Tentu, ini adalah penjelasan mendalam mengenai kode ImageEditorService.cs.

Secara garis besar, kelas ini berfungsi sebagai **Service Layer** atau "jembatan" antara tampilan antarmuka (UI) dengan logika pemrosesan gambar yang lebih rumit. Kelas ini menyimpan gambar yang sedang diedit dan memanggil fungsi-fungsi pemrosesan spesifik (seperti mengubah kecerahan, binerisasi, histogram, dll).

Berikut adalah penjelasannya baris demi baris (dikelompokkan berdasarkan fungsi logisnya):

---

1. Import Library dan Namespace

C#

```
using System.Drawing;
using System.Windows.Forms;
using MiniPhotoshop.Logic.Helpers;
using MiniPhotoshop.Logic.ImageProcessing;
using MiniPhotoshop.Logic.Histogram;

namespace MiniPhotoshop.Logic
```

- using System.Drawing;: Mengimpor pustaka standar .NET untuk manipulasi grafik dasar, seperti Bitmap, Color, dan Point.
- using System.Windows.Forms;: Mengimpor pustaka untuk komponen antarmuka Windows (kemungkinan digunakan untuk tipe data umum UI, meskipun tidak terlihat penggunaan kontrol UI secara eksplisit di sini).
- using MiniPhotoshop.Logic...: Mengimpor *helper classes* buatan sendiri (kode lain dalam proyek Anda) yang berisi rumus matematika untuk histogram dan pemrosesan gambar.
- namespace MiniPhotoshop.Logic: Mendefinisikan bahwa kelas ini berada dalam ruang lingkup (folder logika) aplikasi MiniPhotoshop.

---

2. Definisi Kelas dan Properti Utama

C#

```
public class ImageEditorService
{
    public Bitmap OriginalImage { get; private set; }
    private int[,,] _imageData3D;

    public bool IsImageLoaded => OriginalImage != null;
```

- public class ImageEditorService: Mendefinisikan kelas publik yang bisa diakses dari bagian lain program.
- public Bitmap OriginalImage { get; private set; }: Properti untuk menyimpan gambar asli. public artinya bisa dibaca dari luar, tapi private set artinya hanya bisa diubah nilainya dari dalam kelas ini (untuk keamanan data).
- private int[,,] _imageData3D;: Variabel *private* bertipe array 3 dimensi. Ini kemungkinan besar menyimpan data piksel gambar dalam format matriks (Baris, Kolom, Warna/Channel) untuk mempermudah perhitungan matematis.
- public bool IsImageLoaded => OriginalImage != null;: Properti *boolean* sederhana untuk mengecek apakah gambar sudah dimuat. Bernilai true jika OriginalImage tidak kosong.

---

3. Inisialisasi Gambar

C#

```
    public void InitializeImage(Bitmap image)
    {
        OriginalImage = new Bitmap(image);
        _imageData3D = ImageDataProcessor.LoadTo3DArray(OriginalImage);
    }
```

- InitializeImage(Bitmap image): Fungsi untuk memuat gambar baru ke dalam sistem.
- OriginalImage = new Bitmap(image);: Membuat salinan (clone) dari gambar yang

dimasukkan agar gambar sumber tidak terpengaruh.
- _imageData3D = ImageDataProcessor.LoadTo3DArray...: Mengonversi gambar Bitmap menjadi array 3D menggunakan kelas pembantu ImageDataProcessor. Ini persiapan untuk pemrosesan tingkat lanjut (seperti manipulasi per channel warna).

---

4. Mengambil Gambar

C#

```
public Bitmap GetRestoredImage()
{
    if (!IsImageLoaded) return null;
    return (Bitmap)OriginalImage.Clone();
}
```

- GetRestoredImage(): Fungsi untuk mereset editan atau mengambil gambar versi awal.
- if (!IsImageLoaded) return null;: Pengecekan keamanan (guard clause). Jika belum ada gambar, kembalikan null.
- return (Bitmap)OriginalImage.Clone();: Mengembalikan **salinan** dari gambar asli. Penting menggunakan .Clone() agar jika UI mengubah gambar hasil kembalian ini, gambar OriginalImage yang disimpan di servis tidak ikut rusak.

---

5. Manipulasi Channel dan Grayscale

C#

```
public Bitmap GetChannel(int channel)
{
    if (!IsImageLoaded) return null;
    return ImageDataProcessor.CreateBitmapFrom3DArray(_imageData3D, channel);
}
```

```
public Bitmap ApplyGrayscale()
{
    return GetChannel(3);
}
```

- GetChannel(int channel): Mengambil gambar berdasarkan channel warna tertentu (misalnya 0=Biru, 1=Hijau, 2=Merah). Ia membangun kembali Bitmap dari array 3D yang disimpan.
- ApplyGrayscale(): Fungsi khusus untuk membuat gambar hitam putih (abu-abu).
- return GetChannel(3);: Di sini terlihat bahwa logika program Anda menetapkan indeks 3 sebagai representasi channel Grayscale dalam _imageData3D.

---

6. Transformasi Gambar (Negasi, Binerisasi, Brightness)

Bagian ini mengikuti pola yang sama: Validasi -> Panggil Algoritma -> Return Hasil.

C#

```
public Bitmap ApplyNegation(Bitmap currentImage)
{
    if (!IsImageLoaded) return null;
    return ImageNegation.Apply(currentImage);
}

public Bitmap ApplyBinarization(int step)
{
    if (!IsImageLoaded) return null;
    return BlackWhite.ApplyBinarization(OriginalImage, step);
}

public Bitmap ApplyBrightness(int adjustment)
{
    if (!IsImageLoaded) return null;
    return Brightness.AdjustBrightness(OriginalImage, adjustment);
}
```

- ApplyNegation: Memanggil kelas statis ImageNegation untuk membalikkan warna (efek klise film).
- ApplyBinarization: Mengubah gambar menjadi murni hitam dan putih (tanpa abu-abu) berdasarkan nilai ambang batas (step). Menggunakan kelas helper BlackWhite.
- ApplyBrightness: Mengatur kecerahan gambar berdasarkan nilai adjustment (+ atau -), menggunakan kelas helper Brightness.

---

7. Seleksi Warna dan Histogram

C#

```
public Bitmap ApplyColorSelection(Point imagePoint)
{
    if (!IsImageLoaded) return null;
    Color clickedColor = OriginalImage.GetPixel(imagePoint.X, imagePoint.Y);
    return SelectionColor.ApplySelection(OriginalImage, clickedColor);
}
```

- ApplyColorSelection(Point imagePoint): Menerima koordinat (X, Y) di mana pengguna mengklik gambar.
- Color clickedColor = ...: Mengambil data warna pixel tepat di koordinat tersebut.
- SelectionColor.ApplySelection...: Memanggil algoritma untuk menyeleksi/menonjolkan area lain di gambar yang memiliki warna mirip dengan warna yang diklik.

C#

```
public Bitmap GetHistogram(int channel, int width, int height)
{
    if (!IsImageLoaded) return null;
    int[] hist = HistogramCalculator.Calculate(OriginalImage, channel);
    return HistogramDrawer.Draw(hist, channel, width, height);
}
```

- GetHistogram(...): Meminta grafik histogram untuk channel warna tertentu dengan ukuran

gambar output width x height.

- HistogramCalculator.Calculate...: Menghitung frekuensi kemunculan setiap nilai pixel (statistik data gambar) dan menyimpannya di array integer.
- HistogramDrawer.Draw...: Menggambar grafik batang visual berdasarkan data array tersebut menjadi sebuah gambar Bitmap.

---

8. Pembersihan Memori

C#

```
public void ClearImage()
{
    OriginalImage = null;
    _imageData3D = null;
}
```

- ClearImage(): Menghapus referensi ke gambar dan data array.
- Ini penting untuk membebaskan memori (RAM) ketika pengguna menutup gambar atau ingin memuat gambar baru, agar aplikasi tidak menjadi berat (Memory Leak).

Langkah Selanjutnya:
Apakah Anda ingin saya menjelaskan logika matematika di balik salah satu kelas helper yang dipanggil di sini, misalnya bagaimana ImageDataProcessor mengubah gambar menjadi Array 3D?