ITIS-LS "Francesco Giordani" Caserta

prof. Ennio Ranucci

a.s. 2021-2022

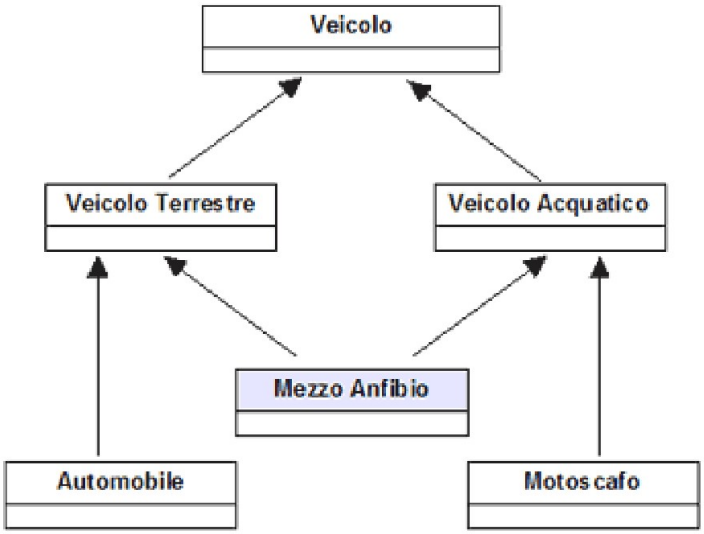
La programmazione orientata agli oggetti: Ereditarietà e Polimorfismo

- Sia Fido che Cocorito sono animali
- Essi hanno in comune attributi e metodi
  - nome
  - getNome()
  - mangia()
  - muoviti()
  - parla()

Si tratta degli stessi attributi e metodi della classe Animale!

Devo duplicarlo in Cane e in Papagallo??

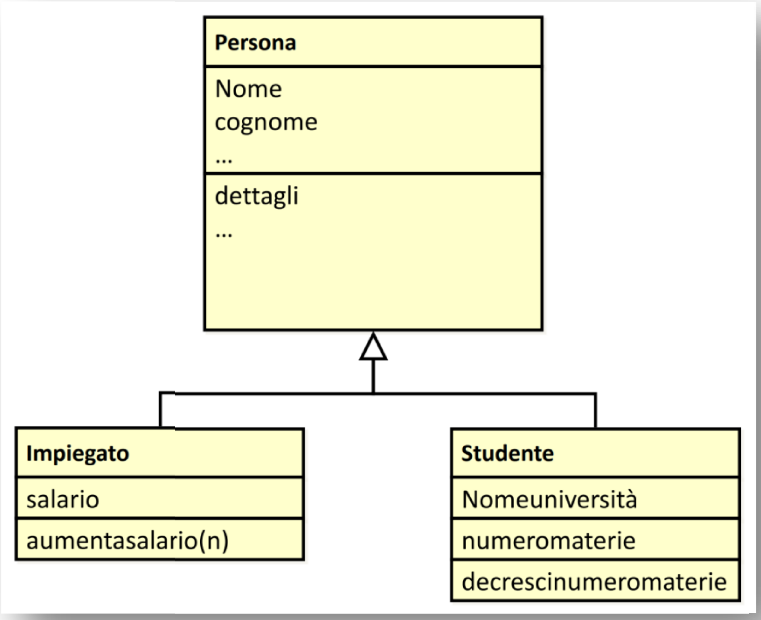
- Sarebbe molto comodo disporre di un meccanismo che consenta di **riutilizzare codice** già scritto senza doverlo duplicare!
- Questo meccanismo è previsto dalla programmazione orientata agli oggetti ed è l'**ereditarietà**

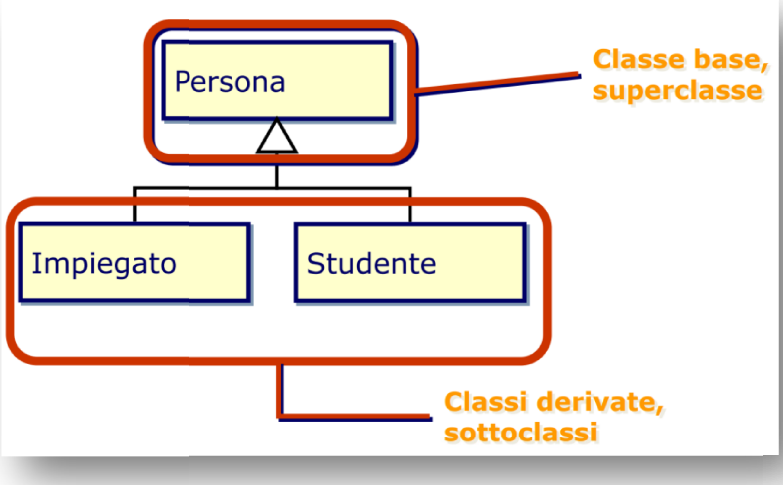


Ereditarietà e Polimorfismo: come riusare il software

A volte si incontrano classi con funzionalità simili.
È possibile creare classi disgiunte copiando gli attributi e i metodi in comune.
L'approccio "Copia e Incolla", però, non è una strategia vincente per le difficoltà di manutenzione, meglio "specializzare" il codice già funzionante aggiungendo il minimo necessario, in pratica definiamo una nuova classe (classe derivata) come specializzazione di un'altra (classe base).

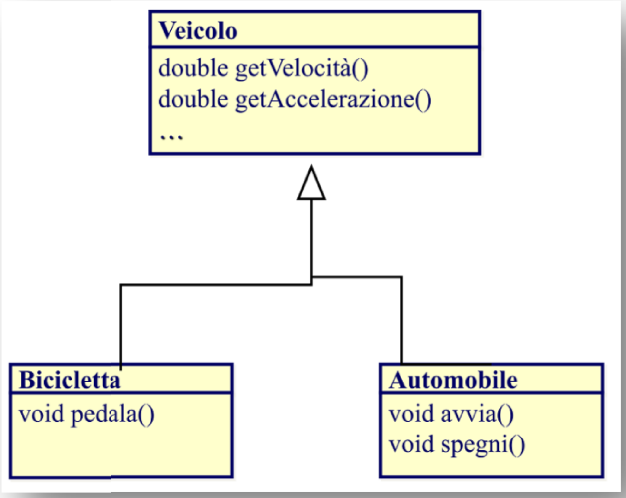
La classe base modella un concetto e la classe derivata modella un concetto più specifico.
La classe derivata quindi dispone di tutte le funzionalità (attributi e metodi) di quella base e aggiunge funzionalità proprie.
Può anche ridefinirne il funzionamento di metodi esistenti (polimorfismo).





Processo di astrazione

– Si introduce la superclasse che “astrae” il concetto comune condiviso dalle diverse sottoclassi
– Le sottoclassi vengono “spogliate” delle funzionalità comuni che migrano nella superclasse



Vantaggi dell’ereditarietà

- Evitare la duplicazione di codice
- Permettere il riuso di funzionalità
- Semplificare la costruzione di nuove classi
- Facilitare la manutenzione
- Garantire la consistenza delle interfacce

Ereditarietà in Java

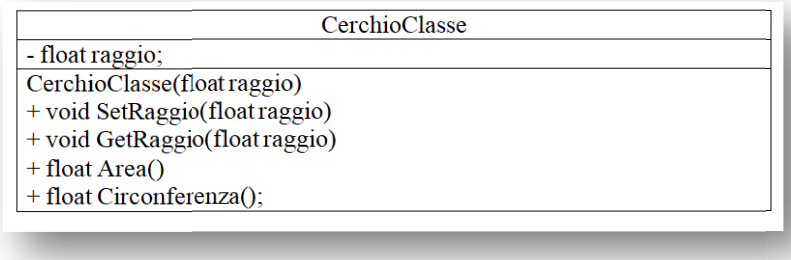
• Si definisce una classe derivata attraverso la parola chiave “extends” seguita dal nome della classe base
• Gli oggetti della classe derivata sono, a tutti gli effetti, estensioni della classe base anche nella loro rappresentazione in memoria

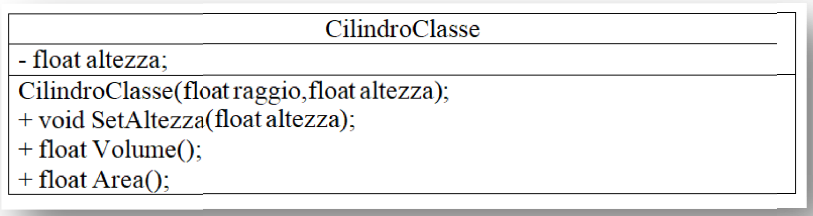
```
class Impiegato {
  String nome;
  double stipendio;

  Impiegato(String n) {
       nome = n;
       stipendio= 1500;
  }
}
```

```
class Funzionario
  extends Impiegato {

   Funzionario(String n) {
      super(n);
      stipendio = 2000;
  }
}
```

L’oggetto derivato contiene tutti i componenti (attributi e metodi) dell’oggetto da cui deriva.
I suoi metodi non possono operare direttamente su quelli definiti privati.
La restrizione può essere allentata: La superclasse può definire attributi e metodi con visibilità “protected”. Questi sono visibili alle sottoclassi.

Anno scolastico 2014/2015
Classe 4^ sez.D spec. Informatica
Data: 23/02/2015
Numero es:55
Versione:1.0 Programmatore/i: Sebastiano Fusco
Sistema Operativo:Windows XP
Compilatore/Interprete:Dev-C++ 4.9.9.2
Obiettivo didattico: L'alunno e' in grado di definire le classi ed utilizzare gli oggetti in c++
Obiettivo del programma: Codifica c++ La classe “Cilindro" derivata dalla classe “Cerchio”;






```
#include<iostream>
#include<cstdlib>
using namespace std;
class CerchioClasse
{
 private:
   float raggio;
 public:
   CerchioClasse(float raggio) { this->raggio=raggio; };
   void SetRaggio(float raggio) { this->raggio=raggio; };
   float Area() { return(raggio*raggio*3.14); };
   float Circonferenza() { return(raggio*2*3.14); };
};
```

```
class CilindroClasse : public CerchioClasse  //la classe cilindro deriva dalla classe cerchio
{
  private:
    float altezza;
  public:
    CilindroClasse(float raggio,float altezza) : CerchioClasse(raggio)
        { this->altezza=altezza; };
    void SetAltezza(float altezza) { this->altezza=altezza; };
    float Volume() { return (CerchioClasse::Area()*altezza); };
    float Area();
};
int main()
{
  CilindroClasse CilindroObj(4,10);
  cout<<"L'area del cilindro e': "<<Cilindro.Area()<<endl;  //351,68
  cout<<"Il volume del cilindro e': "<<Cilindro.Volume()<<endl; //502,4
  system("pause");
}

float CilindroClasse::Area()
{
  float AreaBase,AreaLaterale;
  AreaBase=CerchioClasse::Area()*2;      //funzione Area() della classe Cerchio
  AreaLaterale=CerchioClasse::Circonferenza()*altezza;
  return (AreaBase+AreaLaterale);
};
```

Codifica Java

```
class Cerchio
{
  private double raggio;
  public Cerchio(double raggio)
      {
        this.raggio=raggio;
      }
  public void setRaggio(double raggio)
     {
       this.raggio=raggio;
     }
  public double area()
    {
      return (raggio*raggio* Math.PI);
    }
public double circonferenza()
  {
     return(raggio*2*Math.PI);
  }
}

class Cilindro extends Cerchio
{
   private double altezza;
   public Cilindro(double raggio, double altezza)
      {
         super(raggio);
         this.altezza=altezza;
      }
  public void setAltezza (double altezza)
     {
       this.altezza=altezza;
     }
  public double volume()
   {
      return area()*altezza;
   }
}
```

```
class ProgCilindro
{
  public static void main(String args[])
     {
       Cilindro cilindroObj=new Cilindro(4.0,10.0);
       System.out.println("area di base= " + cilindroObj.area());
       System.out.println("volume = " + cilindroObj.volume());
     }
}
```