CLASSI
OOP
La programmazione orientata agli oggetti(in inglese object-oriented programming, in acronimo OOP) è un paradigma di programmazione che permette di definire oggetti software in grado di interagire gli uni con gli altri attraverso lo scambio di messaggi.
OGGETTI
Gli oggetti sono la chiave per comprendere la tecnologia Object Oriented.
Un oggetto rappresenta un singolo esemplare di una certa categoria.
Le caratteristiche di un oggetto sono :

- •Identità
  Ogni oggetto è indicato dalle sue qualità (attributi),dai suoi stati e dai suoi comportamenti .

- •Attributi
  Queste qualità possono essere fisse, ovvero non cambiano dalla creazione dell'oggetto, o subire un cambiamento a seconda del comportamento dell'oggetto o del comportamento di altri oggetti esterni
- •Stato
  Rappresenta la situazione in cui si trova l'oggetto in un dato tempo o spazio .
- •Comportamento
  rappresenta la situazione in cui si trova l'oggetto in un dato tempo o spazio .

Un sistema è un'unità fisica e funzionale costituita da più parti o sotto inisemi che interagiscono tra loro con l'obiettivo di raggiungere una finalità comune.
In un sistema possiamo avere diversi oggetti che interagiscono tra loro.
Ogni oggetto può essere replicato in diverse «copie» che rappresentano le istanze di tale oggetto .
Ogni istanza ha un suo diverso particolare stato ed, oltre allo stato, ha anche suoi propri metodi(comportamenti) che operano sullo stato stesso.
CARATTERISTICHE OOP
Un linguaggio si definisce ad oggetti se gli oggetti prevedono caratteristiche come:

- •Astrazione
  Essa rappresenta i diversi punti di vista di un determinato oggetto.
  Il termine astrazione usato nell'ambito dello sviluppo di sistemi ,indica che solo gli oggetti che sono effettivamente necessari dovrebbero essere disponibili
- •Classificazione
  Gli oggetti vengono creati in base a una definizione di oggetti dello stesso tipo, chiamata classe .
  La classe è un modello o prototipo che definisce un tipo di oggetto, cioè è un modello per tutti gli oggetti dello stesso tipo; definisce la struttura e il comportamento degli oggetti appartenenti alla classe.
  La creazione di un oggetto a partire da una classe si dice istanziazione o creazione di un'istanza della classe.
  La classe è il codice, è lo "stampo" per istanziare oggetti.
  Un oggetto è un'istanza(o esemplare) della classe.
  I termini"oggetto"e"istanza"sono sinonimi.
  La differenza principale consiste nel riferimento alla classe.
  L'Istanza indica la relazione di un oggetto con la sua classe.
  Gli oggetti sono legati al mondo reale, e rappresentano istanze di classi che si trovano nella memoria di un programma durante l'esecuzione.

Una classe è invece un concetto statico, cio è una volta impostato rimane come è nel programma, mentre l'oggetto è un concetto dinamico, il che significa che può essere creato, modificato e cancellato mentre il programma è in esecuzione.

•Ereditarietà

Gli oggetti possono essere suddivisi in sottoclassi che «ereditano» gli stessi attributi della superclasse,favorendo il riutilizzo del codice.

•Incapsulamento

Limita la visibilità delle informazioni di una classe per una maggiore chiarezza ,contribuendo a migliorare la manutenzione e favorendo l'utilizzo del codice.

L'incapsulamento si ha quando l'oggetto mantiene il suo stato privato all'interno della classe.

Altri oggetti non hanno accesso diretto a questo stato.

Invece,possono solo chiamare un elenco di funzioni pubbliche, chiamate metodi.

Quindi, l'oggetto gestisce il proprio stato tramite i metodi e nessun' altra classe può modificare lo stato se non esplicitamente consentito.

Se si desidera comunicare con l'oggetto, è necessario utilizzare i metodi pubblici forniti.

Per default, non è possibile modificare lo stato.

•Polimorfismo

L'oggetto può assumere il tipo di una sua superclasse.

È inteso come la capacità di un codice di avere diversi comportamenti oppure la capacità di uno stesso codice di essere utilizzato da più classi.

•Persistenza

Indica che lo stato e il comportamento di un oggetto variano con il tempo

ATTRIBUTI E METODI

Le classi e gli oggetti sarebbero inutili se non fossimo in grado di manipolarli.

Attributi e metodi sono il modo orientato agli oggetti per manipolare le istanze di una classe .

Gli Attributi sono elementi che definiscono la struttura di una classe.

Sono le variabili che memorizzano i tipi di dati specifici come numeri interi,stringhe,numeri reali.

Il valore di questi attributi nel tempo indica lo stato di tale oggetto in un determinato momento.

Un attributo può essere di istanza (ovvero relativo a quella particolare copia della classe) oppure di classe (ovvero relativo alla classe in maniera generale,valido quindi per tutti gli oggetti di quella classe).

I metodi definiscono le operazioni eseguibili sull'oggetto, cio è il comportamento

L'utilità dei metodi è di evitare inutili copie di codice, essendo sufficiente richiamare i metodi quando lo si desidera.

Inoltre è possibile utilizzare il codice di classi esistenti e velocizzare intal modo l'implementazione del sistema finale.

CLASSI ,OGGETTI E INTERFACCE

Le classi Apex sono modellate sulle loro controparti in Java. Potrai definire, istanziare ed estendere le classi e lavorerai con interfacce, versioni di classi Apex, proprietà e altri concetti di classe correlati.

Classi

Una classe è un modello da cui vengono creati gli oggetti.

Un oggetto è un'istanza di una classe.

Interfacce

Un'interfaccia è simile a una classe in cui nessuno dei metodi è stato implementato: le firme del metodo sono presenti, ma il corpo di ciascun metodo è vuoto

Per utilizzare un'interfaccia, un'altra classe deve implementarla fornendo un corpo per tutti i metodi contenuti nell'interfaccia.
Una variabile può essere assegnata ad:
•classe stessa (variabile statica) utilizzando la dot-notation
•oggetto creato dalla classe utilizzando la notazione a punti.

In Apex, è possibile definire le classi di livello superiore (chiamate anche classi esterne) e le classi interne, ovvero una classe definita all'interno di un'altra classe .
Modificatori di accesso:

- È necessario utilizzare uno dei modificatori di accesso (come public or global) nella dichiarazione di una classe di livello superiore.
- Non è necessario utilizzare un modificatore di accesso nella dichiarazione di una classe interna.

Private access
Il modificatore dichiara che questa classe è conosciuta solo localmente, cioè solo da questa sezione di codice.
Questo è l'accesso predefinito per le classi interne, ovvero, se non si specifica un modificatore di accesso per una classe interna, viene considerato privato.
Questa parola chiave può essere utilizzata solo con classi interne (o con classi di test di livello superiore contrassegnate con l'annotazione @isTest).

Public access
Il modificatore dichiara che questa classe è visibile nell'applicazione o nello spazio dei nomi.

Global access
Il modificatore dichiara che questa classe è conosciuta da tutti i codici Apex ovunque.
Tutte le classi che contengono metodi definiti con la parola chiave webservice devono essere dichiarate globali.
Se un metodo o una classe interna sono dichiarati globali, la classe esterna di livello superiore deve anche essere definita globale.

With sharing e without sharing
Le parole chiave specificano la modalità di condivisione per questa classe.

Virtual
Il modificatore dichiara che questa classe consente l'estensione e l'override.
Non è possibile sovrascrivere un metodo con la parola chiave override a meno che la classe non sia stata definita come virtuale.

Abstract
Il modificatore di definizione dichiara che questa classe contiene metodi astratti, cioè metodi che hanno solo la propria firma dichiarata e nessun corpo definito.

METODI
Metodi di classe
Per definire un metodo, bisogna specificare quanto segue:

Optional
modificatori come public, private,protected, global.

Required:
•il tipo di dati del valore restituito dal metodo, ad esempio String o Integer.
Utilizzare void se il metodo non restituisce un valore.
•un elenco di parametri di input per il metodo,separati da virgole, ciascuno preceduto dal relativo tipo di dati e racchiuso tra parentesi ().
Se non ci sono parametri, usa una serie di parentesi vuote.

Un metodo può avere solo 32 parametri di input.
•il corpo del metodo, racchiuso tra parentesi graffe{}.
Tutto il codice per il metodo, incluse le dichiarazioni delle variabili locali, è contenuto qui.

CONSTRUCTORS

Un costruttore è un metodo che viene invocato quando viene creato un oggetto della classe.
Non è necessario scrivere un costruttore per ogni classe.
Se una classe non ha un costruttore definito dall'utente, viene utilizzato un costruttore pubblico predefinito senza argomento.
La sintassi per un costruttore è simile a un metodo,ma differisce da una definizione di metodo in quanto non ha mai un tipo di ritorno esplicito e non è ereditata dall'oggetto creato da esso. Inoltre ha sempre nome uguale al nome della classe.
Dopo aver scritto il costruttore per una classe, è necessario utilizzare la parola chiave ‘’new’’ per creare un'istanza di un oggetto da quella classe,utilizzando tale costruttore.

STATIC METHODS E VARIABLES

È possibile utilizzare metodi e variabili statici solo con le classi esterne.
Le classi interne non hanno metodi o variabili statici. Un metodo statico o una variabile non richiede un'istanza della classe per poter essere eseguita.
Prima che venga creato un oggetto di una classe,tutte le variabili vengono inizializzate e tutti i blocchi di codice di inizializzazione statici vengono eseguiti.
Un metodo statico viene utilizzato come metodo di utilità e non dipende mai dal valore di una variabile membro di istanza. Poiché un metodo statico è associato solo a una classe, non può accedere ai valori della variabile membro della sua classe.
Una variabile statica è statica solo nell'ambito della transazione Apex.
Non è statico sul server o sull'intera organizzazione. Il valore di una variabile statica persiste nel contesto di una singola transazione e viene ripristinato oltre i limiti della transazione.

STATIC VARIABLES

Le variabili statiche sono variabili che appartengono alla classe stessa, non alle istanze della classe(oggetti).
Per memorizzare le informazioni che sono condivise tra istanze di una classe.
Ad esempio, può essere utilizzato come contatore.

STATIC METHODS

I metodi statici possono essere richiamati direttamente dalla classe, non da un'istanza della classe (oggetto).
In questo modo, non è necessario creare un'istanza di una classe (oggetto) per chiamare un metodo statico.
Un metodo statico non può essere chiamato da un'istanza di classe (oggetto), può essere chiamato solo dalla classe stessa.

INTERFACES

Un'interfaccia è simile a una classe in cui nessuno dei metodi è stato implementato: le firme del metodo sono presenti, ma il corpo di ciascun metodo è vuoto.
Per utilizzare un'interfaccia, un'altra classe deve implementarla fornendo un corpo per tutti i metodi contenuti nell'interfaccia.
Le interfacce possono fornire uno strato di astrazione al codice.
Separano l'implementazione specifica di un metodo dalla dichiarazione per quel metodo.
In questo modo puoi avere diverse implementazioni di un metodo basato sulla tua specifica applicazione.
Definire un'interfaccia è simile alla definizione di una nuova classe.