Ereditarietà

Alberto Ferrari

Ereditarietà

- Una nuova classe (**classe derivata**) viene creata a partire da una classe esistente (**classe base**)
- La classe derivata **eredita** le variabili membro e le funzioni membro della classe base
- La classe derivata può **aggiungere** variabili membro e funzioni membro

Ridefinizione delle Funzioni Membro Ereditate

- Una classe derivata può cambiare la definizione di una funzione membro ereditata
- In questo caso la definizione della classe derivata deve contenere la dichiarazione della funzione membro ereditata
- Possiamo avere ereditarietà per
  - estensione
    (aggiunta di nuove variabili e/o funzioni)
  - ridefinizione
    (overloading di funzioni)

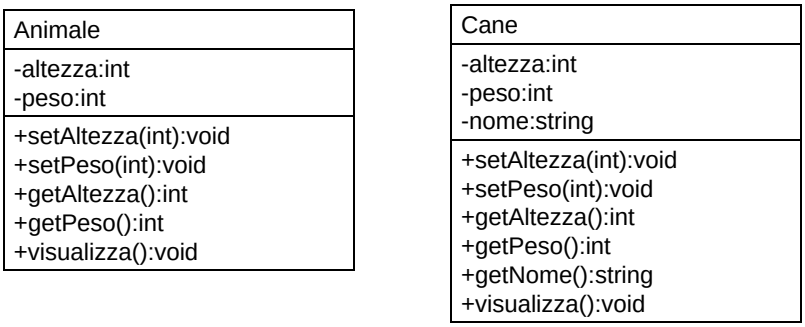
Alberto Ferrari

Ereditarietà e Riuso

- La classe base contiene il codice comune alle classi derivate
- L'ereditarietà consente di riusare il codice della classe base

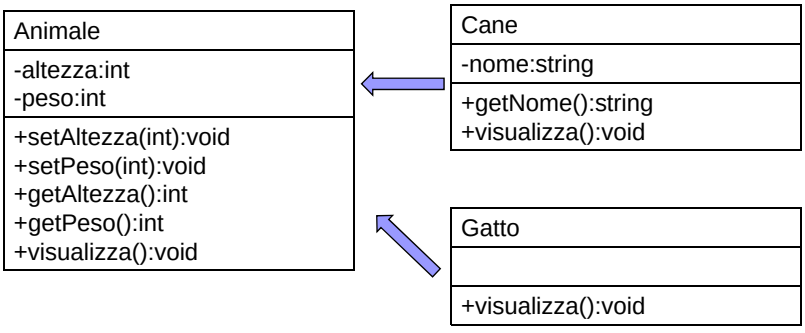
Alberto Ferrari

Un esempio senza ereditarietà



Alberto Ferrari

Un esempio con ereditarietà



Alberto Ferrari

Le classi della gerarchia in C++ Animale

```
class Animale
{
public:  Animale( const int = 0, const int = 0 );
    void setAltezza( int );
    int getAltezza();
    void setPeso( int );
    int getPeso();
    void visualizza();
private:
    int altezza;
    int peso;
};
```

Alberto Ferrari

Le classi della gerarchia in C++ Cane

```
class Cane: public Animale
{
public:
    Cane( const int = 0 , const int = 0, string = "Bill");
    void visualizza();
    void setNome( string );
private:
    string nome;
};
-------------------------------------------------------------
Cane::Cane( const int a, const int p, string n ): Animale( a, p)
{
   setNome( n );
} …
void Cane::visualizza()
{  cout << "Sono un cane di nome: " << nome << endl;
   Animale::visualizza();
}
```

Alberto Ferrari

Le classi della gerarchia in C++ Gatto

```
class Gatto: public Animale
{
public:
    Gatto ( const int = 0, const int = 0);
    void visualizza();
};

Gatto::Gatto( const int a, const int p)  : Animale(a, p)
{}
void Gatto::visualizza()
{
   cout << "Sono un gatto";
   Animale::visualizza();
}
```

Alberto Ferrari

Costruttori nelle Classi Derivate

- Un costruttore della classe base non viene ereditato
- Può essere invocato nella definizione del costruttore della classe derivata per inizializzare le variabili ereditate
- Se non è invocato, il costruttore di default della classe base viene invocato automaticamente

Alberto Ferrari

Ordine di Chiamata dei Costruttori

- La chiamata del costruttore della classe base è la prima azione del costruttore della classe derivata
- Se A->B->C quando viene creato un oggetto di classe C prima viene chiamato un costruttore della classe A, poi un costruttore della classe B, poi vengono intraprese le rimanenti azioni del costruttore di classe C

Alberto Ferrari

Uso dei Membri Privati della Classe Base

- I membri privati della classe base non sono referenziabili nelle definizioni delle funzioni membro della classe derivata
- Diversamente verrebbe violato il principio di incapsulamento
- Le funzioni membro della classe derivata possono accedere alle variabili membro private della classe base tramite le funzioni accessor e mutator
- Le funzioni membro private della classe base non sono accessibili (di fatto non sono ereditate)

Alberto Ferrari

Il Qualificatore protected

- Una variabile o funzione membro qualificata come **protected** può essere referenziata nelle funzioni membro di una classe derivata
- Le variabili membro **protected** agiscono come se fossero **protected** in ogni classe derivata
- Molti ritengono che l'uso di variabili membro **protected** comprometta l'incapsulamento
- E' quindi buona norma utilizzare **protected** solo quando assolutamente necessario

Alberto Ferrari

Ridefinizione (Overriding) e Sovraccarico (Overloading)

- Una funzione **ridefinita** in una classe derivata ha lo stesso numero e tipo di parametri della funzione della classe base
- Una funzione **sovraccaricata** in una classe derivata ha un diverso numero e/o tipo di parametri rispetto alla funzione della classe base e la classe derivata ha entrambe le funzioni

Alberto Ferrari

Accesso a una Funzione della Classe Base Ridefinita

- Una classe derivata può ridefinire una funzione della classe base
- È comunque possibile invocare su un oggetto della classe derivata la versione della funzione data nella classe base
- Per fare ciò si utilizza l'operatore ::, che in questo caso è **obbligatorio**, altrimenti la funzione chiamante continuerebbe in realtà a chiamare se stessa generando un loop

Alberto Ferrari

Relazione "is a"

- Un oggetto di una classe derivata può essere usato ovunque può essere usato un oggetto della classe base
- Un oggetto di una classe derivata ha più di un tipo
- Cane **is a** Animale

Alberto Ferrari

Funzioni che non Vengono Ereditate

- Oltre alle funzioni membro private non vengono ereditati
  - Costruttori
  - Distruttori
  - Costruttori di copia
  - Operatori di assegnamento
- Se non vengono definiti vengono creati quelli di default

Alberto Ferrari

Operatori di Assegnamento in Classi Derivate

- Una classe derivata quando sovraccarica l'operatore di assegnamento può fare uso dell'operatore di assegnamento della classe base

```
Derived& Derived::operator =(const Derived& rtside)
{
  Base::operator =(rtSide)
   …
}
```

Alberto Ferrari

Costruttori di Copia in Classi Derivate

- Una classe derivata quando sovraccarica il costruttore di copia può fare uso del costruttore di copia della classe base

```
Derived::Derived(const Derived& obj) :
  Base(obj), …
{
  …
}
```

Alberto Ferrari

Distruttori in Classi Derivate

- Quando il distruttore di una classe derivata è invocato, viene invocato automaticamente il distruttore della classe base
- Se A->B->C, quando termina lo scope di un oggetto di classe C viene chiamato prima il distruttore della classe C, poi quello della classe B, infine quello della classe A
- I distruttori sono chiamati in ordine inverso rispetto ai costruttori

Alberto Ferrari

Relazioni tra Oggetti

- Relazione “is a”
  - Esempio: un Gatto **is a** Animale
- Relazione “has a”
  - Esempio: an Computer **has a** Processore

Alberto Ferrari

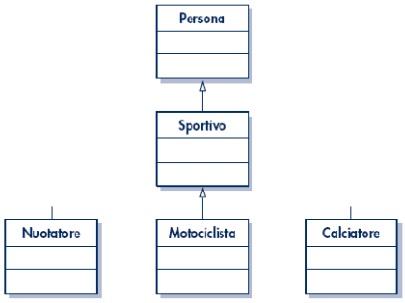
Ereditarietà Protetta e Privata

- Ereditarietà protetta: i membri pubblici della classe base sono protetti nella classe derivata quando sono ereditati
- Ereditarietà privata: nessun membro della classe base può essere referenziato nella classe derivata
  - La relazione “is a” non è valida
  - Sono raramente usate

Alberto Ferrari

Gerarchia di classi

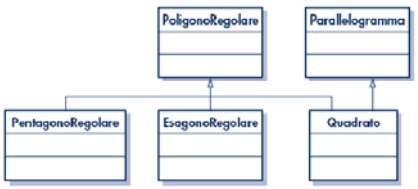
- L'ereditarietà può estendersi a più livelli generando quindi una **gerarchia di classi**.
- Una classe derivata può, a sua volta, essere base di nuove sottoclassi.
- Sportivo è sottoclasse di Persona ed è superclasse di Nuotatore, Motociclista e Calciatore.
- Nella parte alta della gerarchia troviamo le **classi generiche**, scendendo aumenta il **livello di specializzazione**.



Alberto Ferrari

Ereditarietà Multipla

- Una classe derivata può avere più di una classe base
- Possono esserci situazioni ambigue
- Richiede una conoscenza approfondita del linguaggio
- In alcuni linguaggi (es Java) non è ammessa l'ereditarietà multipla



Alberto Ferrari