**OOP**

**La OOP è un tipo di programmazione non procedurale (quindi non esegue istruzione per istruzione) in cui vengono creati degli oggetti che interagiscono tra loro. Ha 4 proprietà fondamentali:**

1. **INCAPSULAMENTO:**

**E’ il modo che abbiamo per raggruppare e incorporare le informazioni e le funzionalità di un oggetto, all’interno dell’oggetto stesso (O CLASSE)**

**INFORMATION HIDING: è un metodo con cui possiamo nascondere e quindi rendere privati o protetti attributi e metodi.**

**Rendendoli privati ci assicuriamo che siano inaccessibili dall’esterno.**

**Quelli protetti sono inaccessibili dall’esterno ma visibili dalle classi che ereditano.**

**Quelli pubblici sono accessibili da ogni parte del programma.**

1. **POLIMORFISMO:**

**\*\* E’ la capacità di un oggetto di assumere più forme. Con il polimorfismo possiamo dichiarare degli oggetti con la loro superclasse e in automatico ognuno riconsocerà il proprio metodo da utilizzare\*\***

**TIPI DI POLIMORFISMO:**

**Esistono 3 tipi di polimorfismo:**

1. **tramite extends con superclasse e sottoclasse,**
2. **tramite classe astratta**
3. **tramite interfaccia**
4. **EREDITARIETA’:**

**CLASSE**

Una classe è un tipo (o modello base) di un oggetto, che ha degli attributi (ovvero delle caratteristiche) e dei metodi (delle funzionalità).

**ISTANZA:**

Oggetto concreto creato a partire da una classe

**A COSA SERVONO I DUE PUNTI QUANDO CREIAMO UNA CLASSE? Studente : Persona**

Significano EXTENDS ( estendere ) perché con la sottoclasse estendiamo attributi e metodi della superclasse.

Quindi Studente estende Persona ed eredita attributi e metodi

**PROPERTY**

Sono un insieme di metodi get e set.

Le property servono per dichiarare un attributo di una classe che sarà privato ma che avrà i metodi get e set (che serviranno per leggerlo e modificarlo) che saranno pubblici e quindi accessibili dall’esterno.

**CLASSE OBJECT**

La classe Object è la classe genesi da cui derivano ed ereditano tutti i tipi

**CLASSE ASTRATTA**

E’ una classe non permette la sua istanziazione. Può contenere attributi e metodi che possono essere estesi dalle sue sottoclassi. Può avere dei metodi astratti:

**METODI ASTRATTI:**

sono dei metodi senza corpo, che devono essere ridefiniti nelle sottoclassi.

**VIRTUAL:**

La keyowrd virtual si utilizza per permettere ai metodi una superclasse di essere sovrascritti in una sottoclasse.

**OVERRIDE:**

La keyword Override si utilizza per sovrascrivere (o ridefinire) dei metodi

**COSTRUTTORE:**

Il **costruttore** è un tipo speciale di funzione in una classe che viene eseguita automaticamente ogni volta che crei un nuovo oggetto di quella classe

**COSTRUTTORE DI DEFAULT:**

Un **costruttore di default** non ha parametri (o utilizza solo valori predefiniti) e viene eseguito senza passare argomenti durante la creazione dell'oggetto.

Permette di allocare in memoria gli attributi della superclasse quando genero un’istanza della sottoclasse

**SPIEGA BASE E THIS?**

Base si riferisce agli attributi della classe genitore (superclasse) mentre this si riferisce a questo oggetto (sicuramente sottoclasse)

**SIGNATURE:**

E’ la descrizione completa delle informazioni per richiamare una funzione, quali nome, numero , tipo di parametri e valori di ritorno.

**OVERLOAD:**

I metodi possono avere lo stesso nome ma se hanno parametri o tipi diversi , appunto la signature, possono avere diverse funzioni

**CASTING:**

E’ una sorta di conversione, ma si può fare solo tra tipi uguali e tra superclassi e sue sottoclassi

**DIFFERENZA TRA ARRAY E LIST?**

L’array è una collezione di dati (o struttura dati) che ha lunghezza fissa, e al suo interno ha elementi dello stesso tipo.

La lista ha lunghezza variabile e può avere elementi diversi.

**AGGREGAZIONE:**

è quando un contenitore esiste a prescindere dal suo contenuto

**COMPOSIZIONE:**

è quando il contenitore non può esistere se il suo contenuto non esiste

**PATTERN:**

Soluzione strategica per problemi noti e ricorrenti

**INTERFACCIA:**

E’ un costrutto che ha solamente metodi ed obbliga le classi che la ereditano ad implementare i suoi metodi

**INTERFACCIA<T>:**

L’interfaccia con il tipo T è un’interfaccia generica che grazie a questo parametro <T> mi consente di richiamare al suo interno uno dei tipi che implementano la data interfaccia. (Interface<Automobile>)

**ARCHITETTURA THREE TEAR:**

User interface, business logic, DBMS

Serve a dividere le responsabilità di ogni livello (primo principio SOLID, single responsability principle)

**FACADES**

Modo che abbiamo per nascondere tutto il dettaglio costruttivo che c’è dietro il nostro programma, appunto mettiamo una facciata.

**TUPLA:**

sequenza di dati eterogenei (tutti diversi tra loro)

**ECCEZIONI:**

Eventi che si verificano quando va qualcosa storto nel codice

**COS’è STATIC?**

Attributo di classe

E’ una keyword per definire un metodo o attributo che appartiene alla classe e non all’oggetto istanziato, quindi sono utilizzabili anche senza creare un oggetto (CLASSE.METODOSTATIC())

I metodi static sono i primi che vengono allocati in memoria e quindi vengono richiamati per primi.

**HASHTABLE:**

collezione di dati composti da coppia chiave-valore

**DICTIONARY:**

collezione di dati composta da coppie chiave-valore TIPIZZATE

**INTERFACCIA DI LOOPBACK:**

permette la comunicazione tra processi sulla stessa macchina, ad esempio durante il test di server web: connettiamo il client con il server ed entrambi saranno eseguiti sulla stessa macchina.

**SOLID:**

I principi SOLID sono intesi come linee guida per lo sviluppo di software leggibile, estendibile e [manutenibile](https://it.wikipedia.org/wiki/Manutenzione_(software)).

5 principi fondamentali della OOP:

* **S: principio di singola responsabilità**
  + ogni classe dovrebbe avere una ed una sola responsabilità incapsulate al suo interno
* **O: principio aperto chiuso**
  + Le entità devono essere aperte alle estensioni (devono essere ereditabili) ma chiuse alle modifiche
* **L: principio di sostituzione di liskov**
  + Gli oggetti devono poter essere sostituiti con dei loro sottotipi ( polimorfismo)
* **I: principio di segregazione delle interfacce**
  + È preferibile avere più interfacce specifiche che una singola generica (una per la lettura, e una per la scrittura ad esempio)
* **D: principio di inversione delle dipendenze**

**DAO**

Il pattern DAO o DATA ACCESS OBJECT si utilizza per separare la logica di accesso ai dati del database, ad esempio le operazioni CRUD dalla logica di business.

Data Access Object, è un oggetto dedicato solo all’interazione con il database ed è relativo ad una singola classe. All’interno del file relativo al DAO ci sono tutte le operazioni di CRUD.

**SINGLETON**

Il singleton è un pattern utilizzato per garantire che una data classe abbia una sola istanza. In pratica se si prova a creare più oggetti di questa classe il singleton ci restituirà sempre lo stesso oggetto.

Innanzitutto bisogna

* rendere impossibile l’uso del costruttore new rendendo private la funzione del costruttore in modo che non sia accessibile dall’esterno ma solo all’interno della classe stessa;
* creare un metodo pubblico che crei l’istanza della classe se già non esiste, oppure restituire l’istanza già esistente

Questo pattern è utile per risparmiare memoria e tempo di elaborazione.

Di solito si utilizza quando una classe si occupa di eseguire delle funzioni, ad esempio le interfacce DAO, che si occupano di interagire con il database con le CRUD etc…

**DIFFERNZA TRA INTERFACCIA E CLASSE ASTRATTA:**

L’interfaccia non ha attributi, ha solamente metodi astratti e costringe chi la implementa ad implementare i suoi metodi.

La classe astratta .

**DATABASE**

**QUAL È IL PRIMO ELEMENTO QUANDO CREIAMO UNA TABELLA?**

L’id

**CHE COS’è UNA PRIMARY KEY?**

Una primary key è un indice e una struttura dati che serve per ottimizzare le ricerche

**A COSA SERVE UNA PRIMARY KEY?**

La PK identifica in modo univoco ogni record nella tabella

**COS’è UN INDICE?**

Identificatore univoco di ogni record. E’ una struttura di memoria di rapida consultazione di un dato che porta con se tutte le informazioni del record.

**QUAL È LA DIFFERENZA TRA PRIMARY KEY E UNIQUE?**

La primary key è univoca quindi può essercene solo una, mentre di unique possono essercene tante.

La primary key è più ottimizzata e di conseguenza è più velocemente reperibile.

COS’è LO SCHEMA?

Lo schema è la parte invariante del database

**COS’è UNA FOREIGN KEY?**

E’ un vincolo di integrità relazionale, quindi controlla che il puntamento (la referenza ) ad un’altra tabella esista.

COS’è IL DDL?

Il Data Definition Language è l’insieme di istruzioni sequel per la creazione ,modificare o eliminare oggetti sul database, ovvero agire sullo SCHEMA (CREATE,DROP,ALTER)

COS’è IL DML?

Il Data Manipulation Language è l’insieme di istruzioni che serve per manipolare i dati in un database (INSERT, UPDATE, DELETE etc.. )

**HTML,CSS**

**CSS:** foglio di stile che serve per applicare regole grafiche (di stile) alla pagina html

**DIV:** Contenitore logico di elementi

**DOM:** Document Object Model, documento web che contiene tutte le entità(tag) html.

**CDN**: Content delivery networks, rete di server che ci mette a disposizione contenuti, quali le librerie (ad esempio le librerie grafiche o di icone)

COSA SIGNIFICA CHE HTML è STATELESS?

**JAVASCRIPT**

**Javascript engine?**

Js engine V8 è un motore JavaScript open source sviluppato da Google, attualmente incluso in Google Chrome.

**ARROW FUNCTION**

Sono un tipo compatto di funzioni e sono anonime, cioè non hanno il nome.

(parametri) => {funzione}

**DIFFERENZA TRA LET E VAR**

Entrambe hanno scope di blocco ( o di funzione) , ovvero sono visibili solo all’interno della funzione in cui sono dichiarate.

La vera differenza sta nel fatto che var è ridichiarabile mentre let non può essere ridichiarata ma solo aggiornata.

Un’altra differenza è nell’hoisting, ovvero il sollevamento all’inizio del codice, quindi se provo ad accedere ad una var prima della sua dichiarazione otterrò undefined, mentre se provo ad accedere ad una let prima della sua dichiarazione riceverò un messaggio di errore in cui c’è scritto che deve essere ancora definita.

**DIFFERENZA TRA == E ===**

Due uguali comparano solo il contenuto, mentre tre uguali (strettamente uguali) comparano contenuto e tipo del contenuto

**GIT**

COS’è GIT?

COS’è LA STAGING AREA?

**COMMIT**

(un insieme di cambiamenti)

prende ciò che abbiamo nella staging area, allega un messaggio e memorizza un'instantanea di tutti i file

FAST FORWARD: modalità per riversare un commit su un altro branch senza avere conflitti, quindi il branch su cui stiamo eseguendo il merge eredita il commit del branch