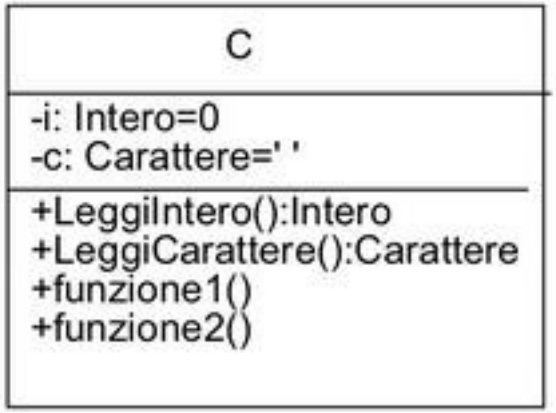
UNIVERSITÀ PARTHENOPE
DiST
Dipartimento di Scienze e Tecnologie

Programmazione II e Lab di PII

UML e C++

Angelo Ciaramella

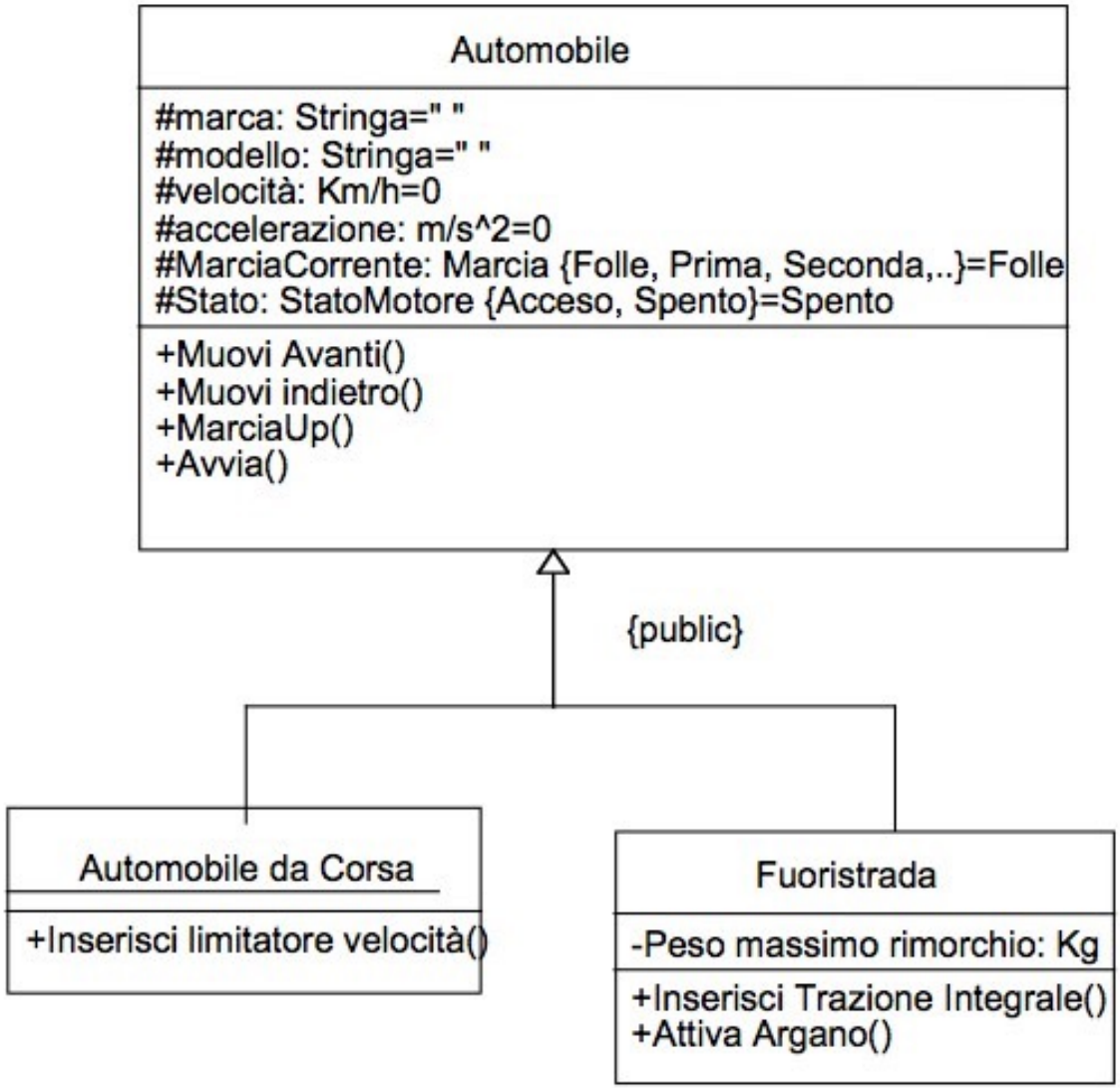
Design Pattern



Esempio di classe

```
class C {
public:
      C(int iI=0, char cI=' ');
      ~C();
      int LeggiIntero() const;
      char LeggiCarattere() const;
      void funzione1();
      void funzione2();
private:
      int i;
      char c;
};
```


Design Pattern



Esempio di ereditarietà

Ereditarietà

```
class Automobile {
public:
        Automobile(Stringa marcaI = "", Stringa modI = "" ,
        ...);
        ~Automobile();
        void Muovi_Avanti();
        void Muovi_indietro();
        Marcia MarciaUp();
        void Avvia();
protected:
      Stringa marca, modello;
      Km_ora velocita;
      MetriSecondoQuadro accelerazione;
      Marcia  marciacorrente;
      StatoMotore stato;
};
```

Esempio di classe

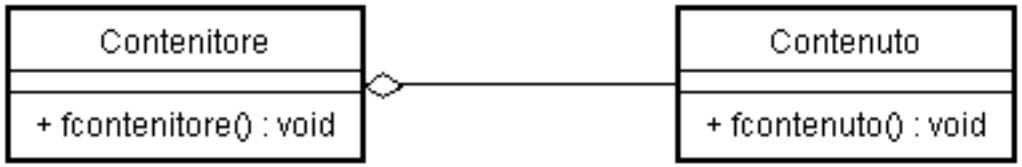
Ereditarietà

```
class Fuoristrada : public Automobile
{
Kg Peso_massimo_rimorchio;

public:
        Fuoristrada();
        ~Fuoristrada();
        void Inserisci_Trazione_Integrale();
        void Attiva_Argano();
}
```

Esempio di estensione di classe

Aggregazione




```
class Contenuto {
public:
        Contenuto();
        void fcontenuto();

private:
        // variabili membro
};
```

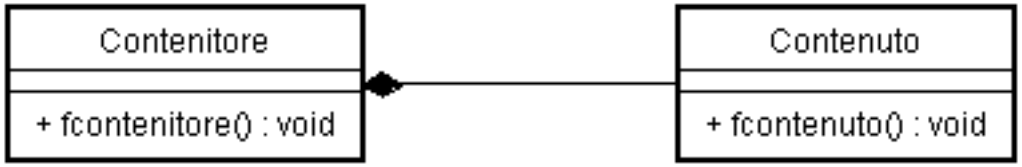
```
class Contenitore {
private:
        Contenuto* c; // può esistere anche indipendentemente dal
contenitore
public:
        Contenitore(Contenuto* q):c(q){};
        void fcontenitore()
         { c -> fcontenuto(); }
};
```


Aggregazione

```
main() {
        Contenuto c(lista valori iniziali);
        Contenuto* ptr=&Contenuto;
        Contenitore x(ptr);
        x.fcontenitore();
        // tale chiamata opera su un oggetto contenitore e
        // indirettamente sul contenuto.
}
```

Esempio di utilizzo di classi aggregate

Composizione



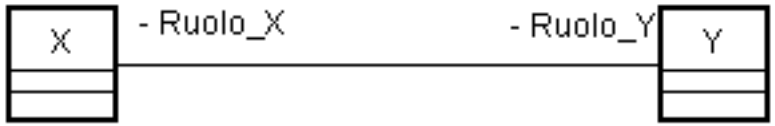

```
class Contenitore {

        Contenuto x; // appartiene all'oggetto Contenitore
...

public:
        Contenitore(int i): x(i){};
        ~Contenitore();
        void fcontenitore() {
            x.fcontenuto(); }
}
  . . .

Contenitore contenitore(5);
contenitore.fcontenitore();
```


Associazione uno a uno




```
class X {
        Y* ruolo_Y;
public:
        linker(Y* ptr): ruolo_Y(ptr){}
};
```

```
class Y {
        X* ruolo_X;
public:
        linker(X* ptr): ruolo_X(ptr){}
};
```

```
void main {
        X x;
        Y y;
       x.linker(&y);
       y.linker(&x);
};
```


Associazione una a molti

- non differisce da quello precedente
  - dal lato ove compare 1 come molteplicità bisogna gestire una lista di puntatori all'altro tipo di oggetto