoda è detto costruttore e crea l'oggetto q
  - Anche il distruttore, che distrugge un oggetto, fa parte delle operazioni applicabili ad un oggetto

Linguaggi Object Oriented

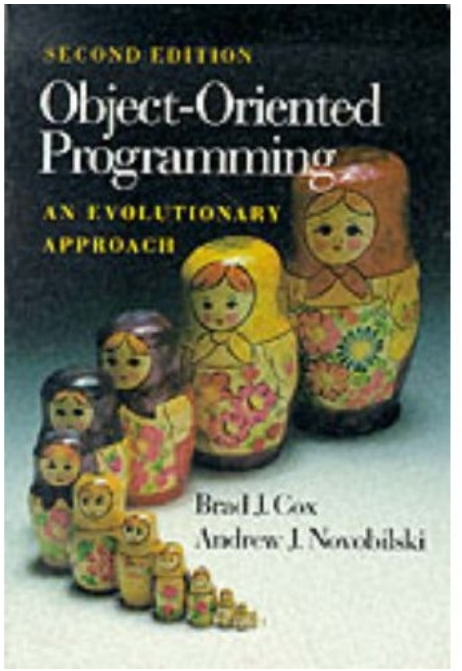
- Un linguaggio object-based supporta
  - Information hiding (encapsulation)
  - Data abstraction
  - Message passing (polymorphism)
- Un linguaggio object-oriented implementa anche
  - l'ereditarietà, incluso il dynamic binding
    - organizzazione di oggetti in una gerarchia di classi
      - un oggetto può ereditare le proprietà della sua classe genitore
    - caratteristica distintiva degli approcci object-oriented

Programmare con gli oggetti

- Un oggetto è una entità dinamica
  - potrebbe cambiare, ma resta sempre lo stesso oggetto
- Un programmatore object oriented affronta un problema
  - dividendolo in agenti (detti oggetti) che fanno qualcosa ed
  - interagiscono fra di loro
- Quando si usa uno stile top-down, si procede in modo algoritmico, delegando la responsabilità di ogni passo ad una procedura
- Un problema può essere decomposto algoritmicamente o in modalità object oriented, ma i due approcci non si possono mischiare. Sono interamente differenti
- Gli oggetti sono indipendenti gli uni dagli altri, dunque più facile verificare, portare, e mantenere procedure interdipendenti. Essi permettono il riuso del codice

I metodi

- Gli oggetti sono costituiti da dati privati e da operazioni permesse su tali dati
- Gli oggetti comunicano fra di loro con il passaggio di messaggi
  - la richiesta ad un oggetto di effettuare una delle sue operazioni
- Un messaggio non è nient'altro che una chiamata a procedura, detta metodo (funzioni membro in C++), che appartiene all'oggetto e potrebbe essere nascosta all'utente



Brad J. Cox, Andrew J. Novobilski, Object-Oriented Programming: An Evolutionary Approach

Messaggi e metodi

- Il passaggio di messaggi permette la comunicazione
  - fra oggetti
  - fra oggetti e programma client
- Quando viene inviato un messaggio ad un oggetto, viene selezionato un metodo, fra quelli disponibili, per rispondere
- Un metodo in un oggetto non può invocare un metodo di un altro oggetto, ma solo inviare dei messaggi
- Così come il tipo è un template per le variabili, la classe è un template per gli oggetti

Esempio in C++

```
typedef int numSides;
typedef int sideLength;
#include <math.h>

class Square {
public:                          // metodi accessibili da qualsiasi punto
    Square(sideLength side): s(side), n(4) {};          // costruttore
    sideLength getSide() { return s;}
    sideLength perimeter() { return n * s;}
    double area() { return (double) s * s;}
private:                        // metodi accessibili da oggetti Square
    sideLength s;
    const numSides n;
};

class Triangle {
public:
    Triangle(sideLength side): s(side), n(3) {};      // costruttore
    sideLength getSide() { return s;}
    sideLength perimeter() { return n * s;}
    double area() { return sqrt(3.0) *getSide() * getSide() / 4.0;}
private:
    sideLength s;
    const numSides n;
};
```

Esempio in C++ /2

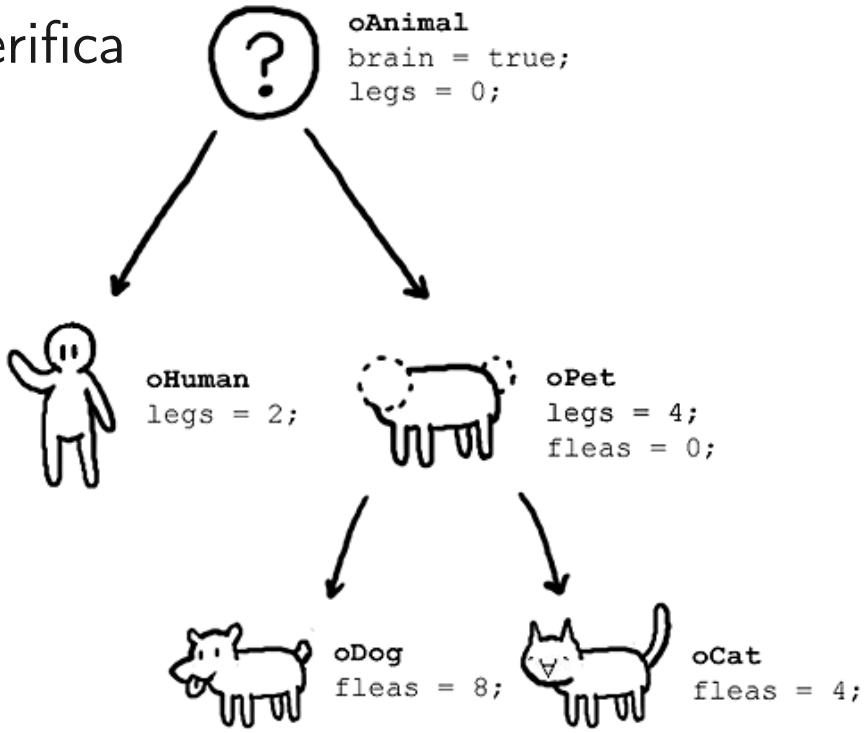
- la parte pubblica elenca la dichiarazione dei metodi
- la parte privata contiene le variabili
- le variabili e i metodi costituiscono gli attributi dell'oggetto
- Square square1(5); (In C++ gli oggetti vengono creati quando dichiarati)
- square1.perimeter(), square1.area()
- per accedere ad s che è privato si usa il metodo getSide()

Classi e polimorfismo

- Una classe è una descrizione di oggetti da "istanziare"
- Un tipo è una descrizione di variabili da dichiarare
- Nei linguaggi object-oriented una classe è una collezione di oggetti, in cui ogni oggetto della classe include gli stessi metodi e le stesse variabili ma può includere diversi valori per i dati
  - oggetto: metodi e dati incapsulati
  - classi di oggetti: tutti gli oggetti cha hanno gli stessi attributi
- Il messaggio draw, che dovrebbe invocare diversi metodi, è detto polimorfico (più forme)
  - operatore aritmetico +: (1+3), (1.5+3)
  - + è un operatore polimorfico o generico
  - + ha diversi significati in base al contesto (overloaded)

Ereditarietà

- I linguaggi object-oriented supportato
  - oggetti
  - classi di oggetti
  - ereditarietà di attributi di una sottoclasse da una classe ad un livello più alto nella gerarchia
- Quando un oggetto riceve un messaggio, verifica l'esistenza di un metodo per rispondere al messaggio. Se non c'è verifica se esiste nella gerarchia delle classi



- Animal (name,habitat,picture)
  - Bird (flyingStatus)
    - Parrot (vocabulary), Ostrich ()
  - Mammal ()
    - Dog (barksAlot), Whale (habitat)

Ereditarietà: gli attributi

- Gli attributi possono essere
  - **ridefiniti** – un attributo che ha lo stesso nome come in una superclasse, ma è definito in una sottoclasse (habitat è ridefinito nella classe Whale)
  - **specifici** – un attributo definito solo in una sottoclasse (flyingStatus, vocabulary e barksALot sono specifici rispettivamente di Bird, Parrot e Dog)
  - **ereditati** – un oggetto ha un attributo definito solo in una delle sue superclassi (name e picture sono ereditati).

Ereditarietà multilpla

- Alcuni oggetti potrebbero ereditare da più classi
- Un problema che si potrebbe verificare per un oggetto discendente è il **name clashes** fra i metodi
  - cosa accade se Novel e Story hanno lo stesso metodo ListPlot?
- In C++ viene utilizzato l'operatore di scope resolution :: per decidere quale metodo utilizzare.