Quinta lezione

23 March 2022 15:39

Abbiamo tre tipi di sezioni: private, public e protected

- **Costruttore:** serve per allocare la memoria per tutti i suoi membri (dati e funzioni) nello heap o nello stack, andando anche ad inizializzarli. Il nome del costruttore coincide con quello della classe e non ha un valore di ritorno. Se non accetta argomenti oppure sono opzionali allora viene detto di default (ha dei limiti: il valore predefinito per il tipo di ogni dato membro può dipendere dal compilatore utilizzato) e se la definizione della classe non include costruttori allora vengono generati in automatico dal compilatore. L'uso di un costruttore definito, consente di inizializzare tutti i membri della classe assieme, di modo che l'istanziazione di un oggetto sia sempre consistente con il nostro modello di dati e indipendente dalle scelte del compilatore.
- **Distruttore:** il nome del distruttore coincide con quello della classe preceduto dal carattere ~. Non ha valore di ritorno e nemmeno i parametri, infatti non può essere sovraccaricato. E' una funzione come le altre, in cui può essere eseguita qualsiasi istruzione. Il suo compito primario sarebbe quello di rimuovere un oggetto e tutte le sue dipendenze dallo stato del programma in maniera sicura e completa.
  Viene invocato per tutte le variabili automatiche ogni volta che viene raggiunta la fine del loro ambito di visibilità e tutte le volte che le variabili dinamiche vengono deallocate con la parola chiave delete. Se non viene esplicitamente incluso nella dichiarazione della classe, il distruttore viene generato automaticamente dal compilatore; in questo caso tuttavia, l'unica risorsa rilasciata è la memoria occupata dall'oggetto. Se alcuni dei suoi dati membro sono delle reference, o se l'oggetto fa uso di altre risorse potrebbe essere necessario svolgere delle operazioni per il loro corretto rilascio.

1. Nel 99,9% dei casi NON PUO' ESSERE PRIVATO, l'unica volta che accade è il suo uso all'interno di un costruttore della stessa classe resa public.
2. Nel 99,9% dei casi non bisogna chiamarlo perché è presente quando si dealloca (tranne se è presente il delete)

CLASSE, OGGETTO E TEMPLATE

ClassName : [virtual/private/protected/public] Base ClassName {…}

di default — andare ad estendere una classe che mi serve nell'intestazione — tutto ciò che è pubblico sarà visibile al mondo

Metodi:

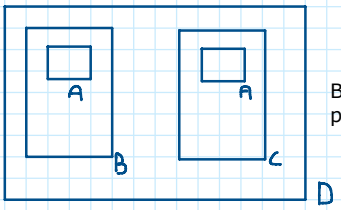
- Riprendono virtual, private, protected e public
- [virtual] type NameFun (lista parametri) [const] [noexcept][ovveride][= assignement] **[]=valore opzionale**

1. [const]= il metodo non modifica la struttura dati su cui agisce
2. [noexcept]= non lancia eccezioni
3. [ovveride]= il compilatore fa ovveride. Serve per far capire al compilatore se c'è un errore o no
4. [= assignement] è pseudo

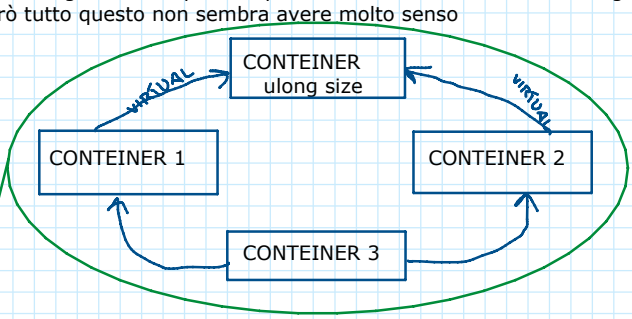
- Uguale a 0: funzione virtuale pura, non da implementazioni e si utilizza quando si devono progettare le interfacce
- Uguale al valore di default: costruttori e distruttori di default soprattutto quando ho variabili dinamiche
- Uguale al delete: se ho una classe astratta A e due classi B,C che ereditano A. Ho il metodo F, alloco un puntatore di tipo A ed accedo una volta a B ed una a C (questa cosa non ha senso perché si chiama F tramite A).

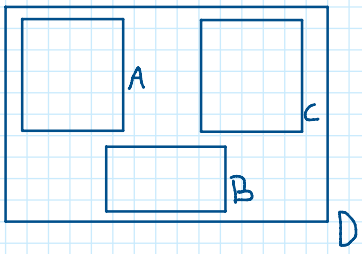
NON HA SENSO FARE I PARAGONI SULLE CLASSE ASTRATTE

Interfaccia Java simile alla Classe Astratta Pura (con almeno un metodo virtuale puro)

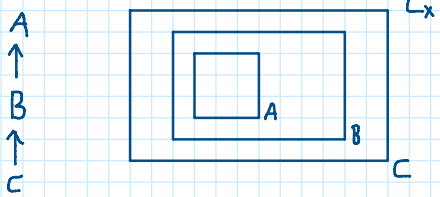


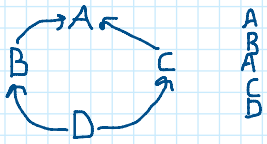
B ha bisogno della copia di A per funzionare cosi come C ha bisogno della copia di A per funzionare però tutto questo non sembra avere molto senso



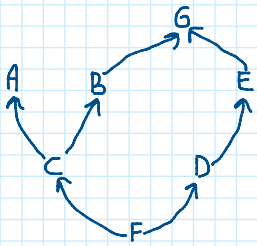


Supponiamo di avere:





A
B
A
C
D



A
G
B
C
G
B
G
E
D
F