**1° lezione programmazione II**

**Introduzione alla OOP (Object Oriented Programming)**

**L’importanza dell’astrazione**

l mondo in cui viviamo è costituito da sistemi molto complessi, in cui oggetti diversi interagiscono tra loro e cambiano il loro modo di agire in funzione di quello che accade.

È difficile gestire una realtà complessa, allo stesso modo è difficile costruire sistemi complessi come, ad esempio, software di grandi dimensioni. Un modo per gestire la complessità è **l’astrazione.** L’astrazione è un procedimento che consente di semplificare la realtàche vogliamo modellare. La semplificazione avviene concentrando. l’attenzione solo sugli elementi importanti del sistema complesso chestiamo considerando.

**Cos’è l’astrazione?**

L’astrazione è un procedimento che consente di semplificare la realtà che vogliamo modellare. La semplificazione avviene concentrando. L’attenzione solo sugli elementi importanti del sistema complesso che stiamo considerando. Questa è un concetto fondamentale nella OOP.

**Gli oggetti in un linguaggio OOP forniscono la funzionalità di astrarre, cioè di nascondere i dettagli implementativi interni**. Quando si creano dei programmi mediante un linguaggio ad oggetti, **la capacità di astrarre, cioè la capacità di semplificare delle entità complesse in oggetti caratterizzati dalle caratteristiche e dalle funzionalità essenziali per gli scopi preposti, può risultare determinante.**

**Linguaggi di programmazione**

Si utilizzano per codificare gli algoritmi. Un linguaggio di programmazione è  un insieme di istruzioni che vengono utilizzate per codificare programmi. Distinguiamo due tipologia di linguaggi di programmazione:

-   **Linguaggi a basso livello:**

o  **Linguaggio macchina:**

o  **Assembly:** è molto simile al linguaggi macchina con la differenza che si utilizzano forme mnemoniche per le corrispondenti istruzioni macchina.

-   **Linguaggi a alto livello(si avvicinano al nostro modo di parlare):** ha i seguenti vantaggi:

· **astrazione**

· **semplificazione**

· **similarità con il ragionamento umano**

**Paradigmi di programmazione**

**I paradigmi sono la filosofia con cui si scrivono i programmi.**

Un linguaggio di programmazione è basato su un paradigma di programmazione che è caratterizzato da:

-   **metodologia** con cui si scrivono i programmi

-   **computazione**  istruzioni o funzioni matematiche

Esistono svariati paradigmi di programmazione come:

-   **programmazione funzionale:** è costituita da valutazione funzione matematiche. Si ha un assenza di side-effect perché non si fa uso di istruzioni imperative (non si posso fare modifiche alla memoria)

-   **programmazione logica:** descrive la struttura logica del programma (logica del primo ordine)

-   **programmazione imperativa:** sequenza di istruzioni

-   **programmazione strutturata :** è basata sul teorema di Bohm Jacobini (**tre costrutti sequenza, selezione e iterazione)**.L'istruzione  Go to permette di fare dei salti in alcune parti del programma.

-   **programmazione procedurale:** blocchi di codice sorgente identificati da un nome ed eventuali argomenti.

-   **programmazione modulare**:  strutturare un programma in moduli ben definiti. mettendo insieme queste funzioni si può pensare di aggregarle logicamente in un modulo. Il modulo è fatto da un certo set di funzioni, alcune di queste saranno dedicate al funzionamento interno del  modulo quindi alle computazioni che avvengono all'interno che non sono visibili all'utente finale. Altre di queste funzioni costituiranno la cosiddetta interfaccia del modulo sono funzioni che si potranno invocare esplicitamente attraverso le quali potremmo attivare certe computazioni.

-   **programmazione orientata agli oggetti(OOP):**Introduce una serie di costrutti. la classe è una estrazione della realtà invece l'oggetto è un'istanza della classe. I metodi sono le funzionalità dell'oggetto. con la programmazione ad oggetti sia un legame tra i  dati e i metodi che vengono incapsulate nelle classi. Si ha anche la possibilità di avere dei metodi che non sono visibili all'utente cosiddetti metodi privati che servono solo per l'implementazione dell'oggetto.

Immagine che contiene testo, schermata, Carattere, design

Descrizione generata automaticamente

**I linguaggi di programmazione sono definiti da un insieme di regole formali cioè la sintassi che include tutti i simboli da utilizzare durante la scrittura del programma. Le regole di sintassi definiscono come comporre i simboli e le parole per comporre istruzioni corrette. La semantica di un'istruzione è il significato di questa.  attenzione!!  se in un programma non ci sono errori sintattici non vuol dire che la semantica del programma.**

**Traduzione**Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Grazie al concetto di traduzione si ha:

* **evoluzione del linguaggio di programmazione**( Maggiormente espressivi)
* **facilità nella scrittura del programma** perché sono vicini al linguaggio parlato
* **Indipendenza dalla piattaforma:** Il programmatore che sta programmando in c non si deve preoccupare della piattaforma(CPU).  Questo problema se la deve porre chi scrive e progetta i compilatori  per le varie CPU.

**si hanno due tipi di traduttori:**

1. **L'interpretazione** traduce ed esegue ogni singola istruzione del programma. Legge ed esegue il codice sorgente del programma senza creare un file oggetto eseguibile. E' più lenta rispetto alla compilazione**.**
2. **compilatore:**  il compilatore prende in pasto un programma scritto con un linguaggio alto livello e lo traduce in linguaggio macchina. Il compilatore opera su una architettura che si dice architettura " target ” se su una macchina si installa il compilatore e si va a compilare un programma si ottiene un programma eseguibile per qualunque altra macchina che abbia la stessa architettura.  Se invece il programmatore Dovesse compilare questo programma per architetture diverse dall'architettura della macchina di sviluppo, allora Dovrebbe effettuare una procedura  chiamata cross compilazione.

Nel processo di traduzione del compilatore si hanno almeno due parti:

1. **back end**:  è la parte che cambia da architettura ad architettura e si occupa di generare  istruzioni macchina
2. **front end**: ha la funzione di eseguire l'analisi sintattica del programma di fare il parsing.

I linguaggi interpretati e compilati hanno dei vantaggi degli svantaggi:

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

Nei linguaggi interpretati il programmatore **non si deve preoccupare nell'architettura su cui verrà eseguito il programma.** infatti il programmatore scrive il programma in Java la compilazione produce un **programma in byte code**  che è un linguaggio simile al linguaggio macchina ma è indipendente dall'architettura e **il  byte code  viene eseguito mediante un interprete Java installato su una macchina.**

**JVM  è la macchina virtuale che interpreta il codice Byte code**

**Evoluzioni dei linguaggi di programmazione**

**I linguaggi hanno avuto un’evoluzione a partire dagli anni’50**: **i linguaggi macchina sono stati abbandonati per utilizzare i linguaggi più vicini al linguaggio naturale. Questo sviluppo è stato possibile grazie al processo di astrazione.**

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

**Programmazione procedurale VS OOP**

Nello sviluppo software, usando **la metodologia della programmazione procedurale**, **l’interesse principale è rivolto alla sequenza di operazioni da svolgere**: **si crea un modello indicando le procedure da eseguire in maniera sequenziale per arrivare alla soluzione.** Lo spostamento di attenzione dalle procedure agli oggetti ha portato all’introduzione della programmazione ad oggetti. **Gli oggetti sono intesi come entità che hanno un loro stato e che possono eseguire certe operazioni.**

* **Un algoritmo è un insieme di istruzioni che a partire dai dati di input permettono di ottenere i risultati di output. Un algoritmo deve essere riproducibile, deve avere una durata finita e non deve essere ambiguo. Il modo di programmare pone attenzione sulla sequenza di esecuzione**
* **Un sistema è una parte del mondo che si sceglie di considerare come un intero, composto da componenti. Ogni componente è caratterizzata da proprietà rilevanti, e da azioni che creano interazioni tra le proprietà e le altre componenti.**

**Quali sono gli svantaggi nell’utilizzo del paradigma procedurale**

* + I linguaggi procedurali hanno dei limiti nel creare componenti software riutilizzabili.
  + I programmi sono fatti da funzioni, che rappresentano codice riutilizzabile, ma che spesso fanno riferimento a headers e/o variabili globali che devono essere importate insieme al codice delle funzioni.
  + I linguaggi procedurali non si prestano bene alla modellazione di concetti ad alti livelli di astrazione, utili per rappresentare entità complesse che interagiscono in un sistema reale.

**I linguaggi procedurali separano le strutture dati e gli algoritmi**

**Quali sono le fasi per programmare ad oggetti?**

Si hanno 3 fasi:

1. **Analisi (OOA):** identificazione dei requisiti funzionali, delle classi e delle loro relazioni logiche.
2. **Design (OOD):** specifica delle gerarchie tra classi, e delle loro interfacce e comportamenti.
3. **Programmazione (OOP):** implementazione del design, test ed integrazione(fase di scrittura effettiva del codice).

**Cos’è la modalità OO**

**La metodologia OO è un modo di pensare al problema in termini di sistema, quindi parte dall’analisi del problema e dalla progettazione della sua soluzione.**

**Durante la fase di analisi si crea un modello del sistema, individuando gli elementi di cui è formato e i comportamenti che devono avere.**

In questa fase **non interessano le modalità con le quali i comportamenti vengono implementat**i, **ma soltanto gli oggetti che compongono il sistema e le interazioni tra essi.**

**Qual è l’elemento base della OOP?**

**L’ elemento base della OOP è l’ oggetto. Questo contiene al suo interno:**

* **Le caratteristiche o attributi**: rappresentano gli elementi che caratterizzano l’oggetto, utili per descrivere le sue proprietà e definirne lo stato.
* **I comportamenti o metodi:** rappresentano le funzionalità che l’oggetto mette a disposizione: chi intende utilizzare l’oggetto deve attivare i comportamenti dell’oggetto stesso.

Una classe incapsula sia le caratteristiche (attributi) sia i comportamenti (metodi) degli oggetti che rappresenta. Inoltre, fornisce una interfaccia pubblica per

poter interagire con gli oggetti definiti dalla classe.

**Da cosa è caratterizzato un programma ad oggetti?**

Un programma ad oggetti è caratterizzato dalla presenza di tanti oggetti che comunicano e interagiscono tra loro **(modello di sistema).**

Nella OOP l’interazione tra oggetti avviene con un meccanismo chiamato **scambio di messagg**i. **Un oggetto, inviando un messaggio ad un altro oggetto,**

**può richiederne l’esecuzione di un metodo.**

**Da cosa è costituito un messaggio?**

* + **Un destinatario**: l’oggetto verso il quale il messaggio è indirizzato
  + **Un selettore**: identifica il metodo che si vuole attivare
  + **Un elenco di argomenti**: insieme dei parametri che vengono passati all’oggetto quando si richiede l’attivazione del metodo.

Quali sono le caratteristiche della promozione ad oggetti

* 1. Incapsulamento
  2. **Interfaccia di un oggetto**
  3. **Classe**
  4. **Ereditarietà**
  5. **Polimorfismo**
  6. **Collegamento dinamico**

Cos’è l’incapsulamento?

L’incassi mento indica la proprietà degli oggetti di incorporare al loro interno sia attributi che metodi.

Si dice che gli attributi e i metodi sono incapsulati nell’oggetto. In questo modo tutte le informazioni utili che riguardano oggetto sono ben localizzate.

L’incazzamento non va confuso con information giding che consiste nel nascondere all’esterno i dettagli implementati di un oggetto rendendo i metodi e o gli attributi privati.

**Le classi permettono di definire dei tipi di dati astratti TDA.**

**Cos’è una sezione pubblica e una sezione privata?**

I linguaggi OO permettono di definire una sezione **pubblica e una sezione privata** dall’interno dell’oggetto a queste possono fare parte sia i metodi che gli attributi.

* + Sezione pubblica: in questa vengono messi attributi e i metodi che si vogliono rendere visibili all’esterno e quindi utilizzabile dagli altri oggetti. Sia in questo modo l’interfaccia.
  + Sezione privata: ci sono gli attributi e i metodi che non sono accessibili e che vengono usati solo internamente all’oggetto per implementare i suoi comportamenti.

**Riassumendo quindi un oggetto è un’istanza di una classe e una classe e l’astrazione della realtà.**

**Come si rappresentano le classi graficamente?**

**Le classi si rappresentano con un diagramma di classi UML dove si vanno a specificare gli attributi metodi della classe e anche il loro ambito divisibilità (pubblico, privato, protetto).**

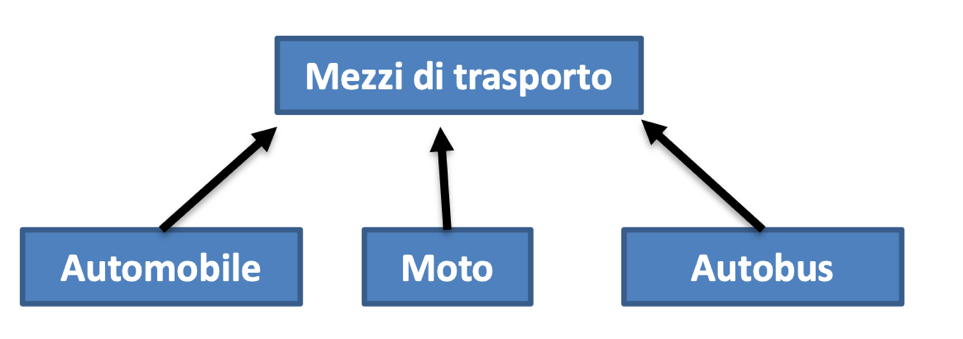
**Cos’è l’ereditarietà?**

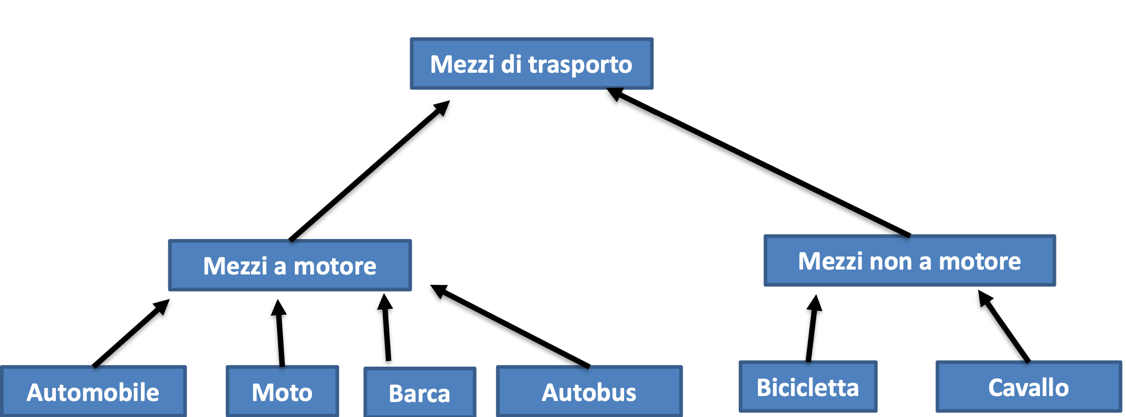
È lo strumento che permette di costruire nuove classi utilizzando quelle già sviluppate. Quando una classe viene creata in questo modo, riceve tutti gli attributi ed i metodi della classe generatrice (li eredita). La classe generata sarà quindi costituita da tutti gli attributi e i metodi della classe generatrice più tutti quelli nuovi che saranno definiti.

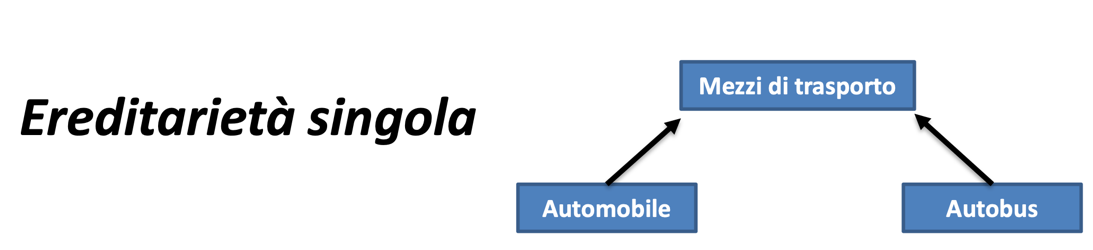
**La classe che è stata derivata** prende il nome di **sottoclasse**, **mentre la classe generatrice si chiama sopra classe.** Queste relazioni individuano una gerarchia che si puòdescrivere usando un grafo di gerarchia.

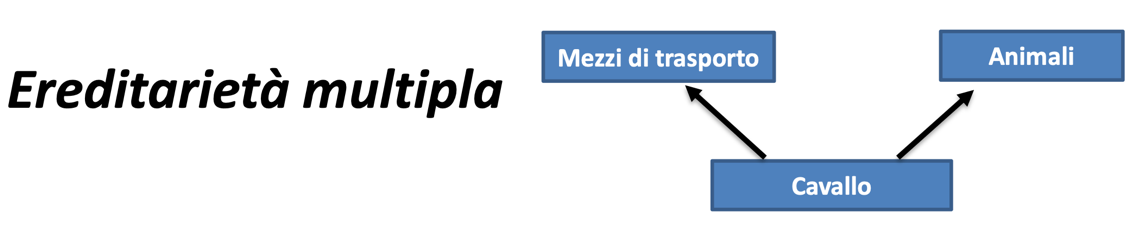
La nuova classe si differenzia dalla sopraclasse in due modi:

* + Per estensione: aggiungendo nuovi attributi e metodi
  + Per ridefinizione: modificando i metodi ereditati,specificando una implementazione diversa di un metodo (override, overload)









**Relazioni d’uso**

* **Association (relazione d’uso)**: Diciamo che una classe A utilizza una classe B se un oggetto di classe A è in grado di inviare dei messaggi ad un oggetto di classe B oppure se un oggetto di classe A può creare, ricevere o restituire oggetti di classe B
* **Aggreagation (relazioni di contenimento)**: Un oggetto di classe A contiene un oggetto di classe B se B è una proprietà (attributo) di A.

L’aggregazione è una forma di associazione più forte: una classe ne aggrega un’altra se esiste tra le due classi una relazione di tipo “intero-parte

* **Composizione:** La composizione è una forma di aggregazione ancora più forte che indica che una ‘’parte’’ può appartenere ad un solo ‘’intero’’ in un certo istante di tempo.

**Cos’è il polimorfismo?**

Indica la possibilità per i metodi di assumere forme, cioè implementazioni, diverse all’interno della gerarchia delle classi.

**Cos’è il collegamento dinamico?**

È lo strumento utilizzato per la realizzazione del polimorfismo. È dinamico perché l’associazione tra l’oggetto e il metodo corretto da eseguire è effettuata a run-time, cioè durante l’esecuzione del programma.