



Introduzione alla libreria di classi BioLab

Raffaele Cappelli
raffaele.cappelli@unibo.it

Contenuti

- Introduzione
- Dati
 - Classe Data
 - Immagini
- Algoritmi
 - Interfaccia IAlgorithm
 - Attributi, eventi ed eccezioni
 - Esempi
- Anteprima ed esecuzione di algoritmi
 - Classe AlgorithmPreviewForm
- Visualizzazione dei dati
 - Classe DataViewer e sue derivate
- Il programma PRLab

Introduzione

■ Libreria BioLab

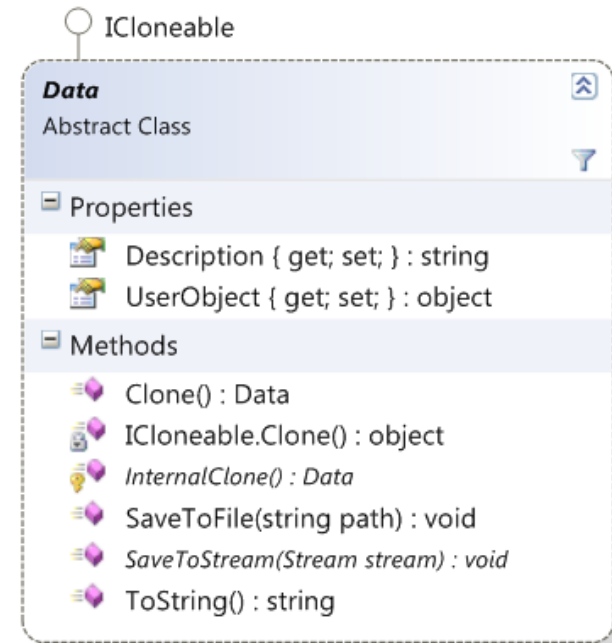
- Insieme di classi per elaborazione di immagini, classificazione e riconoscimento di forme
- Classi sviluppate principalmente per la didattica
 - Semplifica lo sviluppo di algoritmi di elaborazione immagini e pattern recognition
- Utilizzata anche nel corso «Visione Artificiale e Riconoscimento» della laurea magistrale
- È utilizzata con successo anche per scopi di ricerca
 - Sviluppo rapido di nuovi algoritmi e prototipi software
- Sviluppata in C#

■ Libreria BioLab.Biometrics

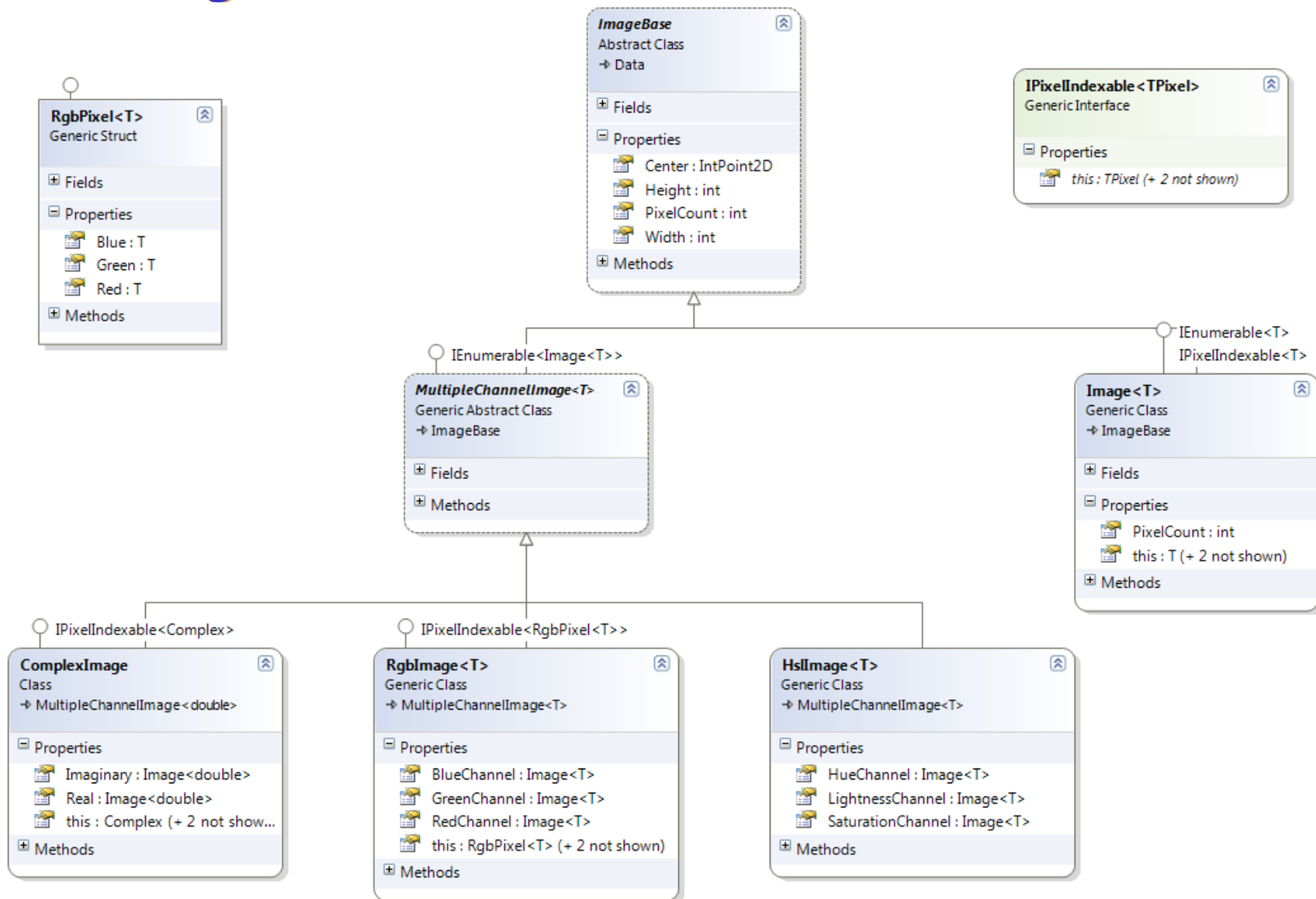
- Contiene dati, visualizzatori e algoritmi per applicazioni di riconoscimento biometrico (impronte digitali, volto)
- Utilizzata nel corso «Sistemi Biometrici» della laurea magistrale

Classe Data (BioLab.Common.Data)

- I principali dati su cui la libreria opera sono classi derivate da questa classe astratta
 - Implementa ICloneable.
 - Contiene metodi per salvare su file o stream.
 - Contiene una proprietà (Description) per aggiungere una descrizione testuale del contenuto e una (UserObject) per associare un eventuale riferimento a un altro oggetto.
- Le classi derivate devono:
 - Implementare i metodi astratti *SaveToStream* (salvataggio dei contenuti su uno stream) e *InternalClone* (duplicazione del contenuto).
 - Fornire metodi statici per il caricamento da stream (*LoadFromStream*) e da file (*LoadFromFile*).



Immagini



Immagini a livelli di grigio – Esempi

```
// Creazione di una nuova immagine con dimensioni 200x100
var img1 = new Image<byte>(200, 100);

// Duplicazione di un'immagine
var img2 = img1.Clone();

// Proprietà Width ed Height
MessageBox.Show(String.Format("Dimensioni: {0}x{1}", img1.Width, img1.Height));

// Accesso a un pixel mediante indice lineare
img1[200] = 3;

// Accesso a un pixel mediante coordinate [y,x]
img1[10, 20] = 30;

// E' anche possibile scorrere (in lettura) tutti i pixel utilizzando foreach
double avg = 0;
foreach (byte pixel in img1)
    avg += pixel;
avg /= img1.PixelCount;
```

Immagini RGB – Esempi

```
// Creazione di una nuova immagine RGB con dimensioni 320x200
var img1 = new RgbImage<byte>(320, 200);

// Copia del canale R come immagine gray scale
var imgR = img1.RedChannel.Clone();

// Proprietà Width ed Height
MessageBox.Show(String.Format("Dimensioni: {0}x{1}", img1.Width, img1.Height));

// Accesso a un pixel mediante indice lineare
img1[200] = new RgbPixel<byte>(255, 0, 128);
byte r = img1[200].Red; // 255
byte b = img1[200].Blue; // 128

// Accesso a un pixel mediante coordinate [y,x]
img1[10, 20] = new RgbPixel<byte>(0, 0, 0);

// Convertete in un'immagine gray scale
var imgG = img1.ToByteImage(); // Extension method

// Crea un'immagine specificando il contenuto dei 3 canali
var img2 = new RgbImage<byte>(img1.RedChannel, imgG, img1.BlueChannel);
```

Algoritmi

■ Algoritmi: interfaccia IAlgorithm

- All'interno della libreria, i principali algoritmi sono implementati non come semplici metodi ma come classi che implementano l'interfaccia IAlgorithm e utilizzano particolari attributi definiti nella libreria stessa.
- È disponibile una classe astratta Algorithm che implementa IAlgorithm e fornisce alcune funzionalità di base utili alla maggior parte degli algoritmi.



Algoritmi (2)

■ Gli attributi:

□ [AlgorithmInfo]

- Va associato alla classe: permette di specificare il nome dell'algoritmo, insieme ad altre informazioni testuali (categoria, descrizione, riferimenti bibliografici)

□ [AlgorithmInput]

- Va associato a ciascuna proprietà della classe che rappresenta un dato di input dell'algoritmo.

□ [AlgorithmOutput]

- Va associato a ciascuna proprietà della classe che rappresenta un dato di output dell'algoritmo.

□ [AlgorithmParameter]

- Va associato a ciascuna proprietà della classe che rappresenta un parametro dell'algoritmo.

Algoritmi (3)

■ Eventi ed eccezioni:

□ Evento ProgressChanged

- Viene generato dall'algoritmo per comunicare il proprio stato di avanzamento.
- È opportuno utilizzarlo solo in quegli algoritmi che potenzialmente richiedono tempi lunghi di esecuzione.

□ Evento IntermediateResult

- Viene generato dall'algoritmo per fornire eventuali risultati intermedi dell'elaborazione.
- È utile principalmente per scopi didattici, è opportuno utilizzarlo solo in algoritmi con risultati intermedi significativi.

□ Eccezione AlgorithmAbortedException

- Indica che l'algoritmo è stato interrotto dall'utente (nel caso l'interfaccia utente che si utilizza permetta questa funzionalità).

Algoritmi - Esempio

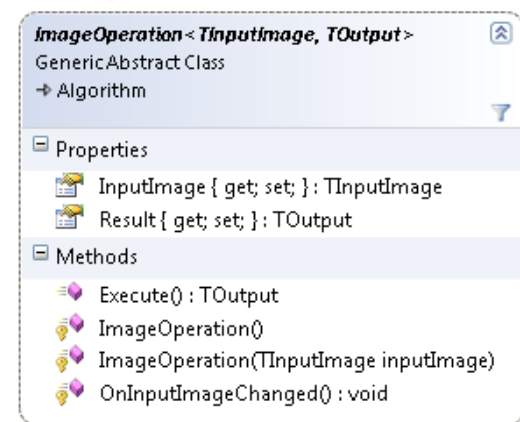
```
[AlgorithmInfo("Calcola livello medio di grigio",Category="FEI",
  Description="Questo algoritmo calcola il valore medio dei valori di grigio " +
    "in un'immagine che sono inferiori alla soglia specificata")]
public class CalcolaLivelloGrigioMedio : Algorithm
{
  [AlgorithmInput] public Image<byte> Immagine { get; set; }
  [AlgorithmOutput] public double Media { get; private set; }
  [AlgorithmParameter] public byte Soglia { get; set; }

  public override void Run()
  {
    int n = 0;
    int sum = 0;
    for (int i = 0; i < Immagine.PixelCount; i++)
    {
      var p = Immagine[i];
      if (p < Soglia) { sum += p; n++; }
      if (i % 100 == 0)
      {
        if (!OnProgressChanged(new AlgorithmProgressChangedEventArgs(
          (double)i / Immagine.PixelCount, "Calcolo media")))
          throw new AlgorithmAbortedException();
      }
    }
    Media = n > 0 ? (double)sum / n : 0;
  }
}
```

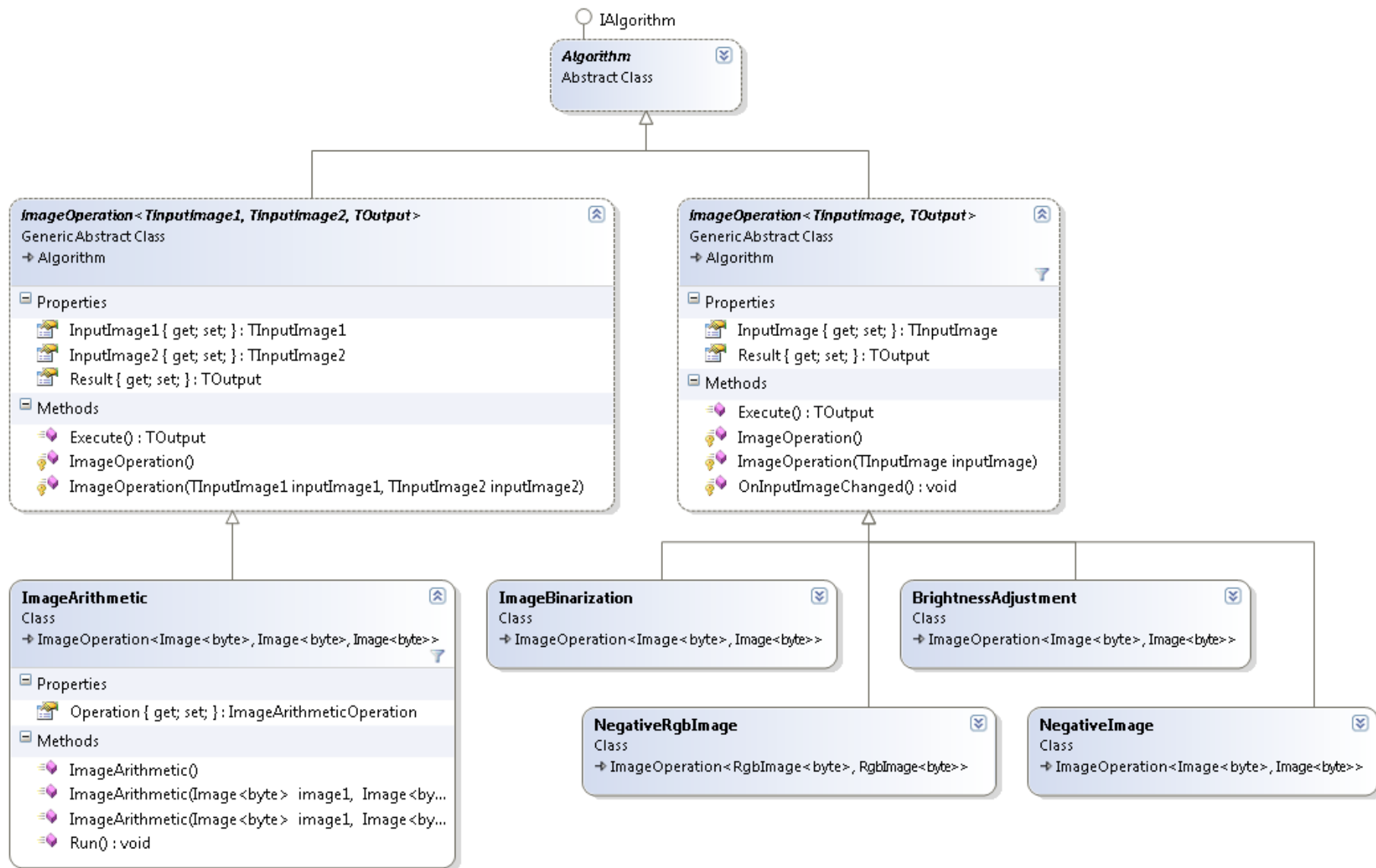
Operazioni sulle immagini: ImageOperation

■ ImageOperation<...>

- Classe generica e astratta
- Utile come classe base per algoritmi di elaborazione immagini
- Type parameters:
 - TInputImage: il tipo dell'immagine di input (es. Image<byte>)
 - TOutputImage: il tipo del risultato (solitamente un'immagine, ma può essere anche un altro tipo di dato)
- Proprietà:
 - InputImage (attributo [AlgorithmInput], di tipo TInputImage)
 - Result (attributo [AlgorithmOutput], di tipo TOutput)
- Metodi:
 - Execute(): chiama il metodo Run() (astratto nella classe base) che esegue l'operazione e ritorna il valore memorizzato in Result.



Operazioni sulle immagini: ImageOperation (2)



Esempio di classe derivata da ImageOperation

■ ImageArithmetic

- Operazioni aritmetiche fra due immagini grayscale

```
// Crea un'istanza della classe per eseguire operazioni aritmetiche
// fra le immagini img1 e img2
// Imposta il tipo di operazione a "And"
var op = new ImageArithmetic(img1, img2, ImageArithmeticOperation.And);

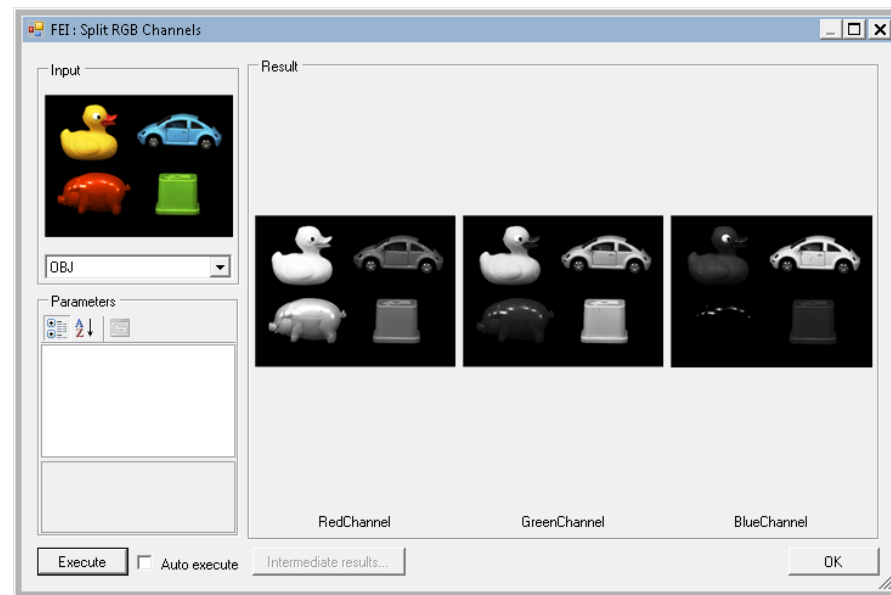
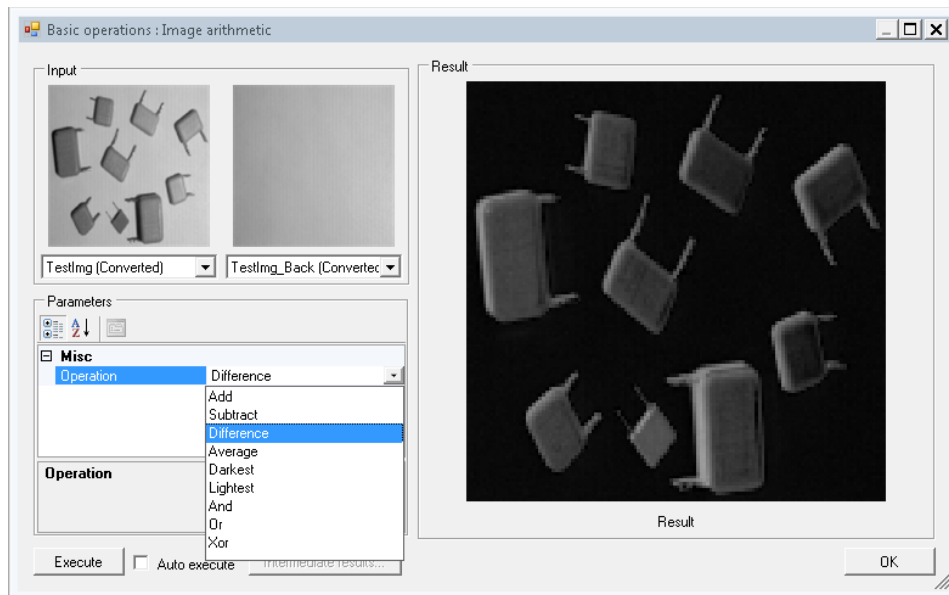
// Esegue l'operazione (restituisce una nuova immagine)
var imgAnd = op.Execute();

// Modifica il parametro tipo di operazione a "Xor"
op.Operation = ImageArithmeticOperation.Xor;

// Esegue l'operazione con il parametro modificato
var imgXor = op.Execute();
```

Esecuzione algoritmi con anteprima dei risultati

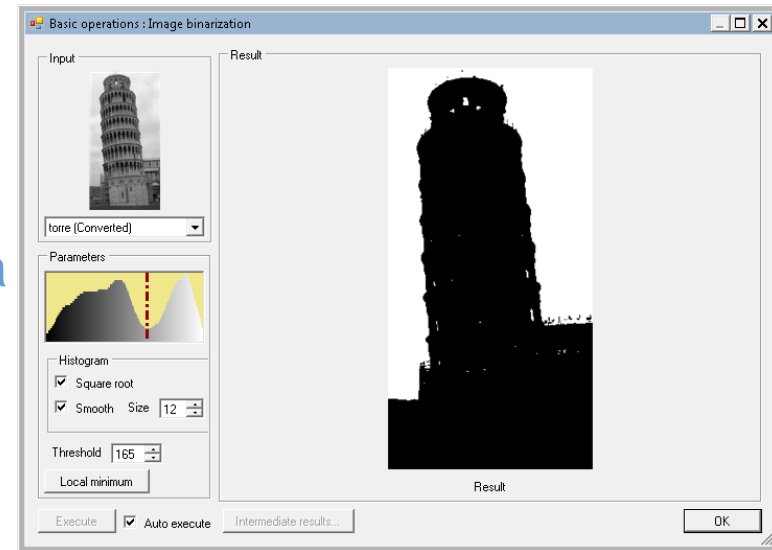
- La classe **AlgorithmPreviewForm** consente di
 - visualizzare un'anteprima dei risultati di un algoritmo (classe che implementa IAlgorithm).
 - selezionare gli input fra quelli disponibili e modificare gli eventuali parametri dell'algoritmo.
 - visualizzare eventuali risultati intermedi dell'algoritmo.



AlgorithmPreviewForm: personalizzazione

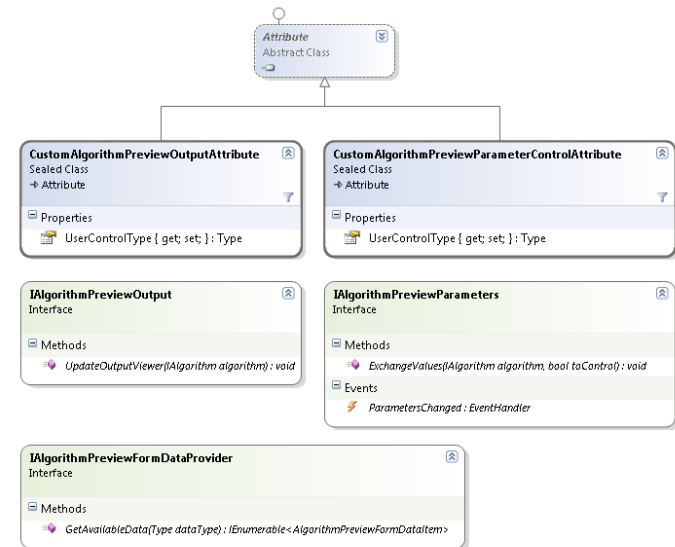
[CustomAlgorithmPreviewParameterControl()]

- Associando tale attributo alla classe che implementa l'algoritmo, è possibile personalizzare la visualizzazione e modifica dei parametri.
- A tal fine è necessario implementare uno UserControl che implementa l'interfaccia IAlgorithmPreviewParameters



Altre possibili personalizzazioni:

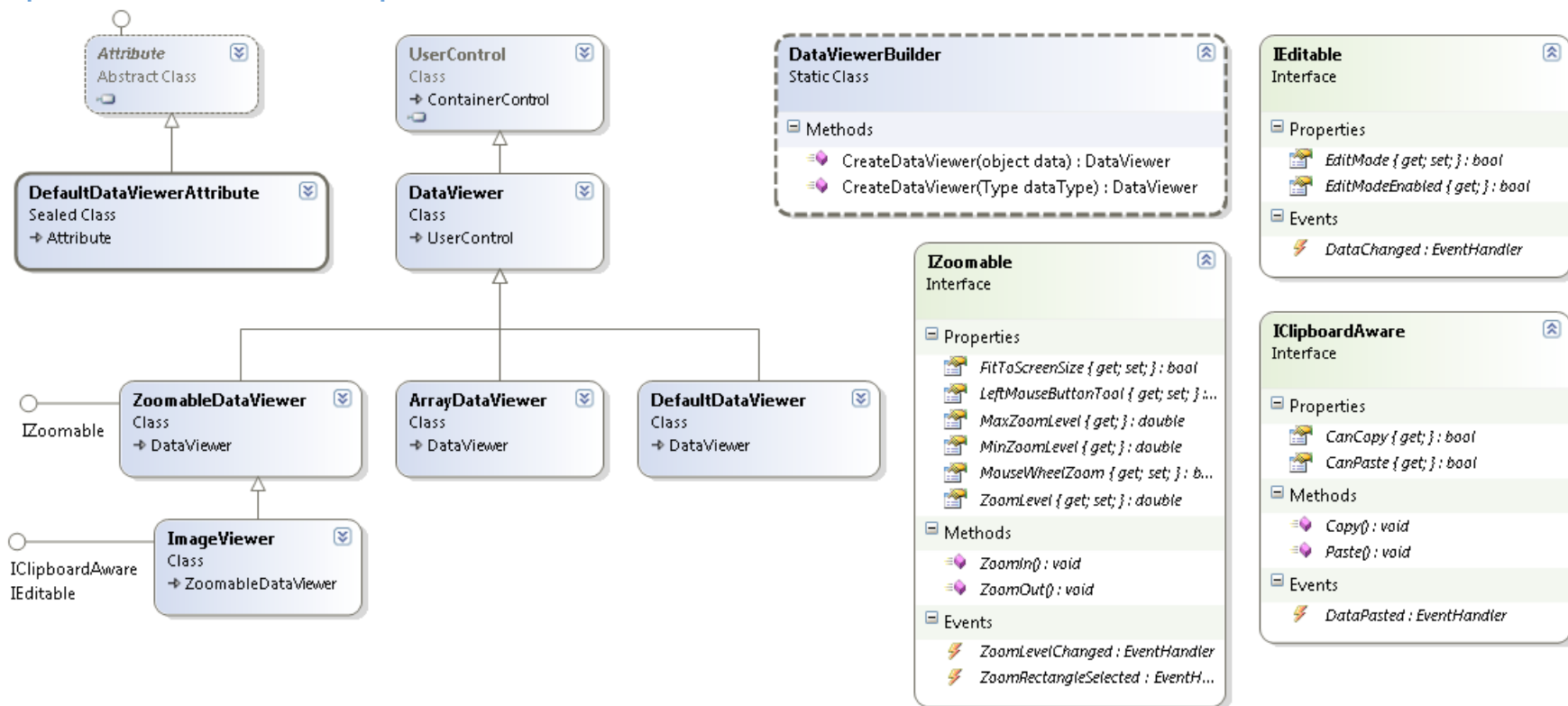
- Fornire un elenco di possibili input (interfaccia IAlgorithmPreviewDataProvider)
- Definire un visualizzatore specifico per i risultati di un algoritmo (attributo [CustomAlgorithmPreviewOutput] da associare alla classe dell'algoritmo e interfaccia IAlgorithmPreviewOutput che lo user control deve implementare)



DataViewer e classi derivate

■ Generico UserControl per la visualizzazione dei dati

- Interfacce IZoomable (funzionalità di zoom), IEditable (possibilità di modifica dei dati), IClipboardAware (copia e incolla dalla clipboard)
- Attributo [DefaultDataViewer(...)] per associare un particolare visualizzatore a un tipo di dati
- La classe statica DataViewerBuilder fornisce metodi per creare il visualizzatore più appropriato per un determinato tipo di dati.



PRLab

- Programma con interfaccia utente basata su Windows Form
 - Permette di utilizzare gli algoritmi contenuti nella libreria BioLab, con particolare riferimento all'elaborazione delle immagini

