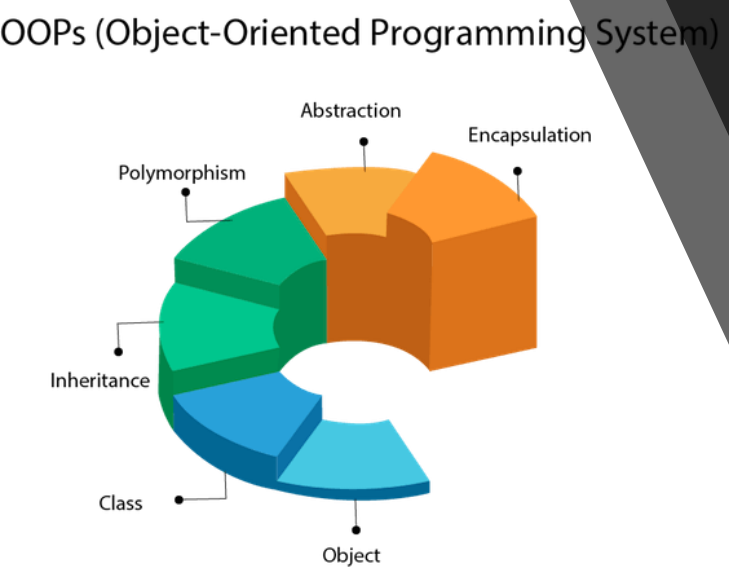
Classi e oggetti

Obiettivi

1. Risolvere un problema individuando gli oggetti e le loro interazioni
2. Definire una classe attraverso i suoi dati e i suoi metodi
3. Realizzare classi flessibili attraverso il polimorfismo
4. Strutturare gerarchie di classi sfruttando l'ereditarietà

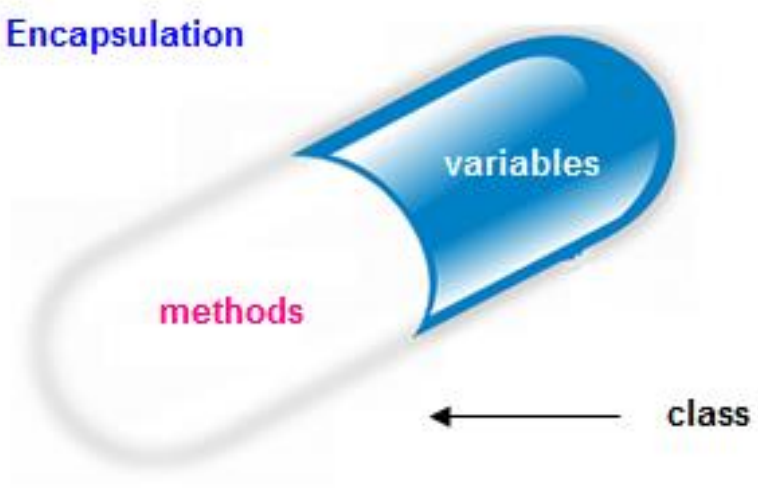
Linguaggio orientato agli oggetti

I programmi sono organizzati intorno ai dati e sono questi a controllare l'accesso al codice



Incapsulazione

Permette di associare ai dati del programma il codice autorizzato ad agire su tali dati



Incapsulazione

Non conosco il codice, ma so cosa posso fare con quell'oggetto

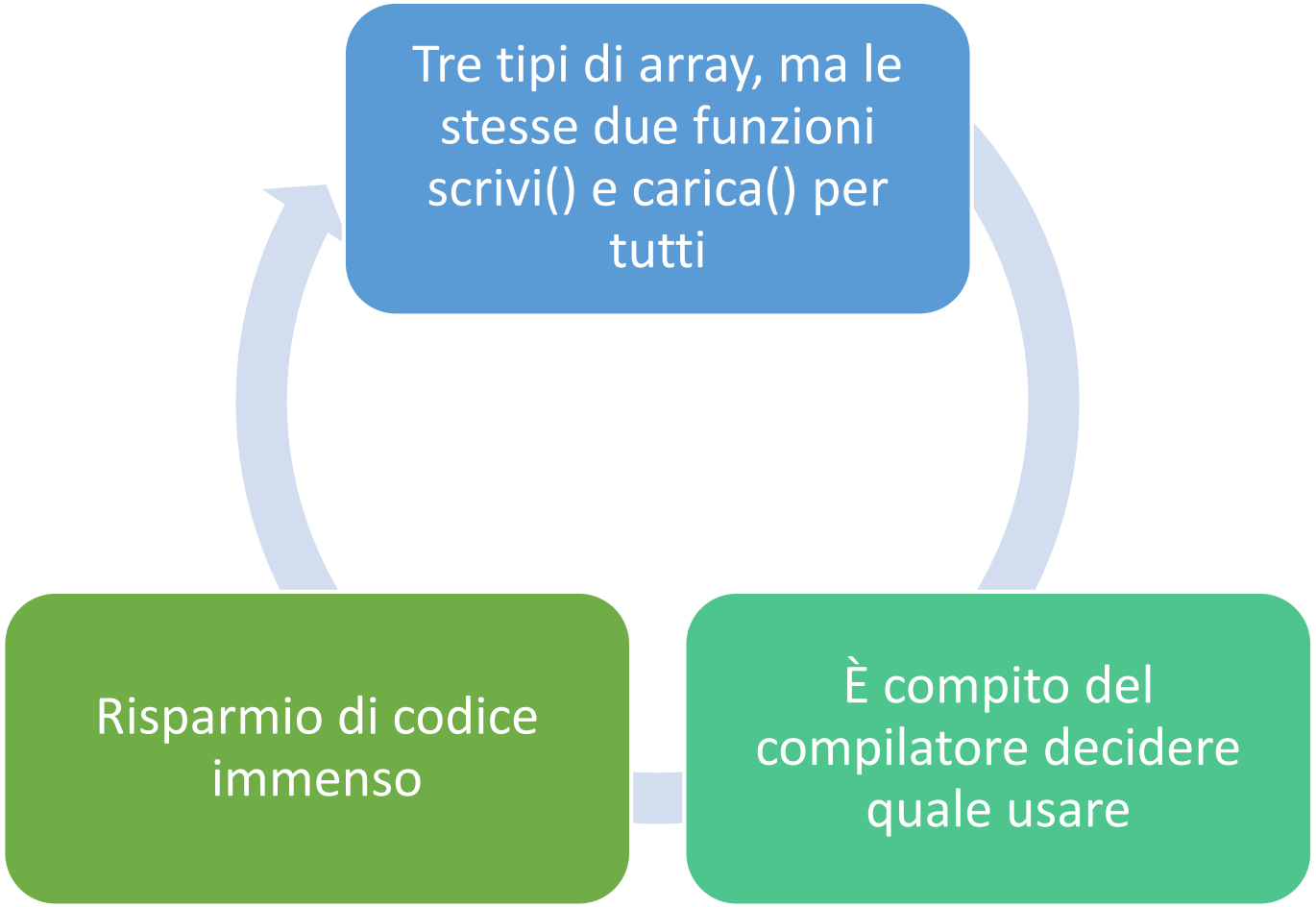
Esempio - Rettangolo

Consente a un'interfaccia di controllare l'accesso a una categoria generale di azioni

Esempio: termostato

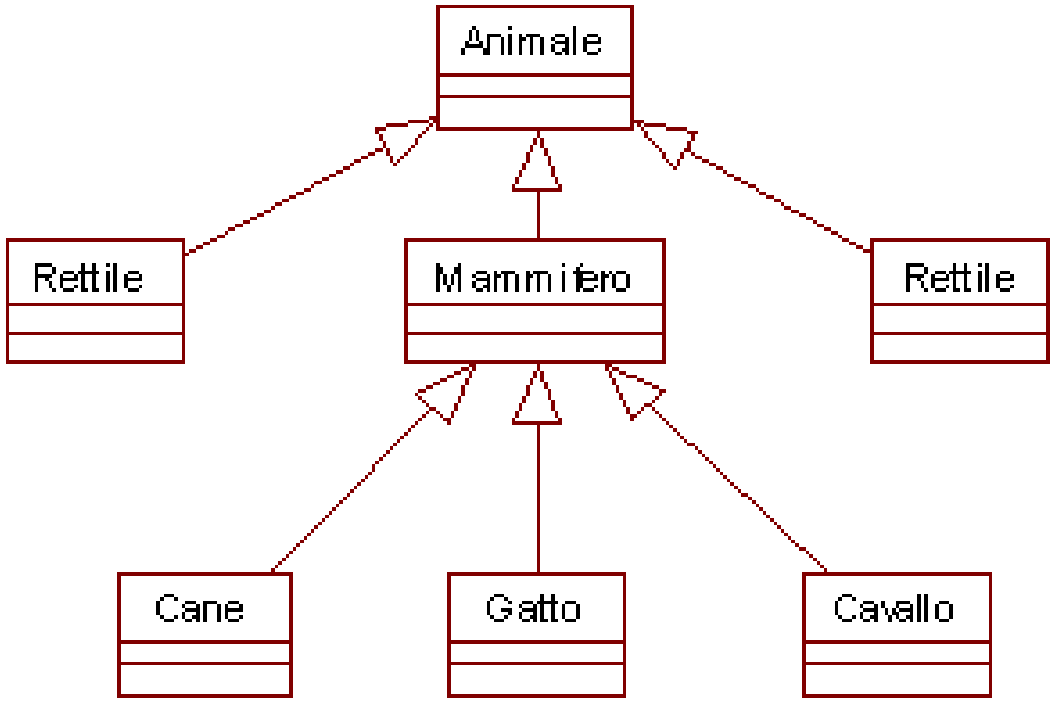
Polimorfismo

Polimorfismo - Array



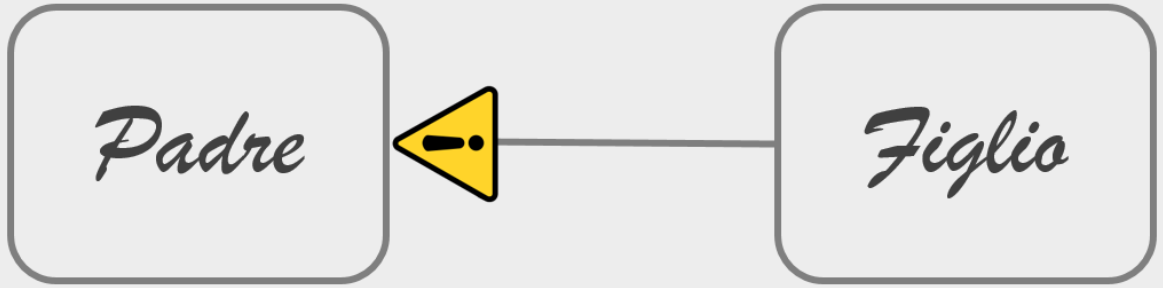
Ereditarietà

- Un oggetto acquisisce le proprietà di un altro oggetto
- È lo strumento che permette di costruire nuove classi utilizzando quelle già realizzate
- Esempio: mela in frutta


Animale
Rettile
Mammifero
Rettile
Cane
Gatto
Cavallo

Sottoclasse

Sopraclasse


Padre
Figlio

Due modalità

Estensione

La sottoclasse aggiunge nuovi attributi e metodi

Arricchisce la sopraclasse

Specializzazione

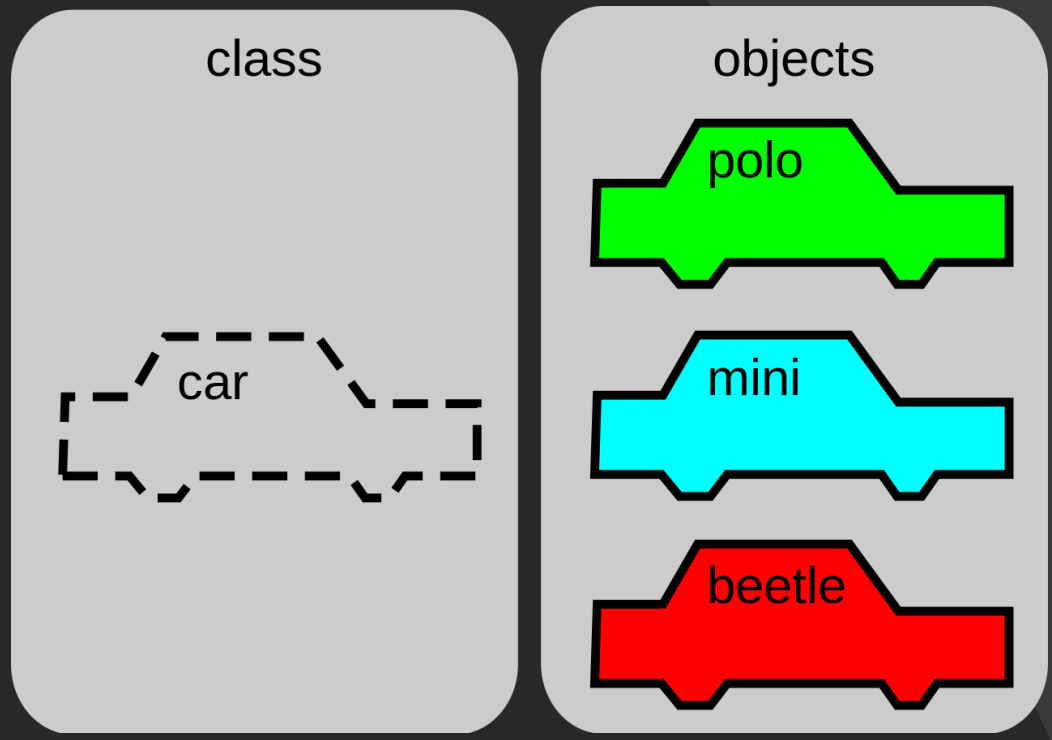
Ridefinizione

La sottoclasse ridefinisce i metodi ereditati

Implementazione diversa dei metodi

Classe

Categoria o gruppo di oggetti che hanno attributi simili e comportamenti analoghi



Esempio codice

Sintassi simile alla struttura

Contiene dati e codice

```
#include <iostream>

using namespace std;

class Rettangolo{
    public:
        float b, h;

        float area(){
            return b*h;
        }

        float perimetro(){
            return 2*(b+h);
        }
}
```

Uso delle classi

```cpp
#include <iostream>

using namespace std;

class Rettangolo{
    public:
        float b, h;

        float area(){
            return b*h;
        }

        float perimetro(){
            return 2*(b+h);
        }
}

int main(){
    Rettangolo rect;

    rett.b = 10;
    rett.h = 6;

    cout << "Area del rettangolo: " << rett.area()<< endl;
    cout << "Perimetro del rettangolo: " << rett.perimetro() << endl;

    return 0;
}
```

Costruttore

Porta lo stesso nome della classe e permette l'inizializzazione automatica dell'oggetto su cui si intende operare

Esempio - Rettangolo

```cpp
#include <iostream>

using namespace std;

class Rettangolo{
    public:
        float b, h;

        float area(){
            return b*h;
        }

        float perimetro(){
            return 2*(b+h);
        }

        public::Rettangolo(int b, int h){
            b = b;
            h = h;
        }
}

int main(){
    Rettangolo rect(10, 6);

    cout << "Area del rettangolo: " << rett.area()<< endl;
    cout << "Perimetro del rettangolo: " << rett.perimetro() << endl;

    return 0;
}
```

Overloading

Consiste nell'impiegare lo stesso nome per due o più metodi

Ogni ridefinizione del metodo deve utilizzare parametri di tipo differente oppure di numero differente

```
class ValoreAssoluto(){
    public:
        int abs(int intero){
            if(intero < 0){
                intero = -intero;
            }
            return intero;
        }

        double abs(double doppio){
            if(doppio < 0){
                doppio = -doppio;
            }
            return doppio;
        }

        long abs(long lungo){
            if(lungo < 0){
                lungo = -lungo;
            }
            return lungo;
        }
};
```

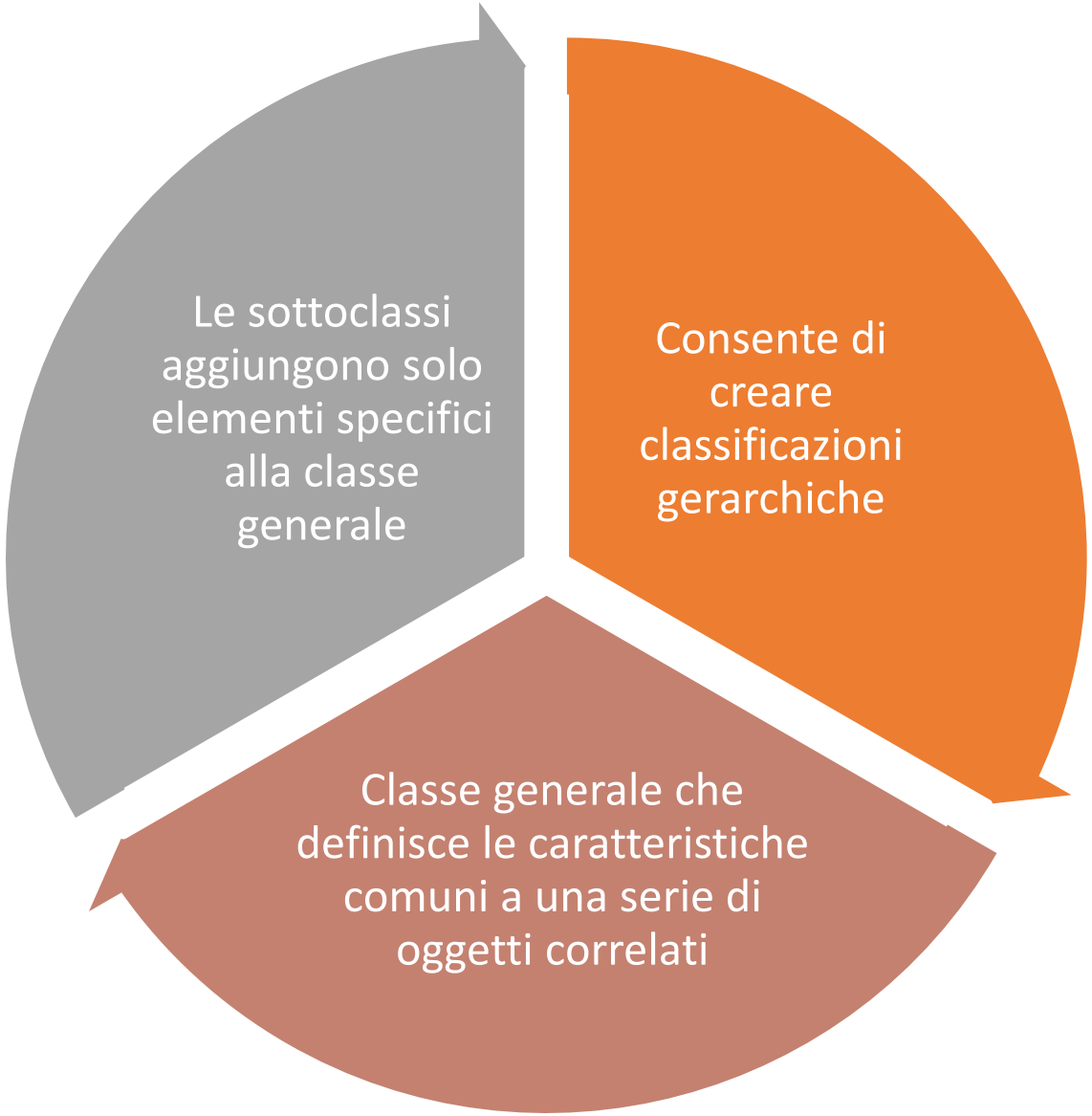
Trova la sua realizzazione nell'uso di metodi e di costruttori modificati tramite overloading

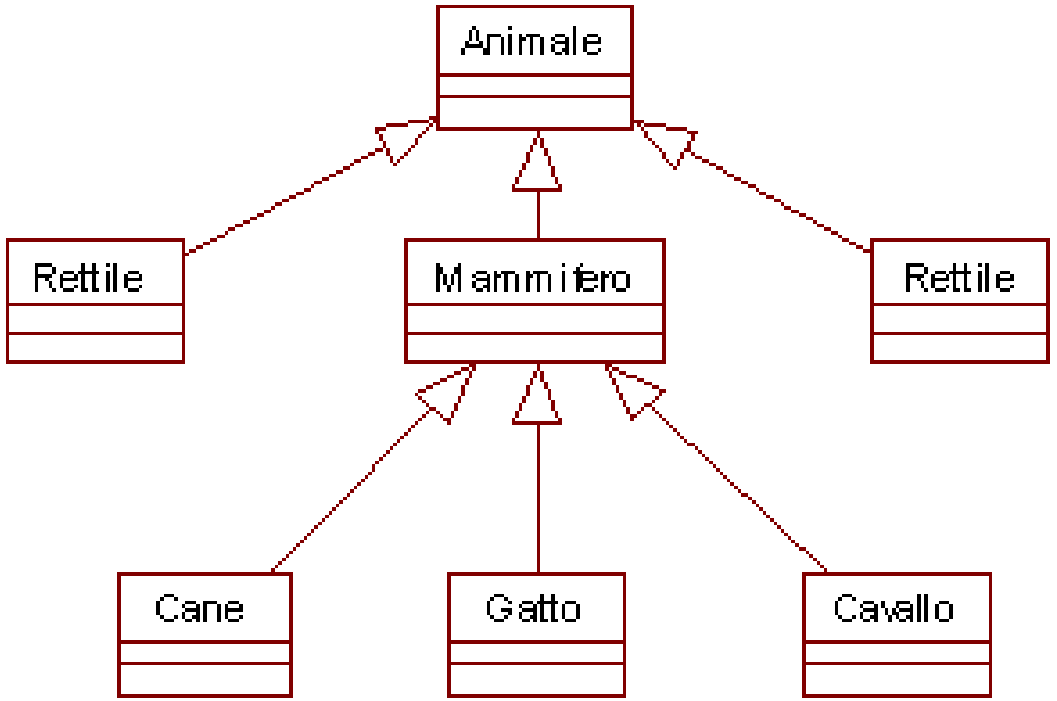
Si aumenta la flessibilità della classe

Se si cerca di creare un oggetto per il quale non esiste un costruttore → errore

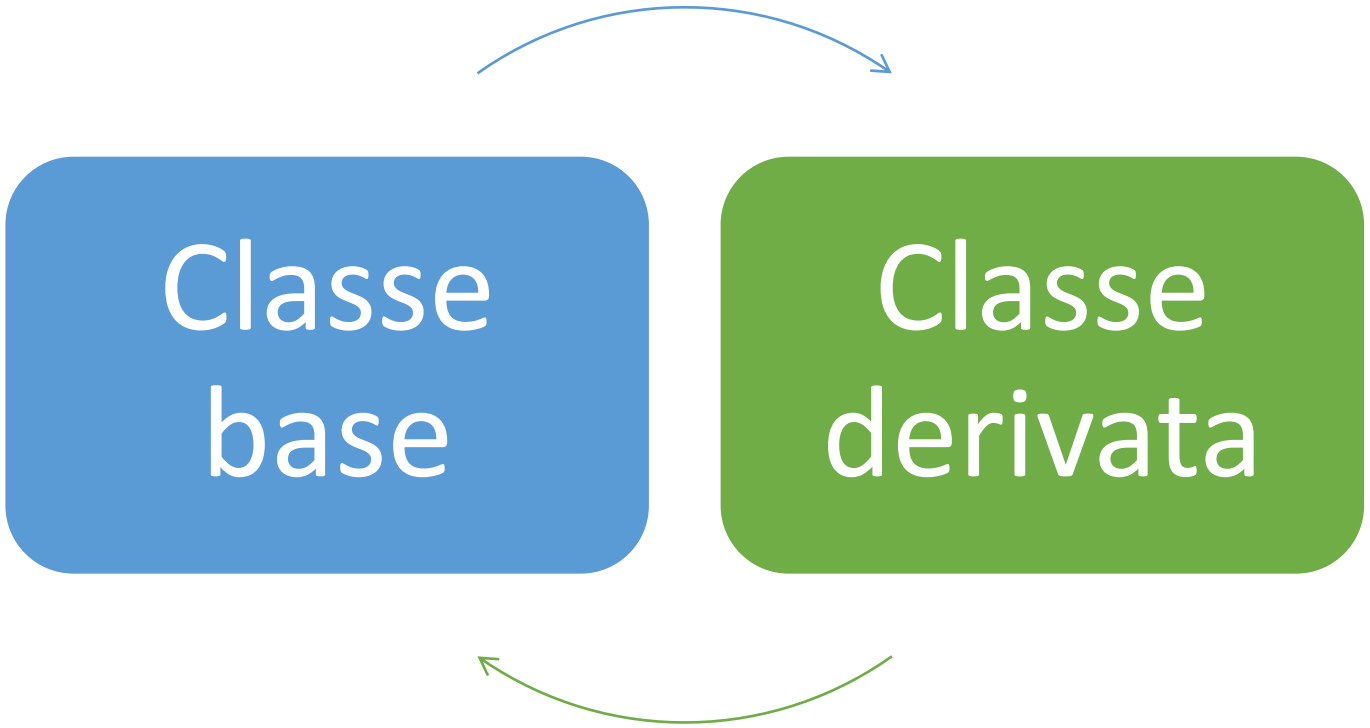
Polimorfismo

Ereditarietà




Animale
Rettile
Mammifero
Rettile
Cane
Gatto
Cavallo

Ereditarietà



Ereditarietà

I membri della classe base diventano anche membri della classe derivata

```
class nome_classe_derivata : nome_classe_base{
    //corpo della classe
}
```

Sintassi ereditarietà


That's all Folks.™
A WARNER BROS. CARTOON
© WARNER BROS. INC. 1989
205/500