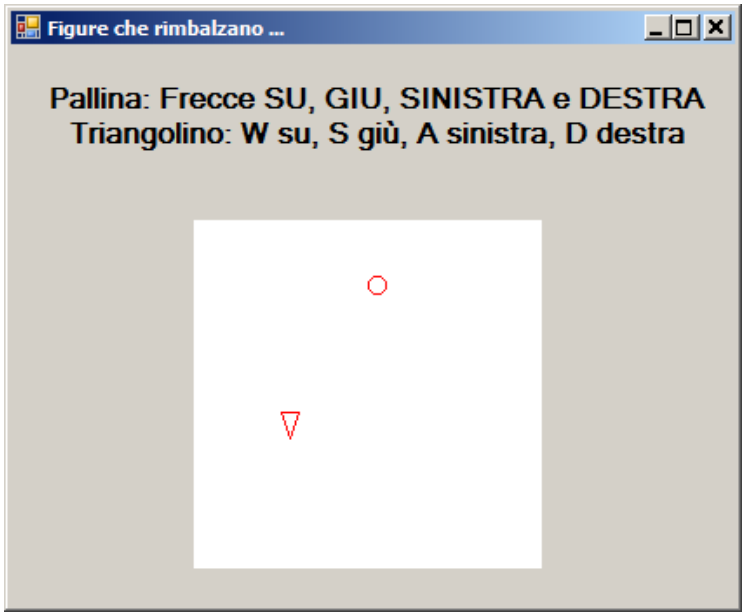
Esempio Pilota:

PallinaCheEredita

Definizione della Form



Codice della Form

```
using System;
using System.Windows.Forms;

namespace PallinaCheEredita
{
   public partial class frmPallinaETriangolino : Form
   {
      public frmPallinaETriangolino()
      {
         InitializeComponent();
      }

      Pallina MiaPallina;
      Triangolino MioTriangolino;

      private void frmPallinaETriangolino_Load
                                    (object sender, EventArgs e)
      {
         // crea le istanze per Pallina e Triangolino ...
         MiaPallina = new Pallina(100, 100, "SU", 10);
         MioTriangolino = new Triangolino(50, 180, "GIU", 10, 15);

         // attiva il Timer ...
         mioTimer.Start();
      }

      private void frmPallinaETriangolino_KeyDown
                                    (object sender, KeyEventArgs e)
      {
        if     (e.KeyCode == Keys.Up) MiaPallina.Direzione = "SU";
      else if (e.KeyCode == Keys.Down) MiaPallina.Direzione = "GIU";
      else if (e.KeyCode == Keys.Right) MiaPallina.Direzione = "DESTRA";
      else if (e.KeyCode == Keys.Left) MiaPallina.Direzione = "SINISTRA";
      else if (e.KeyCode == Keys.W) MioTriangolino.Direzione = "SU";
      else if (e.KeyCode == Keys.S) MioTriangolino.Direzione = "GIU";
      else if (e.KeyCode == Keys.D) MioTriangolino.Direzione = "DESTRA";
      else if (e.KeyCode == Keys.A) MioTriangolino.Direzione = "SINISTRA";

      else { } // ignora tutti gli altri tasti ...
      }

      private void mioTimer_Tick(object sender, EventArgs e)
      {
         // muovi la Pallina ...

         MiaPallina.Muovi (Pnl);

         if (MiaPallina.E_Fuori (Pnl))
         {
            MiaPallina.InvertiDirezione( );
            MiaPallina.Muovi (Pnl);
         }

         // muovi il Triangolino ...

         MioTriangolino.Muovi(Pnl);

         if (MioTriangolino.E_Fuori(Pnl))
         {
            MioTriangolino.InvertiDirezione();
            MioTriangolino.Muovi(Pnl);
         }
      }

   }
}
```

Codice della Classe FiguraGrafica

```
using System.Drawing;
using System.Windows.Forms;

namespace PallinaCheEredita
{
   // FiguraGrafica è una CLASSE ASTRATTA (abstract) perchè è
   // progettata per essere solo una CLASSE BASE
                                        e DEVE ESSERE DERIVATA.
   // N.B.: Non si possono creare istanze di classe FiguraGrafica ...

   public abstract class FiguraGrafica
   {

      // --- ATTRIBUTI ...

      // _X ed _Y indicano la posizione della Figura ...
      protected  int  _X;
      protected  int  _Y;

      // _Dir è la direzione di movimento fra
      // "SU", "GIU", "SINISTRA", "DESTRA" ...
      protected  string  _Dir;

      /* Nota Bene: usiamo il modificatore di accesso PROTECTED
         perchè vogliamo che _X, _Y e _Dir siano accessibili
         dal codice delle classi derivate, ma non dal codice
         esterno (ad esempio, dalla form).                    */
```

```
    // --- COSTRUTTORI ...

    public FiguraGrafica (int nuovaX,
                          int nuovaY,
                          string nuovaDirezione)
    {
       _X = nuovaX;
       _Y = nuovaY;

       _Dir = nuovaDirezione;
    }

    // --- PROPRIETA' ...

    public int X
    {
       get { return _X; }
       set { _X = value; }
    }

    public int Y
    {
       get { return _Y; }
       set { _Y = value; }
    }

    public string Direzione
    {
       get { return _Dir; }
       set {
             if ( ((value == "SU") || (value == "GIU") ) ||
                  ((value == "SINISTRA") || (value == "DESTRA") ) )
             {
                _Dir = value;
             }
             else
                MessageBox.Show("Direzione non riconosciuta.");
       }
    }


    // --- METODI ...

    public void Muovi(Panel P)
    {
       Disegna(P, Pens.White);        // Cancella la figura ...
       ModificaPosizione();           // Aggiorna la sua posizione ...
       Disegna(P, Pens.Red);          // Ridisegna la figura ...
    }

    public void ModificaPosizione()
    {
       // tutte le figure si spostano di 2 pixel alla volta ...

       switch (Direzione)
       {
          case "SU": _Y = _Y - 2; break;
          case "GIU": _Y = _Y + 2; break;
          case "SINISTRA": _X = _X - 2; break;
          case "DESTRA": _X = _X + 2; break;
       }
    }

    // il metodo Disegna è un metodo abstract (astratto) perchè
    // DEVE ESSERE ridefinito (override) in tutte le classi derivate ...

    public abstract void Disegna(Panel P, Pen Penna);

    // il metodo E_Fuori è un metodo virtual. E' definito nella classe
    // base e PUO' ESSERE riscritto (override) nelle classi derivate

    public virtual bool E_Fuori(Panel P)
    {
       return (_X < 0) || (_X > P.Width)   ||
              (_Y < 0) || (_Y > P.Height);
    }

    public void InvertiDirezione( )
    {
       switch (_Dir)
       {
          case "SU":       _Dir = "GIU"; break;
          case "GIU":      _Dir = "SU"; break;
          case "SINISTRA": _Dir = "DESTRA"; break;
          case "DESTRA":   _Dir = "SINISTRA"; break;
       }
    }
  }
}
```

Codice della Classe Pallina

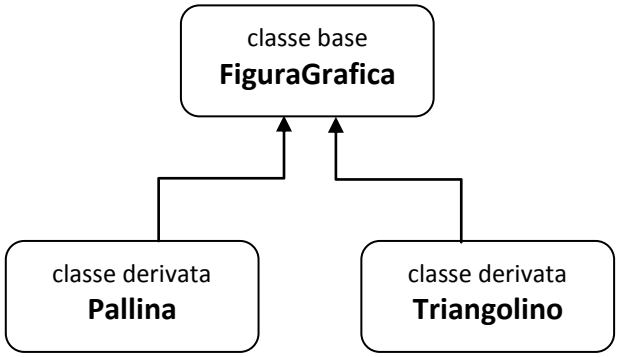
```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace PallinaCheEredita
{
  class Pallina : FiguraGrafica  // … il "due punti" significa "eredita da …"
  {
    // attributi ...

    private int _Diam;

    // costruttori ...

    public Pallina
          (int nuovaX, int nuovaY,
           string nuovaDirezione, int nuovoDiametro)
                        : base (nuovaX, nuovaY, nuovaDirezione)
                    // ": base" chiama il costruttore della classe base
    {
       _Diam = nuovoDiametro;
    }

    // proprietà ...

    public int Diametro
    {
       get { return _Diam; }
       set { _Diam = value; }
    }
```

```
      // metodi ...

      // Ridefinizione del metodo abstract Disegna ...
      public override void Disegna(Panel P, Pen Penna)
      {
         Graphics G = P.CreateGraphics();
         G.DrawArc(Penna, _X, _Y, _Diam, _Diam, 0, 360);
      }

      // Ridefinizione del metodo virtual E_Fuori ...
      public override bool E_Fuori (Panel P)
      {
         return ( _X < 0 )  ||  ( _X > P.Width - _Diam)  ||
                ( _Y < 0 ) ||  ( _Y > P.Height - _Diam);
      }

      /* Nota Bene: in DISEGNA e E_FUORI possiamo usare _X e _Y
         perchè sono stati dichiarati PROTECTED nella classe
         base FIGURAGRAFICA. Se avessimo usato PRIVATE, essi
         non sarebbero accessibili in questo punto del codice */
   }
}
```

Codice della Classe Triangolino

```
using System.Drawing;
using System.Windows.Forms;

namespace PallinaCheEredita
{
   class Triangolino : FiguraGrafica
   {
      // attributi ...

      int _Base;
      int _Altezza;

      // costruttori ...

      public Triangolino
            (int nuovaX, int nuovaY, string nuovaDirezione,
             int nuovaBase, int NuovaAltezza)
                  : base (nuovaX, nuovaY, nuovaDirezione)
      {
         _Base = nuovaBase;
         _Altezza = NuovaAltezza;
      }

      // proprietà ...

      public int Base
      {
         get { return _Base; }
         set { _Base = value; }
      }

      public int Altezza
      {
         get { return _Altezza; }
         set { _Altezza = value; }
      }

      // metodi ...

      // Ridefinizione del METODO ABSTRACT Disegna ...
      public override void Disegna(Panel P, Pen Penna)
      {
         Graphics G = P.CreateGraphics();

         G.DrawLine(Penna, _X, _Y, _X + _Base, _Y);
         G.DrawLine(Penna, _X + _Base, _Y, _X + _Base / 2, _Y + _Altezza);
         G.DrawLine(Penna, _X + _Base / 2, _Y + _Altezza, _X, _Y);
      }

      // Ridefinizione del METODO VIRTUAL E_Fuori ...
      public override bool E_Fuori(Panel P)
      {
         return ( _X < 0 )  ||  ( _X > P.Width - _Base)  ||
                ( _Y < 0)  ||  ( _Y > P.Height - _Altezza);
      }
   }
}
```

Gerarchia delle Classi e Diagrammi UML



classe base FiguraGrafica

attributi

- **_X** int
- **_Y** int
- **_Dir** string

costruttori

+ **FiguraGrafica** (int **nuovaX**, int **nuovaY**, string **nuovaDirezione**)

proprietà

+ **X** int
+ **Y** int
+ **Direzione** string

metodi

+ **Muovi** (Panel **pnl**)
+ **ModificaPosizione** ( )
+ **Disegna** (Panel **pnl**, Pen **Penna**) **metodo abstract**
+ **E_Fuori** (Panel **pnl**) → **bool** **metodo virtual**
+ **InvertiDirezione** ( )

classe derivata Pallina

attributi

- **_Diam** int

*… eredita gli attributi **_X, _Y, _Dir***

costruttori

+ **Pallina** (int **nuovaX**, int **nuovaY**,
string **nuova Direzione**, int **nuovoDiametro**)

proprietà

+ **Diametro** int

*… eredita le proprietà **X, Y, Direzione***

metodi

+ **Disegna** (Panel **pnl**, Pen **Penna**) **override del metodo abstract**
+ **E_Fuori** (Panel **pnl**) → **bool** **override del metodo virtual**

*… eredita i metodi **Muovi, ModificaPosizione, InvertiDirezione***

classe derivata Triangolino

attributi

- **_Base** int
- **_Altezza** int

*… eredita gli attributi **_X, _Y, _Dir***

costruttori

+ **Triangolino** (int **nuovaX**, int **nuovaY**,
string **nuova Direzione**,
int **nuovaBase**, int **nuova Altezza**)

proprietà

+ **Base** int
+ **Altezza** int

*… eredita le proprietà **X, Y, Direzione***

metodi

+ **Disegna** (Panel **pnl**, Pen **Penna**) **override del metodo abstract**
+ **E_Fuori** (Panel **pnl**) → **bool** **override del metodo virtual**

*… eredita i metodi **Muovi, ModificaPosizione, InvertiDirezione***

APPROFONDIMENTO …

```
// Scriviamo il codice dell'evento Tick in modo più "object oriented".
// Il sottoprogramma "MuoviFigura" accetta un parametro di classe
// FiguraGrafica (classe base), quindi è possibile passare ad esso
// qualsiasi oggetto delle classi derivate (pallina, triangolino, ecc.) …

    private void mioTimer_Tick(object sender, EventArgs e)
    {
       MuoviFigura ( MiaPallina );
       MuoviFigura ( MioTriangolino );
    }

    private void MuoviFigura ( FiguraGrafica miaFigura )
    {
       miaFigura.Muovi (Pnl);

       if (miaFigura.E_Fuori (Pnl) )
       {
          miaFigura.InvertiDirezione ( );
          miaFigura.Muovi (Pnl);
       }
    }
```