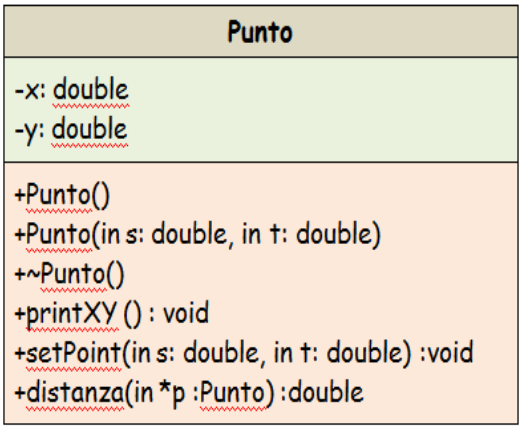
Classe PUNTO

UML: Diagramma della classe




```
// OOP-Punto.cpp      programma completo
#include <iostream>
#include <math.h>
using namespace std;
//---------------------------------------------- Parte dichiarativa della classe
// Definizione della classe Punto
class Punto
{
private:
  // Coordinate del punto
  double x;
  double y;

public:
  // Costruttori
  Punto();                      // 1° costruttore
  Punto(double s, double t);  // 2° costruttore
  // Distruttore
 ~Punto();

  void printXY();                        // Visualizza le coordinate del punto
  void setPoint(double s, double t);   // Imposta il valore delle coordinate del punto
  // distanza da un altro punto
  double distanza(Punto *p);
};
//---------------------------------------------- Parte implementativa della classe
// Costruttore
Punto::Punto()                  //scope resolution ::
{   x = y = 0;
}
// Definizione del 2° costruttore della classe Punto
Punto::Punto(double s, double t)
{     setPoint(s, t);
}
// Definizione del distruttore personalizzato
Punto::~Punto()
{    cout<<"Distrutto oggetto di coordinate "<<this->x<<'\t'<< this->y<<endl;
      system("pause");
}
// Definizione del metodo printXY della classe Punto
void Punto::printXY()
{   cout << "x = " << x << '\t' << "y = " << y << endl;
}
// Definizione del metodo setPoint della classe Punto
void Punto::setPoint(double s, double t)
```

```
{   // Controllo della correttezza delle coordinate
  x = s < 0 ? 0 : (s > 100 ? 100 : s);
  y = t < 0 ? 0 : (t > 100 ? 100 : t);
}
// Definizione metodo distanza
double Punto::distanza(Punto *p)
{  return sqrt((x - p->x)*(x - p->x) + (y - p->y)*(y - p->y));
}


int main()
{
  double a, b;      // coordinate da acquisire
  Punto p1;         // p1 è un oggetto della classe Punto
                    // Chiamata del costruttore senza parametri
  cout<<"p1: ";
  p1.printXY();     // visualizza le coordinate di p1

  // Chiamata del secondo costruttore: x = 15 e y = 68
  Punto p2(15, 68);       //oppure  Punto p2=Punto(15, 68);   //chiamata esplicita del costruttore
                          // p2 è un oggetto della classe Punto
  // Acquisizione delle coordinate del punto
  cout << "Coordinate a [0-100] e b [0-100] \n";
  cout<< "a= ";
  cin >> a;
  cout<< "b= ";
  cin >> b;
  p1.setPoint(a, b);      // imposta le coordinate di p1
  cout<<"p1: ";
  p1.printXY();            // visualizza le coordinate di p1
  cout<<"p2: ";
  p2.printXY();            // visualizza le coordinate di p2
  cout<<"La distanza tra p1 e p2 e': "<<p1.distanza(&p2)<<endl;    // distanza
 system("pause");
}
```

Il modo in cui abbiamo scritto il codice precedente **non permette un facile ri-uso...** scriviamo quindi la nostra classe in modo tale che essa **sia riusabile** in qualsiasi momento.

Per fare ciò basta dividere il codice nelle seguenti parti:

1. **Un file** di tipo **header** dove scrivere la ***dichiarazione*** di classe (esempio **punto.h**)
2. **Un file** di tipo **cpp** dove scrivere l'***implementazione*** della classe (esempio **punto.cpp**).

   In esso deve essere inserito il precedente file header e devono essere *definite* le **funzioni** (**metodi**) il cui prototipo si trova nella dichiarazione di classe.
3. Un **terzo file**, che **includa** il file **cpp** di classe, dove scriveremo il nostro **main**.

```
// ----- punto.h        (sezione dichiarativa della classe) file di tipo header
// Definizione della classe Punto
class Punto
{   private:
      // Coordinate del punto
      double x;
      double y;
   public:
      // Costruttori
      Punto();                          // 1° costruttore
      Punto(double s, double t);     // 2° costruttore
      // Distruttore
     ~Punto();
     // Visualizza le coordinate del punto
     void printXY();
     // Imposta il valore delle coordinate del punto
     void setPoint(double s, double t);

     // distanza da un altro punto
     double distanza(Punto *p);
};
```

```
// ----- punto.cpp      (sezione implementativa della classe)
#include "punto.h"        // inclusione file header
#include <iostream>
#include <math.h>
using namespace std;
// Costruttore
Punto::Punto()
{   x = y = 0;
}
// Definizione del 2° costruttore della classe Punto
Punto::Punto(double s, double t)
{   setPoint(s, t);
}
// Definizione del distruttore personalizzato
Punto::~Punto()
{    cout<<"Distrutto oggetto di coordinate "<<this->x<<'\t'<< this->y<<endl;
     system("pause");
}
// Definizione del metodo printXY della classe Punto
void Punto::printXY()                //scope resolution ::
{
   cout << "x = " << x << '\t' << "y = " << y << endl;
}
```

```
// Definizione del metodo setPoint della classe Punto
void Punto::setPoint(double s, double t)
{  // Controllo della correttezza delle coordinate
   x = s < 0 ? 0 : (s > 100 ? 100 : s);
   y = t < 0 ? 0 : (t > 100 ? 100 : t);
}
// Definizione metodo distanza
double Punto::distanza(Punto *p)
{
  return sqrt((x - p->x)*(x - p->x) + (y - p->y)*(y - p->y));
}
```

```
// ----- main.cpp
#include "punto.cpp"
#include <iostream>
using namespace std;

int main()
{
   double a, b;      // coordinate da acquisire
   Punto p1;         // p1 è un oggetto di classe Punto
                     // Chiamata del 1° costruttore senza parametri: assegna x = 0 e y = 0
   cout<<"p1: ";
   p1.printXY();    // visualizza le coordinate di p1

   // Chiamata del secondo costruttore: x = 15 e y = 68
   Punto p2(15, 68);        //oppure  Punto p2=Punto(15, 68);
                            //p2 è  un oggetto della classe Punto

    // Acquisizione delle coordinate del punto
   cout << "Coordinate a [0-100] e b [0-100] \n";
   cout<< "a= ";
   cin >> a;
   cout<< "b= ";
   cin >> b;
   p1.setPoint(a, b);       // imposta le coordinate di p1

   cout<<"p1: ";
   p1.printXY();            // visualizza le coordinate di p1

   cout<<"p2: ";
   p2.printXY();            // visualizza le coordinate di p2

   cout<<"La distanza tra p1 e p2 e': "<<p1.distanza(&p2)<<endl;    // distanza
  system("pause");
}
```