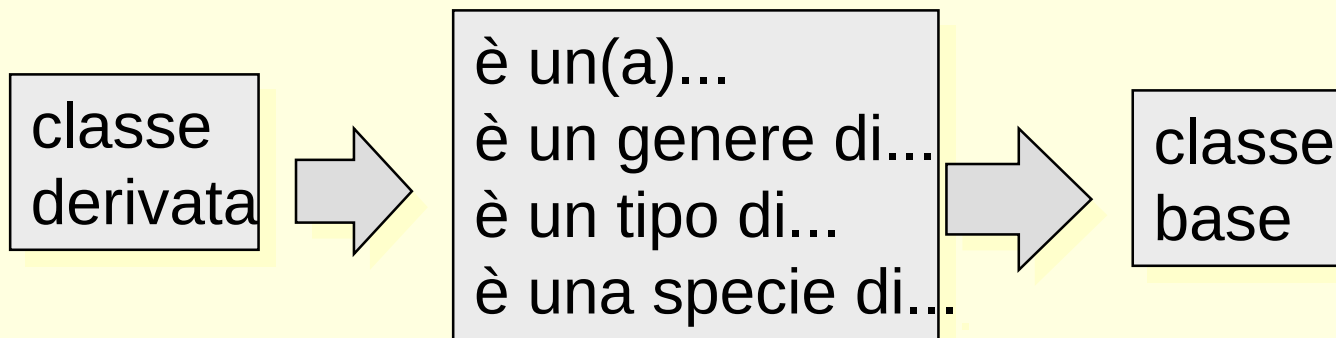


Ereditarietà

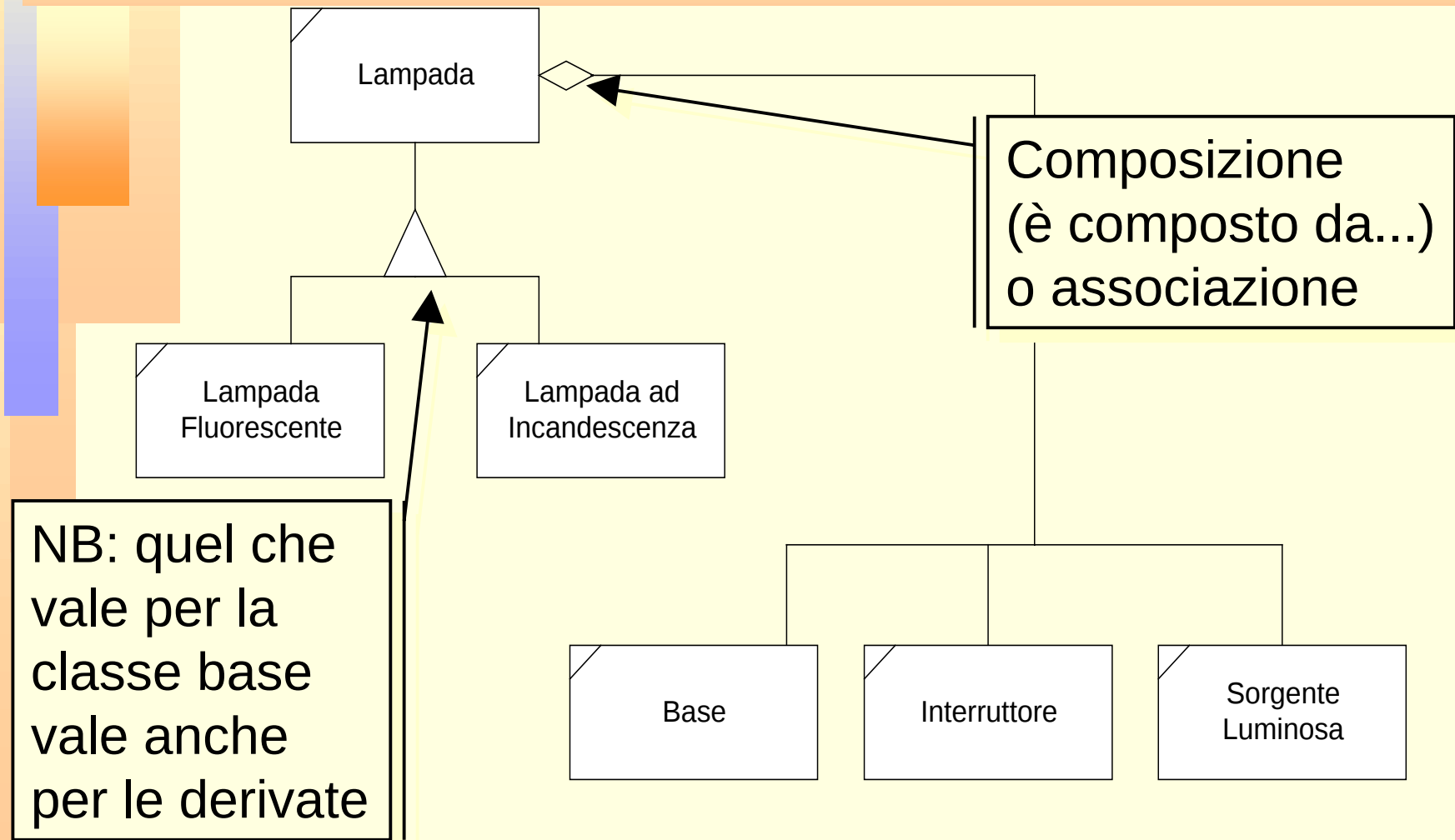
- E' uno dei concetti chiave delle tecniche orientate agli oggetti
- Esprime le somiglianze tra le classi, semplificando la definizione di una classe e riducendola a una o più classi simili
- Rappresenta la generalizzazione e la specializzazione di classi
- Unisce le classi in gerarchie o grafi

Ereditarietà

- Diverse terminologie:
 - superclasse/sottoclasse, classe base/classe derivata
- L'ereditarietà è supportata a livello di linguaggio (C++, Java, Objective-C, Smalltalk...)



Esempio



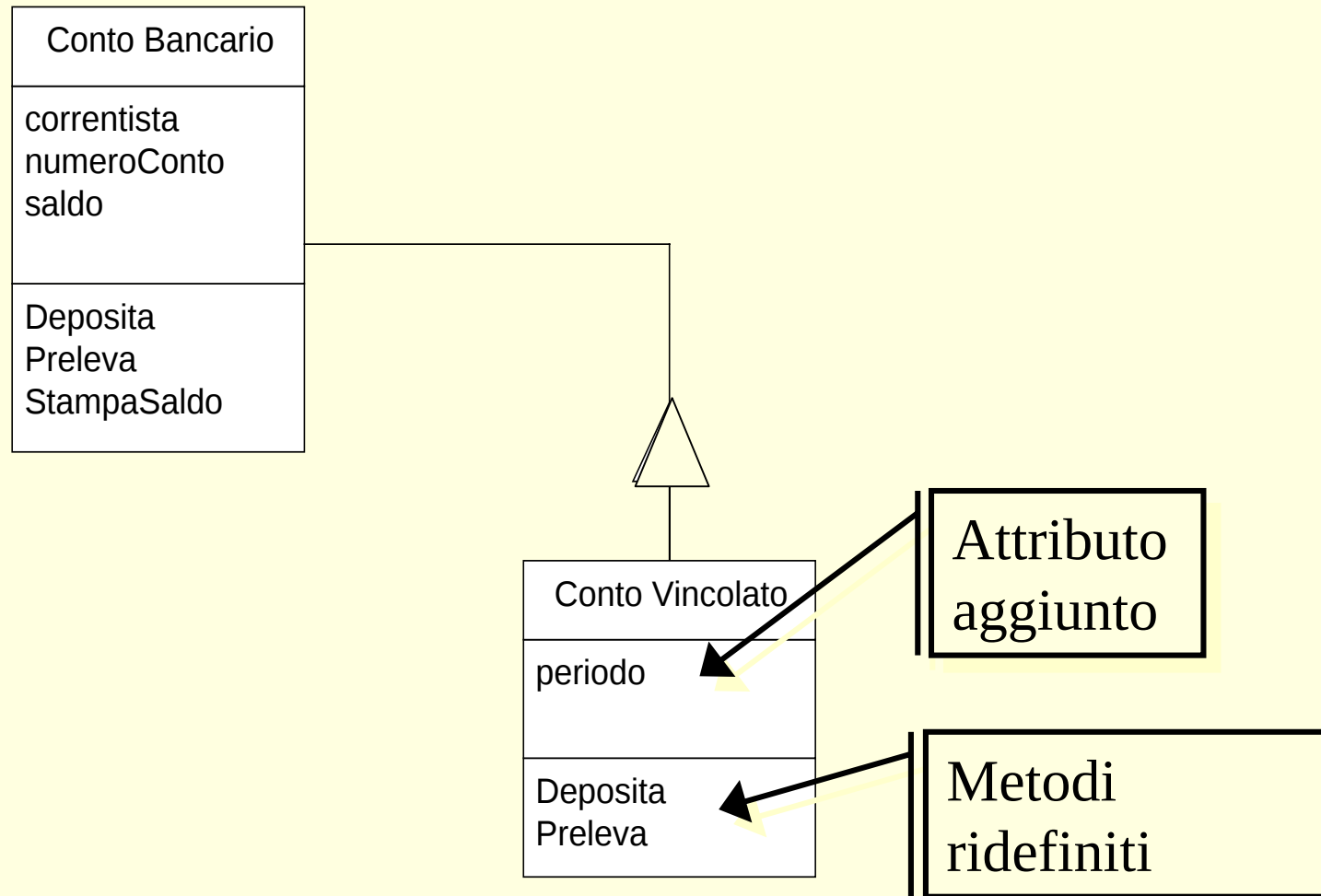
Principio di sostituibilità

- Una classe derivata è utilizzabile in ogni contesto in cui è utilizzabile la classe base
 - Tutte le associazioni valide per la classe base sono valide per le derivate
 - Tutte le operazioni (messaggi) possibili sulla classe base sono possibili sulle derivate
 - Tutti gli attributi della classe base sono attributi delle classi derivate
- Assolutamente non vale il contrario!

Trasmissione dei metodi e attributi

- Tramite una associazione di ereditarietà gli attributi e i metodi della classe base sono automaticamente trasferiti alla classe derivata
- Si possono aggiungere altri attributi e/o ridefinire i metodi (overloading), in modo da specializzare la classe derivata rispetto alla classe base

Esempio



Programming by Difference

- Usando l'ereditarietà è possibile sviluppare nuove parti di un programma “per differenza” rispetto a quelle precedentemente già costruite
- Il costo di ogni modifica è minore rispetto al costo di uno sviluppo da zero
- Purtroppo, in molte situazioni reali è necessaria anche una buona dose di esperienza... il meccanismo da solo non basta...

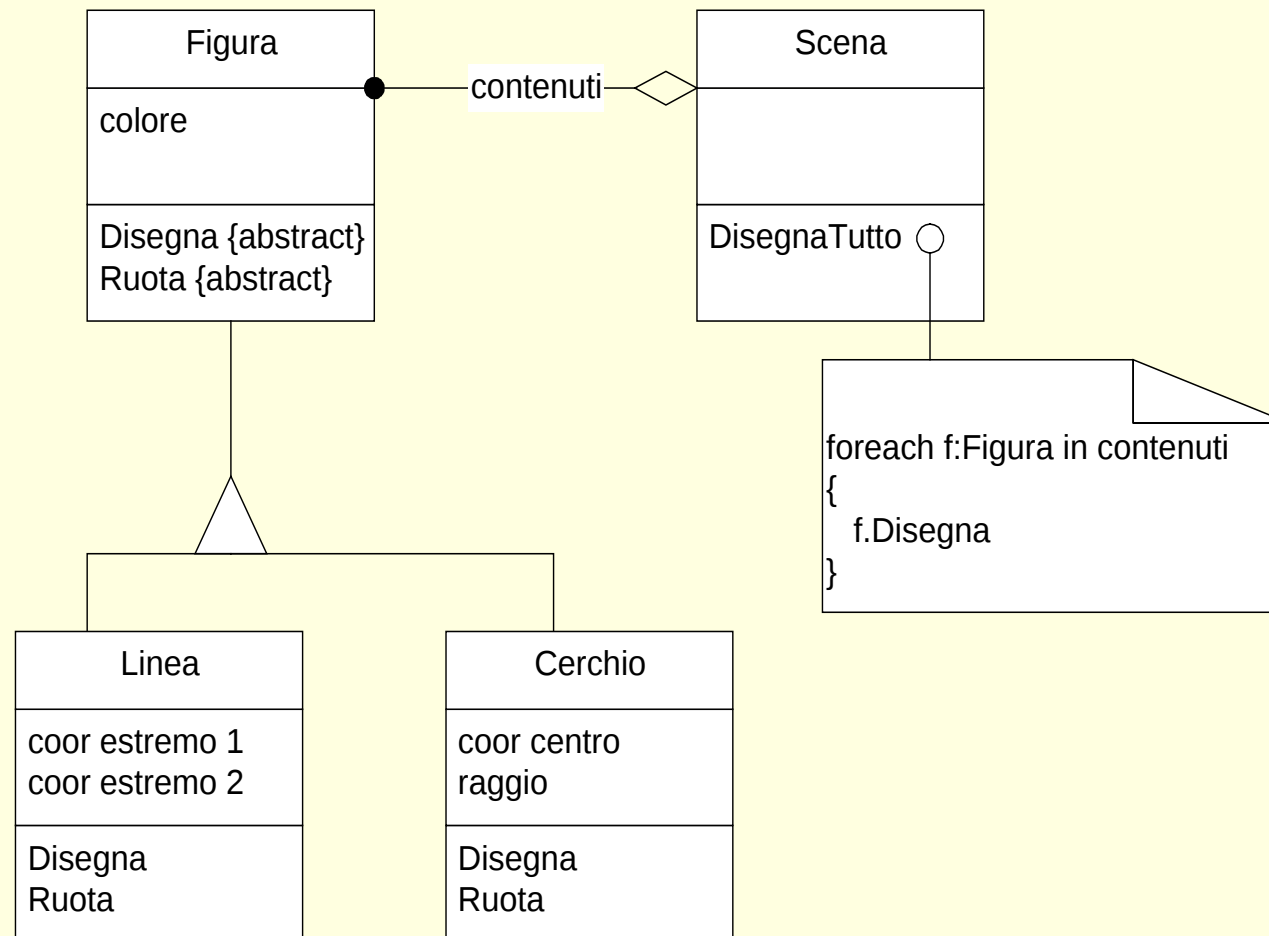
Classi e metodi astratti

- Alcuni metodi in una classe possono essere dichiarati astratti: questo significa che sono vuoti, senza comportamento associato
 - La classe viene allora chiamata classe astratta
 - Una classe astratta viene utilizzata come classe base e i metodi sono sostituiti con metodi “concreti”
 - La classe astratta è una generalizzazione in cui non sono definiti i dettagli - le classi che la specializzano avranno il compito di definirli

Classi e metodi astratti

- Le classi astratte non possono essere utilizzate per istanziare oggetti
 - il compilatore non sarebbe in grado di attivare i metodi dichiarati ma non definiti.
- Le classi derivate da una classe astratta possono essere utilizzate per istanziare oggetti (purché definiscano i metodi astratti)

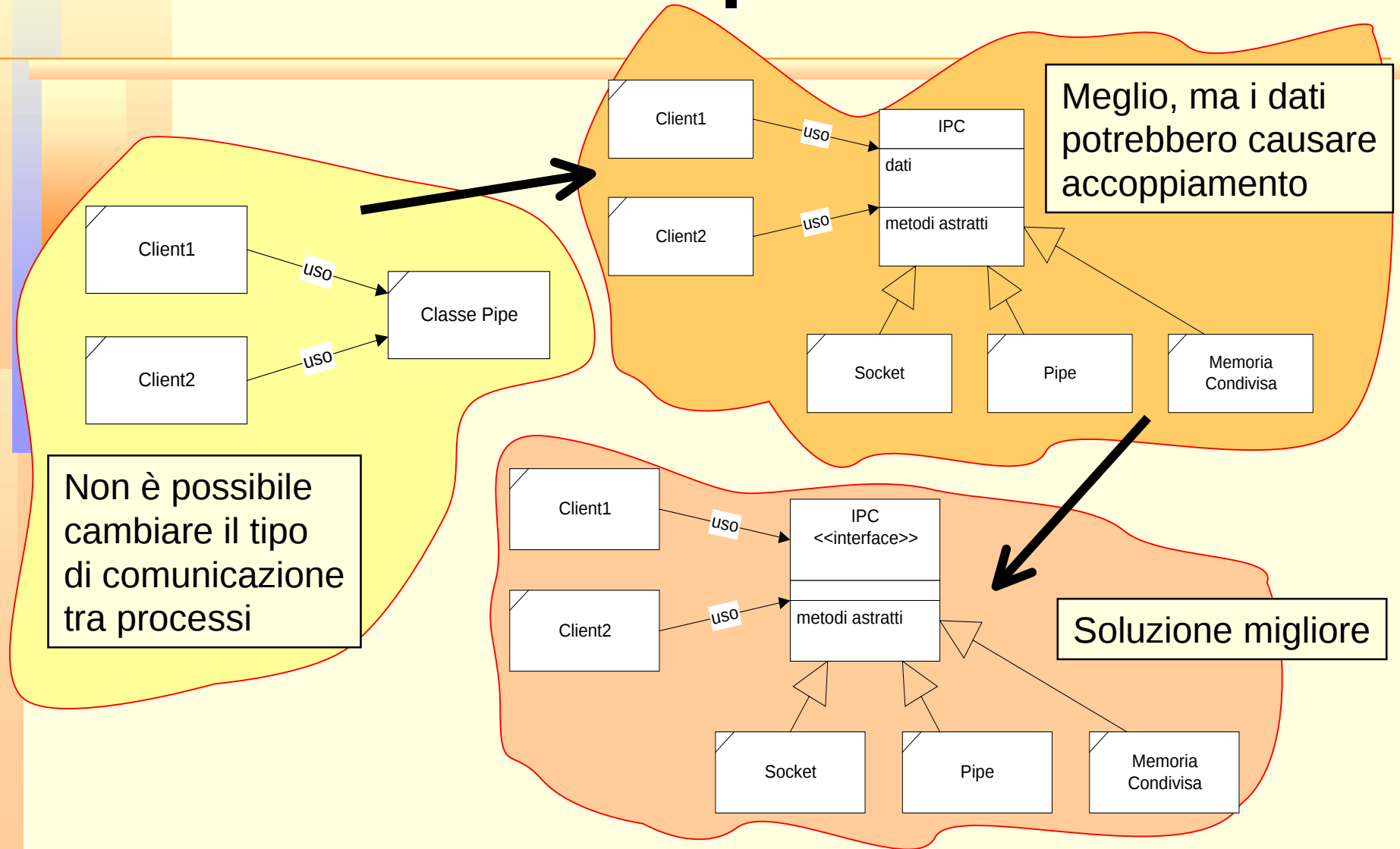
Esempio



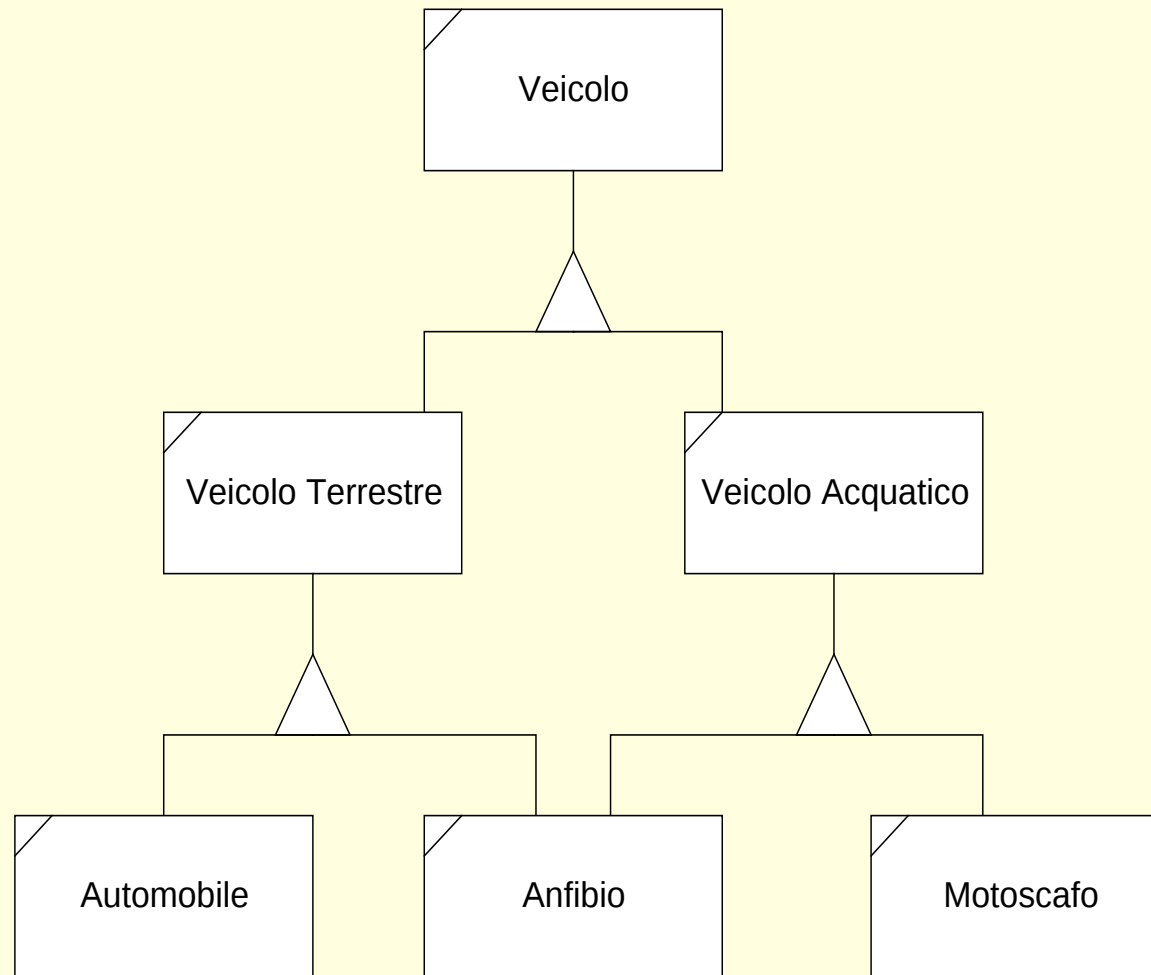
Interfacce

- Una interfaccia è un insieme di funzioni di cui viene specificata l'intestazione (nome, parametri, valori di ritorno) senza specificare l'implementazione
- Portano all'estremo il concetto di metodo e classe astratta
- Sono molto utili per “disaccoppiare” parti diverse di un sistema software, grazie all'assenza di codice o dati
- In C++ non sono supportate direttamente, ma basta definire classi astratte contenenti solo metodi astratti
- Le interfacce sono alla base delle piattaforme per oggetti distribuiti (COM, CORBA)

Esempio



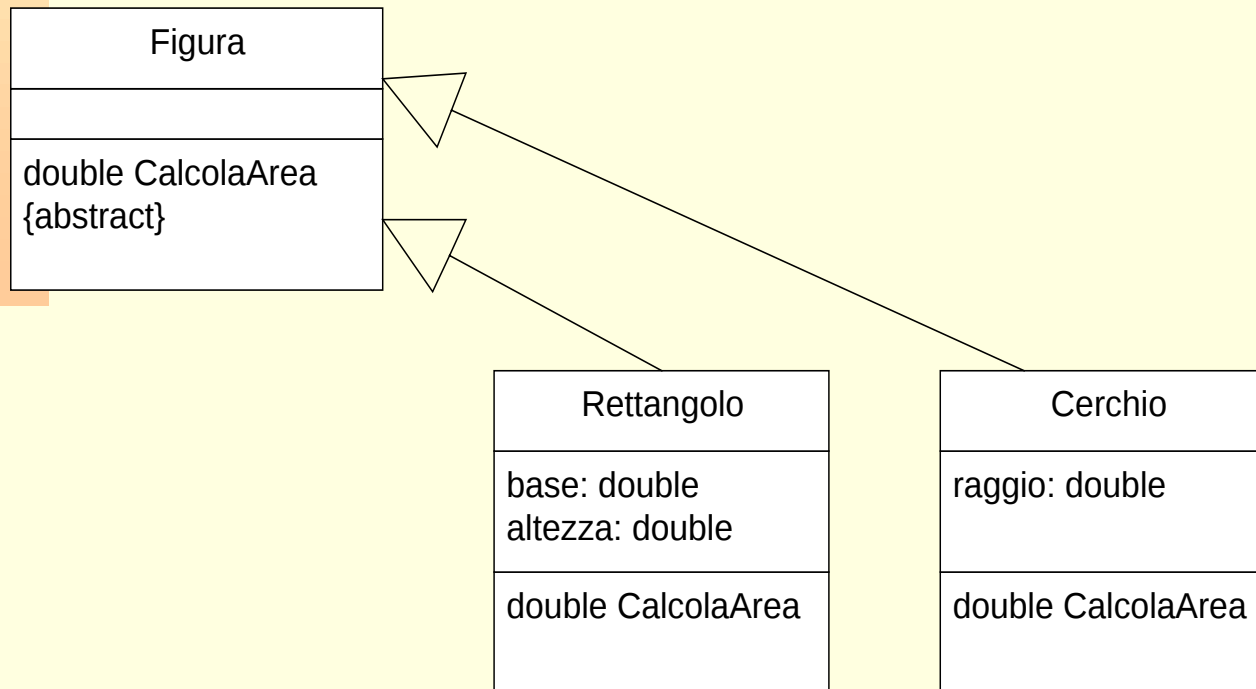
Ereditarietà multipla



Ereditarietà multipla

- Alcuni linguaggi di programmazione non la supportano (C++ sì, ma con qualche problema pratico...)
- In generale è meglio cercare di evitarla e utilizzare altre possibilità
- Di norma complica la situazione durante l'implementazione del sistema

Trasposizione in C++



File Figura.h

```
#ifndef _Figura_h_
#define _Figura_h_
class Figura {
    public:
        virtual double CalcolaArea()=0;
};
#endif
```

Figura.C non contiene nulla, non essendoci metodi “concreti”

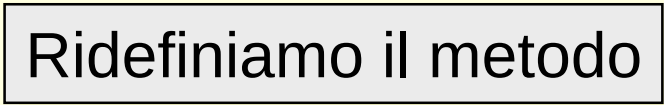
Funzione Virtuale Pura
Metodo Astratto

File Rettangolo.h

```
#ifndef _Rettangolo_h_
#define _Rettangolo_h_
class Rettangolo: public Figura {
    private:
        double base;
        double altezza;
    public:
        Rettangolo(double, double);
        virtual double CalcolaArea();
};
#endif
```



Sintassi per ereditare
l'interfaccia public



Ridefiniamo il metodo

File Rettangolo.cpp

```
#include "Rettangolo.h"
```

```
Rettangolo::Rettangolo(double b, double a)
```

```
{
```

```
    base=b;
```

```
    altezza=a;
```

```
}
```

```
double Rettangolo::CalcolaArea()
```

```
{
```

```
    return base*altezza;
```

```
}
```

Costruttore

Ridefinizione del metodo

Programma Principale

```
#include <iostream.h>
#include "Rettangolo.h"

void main()
{
    Rettangolo r(10.0,40.0);
    cout << r.CalcolaArea();
}
```

Mandiamo un messaggio all'oggetto r che risponde con il suo metodo particolare

Metodi virtuali e non

- Tutti i metodi (astratti e non) segnati virtual nella dichiarazione di una classe possono essere ridefiniti in una classe derivata
- I metodi non virtuali non possono essere ridefiniti - il loro comportamento rimane lo stesso per tutte le classi derivate da quelle in esame
- I metodi astratti devono essere dichiarati virtuali
- I metodi astratti rendono il sistema più elastico, capace di adattarsi alle modifiche, ma non bisogna esagerare ...