OOP
Object Oriented Programming

Alberto Ferrari

Programmazione orientata agli oggetti

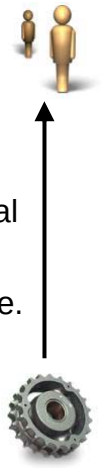
- La programmazione orientata agli oggetti (**O**bject **O**riented **P**rogramming) è un **paradigma di programmazione**
- Permette di raggruppare in un'unica entità (la classe)
  - sia le strutture dati
  - che le procedure che operano su di esse
- Si creano "oggetti" software dotati di proprietà (dati) e metodi (procedure) che operano sui dati dell'oggetto stesso.

OOD (Object Oriented Design)

- La progettazione orientata agli oggetti ha l'obiettivo di formalizzare gli **oggetti** del mondo reale e di costruire con questi un **mondo virtuale**.
- Si avvale del concetto di classe: gli oggetti di una determinata classe hanno le stesse caratteristiche
- Questa parte di mondo che viene ricostruita in modo virtuale è detta **dominio applicativo**.

Alberto Ferrari

Il livello di astrazione



- I linguaggi di programmazione si sono evoluti in modo che i codici sorgenti potessero **astrarsi** sempre più dal modo in cui gli stessi, una volta compilati, sarebbero stati eseguiti.
- Nella OOP non ci si vuole più porre i problemi dal punto di vista del **calcolatore**, ma si vogliono risolvere facendo interagire oggetti del dominio applicativo come fossero oggetti del mondo reale.
- L'obiettivo è di dare uno strumento al programmatore, per formalizzare soluzioni ai propri problemi, pensando come una **persona** e senza doversi sforzare a pensare come una macchina.

Alberto Ferrari

Programmazione imperativa vs Programmazione ad oggetti

Linguaggi procedurali

- Nei linguaggi procedurali (C, Fortran, Pascal) la programmazione è orientata all'**azione**.
- L'unità di programmazione è la **funzione**
- Metodologia: **scomposizione funzionale**

Linguaggi ad oggetti

- Nel linguaggi ad oggetti (C++, Java) la programmazione è orientata all'**oggetto**
- L'unità di programmazione è la **classe**
- Metodologia: **OOD**

Il processo di astrazione: le classi

- Per popolare il dominio applicativo utilizzato dall'applicazione è necessario **creare gli oggetti**, e per fare questo è necessario definire le **classi**.
- Una classe è lo strumento con cui si identifica e si crea un oggetto.

Classi e tipi di dato

- Una classe è a tutti gli effetti un tipo di dato (come gli interi e le stringhe e ogni altro tipo già definito)
- Nella programmazione orientata agli oggetti, è quindi possibile sia utilizzare tipi di dato esistenti, sia definirne di nuovi tramite le classi

Alberto Ferrari

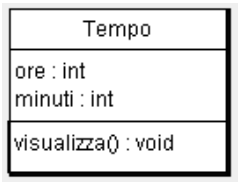
UML (Unified Modeling Language)

- UML ("linguaggio di modellazione unificato") è un linguaggio di modellazione e specifica basato sul paradigma object-oriented.
- Il nucleo del linguaggio fu definito nel 1996 da Grady Booch, Jim Rumbaugh e Ivar Jacobson (detti "i tre amigos") sotto l'egida dello OMG, che tuttora gestisce lo standard di UML.
- Il linguaggio nacque con l'intento di unificare approcci precedenti (dovuti ai tre padri di UML e altri), raccogliendo le best practices nel settore e definendo così uno standard industriale unificato.
- UML svolge un'importantissima funzione di lingua franca nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa UML per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico.
- http://www.uml.org/

Wikipedia

Alberto Ferrari

UML - Diagramma delle classi



- La prima sezione contiene il **nome** della classe
- La seconda sezione definisce i suoi **attributi** (in C++ variabili membro)
- Nella terza sezione sono definiti i **metodi**, le operazioni che si possono compiere sull'oggetto di quel tipo. (in C++ funzioni membro)

Alberto Ferrari

Esempio C++

```
#include <iostream>
using namespace std;

class Tempo
{
public:
   int ore;
   int minuti;
   void visualizza()
   {
    cout<<ore<<":"<<minuti<<endl;
   }
private:
};
```

```
int main()
{
    Tempo t;
    t.ore=9;
    t.minuti=30;
    t.visualizza();
}
```

Alberto Ferrari

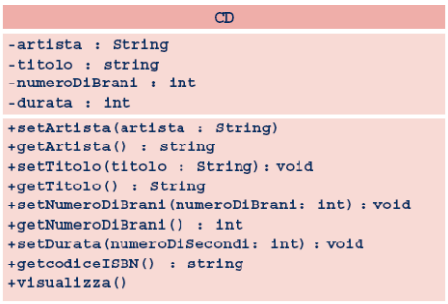
Gli oggetti

- Gli oggetti sono le entità di un programma che interagiscono tra loro per raggiungere un obiettivo
- Gli oggetti vengono creati in fase di esecuzione ed ognuno di essi fa parte di una categoria (di una classe)
- Ogni classe può creare più oggetti, ognuno dei quali pur essendo dello stesso tipo è distinto dagli altri
- Un oggetto è l'istanza di una classe

Alberto Ferrari

Un esempio di classe

- Se vogliamo catalogare i cd musicali in nostro possesso, abbiamo bisogno di implementare un programma nel cui dominio applicativo è presente la classe CD
- I metodi della classe CD servono per impostare e recuperare i valori degli attributi

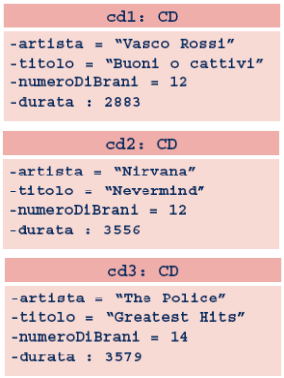


Alberto Ferrari

Diagramma degli oggetti

- I diagrammi che rappresentano gli oggetti (Object Diagram in UML)

  mettono in luce i valori che assumono gli attributi



Alberto Ferrari

Stato di un oggetto

- L'insieme dei valori degli attributi di un oggetto è chiamato stato dell'oggetto e generalmente può variare in funzione del tempo

Alberto Ferrari

Creazione di un oggetto

- Per creare un oggetto si effettua un’istanziazione di una classe.
- In questa fase viene riservato uno spazio di memoria per conservare i valori degli attributi dell’oggetto che si sta creando (per mantenere memorizzato da qualche parte lo stato dell’oggetto)

Alberto Ferrari

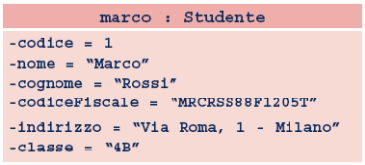
Istanziare un oggetto in C++

- A seconda del linguaggio utilizzato si impiegano diversi costrutti di programmazione per creare un oggetto

Alberto Ferrari

Variabili membro

- Le variabili membro sono quelle posseduti da un oggetto, sono chiamate anche **attributi** dell'oggetto.
- L'attributo di un oggetto è una variabile che ne descrive una caratteristica o proprietà



Alberto Ferrari

Le azioni degli oggetti

- Una funzione membro (metodo) è un'azione che l'oggetto può eseguire.
- La dichiarazione di una funzione è composta da:
  - Modificatore
  - Nome del metodo
  - Tipo di dato da ritornare
  - Tipo e nome dei parametri di ingresso
- Tutto questo è detto **firma del metodo**.

Alberto Ferrari

Funzioni membro

- Una funzione membro, per essere utilizzata, ha bisogno della creazione di un oggetto della classe a cui appartiene su cui essere invocata.

Alberto Ferrari

Esempio di firma

- **`public int studia(string testo)`**
- **`public`** è il modificatore
- **`int`** è il tipo del metodo
- **`studia`** è il nome del metodo
- **`string testo`**
  è il tipo e nome dei parametri

Alberto Ferrari

Un esempio: attributi

- Si vuole realizzare una classe che permetta di gestire e risolvere equazioni di secondo grado
- In una equazione individuiamo tre **attributi**: **a, b, c** che rappresentano i coefficienti di x^2, di **x** ed il termine noto
- L'equazione $3x^2-2x+1=0$ avrà come attributi i valori 3, -2 e 1

Alberto Ferrari

Un esempio: metodi

- Definiamo un insieme di metodi che ci permetta di:
  - Modificare i valori dei coefficienti
  - Ottenere i valori dei coefficienti
  - Conoscere il tipo di equazione
  - Ottenere la prima soluzione
  - Ottenere la seconda soluzione

Alberto Ferrari

Diagramma UML della classe

Equazione
-a : double -b : double -b : double
+setA(in v : double) +getA() : double +setB(in v : double) +getB() : double +setC(in v : double) +getC() : double -delta() : double +pura() : boolean(idl) +spuria() : boolean(idl) +complessa() : boolean(idl) +soluzione1() : double +soluzione2() : double

Esercizio

- Implementare la classe Equazione
- Istanziare due equazioni:
  - $5x^2-3x+2=0$
  - $2x^2-4=0$

eq1 : Equazione
a : double = 5 b : double = -3 b : double = 2

eq2 : Equazione
a : double = 2 b : double = 0 b : double = -4

Metodo costruttore

- Il **costruttore** è un metodo particolare che viene invocato alla creazione dell'oggetto e che contiene tutte le istruzioni da eseguire per la sua inizializzazione.

Alberto Ferrari

Modificatori

- **public**: consente a qualunque classe o oggetto di qualsiasi tipo di avere accesso all'attributo o al metodo a cui è applicato.
- **protected**: consente l'accesso solo alle classi e agli oggetti il cui tipo è una sottoclasse di quella in cui è utilizzato. Le sottoclassi saranno trattate in successive lezioni.
- **private**: consente l'accesso solo agli oggetti della classe stessa in cui è utilizzato.

Alberto Ferrari

Incapsulamento

- L'**incapsulamento (***information hiding)* è un concetto fondamentale dell'ingegneria del software.
- Questo principio prevede che si possa **accedere** alle informazioni di un oggetto **unicamente attraverso i suoi metodi**.
- La tecnica di programmazione che consente di applicare l'incapsulamento si avvale dei **modificatori di visibilità** per nascondere gli attributi di un oggetto al mondo esterno.
- Mettere in atto questa tecnica significa **non avere** mai **attributi** di un oggetto di tipo **public**, salvo eccezioni particolari per costanti o attributi di classe da gestire in base al caso specifico.

Accesso agli attributi

- Per accedere dall'esterno agli attributi, si inseriscono metodi **public** che possono essere chiamati da chiunque per impostare o richiedere il valore dell'attributo.
- I metodi hanno di solito un nome particolare:
  - set (seguito dal nome dell'attributo) per modificarne (settare) il valore
  - get (seguito dal nome dell'attributo) per recuperare (get) il valore

Incapsulamento: perché?

- Potrebbe sembrare che non vi sia alcuna differenza rispetto ad accedere direttamente agli attributi
- Sembra che questa tecnica serva solo a rendere più complessa la loro gestione
- Le motivazioni sono:
  - un maggiore controllo sulle operazioni effettuate sugli attributi, limitando l'utilizzo improprio che se ne può fare e guadagnando così in sicurezza.
  - La possibilità di nascondere il modo in cui i dati sono memorizzati negli attributi

Interazione tra gli oggetti

- Per comunicare, gli oggetti possono utilizzare i metodi, scambiandosi messaggi l'uno con l'altro.
- Quando un oggetto invoca un metodo di un altro, quest'ultimo reagisce eseguendo il metodo opportuno.
- L'invocazione dei metodi può richiedere parametri di input di qualsiasi tipo, compresi quindi oggetti del nostro dominio applicativo.
- Un oggetto potrà quindi essere in grado di passarne un altro attraverso un metodo, o addirittura potrà passare se stesso.
- Un messaggio ha la seguente sintassi:

```
<NomeOggetto>.<nomeMetodo>(<paramteri>)
```