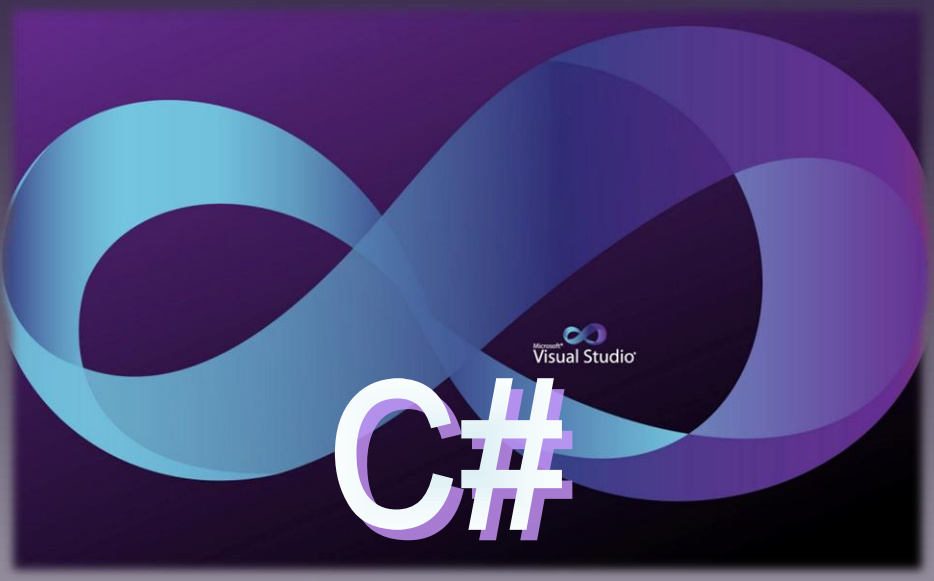
ISTITUTO TECNICO INDUSTRIALE
G. M. ANGIOY
SASSARI



CORSO DI PROGRAMMAZIONE

ELEMENTI STATICI E CLASSI STATICHE

DISPENSA 15.03

15-03_OOP_Static_[15]

Versione del: 07/11/2015
Revisione numero: 15

Prof. Andrea Zoccheddu
Dipartimento di Informatica

DIPARTIMENTO
INFORMATICA E TELECOMUNICAZIONI

ELEMENTI DI CLASSE E CLASSI STATICHE

INTRODUZIONE AGLI ELEMENTI STATICI DI UNA CLASSE

ATTRIBUTI STATICI

PROGETTO GUIDATO SU ATTRIBUTI STATICI

- Si prepari un nuovo progetto e si crei una nuova classe come la seguente:

VC#

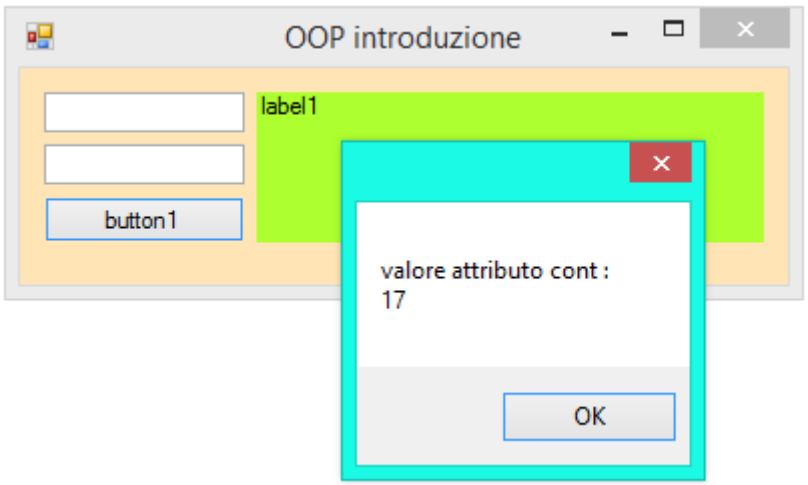
```
class Punto
{
    // attributo statico
    public static int cont;
    // attributi di istanza
    public int x, y;
}
```

- Associa il seguente gestore di evento:

VC#

```
private void button1_Click(object sender, EventArgs e)
{
    Punto p;
    p = new Punto();
    p.x = 13;
    Punto.cont = 17;
    MessageBox.Show("valore attributo cont : \n" + Punto.cont);
    MessageBox.Show("valore attributo x: \n" + p.x);
}
```

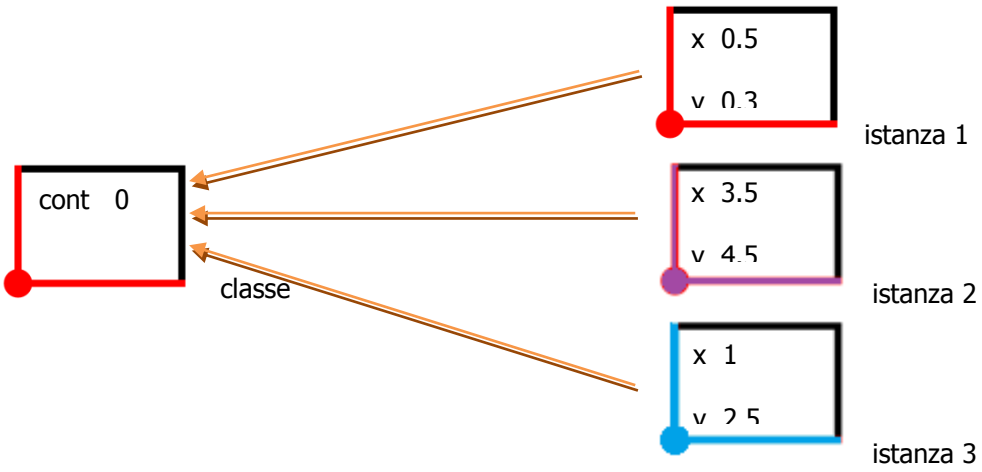
- Prova il progetto



CONCETTO DI ATTRIBUTO STATICO

Si è osservato che gli attributi e i metodi sono utilizzati a partire dalle istanze. In effetti per ogni istanza che viene creata (invocando un costruttore) è allocata specifica memoria (RAM, nello HEAP) e si linkano i metodi in modo opportuno. In alcuni casi, tuttavia, è la stessa classe (intesa come tipo) che può ospitare delle informazioni e dei comportamenti. In una tale situazione è possibile utilizzare attributi e metodi a partire dalla classe stessa e non dalle sue istanze; anzi, in effetti, le istanze non possono disporre direttamente di questi elementi. Attributi e metodi appartenenti alla classe sono detti statici.

Normalmente un attributo è un dato che viene ad occupare memoria per ciascuna delle istanze create di quel tipo; viceversa un attributo statico è un attributo che occupa memoria condivisa per tutte le istanze di quel tipo. In effetti l'attributo è un dato legato alla classe e non all'istanza.



Nella figura è mostrata la classe con il suo attributo **cont** e tre possibili istanze con i rispettivi attributi **x** ed **y**.

A differenza degli attributi di istanza il cui accesso si specifica da una istanza, l'attributo di classe si specifica dalla classe stessa. Ad esempio (se sono attributi pubblici) si avrebbe:

VC#

```
Punto p = new Punto();
p.x = 13;
Punto.cont = 17;
```

SCOPO DI UN ATTRIBUTO STATICO

Per comprendere lo scopo degli attributi statici si può pensare a una classe che rappresenta un pacchetto di informazioni strutturate e accorpate; per esempio per raggruppare insieme le costanti matematiche è possibile dichiarare una classe (Math) nella quale ospitare queste informazioni.

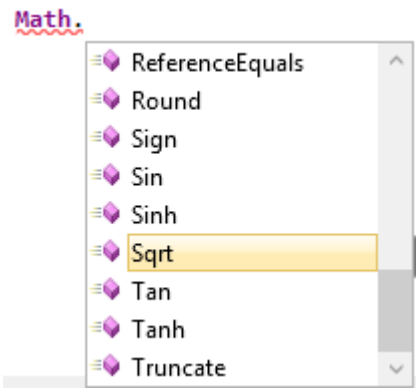
Oltre a costanti è anche possibile avere a disposizione delle locazioni della classe che conservano valori variabili, che evolvono nel tempo, che tengono traccia di informazioni relative all'insieme delle istanze create.

Un esempio classico consiste nel numero di istanze create. Una modalità analoga è mantenere una collezione di riferimenti alle istanze create.

Per esempio in Visual Studio esiste la classe Math che mette a disposizione molte costanti matematiche come:

- E, la costante di Nepero;
- PI, la costante pigreco; Per esempio è possibile usare locazioni come le seguenti:

VC#
```
double d = Math.PI;
if (Math.E < Math.PI)
BackColor = Color.Red;
```



- Lo studente è invitato ad utilizzare la classe Math e a provare le numerose costanti offerte

DEFINIZIONE DI ATTRIBUTI STATICI

Per definire attributi statici in una classe di propria creazione è sufficiente premettere la parola chiave **static**.

VC#
```
class Rectangle
{
    static public int count;
    public Rectangle()
    {
        count++;                //incremento l'attributo statico
    }
}
```

UTILIZZO DI ATTRIBUTI STATICI

L'attributo **static** può essere usato per esempio all'interno del costruttore, come nell'esempio precedente. Se è pubblico può essere usato anche dall'esterno, come da gestori di evento.

VC#
```
Rectangle r = new Rectangle();
MessageBox.Show("" + Rectangle.count);
```

PROGETTO GUIDATO SU ATTRIBUTI STATICI II

- Si prepari un nuovo progetto e si crei una nuova classe come la seguente:

VC#

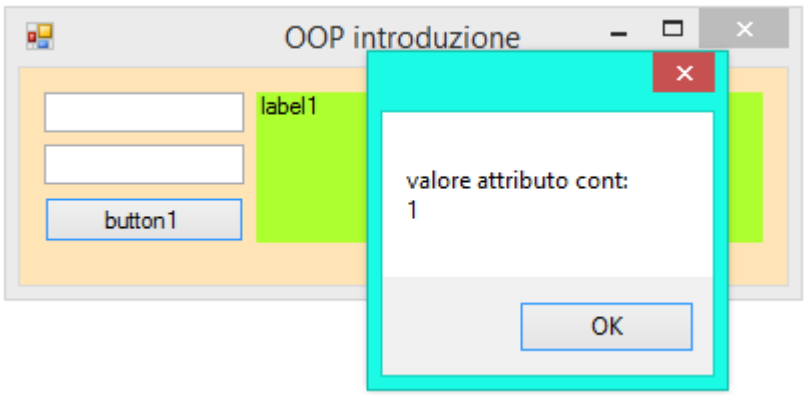
```
class Punto
{
    // attributo statico
    static int cont;
    // attributi di istanza
    int x, y;
    // costruttore
    public Punto()
    {
        x = 3;
        y = 4;
        cont++;
    }
    // metodo pubblico
    public int Quanti()
    {
        return cont;
    }
}
```

- Associa il seguente gestore di evento:

VC#

```
private void button1_Click(object sender, EventArgs e)
{
    Punto p;
    p = new Punto();
    MessageBox.Show("valore attributo cont: \n" + p.Quanti());
    p = new Punto();
    MessageBox.Show("valore attributo cont: \n" + p.Quanti());
    p = new Punto();
    MessageBox.Show("valore attributo cont: \n" + p.Quanti());
}
```

- Prova il progetto



METODI STATICI

PROGETTO GUIDATO SU METODI STATICI

- Si prepari un nuovo progetto e si crei una nuova classe come la seguente:

VC#

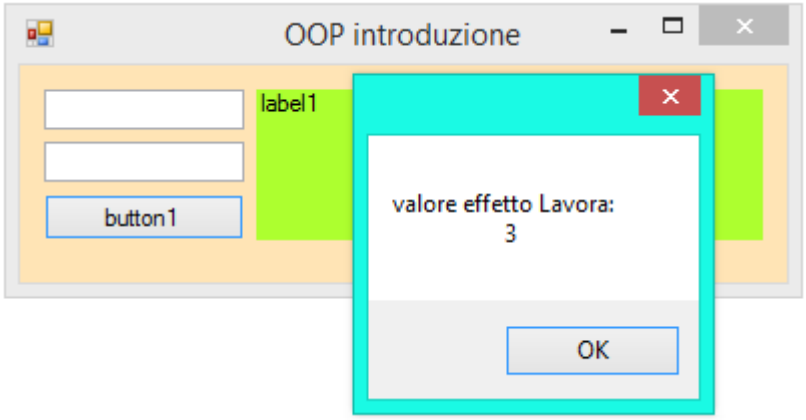
```
class Punto
{
    // attributo statico
    static int cont;
    // attributi di istanza
    int x, y;
    // metodo statico
    static public int Lavora()
    {
        return cont++;
    }
}
```

- Associa il seguente gestore di evento:

VC#

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("valore effetto Lavora: \n\t" + Punto.Lavora());
    MessageBox.Show("valore effetto Lavora: \n\t" + Punto.Lavora());
    MessageBox.Show("valore effetto Lavora: \n\t" + Punto.Lavora());
}
```

- Prova il progetto



CONCETTO DI METODO STATICO

Come già discusso per gli attributi, anche i metodi possono essere dichiarati statici. In questo caso significa che il metodo appartiene alla classe e verrà invocato a partire dalla classe. Il metodo è un comportamento che si riferisce all'intera classe ovvero anche all'insieme delle istanze di quella classe.

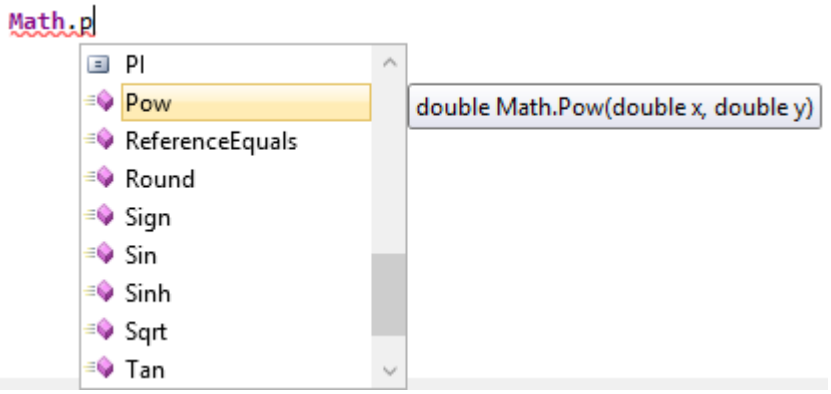
SCOPO DI METODO STATICO

Per comprendere lo scopo degli attributi statici si può pensare a una classe che rappresenta un pacchetto di informazioni strutturate e accorpate; per esempio per raggruppare insieme le costanti matematiche è possibile dichiarare una classe (Math) nella quale ospitare queste informazioni.

Oltre a costanti è anche possibile avere a disposizione delle locazioni della classe che conservano valori variabili, che evolvono nel tempo, che tengono traccia di informazioni relative all'insieme delle istanze create. Un esempio classico consiste nel numero di istanze create. Una modalità analoga è mantenere una collezione di riferimenti alle istanze create.

In riferimento alla classe Math già discussa, questa propone molti metodi statici tra cui:

- **Pow**, che calcola la potenza
- **Sqrt**, che calcola la radice quadrata



Per esempio è possibile usare locazioni come le seguenti:

VC#
```
double d = Math.Sqrt(18);
if (Math.Sin(1/2.0) < Math.Cos(1/2.0))
BackColor = Color.Red;
```

DEFINIZIONE DI METODI STATICI

Per definire metodi statici in una classe di propria creazione è sufficiente premettere la parola chiave **static**.

VC#
```
class Rectangle
{
    static public int count;        //attributo statico

    static public int Calculate()   //metodo statico
    {
        return count;
    }

    public Rectangle() //costruttore di istanza
    {
        count++; //incremento l'attributo statico
    }
}
```

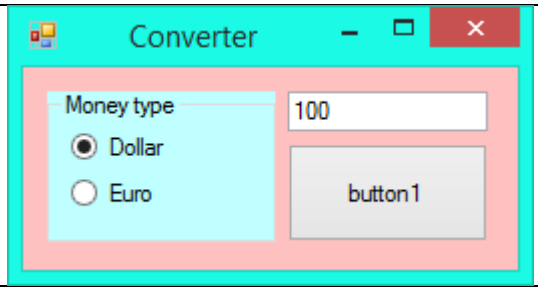
UTILIZZO DI METODI STATICI

Un metodo static può essere invocato sia dall'interno (da altri metodi statici o da costruttori), sia dall'esterno (se pubblico) a partire dalla classe, come da un gestore di evento.

VC#
```
int x = Rectangle.Calculate ();
MessageBox.Show (Convert.ToString (x) );
```

PROGETTO GUIDATO

- Si deve avviare Visual Studio e costruire una finestra come quella della figura a lato
- I radioButton sono inseriti in un GroupBox ma è possibile evitarlo se non lo si desidera
- Definire poi una nuova classe come la seguente:



VC#

```
public class Exchange
{
    static double rate;
    static public void SetRate(double x)
    {
        rate = x;
    }
    static public double DollarToEuros(double x)
    {
        return (x / rate);
    }
    static public double EurosToDollar(double x)
    {
        return (x * rate);
    }
}
```

- Si associ al **button1** il codice seguente:

VC#

```
private void button1_Click(object sender, EventArgs e)
{
    Exchange.SetRate(1.359);
    double y = Convert.ToDouble(textBox1.Text);
    if (radioButton1.Checked)
        y = Exchange.DollarToEuros(y);
    else
        y = Exchange.EurosToDollar(y);
    MessageBox.Show("" + y);
}
```

- Prova il progetto

COSTRUTTORI STATICI

CONCETTO DI UN COSTRUTTORE STATICO

Il costruttore statico è utilizzato per inizializzare una classe ed è richiamato automaticamente dal runtime per inizializzare la classe prima che sia creata la prima istanza e prima che qualsiasi attributo statico sia utilizzato. Il costruttore statico è richiamato una sola volta durante tutto il ciclo di vita dell'applicazione.

SCOPO DI UN COSTRUTTORE STATICO

Se, prima di usare gli attributi statici della classe e di invocare metodi statici della classe, ma anche prima di usare qualsiasi istanza della classe, è necessario che una classe sia preparata (inizializzata, ben formata, costruita, pronta) si può affidare a un costruttore statico il compito di questa preparazione.

Un costruttore statico consente di inizializzare gli eventuali dati statici oppure di eseguire un'operazione specifica che deve essere effettuata una sola volta. Viene chiamato automaticamente prima che ne venga creata la prima istanza o venga fatto riferimento a qualsiasi membro statico.

Non è possibile dichiarare modificatori di accesso (es. **public** o **private**) per costruttore statici; il costruttore è implicitamente pubblico ed è uno solo.

Un costruttore statico viene chiamato automaticamente per inizializzare la classe prima che ne venga creata la prima istanza o venga fatto riferimento a qualsiasi membro statico. Un costruttore statico non può essere chiamato direttamente. L'utente non può controllare in alcun modo il momento in cui il costruttore statico viene eseguito nel programma. Un costruttore statico NON può avere parametri, poiché non ha una invocazione esplicita e non si potrebbero passargli argomenti.

Di conseguenza una classe NON può avere più costruttori statici, che non sarebbero distinguibili gli uni dagli altri. Una classe ammette al massimo un costruttore statico. Se un costruttore statico genera un'eccezione, il runtime non lo richiamerà una seconda volta e il tipo rimarrà non inizializzato per la durata del dominio dell'applicazione in cui il programma è in esecuzione.

DEFINIZIONE DI UN COSTRUTTORE STATICO

Per definire il costruttore statico occorre specificare la parola riservata **static** e evitare di specificare sia la modalità di accesso (**public** o **private**) sia il tipo (un costruttore non ne ha):

PROGETTO GUIDATO CON COSTRUTTORE STATICO

- Si dichiari un nuovo progetto e si definisca una classe come la seguente:

VC#

```
public class Mostro
{
    static int quanti;
    static Mostro() //implicitamente public
    {
        quanti = 0;
    }
    public Mostro() //costruttore di istanza
    {
        quanti++;
    }
    public string Info()
    {
        return ("" + quanti);
    }
}
```

- Poi si disegni un form1 simile alla figura
- Si associ un codice per il pulsante button1 come il seguente:

VC#
```
            Mostro m = new Mostro();
            Mostro n = new Mostro();
            Mostro p = new Mostro();
            MessageBox.Show("Hai creato " + m.Info() + " mostri");
```

- Prova il progetto

UTILIZZO DI UN COSTRUTTORE STATICO

Se una classe Mostro dispone di un costruttore statico, esso sarà richiamato quando viene creata la prima istanza di Mostro (m), il costruttore statico viene richiamato per inizializzare la classe. Il costruttore statico è eseguito una sola volta, anche se vengono create due istanze della stessa classe, e sarà eseguito prima del costruttore di istanza.

Poiché un costruttore statico non può essere chiamato direttamente e l'utente non può invocarlo in alcun modo il momento non si discuterà il suo uso come invocazione; tuttavia al costruttore statico possono essere affidate tutte le operazioni che preparano la classe e le sue istanze prima di ogni altra cosa. In riferimento all'esempio precedente è possibile definire il seguente gestore di evento:

VC#
```
Mostro n;
Mostro m = new Mostro(); //qui si chiamerà il costruttore statico
m.Info();
```

PROGETTO GUIDATO

- Si deve avviare Visual Studio e costruire una finestra come quella della figura
- Visualizzare il codice e definire una nuova classe;
- Nel file aggiungi la libreria delle collezioni

VC#
```
using System.Collections;
```

- E poi definisci la classe nel seguente modo:

VC#
```
    public class Mostro
    {
        static List<Mostro> schiera;
        static int inizio = 5;
        int forza = 0;
        static Mostro() //implicitamente public
        {
            inizio = 10;
            schiera = new List<Mostro>();
        }
//continua →
```

```
        public Mostro() //costruttore di istanza
        {
            inizio += 5;
            forza = inizio;
            schiera.Add(this);
        }
        public string Info()
        {
            string s = "";
            foreach (Mostro m in schiera)
                s += m.forza + "; ";
            return s;
        }
    }
```

- Si associ al button1 il codice seguente:

VC#

```
        private void button1_Click(object sender, EventArgs e)
        {
            Mostro m = new Mostro(); //invoca costruttore di istanza
            string ris = m.Info();
            MessageBox.Show(ris);
            m = new Mostro(); //invoca costruttore di istanza
            ris = m.Info();
            MessageBox.Show(ris);
            m = new Mostro(); //invoca costruttore di istanza
            ris = m.Info();
            MessageBox.Show(ris);
        }
```

- Prova il progetto: hai capito cosa accade?

CLASSI STATICHE

CONCETTO DI CLASSE STATICA

Le classi statiche e i relativi elementi (attributi e metodi) vengono utilizzati per creare dati e algoritmi da utilizzare senza creare istanze della classe. È possibile utilizzare i membri delle classi statiche per separare dati e comportamenti indipendenti dall'identità di qualsiasi oggetto. In questo modo, infatti, i dati e le funzioni non cambiano, indipendentemente dalle operazioni eseguite sull'oggetto.

Le classi statiche possono essere utilizzate quando la classe non contiene dati o comportamenti che dipendono dall'identità dell'oggetto. La classe quindi coincide con le informazioni (dati e algoritmi) che mette a disposizione, e non costituisce invece il modello su cui costruire elementi; la classe statica quindi non è una categoria ma un oggetto in sé.

Una classe statica è quindi utilizzata per definire metodi da non associare a particolari istanze, ma i metodi sono funzioni in sé. In definitiva la classe è utile per creare un insieme di metodi che non agiscano sui dati delle istanze e non siano associati a un oggetto specifico da istanziare. Quindi la classe ammette solo metodi statici. Le classi statiche hanno le seguenti caratteristiche:

- Contengono solo attributi e metodi statici.
- Non consentono di creare istanze.
- Sono sigillate (sealed) cioè non permettono di essere ereditate.
- Non possono contenere costruttori di istanze (ma ammettono un costruttore statico).

Tra i vantaggi di usare una classe statica si può annoverare quello di consentire al compilatore di verificare che non vengano aggiunti accidentalmente membri di istanze e di garantire che non vengano create istanze di questa classe.

SCOPO DI CLASSE STATICA

Un motivo per dichiarare una classe statica è per indicare che contiene solo membri statici. Un altro motivo è impedire di creare sue istanze mediante la parola chiave new. Una classe statica è in genere utilizzata per costituire un'unità logica di dati e metodi non associati a oggetti specifici. Rende più semplice e rapida l'implementazione, e rende inutile la creazione di oggetti per invocare i relativi metodi.

Un esempio di classe statica è la classe Math nello spazio dei nomi System. I suoi membri statici sono utilizzati per rappresentare dati e calcoli che non dipendono da specifiche istanze. La libreria matematica contiene per esempio attributi statici che rappresentano Pi Greco e costante di Nepero e metodi statici per il calcolo di seno e coseno.

DEFINIZIONE DI CLASSE STATICA

I membri delle classi statiche vengono dichiarati utilizzando la parola chiave static prima del tipo restituito. Ad esempio consideriamo una classe statica che contiene due metodi che consentono rispettivamente di convertire i valori relativi alla temperatura da gradi Celsius a gradi Fahrenheit e viceversa:

VC#

```
public static class ConvertiTemperature
{
    public static double CelsiusInFahrenheit(double tCelsius)
    {
        // Converte Celsius in Fahrenheit.
        double fahrenheit = (tCelsius * 9 / 5) + 32;
        return fahrenheit;
    }
    public static double FahrenheitInCelsius(double tFahrenheit)
    {
        // Converte Fahrenheit in Celsius.
        double celsius = (tFahrenheit - 32) * 5 / 9;
        return celsius;
    }
}
```

I membri statici vengono inizializzati prima dell'accesso iniziale e prima dell'eventuale chiamata al costruttore statico. Per accedere a un membro di una classe statica, utilizzare il nome della classe anziché il nome di una variabile per specificare la posizione del membro.

UTILIZZO DI CLASSI STATICHE

Una classe statica viene usata senza invocare costruttori di istanze, peraltro vietati.

VC#

```
double c, f;
c = Convert.ToDouble (textBox1.Text);
f = ConvertiTemperature.CelsiusInFahrenheit ( c );
label1.Text = Convert.ToString (f);
```

ESERCIZI

ESERCIZI

ESERCIZI SULLE CLASSI STATICHE

ESERCIZIO 1. CLASSE STATICA RANDOMINT

Si definisca una classe statica `RandomInt` che senza bisogno di creare istanze offra tutti i metodi di generazione di numeri interi casuali previsti per una istanza di tipo Random.

ESERCIZIO 2: CLASSE STATICA RANDOMDOUBLE

Si definisca una classe statica `RandomDouble` che senza bisogno di creare istanze offra i seguenti metodi statici:

1. Metodo senza parametri che rende un numero con la virgola compreso tra zero (incluso) e 1 (escluso);
2. Metodo con un parametro numerico che rende un numero con la virgola compreso tra zero (incluso) e il numero dato (escluso);
3. Metodo con due parametri numerici che rende un numero con la virgola compreso tra i due estremi, il minore incluso ma il maggiore escluso;

ESERCIZIO 3: CLASSE STATICA CONVERTTOBOOL

Si definisca una classe statica `ConvertToBool` che senza bisogno di creare istanze offra i seguenti metodi statici:

1. Conversione da string a bool;
2. Conversione da bool a string;
3. Conversione da int a bool (zero vale falso, il resto a true);
4. Conversione da bool a int (falso rende zero, altrimenti uno);

ESERCIZIO 4: CLASSE STATICA CAMBIAVALUTA

Si definisca una classe statica CambiaValuta che senza bisogno di creare istanze, definisca attributi statici che rappresentano il cambio tra Dollaro, Euro, Yen e Yuan ed i seguenti metodi statici:

1. Tre metodi di conversione da Dollaro nelle altre tre valute;
2. Tre metodi di conversione da Euro nelle altre tre valute;
3. Tre metodi di conversione da Yen nelle altre tre valute;
4. Tre metodi di conversione da Yuan nelle altre tre valute;

ESERCIZIO 5: CLASSE STATICA CONVERTITORE

Si definisca una classe statica Convertitore che senza bisogno di creare istanze, definisca metodi statici che rappresentano:

1. Conversione da intero (sistema decimale) a intero (sistema binario)
2. Conversione da intero (sistema decimale) a intero (sistema ottale)
3. Conversione da intero (sistema decimale) a intero (sistema esadecimale)
4. Conversione da intero (sistema binario) a intero (sistema decimale)
5. Conversione da intero (sistema ottale) a intero (sistema decimale)
6. Conversione da intero (sistema esadecimale) a intero (sistema decimale)

SOMMARIO