OOA/P

Introduzione a Object Oriented Analysis/Design

OOAP/D - definizione

- Tecnica utilizzata per analizzare e progettare un’applicazione o un sistema applicando OOP (Object Oriented Programming);
- Utilizza la modellazione visuale, durante tutto il ciclo di vita del software, per migliorare la comunicazione dei requisiti e la qualità del prodotto;

OOAP/D - definizione

- L’obiettivo di qualsiasi attività di analisi è di creare un modello dei requisiti funzionali che sia indipendente dai dettagli della successiva implementazione;
- La differenza principale tra OOA e le altre forme di analisi è che i requisiti sono organizzati in funzione degli oggetti che integrano sia i processi sia i dati al suo interno.

OOAP/D - definizione

- I modelli utilizzati in OOA/D sono gli use cases e gli object models;
- Gli use cases (casi d'uso) descrivono gli scenari e le funzioni che il sistema deve prevedere;
- Object models (modello delle classi) descrive i nomi, le relazioni tra le classi, le operazioni e le proprietà degli oggetti coinvolti nel sistema;

OOP

Introduzione a Object Oriented Programming

OOP - definizione

- OOP è una filosofia di sviluppo software in cui la struttura di un programma è basata su un insieme di oggetti che interagiscono e collaborano fra di loro per eseguire un compito;
- Ogni oggetto è capace di inviare e ricevere messaggi da altri oggetti e ognuno di essi possiede uno stato e può compiere determinate azioni, strettamente correlati con il tipo di oggetto;
- I concetti degli oggetti nascono nel mondo reale e vengono trasportati nel sistema in fase di sviluppo.

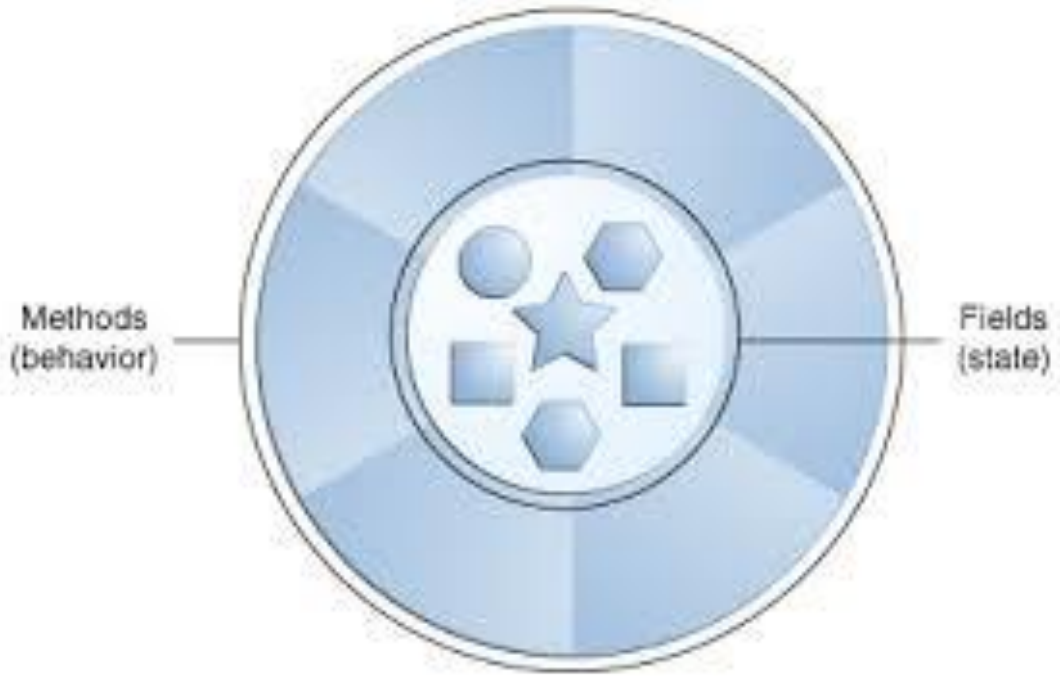
OOP - definizione

- Nei linguaggi OOP esiste un nuovo tipo di dato, la classe. Questo tipo di dato serve appunto a modellare un insieme di oggetti;
- Un oggetto è caratterizzato da un insieme di attributi e da un insieme di funzionalità;
- L'isolamento e l'incapsulamento di funzionalità all'interno di oggetti separati permette di mantenere il codice in maniera molto più semplice, separandone le responsabilità anche in diversi team.
- Pensare a oggetti permette poi un'analisi del modello di business più veloce, anche perché è il cliente stesso che riesce a esporre più facilmente e chiaramente le proprie esigenze.

OOP - oggetto

- Un oggetto è un qualcosa che possiede un suo stato, un suo comportamento e una sua identità (*Grady Booch*);
- Un oggetto è una struttura che mantiene dei dati al suo interno, che fornisce delle funzioni per manipolarli e che risiede in una propria area di memoria riservata, convivendo e comunicando con altri oggetti, dello stesso tipo o di altro tipo;
- La struttura e il comportamento di oggetti simili sono definiti in una classe comune, quindi in termini OOP un oggetto è un'*istanza* di una classe;
- I termini istanza e oggetto sono intercambiabili e sinonimi e la creazione di un oggetto viene detta anche istanziazione dell'oggetto.

OOP - oggetto



OOP – classe vs oggetto

- La differenza fra classe e oggetto è sottile, ma importante da comprendere;
- La classe è qualcosa di astratto, che non esiste, è ciò che definisce come un oggetto sarà costruito, una sorta di documento progettuale (template);
- L'oggetto è concreto, esiste, è costruito in base al progetto definito dalla sua classe, ha uno stato costituito da dati e valori e può eseguire delle funzionalità o scambiare messaggi con altri oggetti;
- Quando si progetta un'applicazione, bisogna pensare ai concetti del dominio che si sta affrontando e ricavarne gli oggetti e le funzionalità che essi dovranno avere, fornendo magari le modalità di comunicazione fra essi o con oggetti di classi diverse.

OOP - caratteristiche

I concetti fondamentali di OOP sono:

- Classe/oggetto
- Astrazione
- Incapsulamento
- Ereditarietà
- Poliformismo

OOP – caratteristiche - astrazione

- L’astrazione (abstraction) non è un concetto peculiare del paradigma a oggetti, esso è presente in qualunque linguaggio che permetta di definire delle strutture dati, con cui il programma dovrà interagire;
- Il processo di astrazione è, dunque, quello con cui si descrive in maniera più o meno essenziale un oggetto, nascondendo i dettagli implementativi interni e riducendo le complessità;
- L’astrazione è un concetto strettamente correlato con quello di incapsulamento;

OOP – caratteristiche - incapsulamento

- Quando si interagisce o si utilizza un oggetto nel mondo reale non ci interessa come esso sia stato costruito internamente e come sta funzionando;
- L'incapsulamento (detto anche encapsulation o information hiding) è il processo mediante il quale si nascondono al mondo esterno i dettagli implementativi e si proteggono i dati interni degli oggetti;
- Attraverso i concetti di proprietà e metodi e i relativi modificatori di accesso, si forniscono al mondo esterno la possibilità di interagire con l'oggetto e modificarne lo stato senza dover conoscere e modificare direttamente i dettagli interni.
- L'incapsulamento rende quindi i dati degli oggetti più sicuri e affidabili, perché esistono e si conoscono quali sono i modi per accedere a essi e quali sono le operazioni ammesse per modificarli.

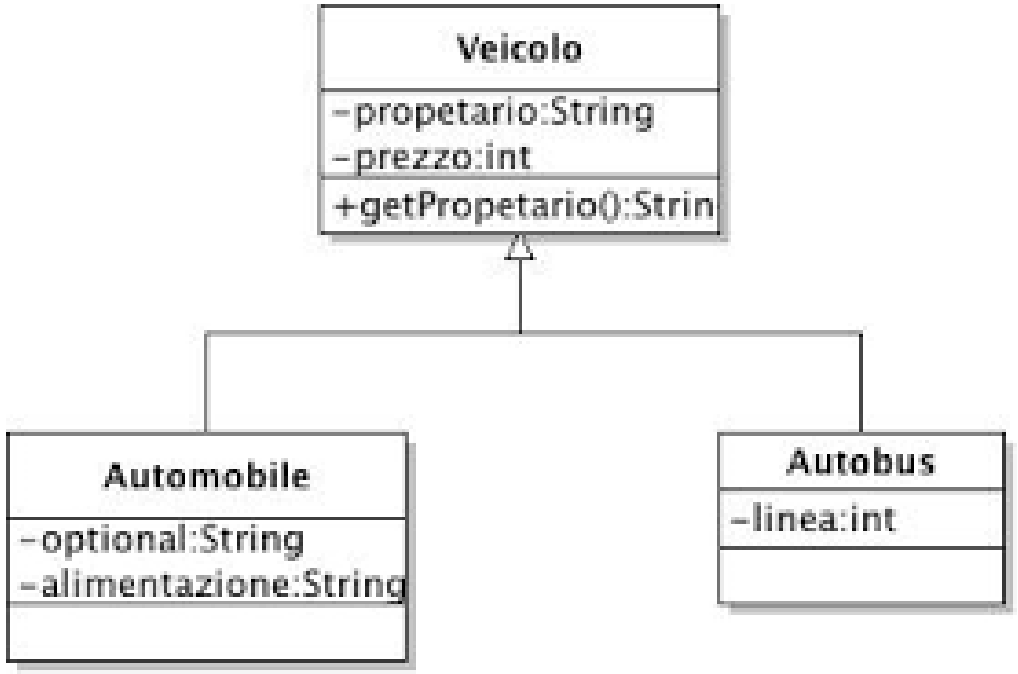
OOP – caratteristiche - ereditarietà

- L'ereditarietà è un meccanismo che permette e facilita il riuso del codice esistente e la manutenzione dello stesso;
- Se una classe possiede determinate caratteristiche possiamo riutilizzarle creando una nuova classe derivata da essa e che erediti alcune di tali caratteristiche, evit;ndo di duplicare codice già scritto;
- In natura molti oggetti possono essere classificati mediante delle gerarchie, in accordo a determinate caratteristiche comuni, o ad altre caratteristiche che vengono invece estese a oggetti più specializzati.
- In termini OOP il meccanismo di ereditarietà permette di definire nuove classi, ereditandole da quelle esistenti e quindi estendendone le caratteristiche mediante nuovi metodi e proprietà, oppure ridefinendone alcune.

OOP – caratteristiche - ereditarietà

- Quando si definiscono e si usano delle gerarchie di classi vengono utilizzate diverse terminologie, con molti termini che indicano lo stesso concetto, cioè sinonimi.
- Una classe che rappresenta la classe madre di una gerarchia viene anche detta classe base o superclasse.
- la classe figlia viene anche detta classe derivata, o ancora classe che estende un'altra classe, perché alla versione base essa aggiunge altre caratteristiche.

OOP – caratteristiche - ereditarietà



OOP – caratteristiche - poliformismo

- Il termine polimorfismo indica letteralmente la possibilità di assumere molte forme e rappresenta la possibilità per differenti oggetti di eseguire una stessa azione o rispondere a uno stesso messaggio in maniera diversa;
- E' strettamente legato all'ereditarietà;
- Metodi per implementare il poliformismo in C# sono:
  - overloading dei metodi, che permette di implementare in una stessa classe più metodi con lo stesso nome e parametri differenti;
  - overriding, che invece permette di ridefinire un metodo presente in una classe base in una classe derivata, quindi con stesso nome e stessi parametri;
- Per tali motivi l'overloading è anche detto polimorfismo a tempo di compilazione, mentre l'overriding è un polimorfismo a tempo di esecuzione.

OOP – classi / oggetti

- Nei linguaggi OOP esiste un nuovo tipo di dato, la classe;
- La classe è una struttura dati che definisce e mantiene lo stato e il comportamento in una singola unità;
- La creazione di una classe in C# avviene mediante l'utilizzo della parola chiave class, la cui sintassi più semplice prevede l'indicazione del nome della classe e quindi del blocco che conterrà i suoi membri:

```
class Dog
{
    //members
}
```

OOP – classi / oggetti

- La sintassi completa per definire una classe può anche essere più complessa e prevedere altri elementi.
- La sintassi completa è:

```
[attributi] [modificatori] [partial] class NomeClasse
[:ClasseBase, Interfacce, ParametroTipoGenerico]
{
   [membri]
}
```

- Quindi una definizione di classe può anche essere parecchio complessa, con diversi attributi, modificatori, il nome della classe da cui deriva e così via.

OOP – classi – modificatori di accesso

Modificatore	Si applica a	Descrizione
public	Tipi e membri	I tipi e membri public non hanno limiti di accesso
protected	Membri e tipi innestati	Accesso consentito solo all'interno della classe che definisce l'elemento e da classi derivate
Private	Membri e tipi innestati	Accesso consentito solo all'interno della classe che definisce l'elemento
Internal	Tipi e membri	Gli elementi internal sono accessibili solo all'interno dell'assembly in cui sono definiti
Protected internal	Membri e tipi innestati	Gli elementi che combinano protected e internal sono accessibili da qualunque tipo nello stesso assembly e in classi derivate anche di altri assembly

OOP – classi – tipi di membri

Membro	Tipo	Descrizione
Campo	Dati	Variabili utilizzate per contenere dati associati alla classe.
Costante	Dati	Valori costanti associati alla classe.
Metodo	Funzione	Azioni eseguibili dalla classe
Proprietà	Funzione	Punti di accesso ai dati della classe, sia in lettura sia in scrittura
Costruttore	Funzione	Azioni eseguite alla costruzione di un’istanza della classe
Tipo innestato	Dati	Tipi definiti all’interno di un altro tipo (classe o struct)
...	...	...

OOP – classi / oggetti

- Un oggetto è una variabile che appartiene ad un particolare tipo di dato definito dall'utente per mezzo del costrutto class
- La variabile di un certo tipo (classe) rappresenta un istanza della classe;
- Lo stato di un oggetto, invece, è rappresento dai valori correnti delle variabili che costituiscono la struttura dati utilizzata per implementare il tipo di dato rappresentato dalla classe

OOP – classi / oggetti

- La classe è dotata di una interfaccia e di un corpo;
- La struttura dati di un oggetto della classe e le operazioni sono tenute nascoste all'interno del modulo che implementa la classe;
- Lo stato di un oggetto viene modificato in relazione alle operazioni previste per la sua modifica;
- Le operazioni sono utilizzabili a prescindere da qualsiasi aspetto implementativo, in tal modo è possibile modificare gli algoritmi utilizzati senza modificare l'interfaccia;

OOP - esercitazione