

Baik 🙌 berikut penjelasan **baris demi baris** untuk program `OperationManager.cs` ini — program ini berfungsi sebagai **pengatur operasi gambar** di aplikasi *MiniPhotoshop*. Penjelasan dibagi berdasarkan blok kode agar tetap rapi dan mudah dipahami.

✖ Bagian `using` dan Namespace

```
using MiniPhotoshop.Logic;
using MiniPhotoshop.Logic.ImageProcessing;
using System.Collections.Generic;
using System.Drawing;
using System.Globalization;
using System.Windows.Forms;
```

- `using MiniPhotoshop.Logic;` dan `using MiniPhotoshop.Logic.ImageProcessing;` → Mengimpor namespace internal proyek yang berisi logika pengolahan gambar seperti `ArithmeticOperations`, `LogicalOperations`, dan `ConstantOperations`.
 - `using System.Collections.Generic;` → Untuk menggunakan koleksi seperti `Dictionary`.
 - `using System.Drawing;` → Untuk manipulasi gambar (`Bitmap`, `Color`, dll).
 - `using System.Globalization;` → Agar dapat membaca angka dari `TextBox` dengan format internasional (misal: titik desimal).
 - `using System.Windows.Forms;` → Untuk elemen GUI seperti `PictureBox`, `RadioButton`, `TextBox`, dan `MessageBox`.
-

🏠 Deklarasi Kelas dan Variabel

```
namespace MiniPhotoshop.Helpers
{
    public class OperationManager
    {
        private readonly ImageEditorService _editorService;
```

- Membuka namespace `MiniPhotoshop.Helpers` yang berisi kelas bantuan.
 - `public class OperationManager` → Kelas utama untuk mengelola operasi gambar.
 - `_editorService` adalah referensi ke **layanan editor utama**, tempat gambar A (gambar utama) disimpan dan dimanipulasi.
-

✖ Variabel Tambahan

```
private readonly Dictionary<RadioButton, PictureBox> _thumbnailMap;
private readonly TextBox _constantTextBox;
```

- `_thumbnailMap` → Peta (dictionary) yang menghubungkan setiap `RadioButton` dengan `PictureBox` terkait (gambar mini).
 - `_constantTextBox` → `TextBox` tempat pengguna memasukkan nilai konstanta K untuk operasi tertentu (misal: *Multiply by K*).
-

⚙ Constructor

```
public OperationManager(ImageEditorService editorService,  
                        Dictionary<RadioButton, PictureBox> thumbnailMap,  
                        TextBox constantTextBox)  
{  
    _editorService = editorService;  
    _thumbnailMap = thumbnailMap;  
    _constantTextBox = constantTextBox;  
}
```

- Konstruktor menerima 3 parameter:
 - `editorService`: pengelola gambar utama.
 - `thumbnailMap`: daftar gambar kecil yang bisa dijadikan gambar B.
 - `constantTextBox`: input konstanta dari pengguna.
 - Semua nilai disimpan ke variabel kelas untuk digunakan oleh metode lain.
-

🔍 Fungsi `GetImageB_FromRadioButtons()`

```
private Bitmap GetImageB_FromRadioButtons()  
{  
    foreach (var pair in _thumbnailMap)  
    {  
        RadioButton rb = pair.Key;  
        PictureBox pb = pair.Value;  
  
        if (rb.Checked && pb.Image != null)  
        {  
            return new Bitmap(pb.Image);  
        }  
    }  
    return null;  
}
```

- Mengecek semua pasangan `RadioButton` dan `PictureBox`:
 - Jika `RadioButton` dalam keadaan *checked* dan `PictureBox` memiliki gambar,
 - Maka dikembalikan gambar tersebut sebagai `Bitmap`.
- Jika tidak ada yang dipilih, `null` dikembalikan.
- Fungsi ini menentukan **gambar B** yang akan digunakan dalam operasi dua gambar (A dan B).

Fungsi `GetImageOperands()`

```
private bool GetImageOperands(out Bitmap imgA, out Bitmap imgB)
```

- Digunakan untuk mendapatkan **dua operand gambar** (A dan B).
- Menggunakan `out` karena nilai dikembalikan lewat parameter.

```
if (!_editorService.IsImageLoaded)
{
    MessageBox.Show("Gambar A (di kanvas utama) masih kosong...");
    return false;
}
```

- Mengecek apakah gambar A sudah dimuat di editor. Jika belum, tampilkan pesan peringatan.

```
imgB = GetImageB_FromRadioButtons();
if (imgB == null)
{
    MessageBox.Show("Gambar B belum dipilih...");
    return false;
}
```

- Mengecek apakah gambar B sudah dipilih lewat radio button.

```
imgA = _editorService.GetRestoredImage();
return true;
```

- Jika semua valid, ambil gambar A dari editor dan kembalikan `true`.

Fungsi `GetOriginalImage()`

```
private Bitmap GetOriginalImage() => _editorService.GetRestoredImage();
```

- Mengambil kembali gambar asli dari editor service.

Operasi Aritmatika

Contoh satu fungsi:

```
public Bitmap PerformAdd()
{
```

```

    if (GetImageOperands(out Bitmap imgA, out Bitmap imgB))
    {
        using (imgA) using (imgB)
            return ArithmeticOperations.Add(imgA, imgB);
    }
    return GetOriginalImage();
}

```

Penjelasan:

- Cek apakah gambar A dan B tersedia.
- Jika ada, lakukan operasi Add dari ArithmeticOperations.
- using memastikan gambar dibuang dari memori setelah selesai.
- Jika gagal, tampilkan gambar asli.

Hal yang sama berlaku untuk:

- PerformSubtract() → Kurangi tiap piksel gambar A dan B.
- PerformMultiply() → Kalikan nilai piksel.
- PerformDivide() → Bagi nilai piksel.

✖ Operasi Logika

Contoh:

```

public Bitmap PerformAnd()
{
    if (GetImageOperands(out Bitmap imgA, out Bitmap imgB))
    {
        using (imgA) using (imgB)
            return LogicalOperations.And(imgA, imgB);
    }
    return GetOriginalImage();
}

```

- And, Or, Xor melakukan operasi logika biner antar gambar.
- PerformNot() hanya bekerja pada satu gambar (A) → membalik warna (*invert*).

❓ Fungsi Membaca Konstanta

```

private bool TryGetConstant(out double valueK)
{
    if (double.TryParse(_constantTextBox.Text, NumberStyles.Any,
        CultureInfo.InvariantCulture, out valueK))
    {

```

```

        return true;
    }

    MessageBox.Show("Nilai Konstanta (K) tidak valid...");
    valueK = 1.0;
    return false;
}

```

- Membaca angka dari `TextBox`.
- Menggunakan format internasional (agar . bisa dibaca sebagai desimal).
- Jika gagal, tampilkan pesan dan gunakan default 1.0.

Fungsi Mengambil Gambar A

```

private Bitmap GetImageA()
{
    if (!_editorService.IsImageLoaded)
    {
        MessageBox.Show("Gambar A (di kanvas utama) masih kosong.", "Error");
        return null;
    }
    return _editorService.GetRestoredImage();
}

```

- Mengecek apakah gambar A tersedia.
- Jika tidak, tampilkan pesan kesalahan.

Operasi Konstanta

```

public Bitmap PerformConstantMultiply()
{
    if (!TryGetConstant(out double k)) return GetOriginalImage();

    using (Bitmap imgA = GetImageA())
    {
        if (imgA == null) return null;
        return ConstantOperations.Multiply(imgA, k);
    }
}

```

- Membaca nilai k dari textbox.
- Melakukan perkalian setiap piksel gambar A dengan konstanta k .

```

public Bitmap PerformConstantDivide()
{
    if (!TryGetConstant(out double k)) return GetOriginalImage();
}

```

```
using (Bitmap imgA = GetImageA())  
{  
    if (imgA == null) return null;  
    return ConstantOperations.Divide(imgA, k);  
}
```

- Melakukan pembagian setiap piksel dengan konstanta k .

Kesimpulan

Kelas `OperationManager` ini bertugas:

1. **Mengelola input gambar A dan B.**
 2. **Menjalankan operasi aritmatika dan logika** (Add, Subtract, Multiply, Divide, And, Or, Xor, Not).
 3. **Mendukung operasi konstanta** dengan input dari `TextBox`.
 4. **Memberi peringatan interaktif** jika pengguna lupa memilih gambar atau mengisi nilai.
-