Principi Fondamentali della Programmazione Orientata agli Oggetti (OOP)

Le fondamenta dell'OOP sono:

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

Ma prima di tutto dobbiamo capire cosa è un oggetto.

Oggetti

E' un qualcosa che rappresenta il mondo reale. Può essere sia un qualcosa con una forma fisica che astratta.
Nell'ambito della programmazione orientata agli oggetti, questi, sono cose di cui si vuole memorizzare e processare dati.

> **Note**
>
> Un altro nome per oggetti è **entità**.

Abstraction

Serve a **semplificare la realtà**. Ossia ci permette di non dovere prendere in considerazione tutte le caratteristiche di un oggetto del mondo reale, ma solo quelle che ci interessano per il programma.

Classi

Le classi sono dei template per gli oggetti. Definiscono gli attributi e le operazioni che possiamo svolgere sugli oggetti.

> **Note**
>
> Gli attributi vengono chiamati **campi**, mentre le operazioni sono dette **metodi.**

I campi sono degli elementi che permette di rappresentare delle proprietà della classe e degli oggetti generate da essa. I metodi invece descrivono le azioni che possono essere svolte dalla/sulla classe.

> **Note**
>
> I campi sono parti di codice mantenute come variabili, mentre i metodi sono parte di codice mantenuti come funzioni.

Una classe può essere usata per creare tanti oggetti dello stesso tipo. Infatti ci si riferisce alle classi come **tipi di dato**. Una classe può creare tante **istanze**, per questo motivo il processo di creazione di un oggetto partendo da una classe si chiama: **istanziazione**.

Ad ogni oggetto possono essere associati diversi attributi rendendo le oggetti unici.

> **Note**
>
> Ci si riferisce agli insiemi degli attributi di un oggetto come: **Stato**.

Possiamo passare gli attributi durante la creazione dell'oggetto tramite il metodo **new**, conosciuto anche come **costruttore**.

Encapsulation

Serve a **nascondere la complessità interna di un oggetto** dal programma e dai programmatori che ne fanno uso.

> **Note**
>
> Si chiama anche *information hiding*, perché le informazioni contenute in un oggetto e le funzioni che manipolano i dati sono collegate e quindi sono al sicuro da interferenze esterne.

Una classe potrebbe essere resa accessibile sotto forma di una libreria. Per creare un oggetto da un classe di questo tipo, non è necessario capire come funziona internamente la classe. L'unica cosa di cui si ha bisogno è dell'**interfaccia della classe**. L'implementazione può rimanere sconosciuta.

Inheritance

Una **classe può derivare i propri metodi e proprietà da un'altra classe**. In questo modo si crea una gerarchia di oggetti che ereditano metodi e proprietà di una classe padre.
Una classe può anche **estendere** un'altra classe da cui ottiene campi e metodi, aggiungendo nuovi metodi utilizzabili solo da sè stessa.

> **Note**
>
> L'Inheritance definisce delle relazioni tra tipi di oggetti.

La classe in cima alla gerarchia è detta **classe base**. Qualsiasi classe che deriva da un'altra classe è detta **sottoclasse**. Ogni classe da cui si derivano altre classi è detta **superclasse**.

Polymorphism

Una classe può **implementare dei metodi ereditati nel modo in cui vuole.** Il polimorfismo permette di riscrivere il funzionamento di un metodo che eredita da una superclasse con una nuova versione.
Differenti forme dello stesso tipo oggetto hanno tutti la stessa interfaccia, ma la implementano in maniera diversa.

Recap

Proprietà	Che cosa fà
Abstraction	Semplificare la realtà e mantenere solo dati e processi necessari.
Encapsulation	Dati e programmi che manipolano i dati sono connessi e la loro complessità è nascosti.
Inheritance	Una classe può derivare i propri metodi e campi da un'altra classe.

Proprietà	Che cosa fà
Polymorphism	Sottoclassi diverse della stessa superclasse implementano la stessa interfaccia in maniera diversa